EPFL

Thèse n° 9112

Bridging the gap between model-driven and data-driven methods in the era of Big Data

Présentée le 4 novembre 2022

Faculté de l’environnement naturel, architectural et construit
Laboratoire transport et mobilité
Programme doctoral en génie civil et environnement

pour l’obtention du grade de Docteur ès Sciences

par

Gael LEDERREY

Acceptée sur proposition du jury

Prof. A. Vassilopoulos, président du jury
Prof. M. Bierlaire, Dr T. M. Hillel, directeurs de thèse
Prof. B. Farooq, rapporteur
Prof. F. Rodrigues, rapporteur
Prof. A. Alahi, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

Acknowledgements

The PhD is often seen as the ultimate goal in a student's career. Indeed, there are no additional degrees after the PhD (except a second one, maybe). However, students have many different views about it:

- Some do not even consider it and are waiting to start a career in the industry.
- Some see it as an exciting opportunity to continue to live as a student (while being paid).
- Some consider it a mandatory step in a fulfilling academic career.

During all my studies, I belonged to the first category. I remember saying many times to my friend, "*I will never do a PhD*". However, here I am, a few years later, writing the acknowledgments for my PhD thesis. So, before thanking the many people who deserve it, I would like to share a little bit of my story:

"*The story of a Master's student who never considered becoming a PhD student and happens to write these words today.*"

Everything leading to this day started many years ago, during my Master's studies. Indeed, I had to undertake multiple projects during my studies. I always thought of these projects as the best way to learn about specific topics, and I always enjoy them a lot. However, I often received the same comment from the people supervising me: "*You should trust your work more*" or "*You should learn how to work alone*". I knew this was my weakness. And it was the first time I thought to myself that a PhD could help me. Indeed, I knew I would have to overcome this issue to succeed in a PhD. However, I was still not interested in doing research. To be honest, I did not know what "research" means.
Then came the time to undertake the Master's thesis. Following an incredible course on Data Science (Thanks, Michele!), I had the pleasure of meeting Prof. Bob West, who recently became a professor at EPFL. He trusted me to become a student in his lab and gave me a fascinating project: studying two datasets about beer reviews (yes, you read that right!). This project taught me that academic research could be exciting and fun. I was convinced! The PhD will teach me how to work alone while being stimulating! Thus, I applied for a PhD with Bob. However, I was not accepted by the Computer Science school. Usually, it's at that moment that we give up. Yet, here I am today. Indeed, in life, there are always moments or people who shape your path. In my case, it was when Prof. Michel Bierlaire offered me a PhD position

after this failure. Indeed, none of this would have been possible without Michel. I owe him many thanks for giving me this opportunity.
My journey as a PhD student was a roller coaster of emotions and motivation. It is well-known that PhD studies are tough on mental health. But I never thought it would be such an ordeal. Fortunately, I had a lot of support in these last five years, and I would have never been able to write these words without the help of some incredible people! The first year of my PhD was challenging but thrilling. I went to my first conference with Bob (I even wrote a blog post about it: http://go.epfl.ch/WWW2018). I spent a lot of time supervising students and preparing courses with Virginie (now Prof. Lurkin). All the while, Nicholas (now Dr. Molyneaux), best friend and officemate then, was always listening to me complain and giving me advice (that I still follow nowadays). The mix of research, student supervision, and course organization were thrilling. But I would have never survived this time without these two! Sadly, not everything is sunshine and rainbows. After the first year, the "highway to hell" started.
The well-known unbounded motivation hole called the second and third years of PhD (look it up on Internet, it is true) became my curse. Indeed, these two years were the most difficult in terms of motivation (Github can prove it!) and mental health. And it ended (and ended me) with the quarantine during the Covid-19 outbreak. A mix of issues in my personal life and in my work put a complete stop to my progress. Yet, it's at that moment that it also became a blessing. Indeed, I got the help and support that I needed from marvelous people:

- Dr. Charpentier, my psychiatrist, made me realize many things about myself and was able to help me despite the urgency of the situation.
- Marion, my roommate, with whom I shared everything about my feelings and was always here to support me.
- Jacques and Sandrine, my parents, never gave up on me despite all the troubles I gave them and always gave me the best possible advice!
- Marija, my girlfriend (and temporary officemate), was always by my side, especially in the worst moments, and always encouraged me to take the best option for my well-being.

Without these people, I would have never been able to push through this nightmare called burnout. It was also at that moment that I started to learn to trust my work and myself. That was when I could fully enjoy the personal experience that is the PhD. So, while writing these words, I can finally say that this path was worth it! I grew and learned a lot about myself. And it would have never been possible if it wasn't for the people cited above.
There are many more people that I need to thank. And I might forget some names. So, please, forgive me. However, I cannot forget my supervisors who contributed to my success: Prof. Michel Bierlaire and Dr. Tim Hillel. First, I would like to thank Michel for offering me this position and believing in me during the more difficult times. I enjoyed the insightful discussions we had over the years, and the multiple beers shared at conferences. Then, I'd like to thank Tim, who became my supervisor while I was already struggling for my PhD. I know

I haven't been the perfect student, and I'm sorry for the harsh emails I (may have) sent. But your resilience, the advice you gave me, the long personal and professional discussions we had, and the beers and laughs shared all contributed significantly to this success. Besides my supervisors, I would like to thank the jury members who provided priceless feedback:

- Prof. Bilal Farooq and Prof. Filipe Rodrigues, the external jury members, with whom I shared a fascinating discussion during my private defense. They also gave insightful comments to improve my manuscript.
- Prof. Alexandre Alahi always liked to challenge me during all these years, often on the same subject, and with whom I shared some fantastic moments at conferences.
- Prof. Anastasios Vassilopoulos presided over the jury and made this stressing process much more bearable.

I would also like to thank all the previous and current members of the TRANSP-OR lab that I have met:

- The previous generation, aka the golden age of the lob. I want to especially thank Flurin, a great supervisor (and an inspiration to me).
- My generation with whom I spent most of my PhD years. I must admit it: Nicholas, you were the best officemate ever, even though we never agreed on the temperature in the office!
- The new generation of the lab. I'm sure you will take good care of the lab.

I also met many people during my studies, PhD, and life:

- The friends from High School, especially François and Nicholas! They have endured my craziness (and sometimes stupidities) for more than 15 years!
- The trio of students I helped during my third year of High School, specifically Kevin, who's always been an inspiration to me!
- The Physicists I met during my Bachelor's. I will never forget the "TP" and the crazy nights we had!
- The Mathematicians and CQFD. I don't think I would have enjoyed my Master's as much as I did without you!
- The friends with whom I drank way too much at Sat, Zelig, and No Name.
- The friends from Nez Rouge with whom I spent many cold nights in December playing cards!
- The friends/colleagues from dlab, LUTS, and VITA with whom I shared many discussions about life and work (and also many beers).

- The many students who trusted me to be their advisor.
- All the current and previous members of PolyDoc, and the ones who attended all our aperos. Annie, Margaux, Nick, Lucas, and Yujin, you helped me a lot during the more difficult times, and I will be forever grateful to you!
- My Serbian crew that is kind of a second family to me.
- My smoking buddies, especially Claudia and Yazan! (We can share a lot with a small cigarette break!)
- My previous and current roommates who had to deal with my ever-changing mood.
- All the virtual friends I made while spending my time on Internet, Discord, and playing games.
- The team at SWISS-SDI who believes in me for the next chapter of my life.
- And all the other friends I met during my life! (Yes, it's an easy way to make sure everybody has been thanked.)

Of course, I will be forever grateful to my parents, Jacques and Sandrine, who pushed me in the right direction when I was taking the wrong path. I wasn't always the easiest child or teenager, but they always coped with my eccentricity and helped me achieve my goals. And I will always admire my brother Yann. We have very different characters (I sometimes envy him), and I know I wasn't always the best big brother. But I've always been incredibly proud of his achievement! Finally, last but not least, I would like to thank Marija, my amazing girlfriend. These past two years have been life-saving with you by my side. I can only hope it will continue like this for a very long time (and I'm sure it will). Volim Te!
To conclude these lengthy acknowledgments (sorry about that), I would like to give final advice to whoever dared to stick with me until now: "**Take care of your mental health!**" We should only care about this. Money, fame, ambition, etc., come and go. We have only one life, and it is too precious to let it go to waste. So, go and seek help if you need it. But never let yourself wither out!

Cheseaux-sur-Lausanne, October 15, 2022 Gael Lederrey

Abstract

Data-driven and model-driven methodologies can be regarded as competitive fields since they tackle similar problems such as prediction. However, these two fields can learn from each other to improve themselves. Indeed, data-driven methodologies have been developed to use advanced methodologies based on Big Data technologies. On the other hand, model-driven methodologies concentrate on developing mathematical models based on theory and expert knowledge to allow for interpretability and control. Through three main contributions, this thesis aims to bridge the gap between these two fields by using their strengths and applying them to its counterpart.

Discrete Choice Models (DCMs) have shown tremendous success in many fields, such as transportation. However, they have not evolved to tackle the growing amount of available data. On the other hand, Machine Learning (ML) researchers have developed optimization algorithms to efficiently estimate complex models on large datasets. Similarly, faster estimation of DCMs on larger datasets would improve the efficiency of modelers as well as enable new research axes. Thus, we take inspiration from the large body of existing research in efficient parameter estimation with extensive data and large numbers of parameters in deep learning and apply it to DCMs. The first chapter of this thesis introduces the HAMABS algorithm, which combines three fundamental principles to enable faster parameter estimation of DCMs (20x speedup compared to standard estimation) without compromising the precision of the parameter estimates.

Collecting large amounts of data can be cumbersome and costly, even in the era of Big Data. For example, ML researchers in Computer Vision have been developing generative deep learning models to augment datasets. DCM researchers face similar issues with tabular data, *e.g.* travel surveys. In addition, if the collection process is not performed correctly, these datasets can contain bias, lack consistency, or be unrepresentative of the actual population. The second chapter of this thesis introduces the DATGAN, a Generative Adversarial Network (GAN) integrating expert knowledge to control the generation process. This new architecture allows modelers to generate controlled and representative synthetic data, outperforming similar state-of-the-art generative models.

Finally, researchers are increasingly developing fully disaggregate agent-based simulation models, which use detailed synthetic populations to generate aggregate passenger flows. However, detailed disaggregate socioeconomic data is usually expensive to collect and heavily restricted in terms of access and usage. As such, synthetic populations are typically either drawn randomly from aggregate level control totals, limiting their quality, or tightly controlled, limiting their application and usefulness. To combat this, the third chapter extends the DATGAN methodology to generate highly detailed and consistent synthetic populations from small sample data. First, ciDATGAN learns to generate the variables in a low-sample highly detailed dataset, *e.g.* household travel survey. It then completes a high-sample dataset with few variables, *e.g.* microdata census, by generating the previously learned variables. The results show that this methodology can correct for bias and may enable the transfer of synthetic populations to new areas/contexts.

Keywords Discrete Choice Model, Stochastic Optimization, Deep Learning, Expert Knowledge, Generative Adversarial Network, Synthetic Tabular Data, Synthetic Population

Résumé

Les méthodologies axées sur les données et celles axées sur les modèles peuvent être considérées comme des domaines concurrentiels puisqu'elles s'attaquent à des problèmes similaires tels que la prédiction. Cependant, ces deux domaines peuvent apprendre l'un de l'autre pour s'améliorer. En effet, les méthodologies orientées données ont été développées pour utiliser des méthodologies avancées basées sur les technologies Big Data. D'autre part, celles guidées par les modèles se concentrent sur le développement de modèles mathématiques basés sur la théorie et les connaissances des experts pour permettre l'interprétabilité et le contrôle. À travers trois contributions principales, cette thèse vise à combler le fossé entre ces deux domaines en utilisant leurs forces et en les appliquant à son homologue.

Les Modèles de Choix Discrets (MCD) ont connu un succès considérable dans de nombreux domaines, tels que le transport. Cependant, ils n'ont pas évolué pour faire face à la quantité croissante de données disponibles. D'autre part, les chercheurs en apprentissage automatique ont développé des algorithmes d'optimisation pour estimer efficacement des modèles complexes sur de grands ensembles de données. De même, une estimation plus rapide des MCD sur de plus grands ensembles de données améliorerait l'efficacité des modélisateurs et permettrait de nouveaux axes de recherche. Ainsi, nous nous inspirons du grand nombre de recherches existantes sur l'estimation efficace des modèles complexes d'apprentissage profond utilisant de large jeux de données et nous les appliquons aux MCD. Le premier chapitre de cette thèse présente l'algorithme HAMABS, qui combine trois principes fondamentaux pour permettre une estimation plus rapide des paramètres des MCD (20x plus rapide que l'estimation standard) sans compromettre la précision des paramètres.

La collecte de grandes quantités de données peut être fastidieuse et coûteuse, même à l'ère du Big Data. Par exemple, les chercheurs en vision par ordinateur ont développé des modèles d'apprentissage profond génératifs pour augmenter les ensembles de données. Les chercheurs en MCD sont confrontés à des problèmes similaires avec les données tabulaires, par exemple les enquêtes sur les voyages. En outre, si le processus de collecte n'est pas effectué correctement, ces ensembles de données peuvent contenir des biais, manquer de cohérence ou ne pas être représentatifs de la population réelle. Le deuxième chapitre de cette thèse présente le DATGAN, un réseau adversarial génératif (GAN) intégrant de l'expertise humaine pour contrôler le processus de génération. Cette nouvelle architecture permet aux modélisateurs de générer des données synthétiques contrôlées et représentatives, surpassant les modèles génératifs de l'état de l'art.

Enfin, les chercheurs développent de plus en plus de modèles de simulation entièrement désagrégés, à base d'agents, qui utilisent des populations synthétiques détaillées pour générer des flux de passagers agrégés. Cependant, les données socio-économiques détaillées et désagrégées sont généralement coûteuses à collecter et fortement limitées en termes d'accès et d'utilisation. En conséquence, les populations synthétiques sont, généralement, soit tirées au hasard à partir de totaux de contrôle de niveau agrégé, ce qui en limite la qualité, soit étroitement contrôlées, ce qui en limite l'application et l'utilité. Pour lutter contre ce problème, le troisième chapitre étend la méthodologie DATGAN pour générer des populations synthétiques détaillées et cohérentes à partir de petits échantillons de données. Tout d'abord, ciDATGAN apprend à générer les variables d'un ensemble de données très détaillées à faible échantillon, par exemple une enquête sur les déplacements des ménages. Il complète ensuite un ensemble de données à fort échantillonnage avec peu de variables, par exemple des microdonnées de recensement, en générant les variables apprises précédemment. Les résultats montrent que cette méthodologie permet de corriger les biais et peut permettre le transfert de populations synthétiques à de nouvelles zones/contextes.

Mots-clef Modèle de Choix Discrets, Optimisation Stochastique, Apprentissage Profond, Expertise Humaine, Réseau Adversarial Génératif, Données Tabulaires Synthétiques, Population Synthétique

Contents

List of Figures

List of Tables

Acronyms

AMABS	Adaptive Moving Average Batch Size.
AUS	Assisted Utility Specification.
BFGS	Broyden-Fletcher-Goldfarb-Shanno.
ciDATGAN	conditional inputs DATGAN.
CMAP	Chicago Metropolitan Agency for Planning.
CNN	Convolutional Neural Network.
CTAB-GAN	Conditional TABular GAN.
CTGAN	Conditional Tabular GAN.
DAG	Directed Acyclic Graph.
DATGAN	Directed Acyclic Tabular GAN.
DCM	Discrete Choice Model.
ELBO	Evidence Lower-BOund.
ERGM	Exponential Random Graph Model.
FCNN	Fully Connected Neural Network.
FFNN	Feed-Forward Neural Network.
GAN	Generative Adversarial Network.
GDPR	General Data Protection Regulation.
GNN	Graph Neural Network.
HAMABS	Hybrid Adaptive Moving Average Batch Size.
IPF	Iterative Proportional Fitting.
JS	Jensen-Shannon.

KL	Kullback-Leibler.
LPMC	London Passenger Mode Choice.
LSTM	Long Short-Term Memory.
LTDS	London Travel Diary Survey.
MAE	Mean Absolute Error.
MCMC	Markov Chain Monte Carlo.
MEV	Multivariate Extreme Value.
ML	Machine Learning.
MTMC	Mobility and Transport MicroCensus.
RMSE	Root Mean Squared Error.
SGD	Stochastic Gradient Descent.
SNM	Stochastic Newton Method.
SRMSE	Standardized Root Mean Squared Error.
STR	Stochastic Trust-Region.
TGAN	Tabular GAN.
TVAE	Tabular VAE.
VAE	Variational AutoEncoder.
VGM	Variational Gaussian Mixture.
VNS	Variable Neighborhood Search.
VoT	Value of Time.
WGAN	Wasserstein GAN.
WMA	Window Moving Average.

1 Introduction

1.1 Context and motivation

Innovation is often regarded as a concept defined on a single axis between scientists revolutionalizing the world with new theories and business professionals delivering new tools or business models to existing markets. However, Satell (2017) instead defines innovation on a 2D scale depending on the domain definition and the problem definition.

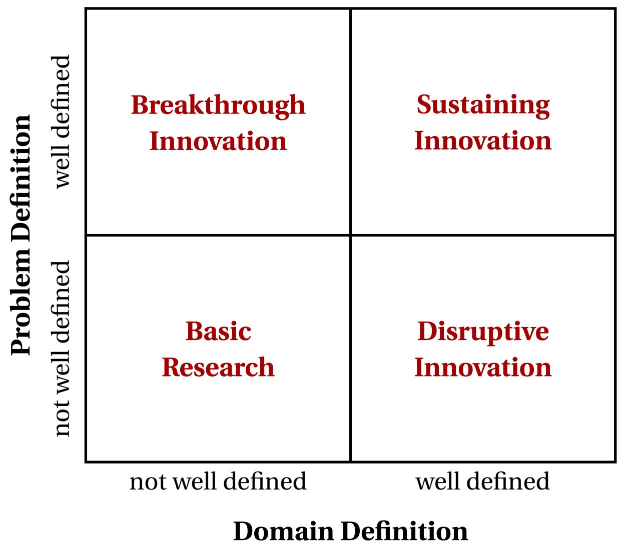


Figure 1.1: Innovation matrix defined by Satell (2017).

He proposes to classify innovation into four quadrants. Sustaining innovations are linked to business-related innovations since low risks are involved. The opposite of sustaining innovations is basic research. This type of innovation is about developing new technologies for future applications, generally linked to academic researchers. However, the most exciting innovations generally occur either when new technologies are applied to existing applications (disruptive innovations) or when existing technologies are applied to new markets (breakthrough innovations). This thesis aims to achieve both by bridging the gap between data-driven and model-driven approaches. The main difference, in academia, compared to the analysis of Satell (2017) is that researchers are not necessarily aiming to bring these innovations to new markets. Instead, the goal is to provide researchers with new tools to expand research in existing or new domains.

Model-driven methodologies use theory and expert knowledge to build a tailored mathematical model to understand a certain phenomenon. Discrete Choice Models (DCMs) (Ben-Akiva and Lerman, 1985) are a prime example of model-driven methodologies. The goal of such models is to understand the choice of individuals amongst a finite set of alternatives. These probabilistic models have two main use-cases: understanding the choice of individuals and forecasting future choices. On the other hand, data-driven methodologies, including Machine Learning (ML) and related fields, use general purpose models fed with specific data for the application. The common rule about ML is: *more data leads to better results*, as long as the quality remains the same. Since data plays a preponderant role in this field, expert knowledge is not necessarily required. However, this lack of knowledge about the data or the application can lead to severe limitations, *e.g.* overfitting or loss of predictive power. Thus, the combination of domain knowledge with ML methodologies, such as physics-informed ML, has become a popular research topic to overcome these limitations. Similarly, this thesis aims at combining ML methodologies and modeling techniques to overcome these limitations.

The gap between these two fields can be bridged in two directions. The first direction is to use the extensive data knowledge of ML methods and apply it to choice modeling. For example, DCMs are estimated using standard optimization algorithms, thus, limiting the usage of large datasets since these algorithms are not suited for large models with extensive datasets. On the other hand, in ML, researchers have developed efficient stochastic algorithms to train deep neural networks containing millions of parameters on enormous datasets. Faster DCMs estimation would make the modeling process faster and create new research axes. For example, DCMs require the modeler to define the utility specification for their model. This process can be cumbersome and repetitive since there are constraints on the variables that need to be fulfilled. Researchers build many models iteratively and estimate them before choosing the most suitable one. Therefore, if researchers can estimate the DCMs more efficiently, they could use external techniques to help them build the model. Ortelli et al. (2021) developed an assisted specification framework for DCMs using optimization techniques. This methodology showed encouraging results. However, more advanced data-driven methodologies could improve assisted specification. While this research axis is out of the scope of this thesis, it shows that applying data-driven methodologies to choice modeling can lead to exciting new research directions.

The second direction uses expert knowledge from modeling practices and integrates it into data-driven methodologies. Indeed, ML algorithms are specialized in the use of data. However, they generally lack interpretability and control. For example, complex ML models, such as deep neural networks, are considered “black-box” models with some inputs and outputs. ML users are working on providing the best possible data to the model through data wrangling and feature engineering. However, ultimately, only the final output matters and little control is used during the training process. Thus, adding expert knowledge to ML models, as it is done in choice modeling, could add more control over these models and make them more interpretable. Furthermore, the control brought by expert knowledge with the processing power of data-driven methodologies could lead to significant improvements depending on the application. For example, in the case of synthetic population generation, both data processing and control are required. Indeed, individuals are characterized by many attributes. Thus, a model capable of handling such complexity in the data is required. On the other hand, control over the model allows the modeler to test multiple hypotheses by generating multiple synthetic populations.

1.2 Summary of contributions

This thesis aims to bring new algorithm frameworks for modelers so they can use the new technologies that arose during this Big Data era. Thus, the contributions proposed in this thesis have been defined around this idea while using knowledge from model-driven and data-driven methodologies. The first contribution is about the estimation of choice models.

1. **Efficient estimation of complex choice models on large datasets** (Chapter 2): As stated in the previous section, choice models software are currently using standard non-linear optimization methods to estimate the models. It significantly limits the models' size and the datasets that can be used. In this era of Big Data, data is ever more available. Thus, this bottleneck prevents modelers from taking advantage of this era fully. On the other hand, ML researchers have been developing very efficient stochastic optimization algorithms to tackle this issue with even more complex models such as deep neural networks. Therefore, this contribution is inspired by the latter while ensuring that the choice model's constraints are met. Indeed, contrary to ML models, DCMs require absolute precision on the parameters since they are used to analyze individuals' behavior. Therefore, this thesis presents the Hybrid Adaptive Moving Average Batch Size (HAMABS) optimization algorithm combining three distinct principles: (i) the use of second-order stochastic methods, (ii) the adjustment of the batch size when the algorithm is stalling, and (iii) the hybridization between algorithms. This new method enables much faster parameter estimation without compromising the precision of the parameter estimates. It thus allows researchers to train many more models in the same amount of time compared to the standard methodologies. Projects such as Assisted Utility Specification (AUS) (Ortelli et al., 2021) considerably benefit from such improvements.

This first contribution allows modelers to estimate DCMs on large datasets in a significantly shorter time, allowing them to use much larger datasets for their model. Indeed, the collection of datasets has been more accessible in recent years thanks to the technologies developed during this Big Data era. However, this does not apply to every type of dataset. For example, household travel surveys are collected by contacting the desired population. It is a long, cumbersome and costly process. Thus, synthetic data can augment existing surveys and faciliate the acquisition of more data. The second contribution focuses on providing easily accessible and sharable data for modelers using data-driven technologies and expert knowledge.

2. **Generating synthetic data from deep learning with expert knowledge** (Chapter 3): This chapter focuses on developing a new generative model based on data-driven methodologies tailored for modelers. Indeed, deep learning models, such as Generative Adversarial Networks (GANs), have shown impressive results when generating synthetic data. However, there are two main issues with these ML models: (i) users do not have any control on the generation process; (ii) models are generic, *i.e.* they are independent of the data and the application. Users only have to feed an original dataset to the model, train it, and obtain generated synthetic data. This contribution integrates expert knowledge by modeling the causal links between the variables in the original dataset in deep learning generative models. It allows modelers to use their data knowledge to generate more representative synthetic data. In addition, modelers want to test multiple hypotheses on given data. Therefore, controlling the generation process provides cheap and easily accessible alternatives to the original dataset. The Directed Acyclic Tabular GAN (DATGAN), presented in this chapter, generates more representative synthetic data compared to state-of-the-art generative models found in the literature. In addition, the expert knowledge added to this model allows users to generate hypothetical datasets that cannot be collected.

In addition to being less expensive, synthetic data have many other use cases, such as bias correction or privacy preservation. Indeed, microdata, such as population census, are challenging to obtain due to privacy issues restricted by the General Data Protection Regulation (GDPR). Thus, synthetic populations are used instead of the original data. For example, agent-based simulation makes extensive use of synthetic population. However, these populations are typically either drawn randomly from aggregate level control totals, limiting their quality, or tightly controlled, limiting their application and usefulness. Thus, the third contribution provides an accessible and efficient framework for generating synthetic populations.

3. **Generation of detailed synthetic populations using deep learning** (Chapter 4): The first step for generating a synthetic population is to ensure that the generated population is representative of the current population it is reproducing. Thus, a generative model tailored for generating representative synthetic data will perform well. DATGAN is the perfect initial step for this contribution. However, generating a synthetic population requires precise control over the data. For example, one might want to correct the age bias if the original census was not collected appropriately. Therefore, the conditional inputs DATGAN (ciDATGAN) integrates the use of microdata during the sampling process.

> The idea is to train ciDATGAN on a highly detailed low-sample dataset, such as a travel survey. Then, the user provides a low detailed high-sample dataset during the sampling phase to control the aggregate totals. This new model uses data-driven methodologies while providing much control to the user. Indeed, the second dataset can correct the bias in the first dataset or generate specific populations that do not exist. In addition, the time efficiency of this model means that modelers can create multiple hypothetical and highly tailored synthetic populations to test different scenarios in their research.

In combination, the three contributions raise the capabilities of existing methodologies to gain insights from large datasets and lower the barriers of entry for highly detailed modeling in data and computing resources. In turn, this opens up several exciting avenues for future applications and investigations of both data-driven and model-driven approaches.

In addition, the implementation details for each of the three main methodological contributions are either already openly available on Github and Pipy, or will be made available on publication of the associated paper.

1.3 Outline

The structure of this thesis is described in the following paragraphs and aims to emphasize the contributions presented above. The core of this thesis is composed of three chapters, followed by the concluding chapter.

Chapter 2 discusses the estimation of DCMs using standard deterministic optimization methods and new methods inspired by the optimization of ML. First, we look at the gaps in the literature for the optimization of DCMs. Then, we introduce the concepts of adaptive batch size and hybridization and test them against more conventional methods. A total of fifteen different optimization methods are tested against each other on ten different models with a different number of parameters and observations. Finally, the best algorithm is tested against the current method used in a state-of-the-art DCM estimation package.

This chapter is based upon the following publication:

> Lederrey, Gael, Lurkin, Virginie, Hillel, Tim and Bierlaire, Michel. Estimation of discrete choice models with hybrid stochastic adaptive batch size algorithms. *Journal of Choice Modelling*, 38:100226, March 2021. ISSN 1755-5345. doi 10.1016/j.jocm.2020.100226. https://www.sciencedirect.com/science/article/pii/S1755534520300257.

Chapter 3 presents a new GAN architecture for generating synthetic tabular data. The DATGAN integrates expert knowledge via a Directed Acyclic Graph (DAG) that represents the causal links between the variables in the training dataset. Each node in the DAG is associated with a Long Short-Term Memory (LSTM) cell that retains information from the previous cells in the graph. We test this new model against multiple state-of-the-art generative models for generating synthetic tabular data on both statistical and ML efficacy metrics. Results show that DATGAN outperforms all these models on both metrics, allowing for more flexibility when generating synthetic data.

This chapter is based upon the following preprint that has been submitted to the *Journal of Transportation Research Part C: Emerging Technologies*:

> Lederrey, Gael, Hillel, Tim and Bierlaire, Michel. DATGAN: Integrating expert knowledge into deep learning for synthetic tabular data. *arXiv:2203.03489 [cs]*, March 2022. https://arxiv.org/abs/2203.03489. arXiv: 2203.03489.

Chapter 4 improves on the DATGAN model presented in the previous chapter to generate synthetic populations. The model has been upgraded to include conditionality, *i.e.* conditional values can be fed to the generator to influence the sampling process. We show that the ciDATGAN can learn from highly detailed datasets with few samples and complete a large dataset with low details. The model can control some variables and effectively remove bias from existing datasets. Finally, we show that ciDATGAN can generate previously unseen individuals without loss of representativity compared to DATGAN.

This chapter is based upon the following technical report:

> Lederrey, Gael, Hillel, Tim and Bierlaire, Michel. ciDATGAN: Conditional Inputs for Tabular GANs. *arXiv:2210.02404 [cs]*, October 2022. https://arxiv.org/abs/2210.02404. arXiv: 2210.02404.

Chapter 5 concludes this thesis by summarizing the contributions and discussing potential ideas for future research.

2 Efficient estimation of complex choice models on large datasets

This chapter is based upon the following publication:

> Lederrey, Gael, Lurkin, Virginie, Hillel, Tim, Bierlaire, Michel. Estimation of discrete choice models with hybrid stochastic adaptive batch size algorithms. *Journal of Choice Modelling*, 38:100226, March 2021. ISSN 1755-5345. doi 10.1016/j.jocm.2020.100226. https://www.sciencedirect.com/science/article/pii/S1755534520300257.

2.1 Introduction

The availability of more and more data for choice analysis is both a blessing and a curse. On the one hand, this data provides analysts with a great wealth of behavioral information. On the other hand, processing the increasingly large and complex datasets to estimate Discrete Choice Models (DCMs) presents new computational challenges. Nevertheless, the Machine Learning (ML) community has been thriving while dealing with vast amounts of data. It, therefore, seems natural to investigate ML optimization algorithms to estimate DCMs.

The predominant approach of these algorithms is the *stochastic gradient*. It approximates the gradient of the log likelihood function (or any goodness of fit measure) using only a small subset of the dataset. The gradient is said to be stochastic because the subset of data used to calculate it is drawn randomly from the entire dataset. A version of the steepest descent algorithm using this stochastic gradient is then applied to maximize the log likelihood function. Many variants have been proposed around this primary principle.

To illustrate the stochastic gradient, we consider applying stochastic gradient descent to a DCM with J alternatives, and a dataset of N observations, each of them containing the vector of explanatory variables x_n and the observed choice i.

$$P_n(i|x_n;\theta) \tag{2.1}$$

provides the probability that individual n chooses alternative i in the context specified by x_n, where $\theta \in \mathbb{R}^K$ is a vector of K unknown parameters, to be estimated from data. Typically, this is done using maximum likelihood estimation, where the log likelihood function $\mathscr{L}(\theta)$ is maximized:

$$\max_{\theta\in\mathbb{R}^K} \mathscr{L}(\theta) = \max_{\theta\in\mathbb{R}^K} \sum_{n=1}^{N} \ln P_n(i|x_n;\theta). \tag{2.2}$$

In the optimization literature, it is custom to define the algorithms for minimization problems. We follow the same convention, and consider the equivalent minimization problem:

$$\min_{\theta\in\mathbb{R}^K} -\mathscr{L}(\theta) \tag{2.3}$$

The gradient of the log likelihood function is

$$\nabla\mathscr{L}(\theta) = \sum_{n=1}^{N} \nabla \ln P_n(i|x_n;\theta). \tag{2.4}$$

To obtain its stochastic version, we draw randomly, without replacement, a subset of N' observations from the data that we call a *batch*, and calculate:

$$\nabla_{N'}\mathscr{L}(\theta) = \sum_{n\in N'} \nabla \ln P_n(i|x_n;\theta). \tag{2.5}$$

As shown in the above analysis, the variants of the stochastic gradient methods that are successful in ML can be used to estimate the parameters of DCMs. They can decrease the time and computational cost of estimating DCMs on large datasets. However, three critical differences between the two contexts must be considered. Firstly, the parameters in DCMs are a substantial output and are used to estimate behavioral indicators such as Values of Time (VoT) and elasticities for the population. Conversely, parameters in ML models typically have no behavioral interpretation, and only the model predictions are treated as a modeling output. It is, therefore, typical to allow choice model parameter estimates to converge during estimation to obtain the highest accuracy and precision of each parameter estimate. Conversely, in ML, it is typical to restrict the model from converging fully on the training data to prevent overfitting due to the overparametrization of ML models. This can be achieved by using early stopping, *e.g.* we stop the optimization process when the model achieves the highest performance on an out-of-sample validation set.

Secondly, DCMs tend to have fewer parameters than ML models. Complex DCMs have hundreds of parameters, while the neural networks used in deep learning can involve millions of unknown parameters. Recent efforts have been made to reduce the number of parameters in ML models. For example, Wu (2019) investigates simplifying Convolutional Neural Networks (CNNs). The author can reduce the number of parameters using matrix decompositions from three million to just above three thousand, with only a small loss of precision. Nonetheless, these models are still complex and exceed the usual number of parameters used in DCMs.

Finally, choice data is typically collected using specific sampling strategies and designs of experiments. The objective is to obtain a representative sample while avoiding redundancies. Furthermore, the analyst wants to test specific hypotheses when designing the data collection. In contrast, ML techniques are often applied to datasets collected automatically or initially collected for other purposes, *e.g.* trip records from contactless payment cards, to detect patterns that have not previously been considered. It is, therefore, common to have a great deal of redundancy in ML data.

Therefore, we introduce a new algorithmic framework for estimating DCMs, which addresses the key differences between the DCM and ML contexts. Our framework includes three primary contributions:

- the use of a stochastic Hessian (that is, second derivative matrix), which is possible thanks to the relatively low number of unknown parameters in discrete DCMs,
- the possible modification of the batch size from iteration to iteration, which is used to allow the high accuracy and precision in individual parameter estimates required in DCMs,
- a change of optimization algorithm depending on the batch size, aiming to find the best trade-off between accuracy and efficiency.

The rest of the chapter is laid out as follows. In the next section, we describe optimization algorithms used both for estimating DCMs and in ML. Then, in Section 2.3, we present in detail the three ideas mentioned above and propose a catalog of optimization algorithms for estimating DCMs. In Section 2.4, we evaluate the performance of various algorithms on various DCMs with large datasets. Finally, in Section 2.5, we conclude this chapter and mention some further ideas that can be investigated in the future.

2.2 Literature review

To the best of our knowledge, the maximum likelihood estimation of the parameters of DCMs exclusively relies on deterministic algorithms that are variants of the line search and trust-region methods. In particular, the main estimation packages written in Python (Pandas Biogeme (Bierlaire, 2003, 2018), PyLogit (Brathwaite et al., 2017), and Larch (Newman et al., 2018)) all use the `minimize` function from the package Scipy (Jones et al., 2014). This makes use of the quasi-Newton Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. While this performs well for estimating small-to-medium-sized models, it struggles with larger, more complex models. With the availability of larger and larger datasets, the performance of these standard methods is completely dominated by the time it takes to calculate the log likelihood function, its gradient, and its possible second derivative matrix. For example, Hillel (2019) shows that fitting a Feed-Forward Neural Network (FFNN) in the Tensorflow Python library is up to 200 times faster than estimating a Nested Logit model in Pandas Biogeme on the same dataset containing 81'086 observations, despite the former having far more parameters.

To understand how we may be able to estimate DCMs more quickly on large datasets, we can look for inspiration from ML. Datasets used in ML, particularly in Computer Vision, can contain millions of observations. For example, ImageNet, a collection of images labeled by hand, contains around fifteen million images. The sheer size of the whole dataset (≈150GB) prevents it from being stored in memory. As such, analyzing the dataset is a significant computational challenge. As the entire dataset cannot be stored in memory, the data must be batch processed. It explains the importance of the stochastic approach for ML, *i.e.* calculate the gradient on a batch of data at each iteration. To better understand the existing approaches used to estimate DCMs and ML models, we conduct a literature review of existing stochastic algorithms in Section 2.2.1. Then, in Section 2.2.2, we identify and discuss the literature gaps for optimizing DCMs. We also link them to the three primary contributions presented in Section 2.1.

2.2.1 Overview of existing studies

In this section, we identify and review twenty-six studies that propose stochastic optimization algorithms, see Table 2.2. The selected studies, while not exhaustive, are believed by the authors to represent a broad overview of the existing optimization algorithms. For each of these algorithms, we analyze four aspects:

- the mathematical order, *i.e.* if it uses a gradient-based (first-order), a quasi-Newton-based (1.5-th order), or a Newton-based step (second-order),
- if it uses an adaptive batch size technique or if the batch size is constant,
- if an experimental (numerical) assessment of the algorithm is conducted,
- a summary of the numerical applications of the algorithm.

Table 2.2: Stochastic optimization algorithms included in the literature review in chronological order. "ABS" stands for Adaptive Batch Size and "Exp." for experimental assessments.

Reference	Name	Order	ABS	Exp.	Application
Robbins and Monro (1951)	SGD	1st			Regression functions

Continues on next page...

Table 2.2 – continued from previous page

Reference	Name	Order	ABS	Exp.	Application
Polyak (1964)	Momentum	1st			Classical optimization
Nesterov (1983)	NAG	1st			Convex optimization
Polyak and Juditsky (1992)	Averaging	1st			Classical optimization
Bordes et al. (2010)	SGDQN	1.5th		✓	Dense and sparse large datasets (PASCAL Large Scale challenge)
Martens (2010)	HF	2nd		✓	Image classification via Deep Learning models
Duchi et al. (2011)	Adagrad	1st		✓	Image, text, and handwritten digit classification
Zeiler (2012)	Adadelta	1st		✓	Handwritten digit classification
Tieleman and Hinton (2012)	RMSProp	1st			None, first shown in a lecture
Schmidt et al. (2013)	SAG	1st		✓	Binary classification
Kiros (2013)	Stochastic HF	2nd		✓	Classification and deep autoencoder tasks
Defazio et al. (2014)	SAGA	1st		✓	Handwritten digit, binary, and multivariate classification
Kingma and Ba (2014)	Adam & AdaMax	1st		✓	Logistic Regression, Neural Networks, and Convolutional Neural Networks
Wang et al. (2014)	SQN & RSQN	1.5th		✓	Convex and non-convex optimization problems
Mokhtari and Ribeiro (2014)	RES	1.5th		✓	Well- and ill-conditioned problems with large scale datasets
You and Xu (2014)	SHF	2nd		✓	Speech recognition
Dozat (2016)	Nadam	1st		✓	Handwritten digit classification
Balles et al. (2016)	CABS	1st	✓	✓	Convolutional Neural Networks

Continues on next page...

Table 2.2 – continued from previous page

Reference	Name	Order	ABS	Exp.	Application
Keskar and Berahas (2016)	adaQN	1.5th		✓	Recurrent Neural Networks
Agarwal et al. (2016)	LiSSA	2nd		✓	Handwritten digit and multivariate classification
Mutny (2016)	ISSA	2nd		✓	Least square estimators
Devarakonda et al. (2017)	AdaBatch	1st	✓	✓	Multiple Neural Networks architecture
Ye and Zhang (2017)	AccRegSN	2nd		✓	Least square regressions
Gower et al. (2018)	hBFGS	1.5th		✓	Matrix Inversion problems
Bollapragada et al. (2018)	PBQN	1.5th	✓	✓	Logistic Regressions and Neural Networks
Reddi et al. (2018)	AMSGrad & AdamNc	1st		✓	Logistic Regression and Neural Networks

Of the twenty-six algorithms considered in this review, fourteen are first-order, six are quasi-Newton (1.5th order), and six are second order. We refer the reader to the article of Ruder (2016) for a precise overview of the first-order methods. While this review focuses on quasi-Newton and second-order algorithms, the literature focuses predominantly on first-order approaches. We believe this focus is due to the speed of first-order algorithms for optimizing large, complex neural networks.

Three of the six second-order algorithms use the Hessian-Free (truncated Newton) optimization technique. This technique approximates the problem using the second Taylor expansion and then solves it using a conjugate gradient. The computation of the Hessian is approximated with a directional derivative and finite differences. The remaining three second-order stochastic methods use a subsampling method of the Hessian to avoid its heavy computation at each step.

Most algorithms use a fixed batch size, with only three out of the twenty-six algorithms using an adaptive batch size technique. All three articles that use an adaptive batch size are recent (2016-2018). None of the second-order methods and only one of the quasi-Newton algorithms use adaptive batch size.

In terms of the analysis within the study, only five references do not provide numerical assessments of their algorithms. The first four references, which do not include a numerical application, are the oldest considered (1951-1992). The fifth reference, the algorithm RMSProp (Tieleman and Hinton, 2012), does not provide any theoretical or numerical results since it was only presented in a lecture. As per standard practice in most of the literature, we present numerical measures in the form of quantitative results for each algorithm. Finally, we can see that the algorithms in the study have been applied in multiple domains, but none explicitly for choice modeling. The earliest algorithms (4/26) were first applied to classical optimization problems. Then, the remainder of the algorithms is applied to ML problems, with the majority (13/26) applied to neural networks. This shows the predominant focus on optimizing neural networks in the literature.

2.2.2 Gaps in knowledge and contributions

As shown by the results of the literature review, the predominant focus of existing optimization research has been the optimization of neural networks, with none of the algorithms explicitly designed for the optimization of DCMs. As discussed, there are substantial differences between the optimization of neural networks and DCMs. In this section, we, therefore, assess the limitations of the existing algorithms in terms of optimizing DCMs. Furthermore, we identify three gaps in knowledge in the existing research, which may enable higher-performing optimization algorithms for DCMs.

Stochastic second order approaches

ML researchers have predominantly focused on first-order stochastic algorithms due to their speed when estimating parameters in large models. Lederrey et al. (2018a) show that first-order stochastic methods cannot achieve convergence on a logit model with ten parameters. The authors compare a gradient descent algorithm, a mini-batch SGD, Adagrad (Duchi et al., 2011), and SAGA (Defazio et al., 2014). They note that normalizing the parameters leads to more accurate results. However, they do not reach a sufficiently high precision in a reasonable time. This failure is mainly due to the required precision for convergence. As stated earlier, the parameter values are a key output of DCMs. These parameters are used to compute behavioral indicators such as the VoT and elasticities. Therefore, it is critical to achieve convergence with high precision.

Of the six second-order stochastic algorithms reviewed, none calculate the exact Hessian, with all using an approximation instead. However, the smaller number of parameters used in DCMs compared to ML models means computing the full Hessian for a batch of data is less computationally complex. Therefore, there is a need to investigate second-order stochastic approaches that compute the exact Hessian. The computation of the full Hessian could be used for DCMs to obtain parameter estimates with the necessary precision for convergence.

Adaptive batch size

None of the second-order methods found in the literature use an adaptive batch size. Furthermore, among the three algorithms proposing adaptive batch size methods, they all couple the batch size with the *learning rate*, a parameter specified in ML estimation. While Goyal et al. (2018) shows that these two parameters are related, this coupling specifically targets ML models, where the learning rate is often set to a fixed value or slightly decreases at each iteration.

The learning rate in ML is similar to the *step size* used in classical optimization problems. Typically, in classical optimization, the step size is computed using more advanced techniques such as line search or trust-region methods. However, since a high degree of precision is required for DCMs, especially at the later stage of the optimization process, the step size should be separated from the batch size. Indeed, close to the optimum value, the optimization algorithm should use a small step and as many data points as possible to achieve the highest precision.

While line search and trust-region methods have already been applied to ML, as demonstrated by Rafati et al. (2018), they have not yet been used in combination with adaptive batch size. Also, Lederrey et al. (2018a); Lederrey et al. (2018b) have demonstrated that the use of a complete batch of data is required at the end of the optimization process to achieve the appropriate precision for DCMs. Therefore, it is required to develop an adaptive batch size technique with a second-order approach that does not interfere with the step size and eventually reaches the whole dataset, as needed to achieve the required precision for DCMs (Lederrey et al., 2018b).

Combined first and second order approaches

All the algorithms presented in the review use the same algorithm throughout the optimization process. However, this can hinder the performance of the optimization since the requirements change during the process. Indeed, an optimization can be faster and less precise at the beginning of the optimization and then become more precise, and slower, close to the optimal solution. This can be partially achieved by using an adaptive batch size technique. However, this could be further addressed by switching the optimization algorithm at the right time to cope with the increased complexity due to larger batch size. There is, therefore, a need to investigate hybrid algorithms, switching between first and second-order approaches at the appropriate point in the optimization process.

Contributions

While researchers have already investigated solutions for some of the aforementioned limitations individually, we could not find any research investigating their combination in a systematic approach. We thus aim to combine the three primary contributions stated in the introduction, Section 2.1, in a new algorithm for optimizing DCMs. Thus, our algorithm incorporates the following concepts:

- the use of a second-order stochastic approach computing the Hessian for each batch,
- the use of adaptive batch size for second-order approaches which do not couple the batch size to the learning rate,
- the use of a hybrid approach that combines first and second-order algorithms and switches between them at the appropriate time.

2.3 Methodology

This section introduces our novel algorithmic framework for estimating DCMs. First, Sections 2.3.1 and 2.3.2 provide a reminder of the two standard optimization techniques: *line search* and *trust regions*. Then, Section 2.3.3 shows how we construct the new hybrid stochastic algorithms with adaptive batch size and Section 2.3.4 summarizes the algorithms presented in the methodology. Table A.1, in the appendix, provides a summary of the notations used in this methodology.

2.3.1 Line search methods

Line search optimization methods combine a descent direction with a line search. The iterates are $\theta_{k+1} = \theta_k + \alpha_k d_k$, where d_k is a descent direction obtained by preconditioning the gradient of $\mathscr{L}$ and α_k is the step size. d_k is defined as:

$$d_k = -D_k \nabla \mathscr{L}(\theta_k) \tag{2.6}$$

where D_k is a positive definite matrix.

There are typically three ways to select D_k:

- **Steepest descent** methods define D_k as the identity matrix. In that case, the descent direction is the opposite of the gradient.
- **Newton's method** assumes that D_k is the inverse of the second derivative matrix (possibly perturbed to make it definite positive).
- Quasi-Newton methods assume that D_k is a secant approximation of (the inverse of) the second derivative matrix, updated at each iteration. Among the many secant methods, we consider the **BFGS** algorithm for which, according to Fletcher (1987), the approximation is given by:

$$B_k = B_{k-1} + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_{k-1} s_k s_k^T B_{k-1}^T}{s_k^T B_{k-1}, s_k}, \tag{2.7}$$

with $s_k = \theta_k - \theta_{k-1}$ and $y_k = \nabla\mathscr{L}(\theta_k) - \nabla\mathscr{L}(\theta_{k-1})$.

A slightly different version of BFGS consists in approximating the inverse of the Hessian. In that case, the name **BFGS^{-1}** is used and the approximation is given by:

$$B_k^{-1} = B_{k-1}^{-1} + \frac{\left(s_k^T y_k + y_k^T B_{k-1}^{-1} y_k\right)\left(s_k s_k^T\right)}{\left(s_k^T y_k\right)^2} - \frac{B_{k-1}^{-1} y_k s_k^T + s_k y_k^T B_{k-1}^{-1}}{s_k^T y_k}. \tag{2.8}$$

The step size α_k is calculated with an inexact line search method, such that it verifies the two Wolfe conditions (Wolfe, 1969, 1971). The first condition guarantees that the step gives a sufficient decrease in the objective function, while the second one makes sure that unacceptably small steps are ruled out. Both Wolfe conditions are given in Equation 2.9.

$$\begin{aligned} &\text{Wolfe 1:} && \mathscr{L}(\theta_k + \alpha_k d_k) \le \mathscr{L}(\theta_k) + \beta_1 \alpha_k d_k^T \nabla\mathscr{L}(\theta_k) \\ &\text{Wolfe 2:} && \frac{\nabla\mathscr{L}(\theta_k + \alpha_k d_k)^T d_k}{\nabla\mathscr{L}(\theta_k)^T d_k} \le \beta_2 \end{aligned} \tag{2.9}$$

with $0 < \beta_1 < \beta_2 < 1$.

2.3.2 Trust-region methods

Trust-region methods define a region around the current search point, where a quadratic approximation of the function value is "trusted" to be correct, and steps are chosen to optimize the function model within this region. There exist multiple ways to compute the quadratic approximation of the function value. The most common choice is a quadratic function of the type:

$$m_k(\theta_k + d_k) = \mathscr{L}(\theta_k) + \nabla\mathscr{L}(\theta_k)^T d_k + \frac{1}{2} d_k^T B_k d_k \tag{2.10}$$

where B_k is either the Hessian $\nabla^2\mathscr{L}(\theta_k)$ or an approximation of it. If the Hessian is chosen, the name **trust-region** algorithm is used. If an approximation of the Hessian is used it is referred to as a **quasi-Newton trust-region**.

The size of the trust region is modified during the search, based on how well the quadratic model agrees with the actual objective function value. Following Conn et al. (2000), the region is modified conditional to the ratio ρ_k of the actual function reduction to the reduction predicted by the quadratic model:

$$\rho_k = \frac{\mathscr{L}(\theta_k) - \mathscr{L}(\theta_k + d_k)}{m_k(\theta_k) - m_k(\theta_k + d_k)}. \tag{2.11}$$

Given the ratio ρ_k, the decision to change the trust region is based on the following rules:

$$\Delta_k = \begin{cases} \gamma_1\Delta_k & \text{if } \rho_k \geq \eta_2, \\ \Delta_k & \text{if } \rho_k \in [\eta_1, \eta_2), \\ \gamma_2\Delta_k & \text{if } \rho_k < \eta_1, \end{cases} \tag{2.12}$$

where $\gamma_1, \gamma_2, \eta_1$ and η_2 are all *a priori* defined parameters.

Intuitively, when the quadratic model is a good predictor of the function value, the ratio ρ_k is close to 1, and the region is not modified. In contrast, when the quadratic model is no longer a good predictor, the ratio is too small or too large, and the region is reduced or extended.

2.3.3 Hybrid stochastic algorithms with adaptive batch size

We present our new algorithmic framework for optimization of DCMs by discussing its three main contributions:

- the use of a stochastic hessian,
- the possible modification of the batch size from iteration to iteration,
- the change of optimization algorithm depending on the size of the batch.

Stochastic Hessian

Inspired by the technique of stochastic gradient, our algorithmic framework includes a stochastic Hessian, defined as:

$$\nabla^2_{N'}\mathscr{L}(\theta) = \sum_{n\in N'} \nabla^2 \ln P_n(i|x_n;\theta), \tag{2.13}$$

where N' is a subset of observations, drawn randomly, without replacement, from the full dataset.

A stochastic Hessian can be included in non linear optimization algorithms to create a **Stochastic Newton Method (SNM)**, as in Lederrey et al. (2018b), or a **Stochastic Trust-Region (STR)** algorithm.

Adaptive Batch Size

Intuitively, this technique increases the batch size when the algorithm can no longer improve the value of the objective function. This generally happens for two reasons. First, either a local optimum in the current neighborhood has been reached, or the stochastic nature of the gradient and Hessian precludes the algorithm from making further progress.

We propose to modify the batch size as the algorithm proceeds with a Window Moving Average

(WMA) technique. WMA averages the values of a time series across a window of W consecutive observations, thereby generating a series of averages. More importance is given to the most recent iterations to capture change in the data better. The WMA at the k-th iteration is given by:

$$\mathrm{WMA}_{k,W} = \begin{cases} \dfrac{\sum_{i=0}^{W-1}(W-i)\mathscr{L}(\theta_{k-i})}{\sum_{i=1}^{W} i} & \text{if } k \geq W, \\ \dfrac{\sum_{i=0}^{k-1}(k-i)\mathscr{L}(\theta_{k-i})}{\sum_{i=1}^{k} i} & \text{otherwise.} \end{cases} \tag{2.14}$$

Note that for all first W iterations, the window size is reduced to the iteration number, *i.e.* $W = k$.

The decision to increase the batch size is then based on a successive lack of progress in the past iterations. The progress at the k-th iteration is defined as:

$$I_k = \frac{\mathrm{WMA}_{k-1,W} - \mathrm{WMA}_{k,W}}{\mathrm{WMA}_{k-1,W}}. \tag{2.15}$$

We consider that there is a lack of progress when I_k is less than a threshold $I_k < \Delta$. After C iterations with a lack of progress, the batch size is increased by a factor τ.

The AMABS algorithm, given in Algorithm 2.1, describes the adaptive batch size process. Therefore, it has to be coupled with a stochastic optimization algorithm. For example, using the principle of stochastic Hessian, an **AMABS Newton's method** or an **AMABS Trust-Region** method can be defined. One potential drawback of this method is that the stopping criterion of the optimization algorithm may be met before the algorithm has observed the full dataset. This could result in a loss of precision in the parameter value estimates. To counter this unwanted behavior, we add a condition on the stopping criterion, which requires that the batch size N'_k at iteration k has the same value as the dataset size N. As an example, this condition is also given for the Hybrid algorithms in Algorithm 2.3, line 7.

Hybridization

Using the AMABS technique naturally allows using different optimization algorithms based on the batch size. Indeed, when small batch size is used at the beginning of the optimization process, it makes sense to use the exact second derivative matrix, as it is relatively cheap to calculate. However, when the batch size becomes large, the calculation time for the exact Hessian is prohibitive, and methods relying on quasi-Newton approximation are preferred. It makes sense to switch to a less precise but faster algorithm. Our hybrid algorithm determines the algorithm to use based on the percentage of data used in a batch. The pseudocode is shown in Algorithm 2.3.

The usual initial parameter value θ_0 is an array of zeros. The last two parameters introduced in Algorithm 2.3, Δ_H and ε, are further discussed in Section 2.4.5. The stopping criterion $\nabla_{rel}\mathscr{L}(\theta)$

Algorithm 2.1 Adaptive Moving Average Batch Size (AMABS)

```
Inputs:
        - Current iteration index: $k$,
        - Function value at iteration $k$: $\mathscr{L}(\theta_k)$,
        - Current batch size: $N'_k$,
        - Size of the full dataset: $N$,
        - Size of the window: $W$ (default: 10),
        - Threshold for successfull iterations: $\Delta$ (default: 1%),
        - Maximum number of unsuccessful iterations: $C$ (default: 2),
        - Expansion factor for the batch size: $\tau$ (default: 2),
Output: New batch size: $N'_{k+1}$
 1: Set the counter at 0 ($c = 0$) at the initialization of the algorithm
 2: function AMABS
 3:     Compute $\text{WMA}_{k,W}$, as in Equation 2.14.
 4:     if $k > 0$ then                                   ▷ At least two iterations required
 5:         Compute $I_k$ as in Equation 2.15 using $\text{WMA}_{k,W}$ and $\text{WMA}_{k-1,W}$
 6:         if $I_k < \Delta$ then                ▷ Count the consecutive steps under threshold.
 7:             $c = c + 1$
 8:         else
 9:             $c = 0$
10:         if $c == C$ then                                         ▷ Update the batch size
11:             $c = 0$
12:             $N'_{k+1} = \min(\tau \cdot N', N)$
13:         else
14:             $N'_{k+1} = N'_k$
15:     return $N'_{k+1}$
```

is discussed at the end of this section. The functions `generateCandidateFirstOrder` (line 4) and `generateCandidateSecondOrder` (line 7) return the parameter values for the next iteration. The generic pseudocode for these methods is given in Algorithm 2.2. The function `generateCandidateFirstOrder` uses the function `generateCandidate` with a first-order or a quasi-Newton method and thus does not use the Hessian ($\nabla^2\mathscr{L}(\theta)$). The function `generateCandidateSecondOrder` uses the function `generateCandidate` with the Hessian.

Stopping Criterion

Both line search and trust-region methods require a stopping criterion to determine when to stop iterating. A relative gradient-based stopping criterion ensures that the algorithm stops when the norm of the relative gradient is below some threshold, ε:

$$||\nabla_{rel}\mathscr{L}(\theta)|| \leq \varepsilon, \tag{2.16}$$

Algorithm 2.2 Pseudocode for the function `generateCandidate`

```
Inputs:
        - Parameter value at iteration $k$: $\theta_k$
        - Log likelihood function: $\mathscr{L}(\theta)$,
        - Gradient of the log likelihood: $\nabla\mathscr{L}(\theta)$,
        - Possibly, Hessian of the log likelihood: $\nabla^2\mathscr{L}(\theta)$
Output: Next parameter value: $\theta_{k+1}$
1: function GENERATECANDIDATE
2:     Compute direction $d_k$ using $\nabla\mathscr{L}(\theta_k)$ and/or $\nabla^2\mathscr{L}(\theta_k)$ as in Equation 2.6.
3:     Compute step size $\alpha_k$ using either a line search or a trust-region method.
4:     Compute $\theta_{k+1} = \theta_k + \alpha_k d_k$
```

Algorithm 2.3 Hybrid algorithm with AMABS method

```
Inputs:
        - Log likelihood function: $\mathscr{L}(\theta)$,
        - Gradient of the log likelihood: $\nabla\mathscr{L}(\theta)$,
        - Hessian of the log likelihood: $\nabla^2\mathscr{L}(\theta)$,
        - Initial parameter value: $\theta_0$,
        - Algorithm generating a candidate for the next iteration using only the gradient (first-
          order or quasi-Newton method): generateCandidateFirstOrder($\theta, \nabla\mathscr{L}(\theta)$),
        - Algorithm generating a candidate for the next iteration using the gradi-
          ent and the Hessian (second-order method): generateCandidateSecon-
          dOrder($\theta, \nabla\mathscr{L}(\theta), \nabla^2\mathscr{L}(\theta)$),
        - Parameters specific to the algorithm:
            * Initial batch size: $N'_0$ (default: 1000),
            * Size of the full dataset: $N$,
            * Size of the window: $W$ (default: 10),
            * Threshold for successfull iterations: $\Delta$ (default: 1%),
            * Maximum number of unsuccessfull iterations with the same batch size: $C$ (de-
              fault: 2),
            * Expansion factor for the batch size: $\tau$ (default: 2),
            * Threshold for hybridization: $\Delta_H$ (default: 30%),
            * Threshold for stopping criterion: $\varepsilon$ (default: $10^{-6}$)
Output: Optimized parameters: $\theta^*$
1: function ITERATION
2:     if $N'_k / N > \Delta_H$ then
3:         $\theta_{k+1}$ = generateCandidateFirstOrder($\theta_k, \mathscr{L}(\theta), \nabla\mathscr{L}(\theta)$)
4:     else
5:         $\theta_{k+1}$ = generateCandidateSecondOrder($\theta_k, \mathscr{L}(\theta), \nabla\mathscr{L}(\theta), \nabla^2\mathscr{L}(\theta)$)
6:     $N'_{k+1} = \text{AMABS}(k, \mathscr{L}(\theta_k), N'_k, N, W, \Delta, C, \tau)$
7:     Stop the optimization if $\nabla_{\text{rel}}\mathscr{L}(\theta_{k+1}) < \varepsilon$ and $N'_k == N$.
```

where the relative gradient is defined by

$$(\nabla_{rel}\mathscr{L}(\theta))_i = \frac{(\nabla\mathscr{L}(\theta))_i\theta_i}{\mathscr{L}(\theta)}. \tag{2.17}$$

A sufficiently small value is chosen for ε, typically in the range $[10^{-6}, 10^{-8}]$. For this work, we used $\varepsilon = 10^{-6}$. A more detailed discussion of the stopping criterion can be found in Dennis and Schnabel (1996) (see Chapter 7.2, page 159). In addition, we restrict the maximal number of epochs of the algorithms. An epoch corresponds to one complete presentation of the dataset. Therefore, to compute the relation between epoch and iteration, we have to calculate it using recursion. We thus define the epoch at iteration 0 being $e_0 = 0$. Then, we can update the number of epochs at each iteration using the following formula:

$$e_{k+1} = e_k + \frac{N'_k}{N} \tag{2.18}$$

where N'_k corresponds to the current batch size and N to the size of the dataset. We therefore see that if we use the full dataset, an epoch corresponds to an iteration.

2.3.4 Summary of algorithms

As depicted in Table 2.3, the new algorithmic framework presented in Section 2.3.3 allows us to compare the performance of our proposed Hybrid Adaptive Moving Average Batch Size (HAMABS) algorithm against 14 standard benchmarks[1]. These algorithms can be split into three categories:

- *standard non-stochastic algorithms* (first 6 algorithms) – deterministic algorithms which are commonly used in the DCM community and are therefore considered as benchmarks. For example, the current version of Biogeme uses the Python package Scipy (Jones et al., 2014) with the BFGS^{-1} implementation. We start by comparing the performance of these standard algorithms before moving to the comparison with stochastic approaches.
- *stochastic algorithms* (next 6 algorithms) – algorithms based on the AMABS method presented in Section 2.3.3. These methods are used to show the improvement in terms of estimation time over their non-stochastic counterparts. The algorithms `NM-ABS` and `TR-ABS` also make use of the stochastic Hessian, as presented in Section 2.3.3. The added value of using stochastic Hessian will therefore also be discussed.
- *hybrid stochastic methods* (last 3 algorithms) – algorithms which combine two separate optimization algorithms with adaptive batch size. Three types of hybridization are investigated and discussed: (i) Newton's method and BFGS, (ii) Trust-Region method and BFGS, and (iii) Newton's method and BFGS^{-1}. The first algorithm always corresponds to

[1]We do not include standard stochastic methods in our benchmarks as these methods were tested in a previous work (Lederrey et al. (2018b)) but were not able to converge to a stable solution.

the function `generateCandidateSecondOrder` and the second to the function `generateCandidateFirstOrder` in Algorithm 2.3.

Table 2.3: Overview of all algorithms used for the optimization of DCMs. A small description of the algorithms is provided as well as their order and if it includes the adaptive batch size method.

Name	Order	AMABS	Description
`GD`	1st		Steepest descent algorithm.
`BFGS`	1.5th		BFGS algorithm using Eq. 2.7.
`BFGS`$^{-1}$	1.5th		BFGS^{-1} algorithm using Eq. 2.8.
`TR-BFGS`	1.5th		quasi-Newton trust-region method with BFGS (Eq. 2.7).
`NM`	2nd		Newton's method.
`TR`	2nd		Trust-region method.
`GD-ABS`	1st	✓	Stochastic steepest descent with AMABS.
`BFGS-ABS`	1.5th	✓	BFGS algorithm (Eq. 2.7) with AMABS.
`BFGS`$^{-1}$`-ABS`	1.5th	✓	BFGS^{-1} algorithm (Eq. 2.8) with AMABS.
`TR-BFGS-ABS`	1.5th	✓	Trust-Region with BFGS (Eq. 2.7) and AMABS.
`NM-ABS`	2nd	✓	Newton with AMABS.
`TR-ABS`	2nd	✓	Trust-region with AMABS.
`H-NM-ABS`	Hybrid	✓	Hybridization: Newton + BFGS (Eq. 2.7).
`H-TR-ABS`	Hybrid	✓	Hybridization: trust-region + BFGS (Eq. 2.7).
`HAMABS`	Hybrid	✓	Hybridization: Newton + BFGS^{-1} (Eq. 2.8).

2.4 Results

This section presents the results of our experiments. Before showing the numerical results, we start by explaining our experimental design, *i.e.*, the algorithms and datasets used in our experiments, and the implementation details.

2.4.1 Experimental design

We collect empirical evidence of the behavior of the 15 algorithms, in Table 2.3, by observing their performance on ten different choice models presented in Table 2.4. Two different data sources are used. The first nine choice models are estimated on data obtained from the London Passenger Mode Choice (LPMC) dataset. This dataset, collected by Hillel et al. (2018), contains mode choices on an urban multi-modal transport network from April 2012 to March 2015. In addition, three sub-datasets have been created to study the impact of the size of the dataset on the performance of the different algorithms:

- a small dataset (S), that contains observations from year 2012 (27'478 observations),
- a medium dataset (M), that contains observations from years 2012 to 2013 (54'766 observations),
- a large dataset (L), that contains all observations, from year 2012 to 2015 (81'766 observations).

In order to analyze the impact of the number of parameters on the estimation time, we compare three logit models from Hillel (2019):

- the `LPMC_DC` model that contains 13 parameters to be estimated,
- the `LPMC_RR` model that contains 54 parameters to be estimated,
- the `LPMC_Full` model that contains 100 parameters to be estimated.

Finally, a tenth choice model was estimated on the Mobility and Transport MicroCensus (MTMC) dataset, a statistical survey of the travel behavior of the Swiss population. The model has been designed by Danalet and Mathys (2018) using the data collected by the Swiss Federal Statistical Office (FSO) and the Swiss Federal Office for Spatial Development (ARE)[2]. The authors also provide more details about the MTMC dataset. We use the most recent version of the survey, collected in 2015. This model provides an exciting opportunity to study the efficiency of all algorithms presented in Table 2.3 for estimating a rather large choice model (almost 250 parameters) on a medium-size dataset (56'915 observations).

2.4.2 Implementation details

All models are estimated on a single node in a supercomputer (18 Cores Skylake Processor@2.30 GHz, 192GB) for each algorithm. We include a stopping criterion on the maximum number of epochs (1,000 epochs). This is done to avoid extremely long computation time for algorithms that would struggle in achieving convergence for certain models. Also, since some of these algorithms are stochastic, convergence speed may differ on the same optimization task. Thus, each stochastic algorithm is used to optimize each model 20 times. We impose an upper limit on the execution time for the 20 estimation process: 12 hours for the models `LPMC_DC`, 24 hours for the models `LPMC_RR`, 36 hours for the models `LPMC_Full`, and 48 hours for the model `MTMC`. Finally, since we want to compare our algorithms' efficiency with state-of-the-art DCM software, we also optimize all the models in Table 2.4 with Biogeme and Scipy within the same rules. All the results are presented and discussed in Section 2.4.3. The code implementing all the algorithms in Table 2.3 can be found on Github at https://github.com/glederrey/HAMABS.

[2]Contact mobilita2015@bfs.admin.ch for accessing the data.

Table 2.4: Summary of the models used for the performance analysis. The number of parameters is provided as well as the dataset and the number of observations used in the model. All the models are logit models.

Names	#Parameters	Data	#Observations
LPMC_DC_S	13	LPMC	27’478
LPMC_DC_M	13	LPMC	54’766
LPMC_DC_L	13	LPMC	81’086
LPMC_RR_S	54	LPMC	27’478
LPMC_RR_M	54	LPMC	54’766
LPMC_RR_L	54	LPMC	81’086
LPMC_Full_S	100	LPMC	27’478
LPMC_Full_M	100	LPMC	54’766
LPMC_Full_L	100	LPMC	81’086
MTMC	247	MTMC	56’915

2.4.3 Performance analysis

In this section, we analyze the performance of the fifteen algorithms reported in Table 2.3. For ease of comparison, we use a graphical approach named *performance profiles* to benchmark these algorithms. As stated by Beiranvand et al. (2017), performance profiles are a great tool to analyze algorithms in terms of efficiency, robustness, and probability of successfully performing a required task. The concept of performance profile is presented in the next section and the results are analyzed in the following section.

Performance profiles

Performance profiles were first introduced by Dolan and Moré (2002) and are now a recurring tool used to compare the performance of optimization algorithms. They are used to compare the performance of a set of optimization algorithms, $\mathscr{A}$, on a set of optimization problems, $\mathscr{P}$. For each pair $(p,a) \in \mathscr{P} \times \mathscr{A}$, they define a performance measure $t_{p,a} > 0$ whose large value indicates poor performance. Classical measures of performance are the execution time or the number of epochs.

In addition, a convergence test $\mathscr{C}_{p,a}$ states if algorithm a was able to optimize problem p. For each optimization problem p and optimization algorithm a, the performance ratio is defined as

$$r_{p,a} = \begin{cases} \frac{t_{p,a}}{\min_{a \in \mathscr{A}} t_{p,a}} & \text{if } \mathscr{C}_{p,a} \text{ passed,} \\ \infty & \text{if } \mathscr{C}_{p,a} \text{ failed.} \end{cases} \tag{2.19}$$

This leads to $r_{p,a} = 1$ for the best algorithm and $r_{p,a} = \infty$ for all algorithms a unable to solve problem p. The performance profile of an algorithm a is finally defined as

$$\rho_a(\pi) = \frac{\left|p \in \mathscr{P} : r_{p,a} \leq \pi\right|}{|\mathscr{P}|} \tag{2.20}$$

where $|\mathscr{P}|$ is the cardinality of the set $\mathscr{P}$. $\rho_a(\pi)$ represents the proportion of problems for which the performance ratio $r_{p,a}$ for algorithm $a \in \mathscr{A}$ is within a factor $\pi \in \mathbb{R}$ of the best possible performance ratio.

Generally, $\pi \in \mathbb{N}^+$ is used to avoid showing too many data points. Furthermore, any upper bound on π can be used. However, if $\mathscr{R} = \max_{p\in\mathscr{P},a\in\mathscr{A}} r_{p,a}$, $\forall r_{p,a} < \infty$, the performance profiles will remain the same for any $\pi \geq \mathscr{R}$. Therefore, $\mathscr{R}$ is used as the upper bound on π.

It is interesting to note that $\rho_a(1)$ corresponds to the percentage of problems for which algorithm a has the best performance. Also, $\rho_a(\mathscr{R})$ represents the percentage of problems that algorithm a was able to solve under the condition of the convergence test $\mathscr{C}$. Therefore, algorithms with high values of $\rho_a(\pi)$ are of interest.

As stated above, a performance profile has the values of π on the x-axis ranging from 1 to $\mathscr{R}$, the maximum of all ratios. The proportion $\rho_a(\pi)$ is situated on the y-axis. There are three specific elements to analyze to interpret these profiles:

- the proportion for each algorithm a at the value $\pi = 1$, *i.e.* at the leftmost side of the graph. Indeed, the proportion $\rho_a(1)$ indicates the percentage of problems for which an algorithm a is the best, based on the performance test $t_{p,a}$.
- the proportion for each algorithm a at the value $\pi = \mathscr{R}$, *i.e.* at the rightmost side of the graph. This proportion indicates the percentage of problems that algorithm a was able to solve within the convergence criterion $\mathscr{C}_{p,a}$.
- how quickly the algorithm a reaches a proportion of 100%, *i.e.* the line reaches the top of the graph. The value of π for which algorithm a reaches 100% indicates its worse relative performance than all other algorithms.

Thus, the ideal performance profile starts at 100% and finishes at 100%. This algorithm a is the best for the performance measure $t_{p,a}$. However, these kinds of results are rare. Therefore, comparing the algorithms by looking at the three points cited above is important to determine which algorithm performs the best.

Results

In our case, the set of problems $\mathscr{P}$ contains the ten models in Table 2.4 and the set of optimization algorithms $\mathscr{A}$ include the fifteen algorithms in Table 2.3. The convergence test $\mathscr{C}_{p,a}$

tells us whether the algorithm was able to converge with the required precision ε in less than 1000 epochs. We selected two performance measures: the execution time and the number of epochs.

Figure 2.1 shows the results for the execution time. While analyzing the lines in Figure 2.1, it is recommended to have a look at the reported execution times in Tables E.1, E.2, and E.3, provided in the appendix. We also provide the maximum ratio, $\mathscr{R} = 114$, for the execution time.

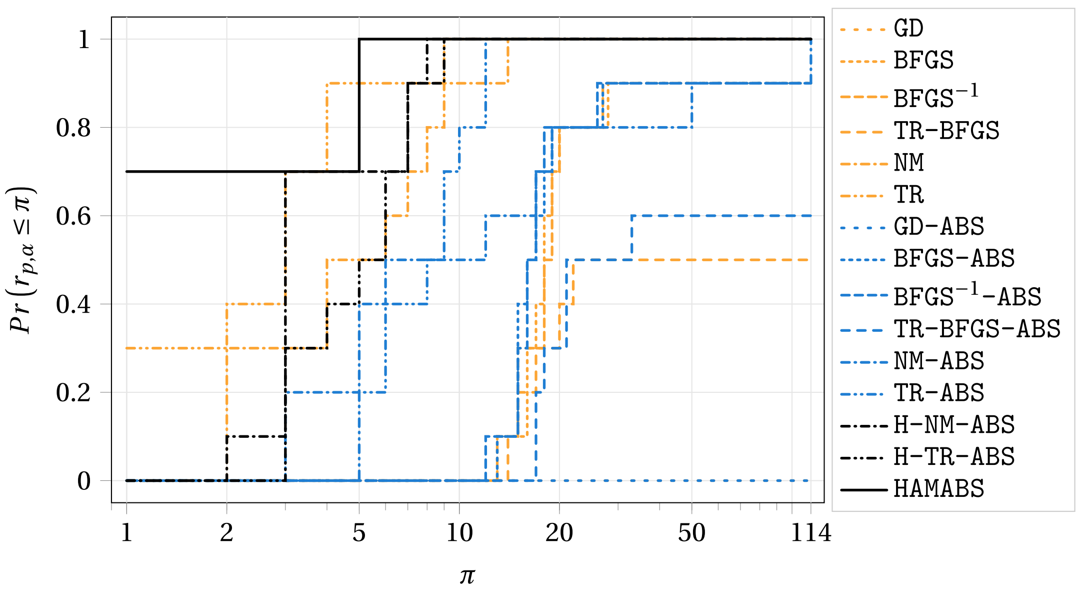


Figure 2.1: Performance profiles on the execution time for all models in Table 2.4 and all algorithms in Table 2.3. The standard non-stochastic algorithms are shown in orange lines, the stochastic algorithms in blue lines, and the hybrid algorithms in black.

As an example, we analyze in detail the line corresponding to the algorithm `HAMABS` in Figure 2.1 based on the three elements cited in Section 2.4.3:

- it reaches a proportion of 70% at $\pi = 1$. It, therefore, means that this algorithm is the fastest for 7 out of the 10 optimized models. Tables E.1, E.2, and E.3, provided in the appendix, report the average time and the standard deviation to optimize each model with each algorithm. As seen in these tables, the `HAMABS` is effectively the fastest algorithm on seven out of ten models.
- it can solve all problems. Indeed, it reaches a proportion of 100% for $\pi = \mathscr{R}$. We can also verify that it is effectively the case in Tables E.1, E.2, and E.3.
- it reaches a proportion of 100% at $\pi = 5$. Thus, this algorithm has, at worst, a relative performance of 5 compared to the fastest algorithm on all the models.

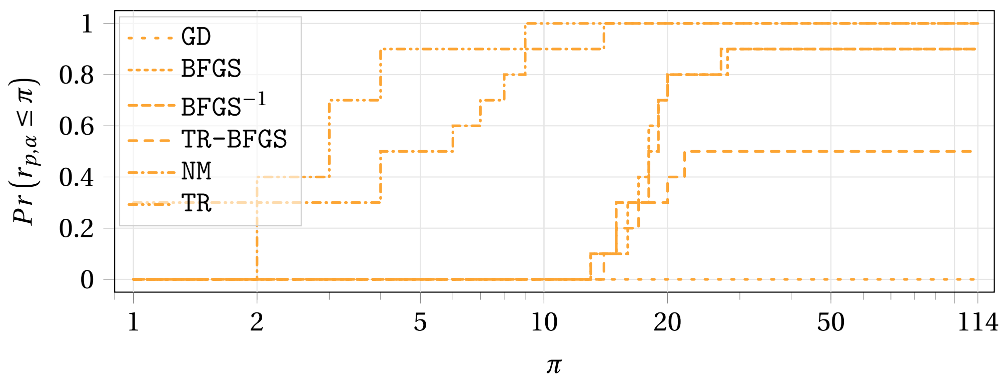
Based on the analysis above and by comparing the `HAMABS` algorithms with the other algorithms, we can conclude that this algorithm is the fastest and the most robust in general. Indeed, it is the fastest on the majority of the models. Besides, the only models on which this algorithm is not the fastest are the `LPMC_DC` models, the smallest models in terms of parameters. Also, we see that this algorithm is the fastest to reach 100% proportion. This thus shows that it is the most robust algorithm across all models. Since there are many algorithms to analyze, we discuss them further by types of algorithms. Figure 2.2 split the lines in Figure 2.1 by the three types of algorithms shown in Table 2.3.

Standard non-stochastic algorithms Figure 2.2a shows the performance profile for all standard non-stochastic algorithms. We observe that these standard algorithms are struggling to optimize the models. Trust-Region and Newton's methods are the fastest and the most robust among the standard ones and reach the 100% proportion with a relative performance of up to 15 times the fastest algorithm. The two BFGS methods are slower than the first two methods. Besides, they also fail to optimize some models. We also see that the algorithm `TR-BFGS` struggles to optimize the models, indicating that the hybrid algorithm `H-TR-ABS` might also struggle. The worst method is the gradient descent algorithm since it does not converge for any model in less than 1000 epochs.

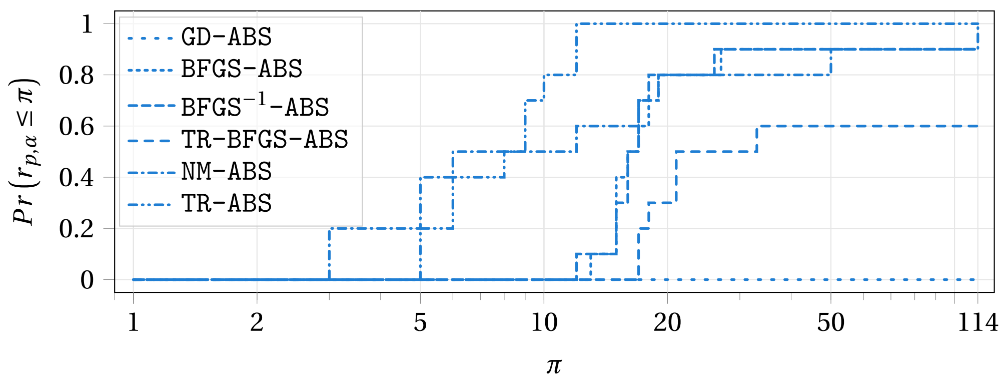
Stochastic algorithms Figure 2.2b shows the results for the algorithms using the AMABS technique. The behavior of these methods does not differ much as the slow standard methods stay slow with the AMABS method. For example, the fastest and most robust algorithms are both the `NM-ABS` and the `TR-ABS`. The algorithm `GD-ABS` cannot optimize any model within the required number of epochs. Tables E.1, E.2, and E.3 also show that, except for the `TR-ABS`, all AMABS methods are faster than the standard ones. It thus shows that the AMABS algorithm can generally speed up the standard algorithms.

Hybrid stochastic algorithms Figure 2.2c shows the results for the three algorithms using hybridization and the AMABS method. These three methods are the fastest algorithms on larger models. While we already discussed the case of the `HAMABS` algorithm, the other two methods are never the fastest for any models. However, they are slightly more robust than the other algorithms. Indeed, the `H-NM-ABS` algorithm is almost as good as the `HAMABS`. However, looking at the times, we still see quite a difference between these two algorithms. Indeed, there is generally a 20% difference in execution time between these two algorithms. The `H-TR-ABS` seems to perform quite well. However, looking at the times, we see that both the `TR` and the `TR-ABS` algorithms are faster. This is most likely due to the use of the Trust-Region method with the BFGS approximation being exceptionally slow.

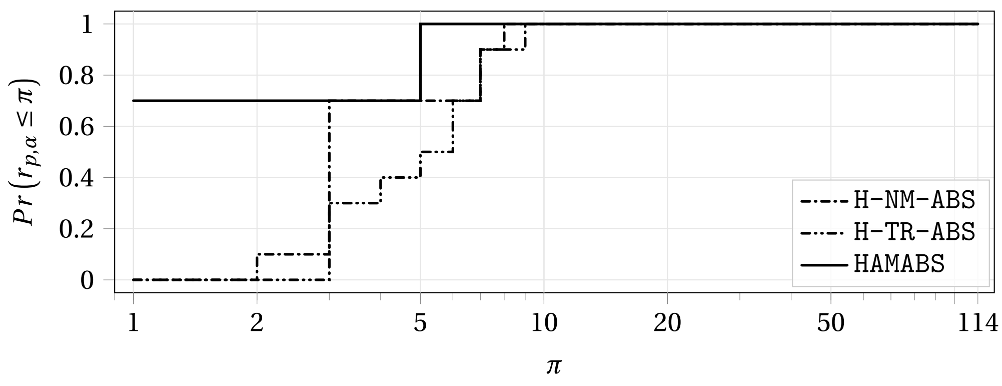
As seen in Figures 2.1 and 2.2, the `HAMABS` algorithm is the fastest to optimize most of the models. The `HAMABS` algorithm has two stages. In the first stage, the exact Hessian is calculated on a small sample. In the second stage, the Hessian is approximated on larger batch size. Each of these stages balances the trade-off between computational cost and precision: calculating



(a) Standard non-stochastic algorithms



(b) Stochastic AMABS algorithms



(c) Hybrid stochastic algorithms

Figure 2.2: Performance profiles on the execution time for all models in Table 2.4 splitted into different groups of algorithms.

the exact Hessian is more computationally expensive and more precise than approximating the Hessian. Using a smaller batch is less computationally expensive but accordingly less precise. By counteracting these effects in each optimization stage, the HAMABS algorithm achieves the fastest overall performance. Figure 2.3 shows the performance profile on the epochs for all algorithms. Since the HAMABS algorithm is the second more robust algorithm after the TR algorithm, it confirms the given interpretation.

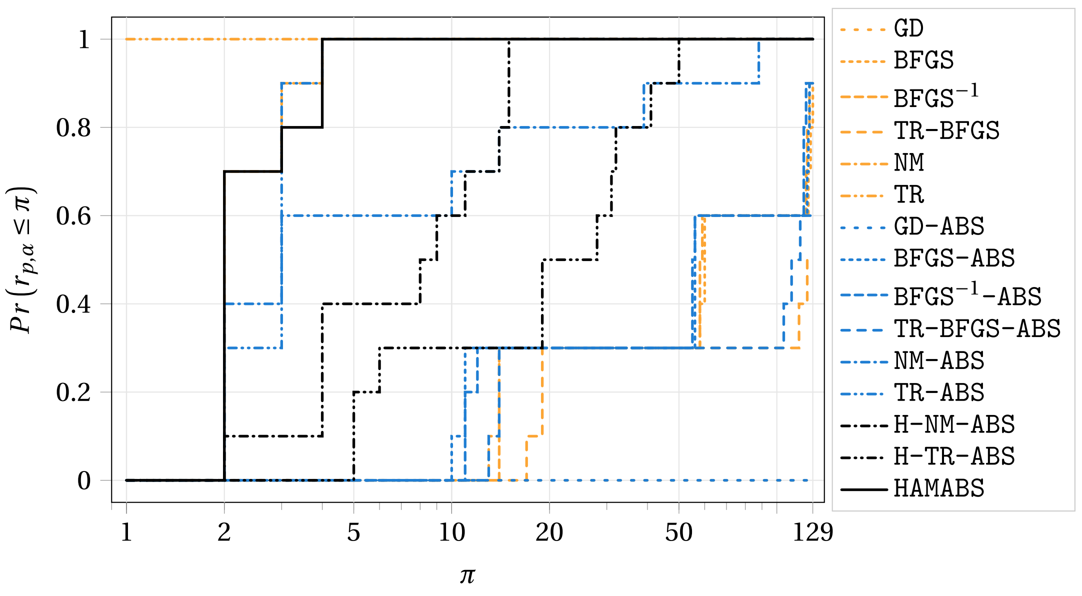
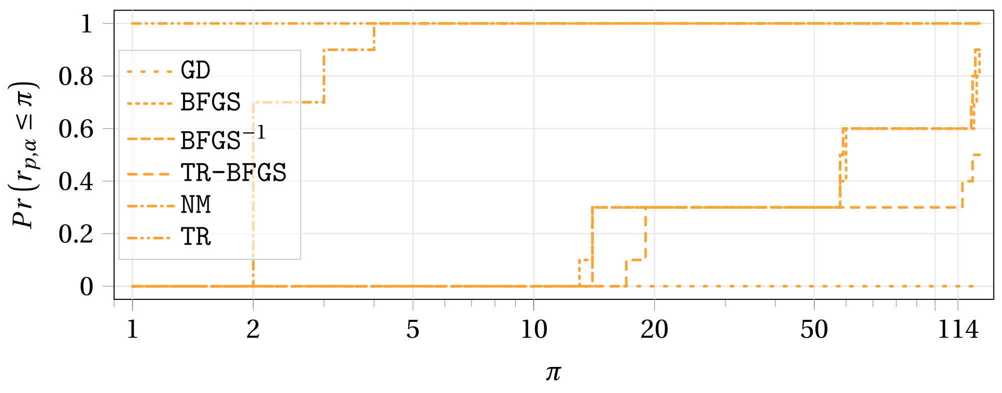


Figure 2.3: Performance profiles on epochs for the all models in Table 2.4 and all algorithms in Table 2.3. The standard non-stochastic algorithms are shown in orange lines, the stochastic algorithms in blue lines, and the hybrid algorithms in black.

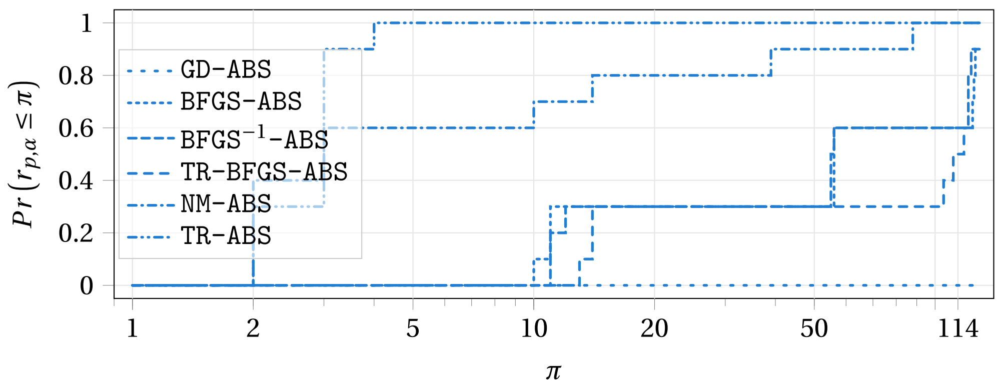
Figure 2.4 splits Figure 2.3 based on the different groups of algorithms. Comparing the different algorithms shows that second-order methods tend to use fewer epochs to achieve convergence. Indeed, we can see in both Figure 2.4a and Figure 2.4b that the methods based on Newton's method (NM/NM-ABS) and Trust-Region method (TR/TR-ABS) are more robust than the other algorithms. This is expected because these methods use the information on the curvature. They thus require fewer steps to complete the estimation process. We also see that second-order methods using the full size dataset, NM and TR, are using less epochs than the stochastic methods, NM-ABS and TR-ABS. This is also expected since the stochastic algorithms use less information per step. They thus need to perform many more steps, often leading to more epochs, to gain the same knowledge. On the other hand, they spend less time on each step, leading to a consequent speedup. It is interesting to note that the HAMABS algorithm is amongst the algorithms using the least number of epochs. Indeed, it reaches a proportion of 100% with a relative performance of 4. It, therefore, explains why this algorithm is that fast compared to the AMABS algorithms. Also, we see that the H-NM-ABS tends to use more epochs than the HAMABS algorithm. This could explain why HAMABS is the fastest algorithm. Tables E.4, E.5, and E.6, in the appendix, provide the numerical results consists of the average number of epochs with the standard deviation used by each algorithm to optimize each model.

2.4.4 Comparison with Biogeme

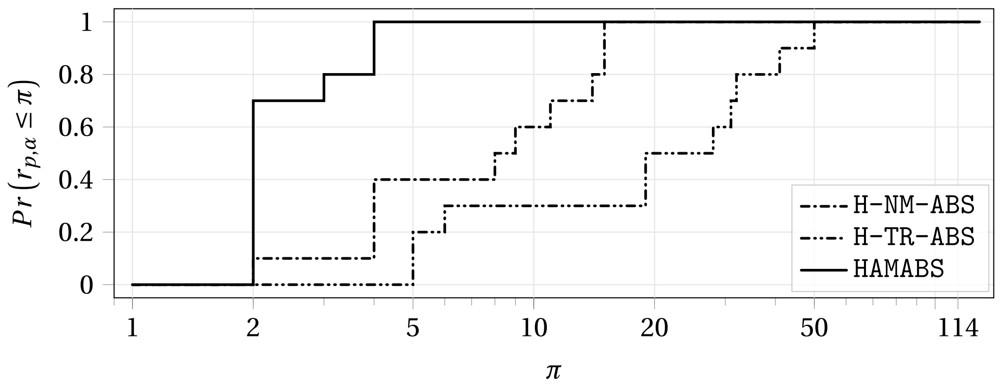
We now compare the performance of our best algorithm, the HAMABS algorithm, to Pandas Biogeme (Bierlaire, 2018), a state-of-the-art choice modeling software. Biogeme uses the Python



(a) Standard non-stochastic algorithms



(b) Stochastic AMABS algorithms



(c) Hybrid stochastic algorithms

Figure 2.4: Performance profiles on epochs for all models in Table 2.4 splitted into different groups of algorithms.

package Scipy to optimize the models. This package's default algorithm for minimization is the BFGS^{-1}. Table 2.5 reports the average time to optimize each model for both Biogeme and the `HAMABS` algorithm. Besides, the last column shows the speedup gained using the `HAMABS` algorithm instead of Scipy. If the `HAMABS` algorithm is faster than the benchmark, the speedup is greater than 1 and shows the ratio between the optimization time. On the other hand, if the speedup is less than 1, the `HAMABS` algorithm is slower than the benchmark.

Table 2.5: Comparison of the optimization time for all models in Table 2.4 between the HAMABS algorithm and Biogeme. The time are reported in seconds. The speedup corresponds to a ratio between the two compared values.

Models	Time [s]		Speedup
	HAMABS	Biogeme/Scipy	
LPMC_DC_S	1.86 ± 0.12	1.62 ± 0.01	0.87
LPMC_DC_M	3.11 ± 0.20	2.79 ± 0.02	0.90
LPMC_DC_L	4.59 ± 0.32	4.07 ± 0.04	0.89
LPMC_RR_S	11.98 ± 1.23	65.17 ± 0.09	5.44
LPMC_RR_M	18.46 ± 1.06	127.67 ± 0.30	6.91
LPMC_RR_L	18.14 ± 1.06	177.09 ± 0.29	9.76
LPMC_Full_S	257.02 ± 42.85	1462.51 ± 14.41	5.69
LPMC_Full_M	405.43 ± 43.77	2480.06 ± 18.27	6.12
LPMC_Full_L	486.31 ± 63.38	4758.28 ± 45.22	9.78
MTMC	1243.95 ± 56.21	28008.10 ± 528.33	22.52

Results presented in Table 2.5 show that the algorithm HAMABS is generally faster than the Scipy package. On the models LPMC_DC, that have few parameters, the HAMABS algorithm is slower with a ratio around 1.15. However, the HAMABS becomes faster than the Scipy package on the models LPMC_RR and LPMC_Full that include more parameters. This implies that a model that previously took minutes, or hours to converge, is now optimized in only a few seconds or minutes, respectively. The most important gain is in optimizing the largest model, the MTMC model, with a speedup ratio exceeding 22. While Biogeme takes around seven and a half hours to converge, our HAMABS algorithm converges in less than 20 minutes.

Table 2.6 compares the two algorithms based on the number of epochs used for optimization. Results show that the computational gains achieved by our HAMABS algorithm are even more important in terms of the number of epochs. For Biogeme and the Scipy package, the number of epochs is directly correlated to the number of parameters. As a result, the number of epochs highly depends on the model's size. For example, the MTMC models uses hundred times more epochs to be optimized than the LPMC_DC models. For our HAMABS algorithm, on the other hand, the number of epochs used is more stable across the different models. Indeed, the average number of epochs doubles between the smallest and the largest models. On the MTMC model, the speedup ratio in number of epochs between Biogeme and our HAMABS algorithm exceeds 600. The stopping criterion is the reason behind these discrepancies in the number of epochs. The Scipy package uses the standard, yet incorrect, gradient value as the stopping criterion. If the objective function and its derivatives are not correctly normalized, this criterion can either stop the algorithm too early or too late. Therefore, using an appropriate stopping criterion makes an important difference in terms of epochs while keeping sufficient precision, as seen in the last column of Table 2.6. Indeed, we see that the relative difference in log likelihood between the two methods is never larger than 2×10^{-4} %.

Table 2.6: Comparison of the epochs used in the optimization process for all models in Table 2.4 between the HAMABS algorithm and Biogeme. The speedup corresponds to a ratio between the two compared values. The last column corresponds to the relative difference in percentage between the log likelihood returned by Biogeme and by the HAMABS algorithm.

Models	**Epochs**		**Speedup**	**$\Delta\mathscr{L}$ [%]**
	HAMABS	**Biogeme/Scipy**		
LPMC_DC_S	13.79 ± 1.70	122	8.85	3.20×10^{-6}
LPMC_DC_M	13.50 ± 2.05	114	8.44	3.13×10^{-6}
LPMC_DC_L	12.57 ± 1.93	123	9.78	5.43×10^{-6}
LPMC_RR_S	15.31 ± 2.53	787	51.40	5.00×10^{-5}
LPMC_RR_M	13.95 ± 1.67	809	57.97	3.28×10^{-5}
LPMC_RR_L	9.55 ± 0.56	772	80.84	2.87×10^{-4}
LPMC_Full_S	24.98 ± 2.16	1786	71.50	1.45×10^{-4}
LPMC_Full_M	20.42 ± 1.84	1531	74.96	2.13×10^{-4}
LPMC_Full_L	21.36 ± 2.05	1996	93.44	8.02×10^{-5}
MTMC	18.63 ± 1.58	11920	639.88	2.01×10^{-4}

The similar ratios between the models LPMC_RR and LPMC_Full, can be due to the added complexity on the LPMC_Full models. Indeed, in these models, multiple parameters are computed on small populations. This leads to an increase in complexity, and the stochasticity might not be that helpful. The algorithm has to perform more steps at full size to find the parameter values for these small groups. We thus lose some time at the end of the optimization process compared to LPMC_RR models.

In definitive, our results showed that the HAMABS algorithm is not only the fastest among the 15 algorithms in Table 2.3, but it is also much faster than the current implementation of the state-of-the-art choice modeling software Biogeme.

2.4.5 Sensitivity analysis

We want now to test the sensitivity of the HAMABS algorithm's parameters to make sure we validate the choice of the default parameters given in Algorithm 2.3. We selected the three following models to perform the sensitivity study: LPMC_DC_L, LPMC_RR_L, and LPMC_Full_L. The sensitivity analysis was performed on the estimation time of these models by the HAMABS algorithm. Each model was trained 20 times for all the test valuesin Table 2.7.

All results are reported using graphs indicating the relative performance on the vertical axis. In addition, all performances are normalized to the execution time obtained with the default parameter values (base value of 1) to ease the comparison of results across the models.

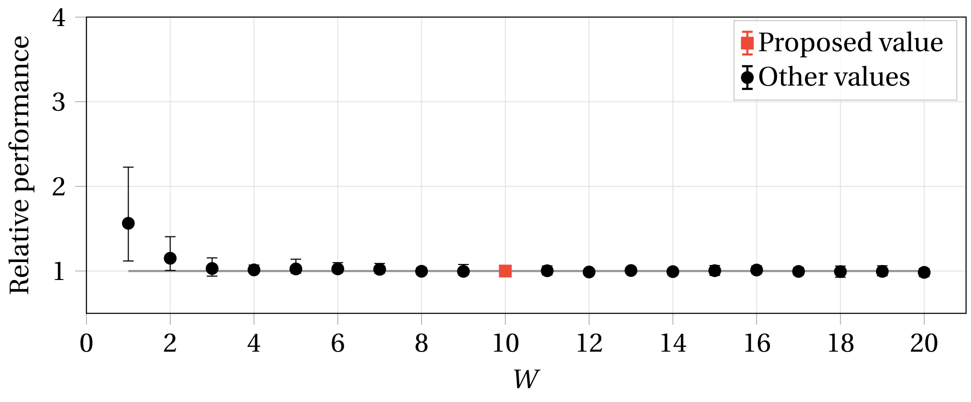
Table 2.7: Parameter values of the HAMABS algorithm used for the sensitivity analysis. We refer the reader to Algorithm 2.3 for a detailed explanation of the parameters.

Parameter	**Default**	**Test values**
W	10	$[1, 2, 3, \cdots, 18, 19, 20]$
Δ	1%	$[0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100]$
C	2	$[1, 2, 3, \cdots, 13, 14, 15]$
τ	2	$[1.1, 1.2, 1.3, 1.4, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8, 9, 10]$
Δ_H	30%	$[0, 5, 10, \cdots, 90, 95, 100]$
ε	10^{-6}	$[10^{-9}, 10^{-8}, \cdots, 10^{-1}, 10^{0}]$
N'_{init}	1.23% (1000)	$[1, 1.23, 2, 5, 10, 20, 50, 100]$

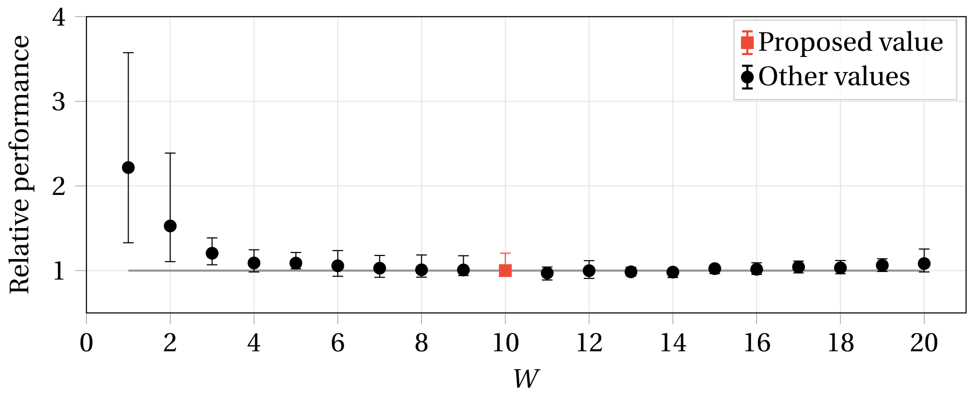
Figure 2.5 shows the analysis for the parameter W; the window size. The results are similar across all models. Indeed, small values of W lead to an increase in the execution time. It can be explained by the fact that the average is noisy if the window is too small. Therefore, the computation of the improvement of the log likelihood is not precise. It thus increases the batch size too soon or too late. This thus leads to an increase in the total execution time. Large values for W do not affect the optimization process as much as small values. We see a small increase in the execution time in Figure 2.5b when W is large. Indeed, if we use large values of W, the average is less influenced by the new data points. Thus, algorithms using AMABS show a slower reaction. Therefore, a good value for W has to be in the middle. We thus propose to use $W = 10$.

Figure 2.6 shows the analysis for the parameter Δ; the threshold for successful iterations. This parameter represents a trade-off between using a small batch size for too long on models that are easy to optimize and switching too soon to a large batch size on models that are difficult to optimize. From Figure 2.6, we select 1% as the value for this parameter as it is the turning point on the curve (where it begins to plateau) and so represents a good balance between these effects.

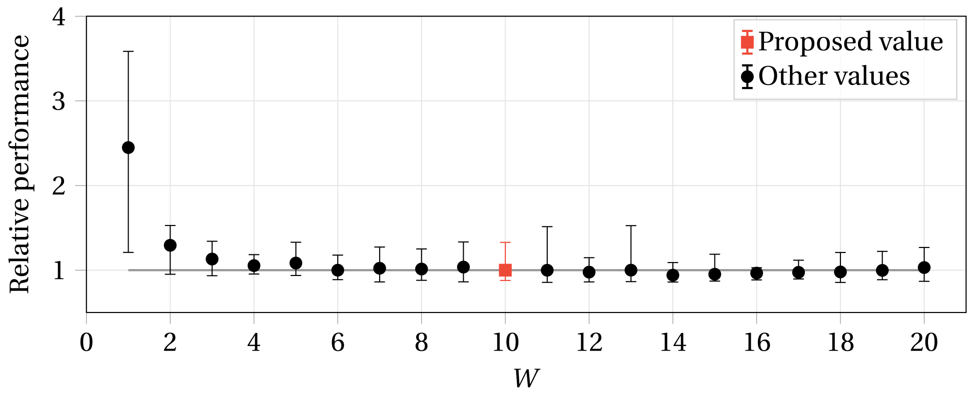
Figure 2.7 shows the analysis for the parameter C; the maximum number of unsuccessful iterations with the same batch size. It is interesting to note that the relationship between the execution time and this parameter value is linear for the three models. Therefore, it is evident that using a value of 1 for C is faster. However, the advantage is reduced with the model's size, as we can see when comparing Figure 2.7a and Figure 2.7c. Besides, if the count is set to 1, it could trigger a false positive. Indeed, if the value of the window, W, is too small, an exceptionally lousy batch of data may increase the batch size while the log likelihood can still be improved. It is thus preferable to slightly slow the execution time but increase the algorithm's robustness. That is the reason we propose to use $C = 2$.



(a) `LPMC_DC_L`

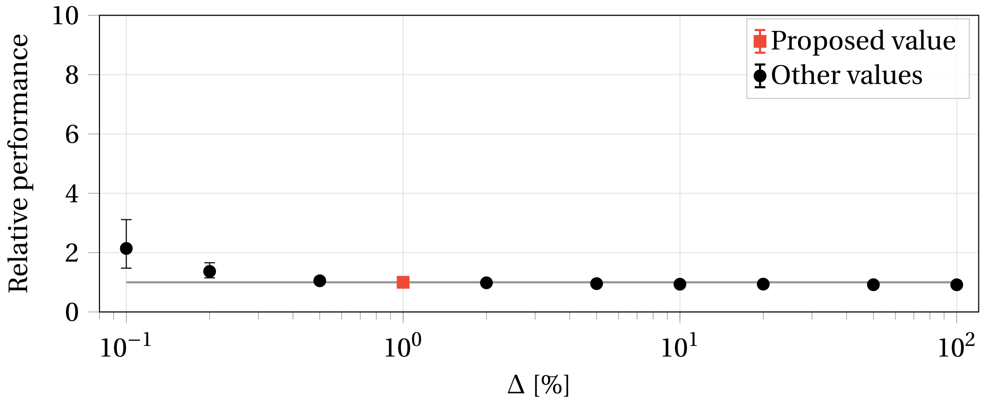


(b) `LPMC_RR_L`

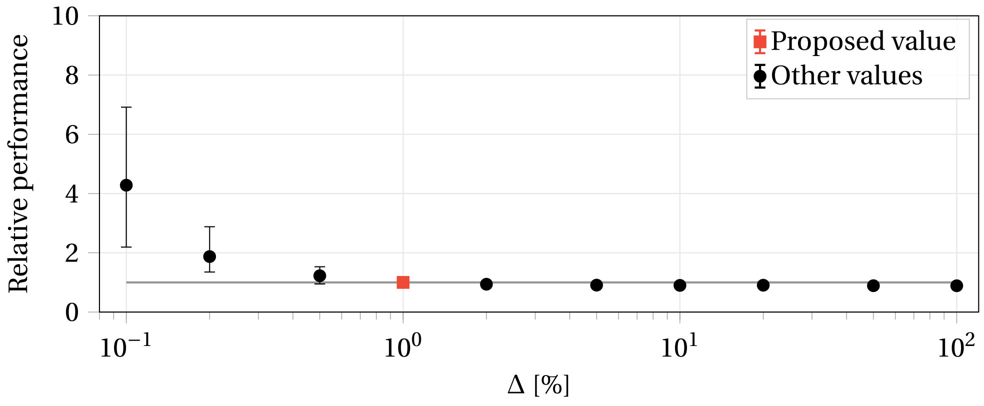


(c) `LPMC_Full_L`

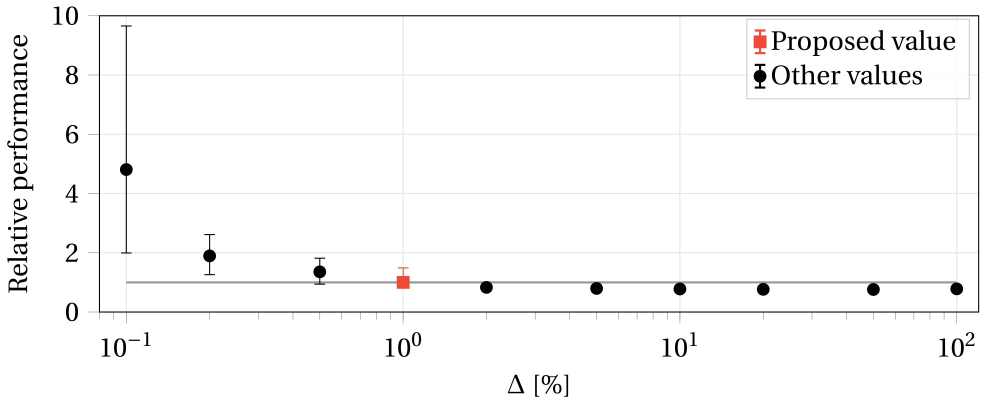
Figure 2.5: Sensitivity analysis for the parameter W for the three LPMC models using the large dataset. The black error bars correspond to the tested values and the red to the proposed value. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.



(a) `LPMC_DC_L`

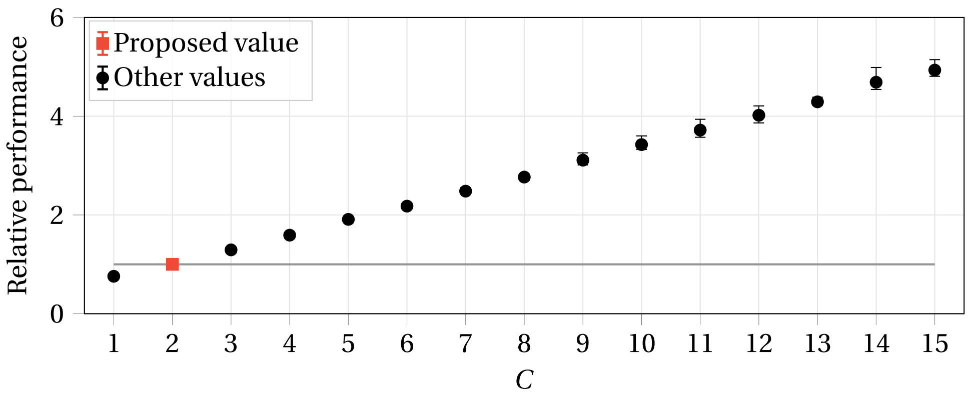


(b) `LPMC_RR_L`

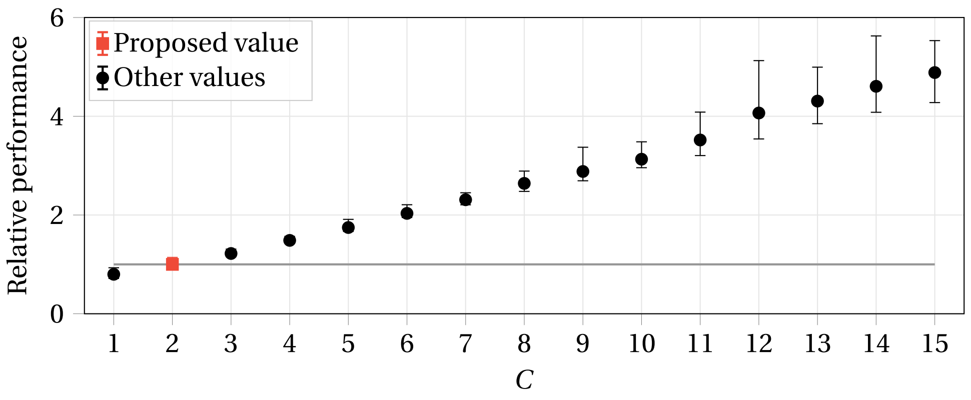


(c) `LPMC_Full_L`

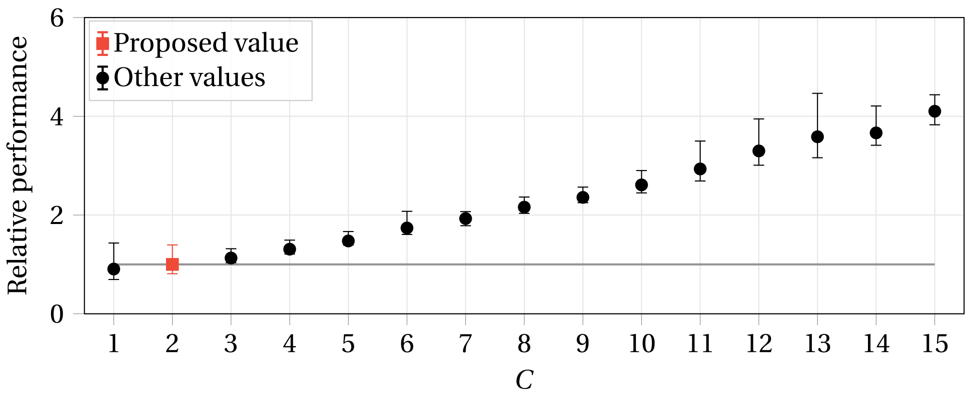
Figure 2.6: Sensitivity analysis for the parameter Δ for the three LPMC models using the large dataset. The black error bars correspond to the tested values and the red to the proposed value. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.



(a) `LPMC_DC_L`



(b) `LPMC_RR_L`



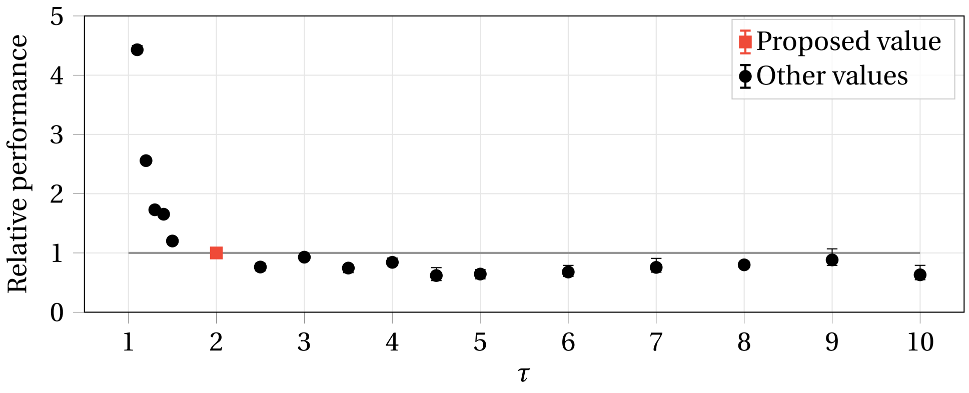
(c) `LPMC_Full_L`

Figure 2.7: Sensitivity analysis for the parameter C for the three LPMC models using the large dataset. The black error bars correspond to the tested values and the red to the proposed value. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.

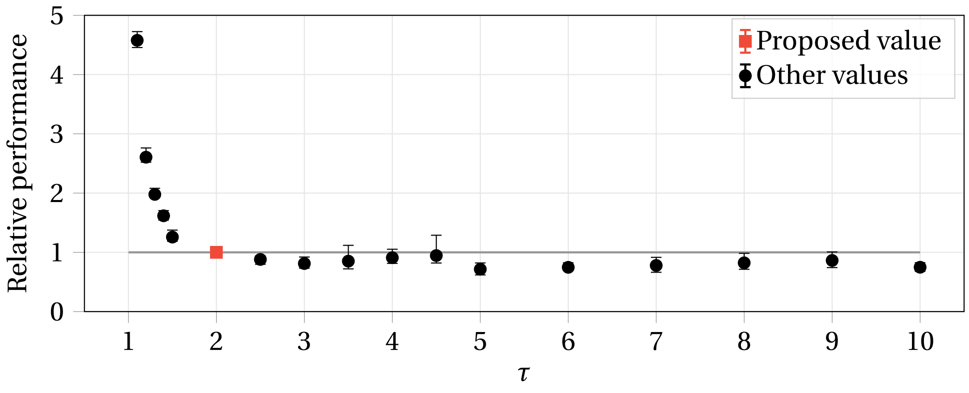
Figure 2.8 shows the analysis for the parameter τ; the expansion factor for the batch size. This parameter is crucial since it decides how fast the algorithm uses the full dataset for its iterations. As expected, a small value for τ leads to a longer execution time for all three models. However, a larger value for τ is more efficient for the smaller models. Indeed, for both the `LPMC_DC_L` and the `LPMC_RR_L`, using larger values lead to around 20% decrease in execution time. Since these models are quite small in the number of parameters, they already profit greatly from the starting batch size. It is more favorable to switch the optimization algorithm and use more data. However, it seems that for the model `LPMC_Full_L`, switching too soon leads to more variance in the execution time. Therefore, staying more robust and using a smaller value for τ is better. We thus propose to use $\tau = 2$.

Figure 2.9 shows the analysis for the parameter Δ_H; the threshold for hybridization. Interestingly, this parameter does not influence the execution time for the smallest model, the `LPMC_DC_L`. As discussed in Section 2.4.3, the `HAMABS` algorithm is not the fastest algorithm for the `LPMC_DC` models. Also, the execution is so small that the difference between Newton's method and BFGS is small. In addition, we see in Table E.1, that Newton's method is faster than BFGS^{-1} on this particular model. Therefore, it is better to switch as late as possible. For the slightly larger model, `LPMC_RR_L`, high thresholds increase the execution time. As shown in Table E.1, the BFGS methods are faster to optimize the models `LPMC_RR`. Therefore, switching to BFGS too late is expected to increase the execution time. However, it seems that switching as quickly as possible to BFGS is the most efficient for this model. This is most likely thanks to the help of the Hessian computation at the first step. Indeed, the BFGS algorithm generally starts with an Identity matrix. However, if we first perform a Newton step with the computation of the Hessian, it gives a good approximation as the starting point. Therefore, a smaller threshold for hybridization is recommended for the medium models. However, since the goal is to optimize large models as soon as possible, it is more important to use parameters specifically proposed for these models. As seen in Figure 2.9c, the best switch appears at around 30% of the data. It is slower to switch too soon since BFGS still takes more time than Newton's method to perform the early steps. Furthermore, it is also slower to switch later since Newton's method takes too much time to compute the Hessian. Therefore, the proposed value is $\Delta_H = 30\%$.

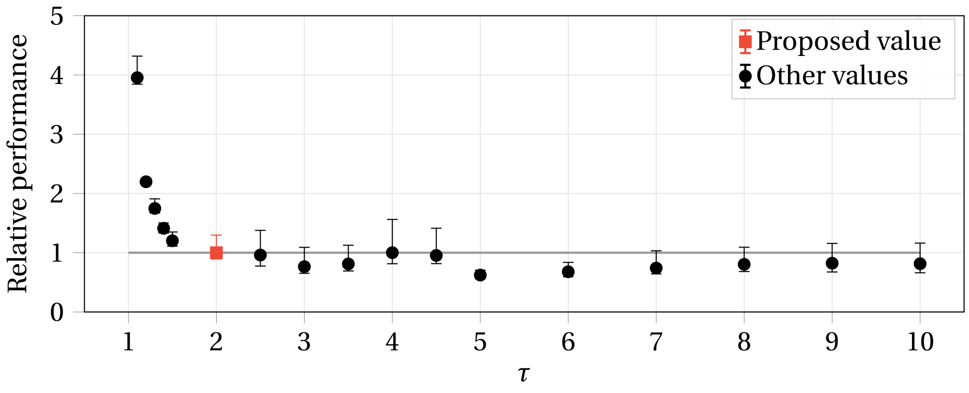
Figure 2.10 shows the analysis for the parameter ε; the threshold for the stopping criterion. As shown in Figure 2.10, using a stopping criterion that is too high is faster. However, it also leads to incorrect results. Indeed, the algorithm stops too soon, and the optimization has not yet converged to the optimal point. This means that we lose accuracy on the parameter values. We also see that using a too small stopping criterion leads to a significant increase in execution time. In this case, this level of required precision is unreasonable. Therefore, a good value is a compromise between precision and speed. As shown in all these figures, a value for the stopping criterion between 10^{-8} and 10^{-5} is acceptable. We propose to use 10^{-6} even if other values can be used.



(a) `LPMC_DC_L`

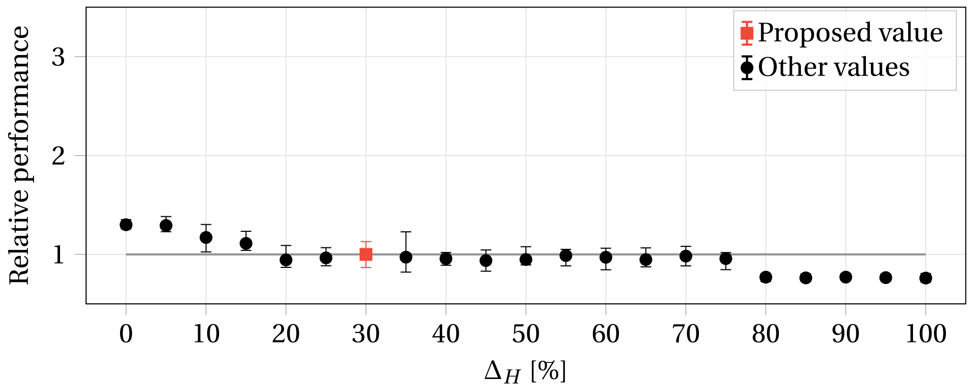


(b) `LPMC_RR_L`

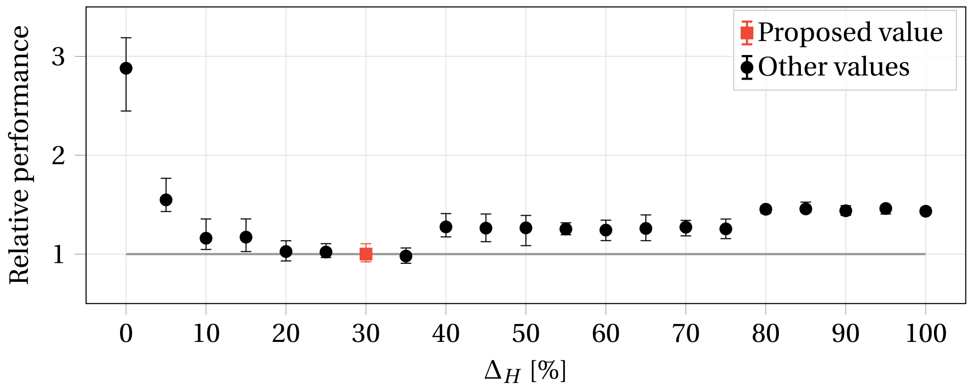


(c) `LPMC_Full_L`

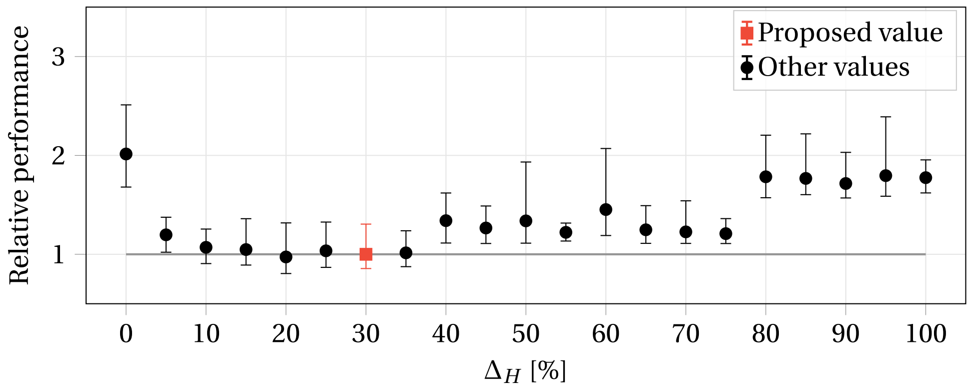
Figure 2.8: Sensitivity analysis for the parameter τ for the three LPMC models using the large dataset. The black error bars correspond to the tested values and the red to the proposed value. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.



(a) `LPMC_DC_L`

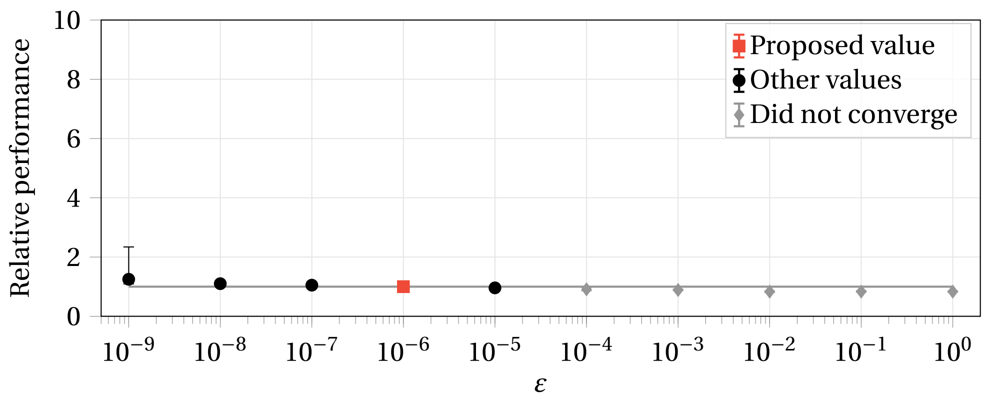


(b) `LPMC_RR_L`

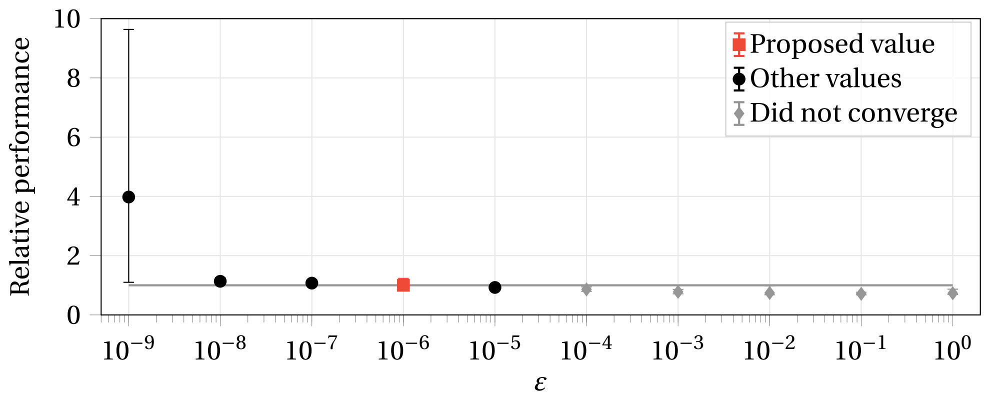


(c) `LPMC_Full_L`

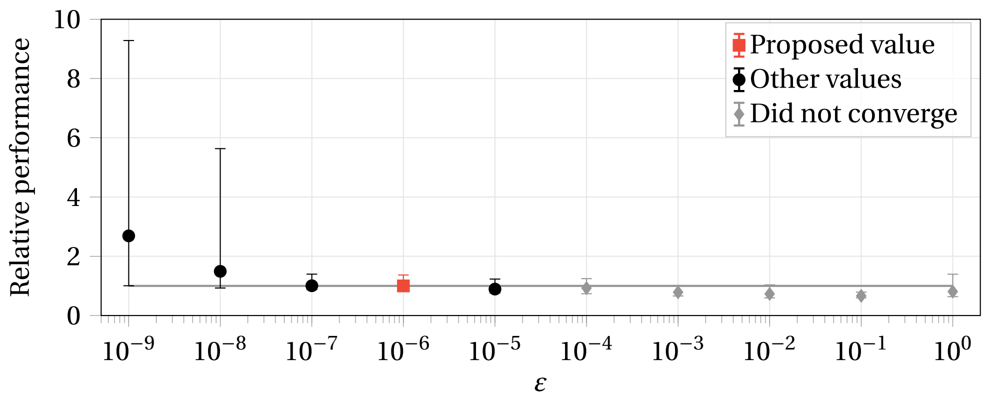
Figure 2.9: Sensitivity analysis for the parameter Δ_H for the three LPMC models using the large dataset. The black error bars correspond to the tested values and the red to the proposed value. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.



(a) `LPMC_DC_L`

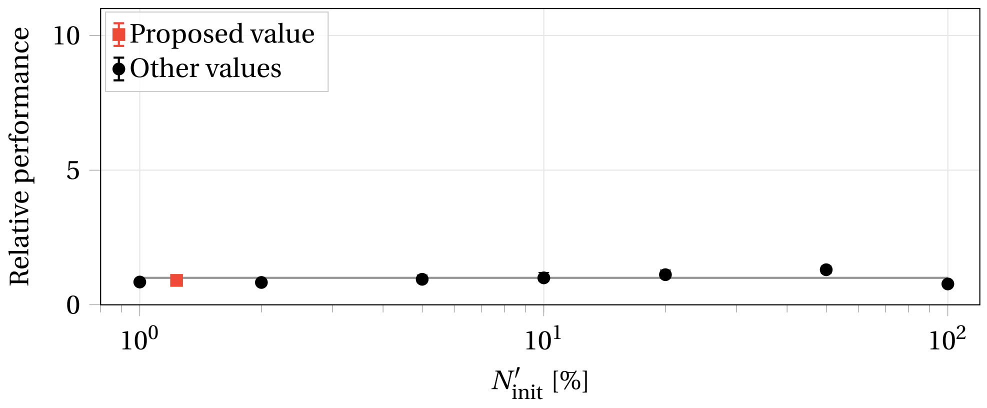


(b) `LPMC_RR_L`

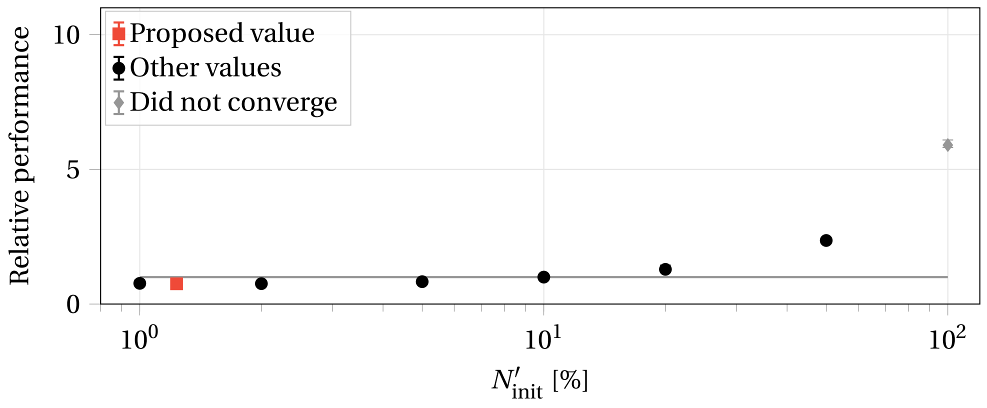


(c) `LPMC_Full_L`

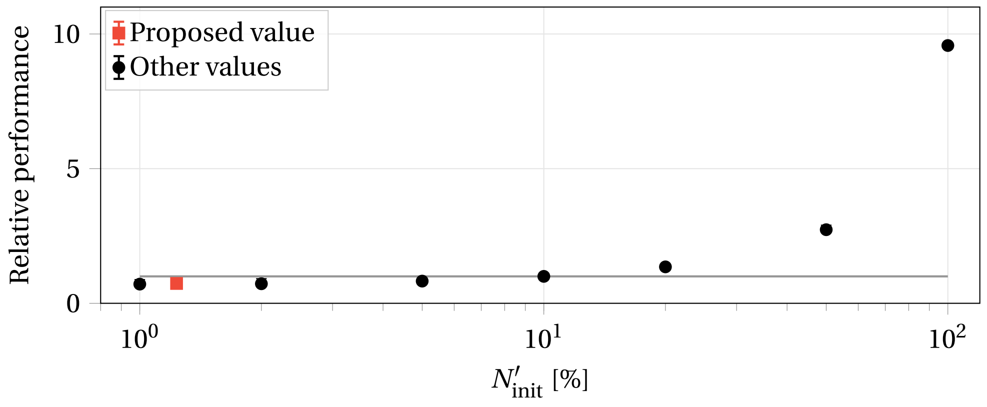
Figure 2.10: Sensitivity analysis for the parameter ε for the three LPMC models using the large dataset. The black error bars correspond to the tested values, the red to the proposed value, and the gray to optimization that were stopped too early and thus did not fully converge. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.



(a) `LPMC_DC_L`



(b) `LPMC_RR_L`



(c) `LPMC_Full_L`

Figure 2.11: Sensitivity analysis for the parameter N'_{init} for the three LPMC models using the large dataset. The black error bars correspond to the tested values, the red to the proposed value, and the gray to optimization that were stopped too early. The gray line correspond to the benchmark with the proposed parameters for the relative performance. The horizontal axis corresponds to the candidate parameter values and the vertical axis to the relative performance measured as time to convergence.

Finally, Figure 2.11 shows the analysis for the parameter N'_{init}; the initial batch size. For the small model, Figure 2.11a, the initial batch size does not impact the final optimization time. This behavior is expected as the optimization of this model using the `HAMABS` algorithm is relatively slow compared to standard methods, *e.g.* BFGS. Indeed, the fastest estimation time of the model corresponds to a batch size of 100%, which is equivalent to the BFGS^{-1} method. For the remaining two models, any initial batch size between 1% and 10% of the dataset reached the same performance. Values above 10% slow down the algorithm.

As the algorithm can easily and quickly increase the batch size if it gets stuck, it is recommended to use the smallest possible batch size when optimizing with stochastic methods. We, thus, decided to use an initial batch size of 1'000 observations, 1.23% for this dataset.

The sensitivity analysis shows that the parameters might depend on the model's size. However, the goal of this article is to speed up the optimization process of large choice models. Therefore, we selected parameters that lead to an improvement on these large models. Besides, half the execution time of a small model would only result in a gain of a few seconds. On the other hand, the same speedup would lead to minutes or even hours on larger models. It is, therefore, more rewarding to speed up the larger models.

2.5 Summary

In this chapter, we present three primary contributions to estimate DCMs: usage of stochastic hessian, an adaptive batch size method, and the hybridization between optimization algorithms. We test 15 different algorithms, from standard to stochastic hybrid adaptive batch size algorithms. We show that using an adaptive batch size technique is beneficial for the optimization time. Besides, since the AMABS method can be used with different optimization algorithms, we created three hybrid AMABS algorithms. We have shown that the fastest algorithm is the HAMABS algorithm. It speeds up the optimization by a factor of up to 23 on larger DCMs. Therefore, using faster algorithms opens the research to new possibilities for the future of choice modeling. As a concrete example, faster optimization time allows researchers to test many more specifications in the same amount of time. This can thus be used to develop Assisted Utility Specification (AUS) techniques to speed up the modelization of DCMs.

In the future, we would like to work on two different improvements. The first one concerns hybridization. The current way of doing it depends on the starting batch size. Indeed, due to the geometrical rule to increase the batch size in AMABS, it would be possible to miss the 30% mark to switch the optimization algorithm. Therefore, we would like to work on a better switch for hybridization. One possible direction is to compare the improvement made by each algorithm in one step over the time it takes to do it. We would then have a metric in the percentage of improvement over seconds, and we could easily decide to switch the algorithm when BFGS leads to more improvements per second. The second improvement can be made on the rule for updating the batch size in AMABS. Indeed, the geometrical rule seems to work well. However, it is possible that combining multiple rules could lead to faster optimization time. The final improvement is to integrate algorithms dealing with bounds. Indeed, nested logit and Multivariate Extreme Value (MEV) models require bounds for some parameters. It is thus essential to integrate them into future optimization algorithms. Finally, we hope to see the HAMABS algorithm integrated in Pandas Biogeme to help all users estimate DCMs more efficiently.

3 Generating synthetic data from deep learning with expert knowledge

This chapter is based upon the following preprint that has been submitted to the *Journal of Transportation Research Part C: Emerging Technologies*:

> Lederrey, Gael, Hillel, Tim and Bierlaire, Michel. DATGAN: Integrating expert knowledge into deep learning for synthetic tabular data. *arXiv:2203.03489 [cs]*, March 2022. https://arxiv.org/abs/2203.03489. arXiv: 2203.03489.

3.1 Introduction

A massive increase in data availability has created tremendous opportunities for targeted modeling and a greater understanding of systems, particularly those involving human behavior. However, reliance on data creates a division based on data. For example, leading international cities in developed nations produce rich data about population movements and interactions with infrastructures. On the other hand, undeveloped nations have much lower data availability. The collection of such data, particularly socio-economic, can be prohibitively expensive. It can, thus, prevent non-data-rich areas from modeling. Furthermore, data can be controlled by certain groups (companies, government, or public agencies), who may be unwilling or unable to make complete data publicly available. In addition, sharing detailed disaggregated socio-economic data has become increasingly complex with the current focus on data privacy via the General Data Protection Regulation (GDPR). Thus, synthetic data generation, *i.e.* the creation of synthetic data samples consistent with the true population, has the opportunity to address many of these limitations.

There are multiple use cases for synthetic tabular data: (i) The most common use case is dataset augmentation. It can allow researchers and modelers to approximate a large population from a smaller sample, thus reducing the cost of data collection. (ii) Secondly, synthetic data can be used for privacy preservation. It can, then, enable the sharing of detailed disaggregate populations without contravening GDPR and other data privacy laws. (iii) Another use case is bias correction. Synthetic data can correct bias in existing samples, allowing for reliable modeling of marginal, minority groups, and behavior. (iv) Finally, synthetic data generation models can be used as transfer learning methods. They can thus be used to transfer data from one city or context to a new context, allowing for detailed modeling where existing high-quality data is not available. In this chapter, we focus on synthetic data generation in the context of synthetic population. Such populations are generally used for simulation in agent-based models, particularly for activity-based transport models. However, the techniques proposed and reviewed in this chapter can be used in any context where there is a need for detailed tabular datasets.

Many methods have been developed to generate such synthetic populations in existing studies. The two main approaches are statistical techniques such as Iterative Proportional Fitting (IPF) (Deming and Stephan, 1940) or simulation using Gibbs sampling (Geman and Geman, 1984), and machine learning techniques. While the first approaches have been well studied within the transportation community, the latter comes from the Machine Learning community and generally focuses on general synthetic data. These deep learning methods, such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), have already been tested against standard statistical techniques and outperform them while generating correlations in synthetic datasets. However, these methods are data-driven. They, therefore, lack control over the generation process. Without controlling the latter, it is impossible to know how well the deep learning models have understood the original sample, *i.e.* which correlations between the variables in the original dataset the models have learned. It can, thus, lead to spurious correlations or propagation of existing bias in the sample. In addition, there is generally no focus on the representativity of the output, which is crucial for the accurate understanding of socio-economic characteristics.

This chapter, thus, proposes a novel model that controls the generation process of such synthetic tabular data. We propose to let the researcher or modeler design a network to represent the interactions between the variables with a Directed Acyclic Graph (DAG). This DAG is then used to model the network structure that will generate the synthetic data. Allowing researchers to control the process has three main advantages: they can tinker with the data generation process, create hypothetical datasets, and control the dependencies for forecasting. In this chapter, we thus present our new GAN model named Directed Acyclic Tabular GAN (DATGAN). We show that it outperforms state-of-the-art synthetic data generators on multiple metrics. These metrics have been created to allow for systematic testing using formal statistical analysis and supervised learning-based approaches. We also provide a sensitivity analysis on the DAG to show its effect on the data generation process. Finally, we show how the DAG can create hypothetical situations and generate a synthetic dataset based on the new rules.

The rest of this chapter is laid as follows. In the next section, we present the literature review. We first introduce the existing approaches for population synthesis and then discuss the different research axes. Finally, we conclude the literature review with the opportunities and limitations of existing research. In Section 3.3, we present the whole methodology for the DATGAN. We discuss how to preprocess the data, what models are used for the generator and the discriminator, and how to use the DAG to create the generator's structure using Long Short-Term Memory (LSTM) cells. Section 3.4 presents the case studies and Section 3.5 shows the results. We conclude this chapter in Section 3.6 and give ideas for future work.

3.2 Literature review

There are five main research axes for synthetic tabular data generation: simulation/activity-based modeling, Machine Learning (ML) efficacy, bias correction, privacy preservation, and transfer learning. These research axes are discussed in detail in Section 3.2.2. The literature review first focuses on population synthesis with older methods such as IPF and Gibbs sampling. Then, we look at more general ML techniques for synthetic tabular data generation. Then, in Section 3.2.3, we discuss in more detail some state-of-the-art models that we selected to compare to the model presented in this article. Next, Section 3.2.4 is dedicated to model evaluation and shows how the transportation and ML communities are evaluating generated synthetic datasets. Finally, in Section 3.2.5, we discuss the opportunities and limitations of these techniques linked to the five research axes.

3.2.1 Existing approaches for synthetic tabular data generation

One of the primary uses of synthetic tabular data has been for the creation of synthetic populations, in particular for transportation research. As a result, many research contributions focus specifically on this topic, using different techniques. Table 3.1 summarizes all the methodologies seen in the literature used to generate synthetic tabular data or synthetic populations. In this section, we provide a detailed discussion of all these methods.

Resampling and simulation-based approaches are the two main methods to generate synthetic populations within the transportation community. The first one is based on IPF methods (Deming and Stephan, 1940). It consists of adjusting a matrix proportionally to produce a new table so that the specified marginals are individually conserved. Beckman et al. (1996), first, use this methodology to create a synthetic population based on the SF3 (San Francisco area) census data. Auld et al. (2009) and Barthelemy and Toint (2013) both propose to improve the IPF methodology using a multi-step procedures. While IPF methods are simple to implement, this technique has multiple significant limitations in generating highly detailed and realistic synthetic tabular data. Firstly, there is no interaction between the variables with the basic algorithm. It is possible to add these interactions by adding more dimensions to the table. However, for each level of interaction, one more dimension has to be added to the table. It, thus, quickly becomes a computationally expensive algorithm. In addition, IPF cannot differentiate between structural and sampling zeros. Multiple methods have been suggested to avoid sampling zero issues in the literature, such as Auld et al. (2009). Finally, IPF cannot differentiate between the different data types (categorical and continuous). Thus, researchers

Table 3.1: Main methods for synthetic tabular data generation found in the transportation literature and in the Machine Learning community.

Methods	Description	References	Advantages	Disadvantages
IPF	Method using a starting synthetic table and iteratively update it to match the marginals of an original table.	Auld et al. (2009); Barthelemy and Toint (2013); Rich (2018)	• Efficient in its basic form • Simple to implement	• No interactions between variables • Computationally expensive if more complexity added • Prone to sampling zero issue • No differences between data types
Gibbs sampling	Gibbs sampler trained until reaching stationary state on prepared conditionals.	Farooq et al. (2013); Casati et al. (2015); Kim and Lee (2016)	• Learn from marginals • Outperforms IPF • Can be linked to other methods	• No interactions between variables • Computationally expensive • Probability distributions are an assumption
Bayesian networks	Probabilistic graphical model used to determine probabilistic inferences between the variables.	Sun and Erath (2015); Zhang et al. (2019a)	• Dependencies of variables defined prior to training • Probabilistic model	• Requires prior information on the dataset • Computationally expensive when dealing with large and sparse datasets
VAE	Pair of neural networks composed of an encoder and a decoder. Transforms data in a latent space to reduce its dimensionality. The encoding-decoding scheme has to be learned.	Garrido et al. (2019); Xu et al. (2019)	• Aims to learn a latent representation of the variables • Latent space is suitable for inference and completion of data	• Might not be able to learn the true posterior distribution • Outperformed by GANs
GAN	Pair of neural networks composed of a generator and a discriminator. The generator is trained to fool the discriminator. Learning process is a two players minimax game.	Goodfellow et al. (2014); Xu and Veeramachaneni (2018); Xu et al. (2019); Zhao et al. (2021)	• Generator never sees true data (privacy ensured) • Architectures of both neural networks are flexible • Current state-of-the-art generative model	• Equilibrium between both neural networks difficult to achieve • Dependencies of variables cannot be controlled

have developed new techniques to generate synthetic populations, such as Markov Chain Monte Carlo (MCMC) simulation.

Farooq et al. (2013) proposes to use a MCMC simulation using Gibbs sampling to generate synthetic populations. The idea is to draw from an (unknown) multi-dimensional random variable characterizing the distribution of individuals in the population using a Gibbs sampler. The marginals and conditional distributions used by the Gibbs sampler are generated from real data. Since the full-conditionals are rarely available for all the attributes in the original data, the authors use a parametric model to construct the missing conditional distributions. The authors show that this simulation technique outperforms IPF methods using multiple statistical metrics such as R^2 and Standardized Root Mean Squared Error (SRMSE) (Müller and Axhausen, 2010). Multiple improvements have been made on the original method (Casati et al., 2015; Kim and Lee, 2016; Philips et al., 2017). However, while simulation-based techniques outperform IPF techniques, these methods still have limitations in the context of synthetic population generation. The main issue is that the models are working with conditionals only. This can be an advantage if only this information is available. However, since MCMC methods must assume the type of probability distributions the variables follow, wrong assumptions can lead to fundamentally incorrect distributions.

While these statistical methods have been widely used in the transportation community, they are outdated compared to ML techniques. For example, Borysov et al. (2019) show that their ML-based approaches outperform MCMC-based approaches on multiple criteria. Indeed, recent advances in Machine Learning and data generation techniques have enabled new approaches for generating synthetic data. We identify three primary ML-based approaches that have been used for this purpose: Bayesian networks, VAE, and GANs.

Sun and Erath (2015) use Bayesian networks to generate such populations. Bayesian networks are graphical models that encode probability distributions for a set of variables. They use a DAG to represent the dependencies between the variables and a set of local probability distributions for each variable in the original table and given conditional probabilities. The authors show that their model outperforms both IPF and Gibbs sampling. Zhang et al. (2019a) extended this concept further by using a three-step procedure to generate a population and its social network. They use a Bayesian network to create a synthetic population of households, an integer problem with Langrangian relaxation for the assignment problem, and an Exponential Random Graph Model (ERGM) for the social network simulation.

VAEs (Kingma and Welling, 2014) aim to reduce the dimensionality of the data into an encoded vector in the latent space. Data can then be generated more easily in this latent space since it is smaller in dimensionality. For example, Borysov et al. (2019) have used a VAE to generate a synthetic population. They demonstrated that their VAE model outperforms both IPF and Gibbs sampling for generating complex data. However, this type of method has quickly been outperformed by the current state-of-the-art method for generating synthetic data: GANs.

GANs (Goodfellow et al., 2014) are considered the state-of-the-art methodology to generate synthetic data. It consists of a two-player game between two neural networks, the *generator* and the *discriminator*. Both neural networks compete against each other on independent, unsupervised tasks. The generator processes random noise to produce synthetic data. Its goal is to generate synthetic data that cannot be distinguished from original data to fool the discriminator. On the other hand, the discriminator (or critic) evaluates the synthetic data against original data to provide a classification or continuous score on each data point on whether the data is original or synthetic. The generator is then trained through backpropagation. Since the generator and the discriminator can always improve, no specific methodology exists to stop this game. Thus, one stops it when the sampled data are considered good enough. Figure 3.1 shows the schematic representation of a GAN. The interactive visualization GAN Lab (Kahng et al., 2019) provides an interesting tool to visualize and understand how GANs operate at https://poloclub.github.io/ganlab/.

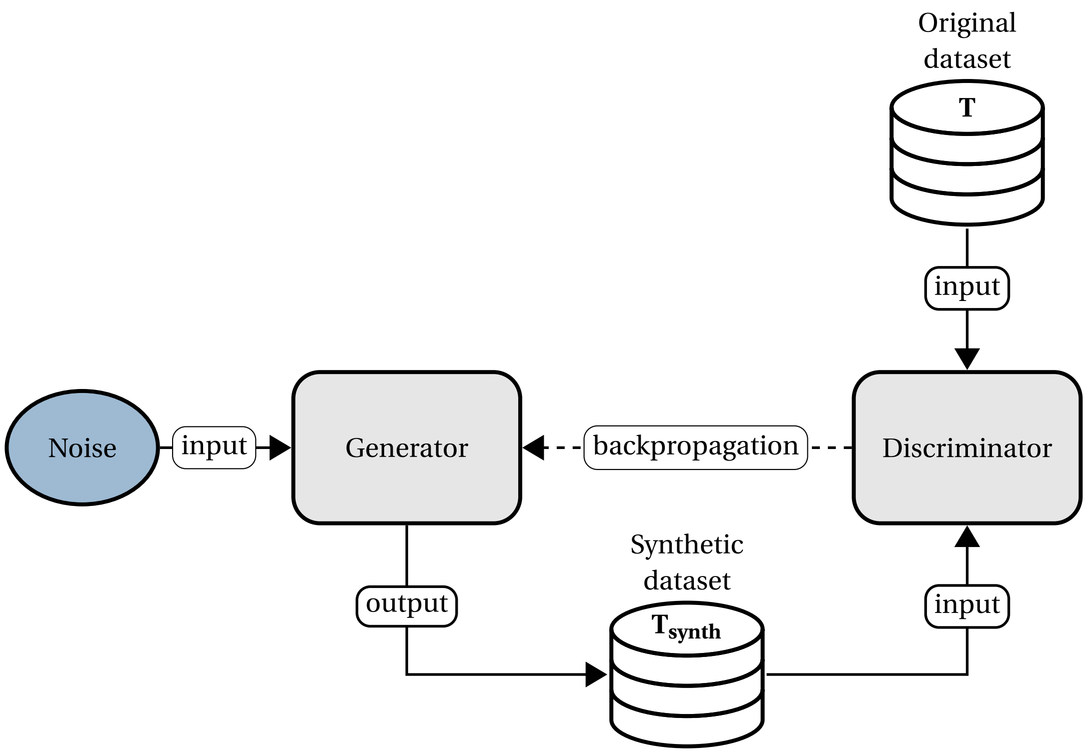


Figure 3.1: Schematic representation of the standard GAN structure.

GANs have quickly evolved to become more specialized. For example, Arjovsky et al. (2017) demonstrate that using a discrete loss function results in issues such as vanishing gradients. They thus propose an alternative continuous loss function based on the Wasserstein distance. This GAN is therefore named Wasserstein GAN (WGAN). Further key developments in GAN research include the introduction of a penalty on the gradient during model training (Gulrajani et al., 2017) or the addition of conditionality (Mirza and Osindero, 2014). While the primary application of GANs has been the generation of image data, with a particular focus on human faces (Alqahtani et al., 2021), researchers have also developed specific architectures for tabular data. It, thus, allowed transportation researchers to switch their focus to more general synthetic tabular data generation rather than synthetic population generation.

TableGAN (Park et al., 2018) and Tabular GAN (TGAN) (Xu and Veeramachaneni, 2018) are two specific GAN models for tabular data. TableGAN has been developed with privacy-preservation techniques in mind. This model is based on Deep Convolutional GAN (DCGAN) (Radford et al., 2016). On the other hand, TGAN has been developed to reproduce tabular data as realistically as possible using LSTM cells for the generator (Hochreiter and Schmidhuber, 1997). The authors demonstrated that TGAN outperforms tableGAN. Researchers have also developed their GAN structures to generate synthetic populations in the transportation community. For example, Garrido et al. (2019) develop their own GAN structure based on WGAN to use tabular data. They show that this new model was statistically better than IPF techniques, Gibbs sampling, and the VAE of Borysov et al. (2019). Finally, Badu-Marfo et al. (2020) created a new GAN named Composite Travel GAN (CTGAN). Their GAN is based on Coupled GAN (CoGAN) (Liu and Tuzel, 2016) and is used to generate the table of attributes for the population and the sequence of Origin-Destination segments. They show that CTGAN outperforms VAE statistically. While these models are showing outstanding performances compared to previous methods, the switch to data-driven methods has hindered the control of the researchers or modelers on the generation process. The lack of control during this process can hinder the final results depending on the research axis. Thus, in the next section, we discuss multiple axes found in the literature, some requiring high control of the generated synthetic data.

3.2.2 Research axes

The primary focus of existing population synthesis in transportation has been for direct use in simulation models. On the other hand, the deep learning community motivates their research by stating that using more data improves the efficacy of ML models. For example, Jha et al. (2019) show that a larger and more complete dataset leads to better validation and fewer uncertainties. Other examples discussing the dataset size can be found in the literature (Barbedo, 2018; Linjordet and Balog, 2019). However, this is not the only synthetic tabular data generation research axis. In the remainder of this section, we present and discuss five different research axes for synthetic generation in the literature.

Simulation and agent-based modeling

Agent-based models (Bonabeau, 2002) are used to simulate the actions and interactions of autonomous agents in order to understand the behavior of a system. These models are extensively used in the transportation community (Kagho et al., 2020) and require a large amount of data to be adequately trained. Synthetic data are, thus, often used to replace scarce and expensive original data.

In the transportation community, the predominant focus for the population synthesis papers introduced in Section 3.2.1 is for direct use in simulation. For example, Beckman et al. (1996), Barthelemy and Toint (2013), Farooq et al. (2013), Borysov et al. (2019), and Garrido et al. (2019) all motivate their work by discussing the generation of synthetic population for (agent-based) simulation models.

Privacy preservation

Privacy preservation techniques ensure that private information is not disclosed using data or ML models. Synthetic data can, thus, replace highly sensitive original data when privacy is of concern. Methods using only the conditionals, such as IPF or Gibbs sampling, are especially effective since the methods never use the original data to generate the synthetic data.

For example, Barthelemy and Toint (2013) motivate their model (a three-step procedure based on simulation techniques) to improve the privacy preservation of the standard IPF methods. They state that the standard method tends to repeat observations, and thus it is possible to retrieve information from the original dataset. More recently, tableGAN (Park et al., 2018) has been specifically designed to preserve the original datasets' privacy. Multiple GAN models have been created in computer vision, with privacy preservation as the core motivation. For example, Liu et al. (2019) created the Privacy Preserving GAN (PPGAN). This GAN uses differential privacy by adding specifically designed noise to the gradient during the learning procedure. Yin and Yang (2018), on the other hand, directly generated protected data within the generator of their GAN by removing some sensible information and encoding them in the generated data. They tested their synthetic data against attack models to show that their GAN could generate more complex data to be deciphered. More recently, Zhao et al. (2021) motivate and test their model on privacy preservation.

Machine Learning efficacy

The generation of synthetic populations also enables the augmentation of existing real-world datasets with synthetic individuals, increasing the dataset's size and variability. Several studies have investigated the estimation of ML techniques on augmented or fully synthetic data. This concept is already widely used on images (Shorten and Khoshgoftaar, 2019). While simple techniques such as rotating or scaling images can be used in Computer Vision, applying such simple tricks to tabular data is impossible. Thus, researchers have been developing models aiming at augmenting tabular data.

For example, Xu and Veeramachaneni (2018) motivate the development of TGAN because organizations are using ML on relational tabular data to augment process workflows carried out by humans. Furthermore, they state that these synthetic datasets can either be used as an augmentation for the existing datasets or as a means to preserve privacy. On the other hand, Xu et al. (2019) do not provide a clear motivation for using synthetic datasets. However, they test their models on ML efficacy by replacing the training data with the generated synthetic data. More recently, both Wen et al. (2021) and Zhao et al. (2021) motivate their models using ML efficacy as the core method to assess their generated synthetic datasets.

Bias correction

Synthetic data can also correct bias in existing datasets by controlling the data generation process. It builds on standard resampling methods (Rubin, 1973) to rebalance the dataset, which reduces the signal-to-noise ratio of the existing data (by removing oversampled data or resampling undersampled data). Indeed, data generation techniques can also be used to augment and rebalance an existing dataset.

For example, Conditional GANs (Mirza and Osindero, 2014) have been created to tackle such imbalance issues. The idea of such GANs is to generate synthetic data using prior information. They, thus, increase the probability of generating synthetic data with the given information. Xu et al. (2019) have adapted this methodology to tabular data with Conditional Tabular GAN (CTGAN). They show that the conditionality is truly efficient for ML models when the data is highly imbalanced. They create synthetic datasets addressing the imbalance and trained ML models on this synthetic and the original dataset. The models trained on the synthetic datasets perform better than those trained on the original dataset. Previously, Farooq et al. (2013) motivate their research on population synthesis with Gibbs sampling using the fact that it can complete datasets. However, the authors have not formally evaluated this use of synthetic data.

Transfer learning

The use of Conditional GANs and other conditional data generation approaches enables the possibility for transfer learning, where knowledge from one context with large data availability can be transferred to another context with lower data availability.

For example, Noguchi and Harada (2019) propose a new method using BigGAN to transfer the knowledge learned on large datasets and apply this knowledge to a dataset with only 25 images. They show that they can add a new class to a pre-trained generator without disturbing the performance of the original domain. Wang et al. (2020) propose to use a miner network that identifies which distribution of multiple pre-trained GANs is the most beneficial for a specific target. This mining pushed the sampling towards more suitable regions in the latent space. Therefore, MineGAN can transfer the knowledge of multiple GANs such as BigGAN and Progressive GAN to a domain with fewer images. Other relevant methods for transfer knowledge can be found in the articles of Jeon et al. (2020) and Frégier and Gouray (2020).

While this research axis has already been explored in the computer vision community, it has not been explored in population synthesis. Indeed, while transferring knowledge between two tabular datasets might not make sense, it could be used for tabular data of populations. For example, census data are often collected regularly, *e.g.* every two to five years. We could, thus, imagine using GANs trained on data from previous years to transfer their knowledge to the most recent years.

3.2.3 State-of-the-art models

We present a detailed overview of four approaches demonstrated to achieve the best performance when generating synthetic tabular data. As such, these approaches represent the state-of-the-art in this field. The four approaches, which all make use of deep learning algorithms, are introduced across three key articles: Xu and Veeramachaneni (2018), Xu et al. (2019), and Zhao et al. (2021). A summary of the models is given in Table 3.2.

Table 3.2: Summary of the state-of-the-art models selected for comparison with DATGAN.

Model	Article	Information
TGAN	Xu and Veeramachaneni (2018)	• *Generator:* LSTM cells in linear arrangement • *Discriminator:* Fully-connected neural network • *Data preprocessing:* Continuous vs categorical • *Loss function:* Cross-entropy loss • *Conditionality:* None
CTGAN	Xu et al. (2019)	• *Generator:* Fully-connected neural network • *Discriminator:* Fully-connected neural network • *Data preprocessing:* Continuous vs categorical • *Loss function:* Wasserstein loss with gradient-penalty • *Conditionality:* On categorical variables
TVAE	Xu et al. (2019)	• *Encoder:* Updated structure for preprocessed data • *Decoder:* Similar to conventional VAE • *Data preprocessing:* Continuous vs categorical • *Loss function:* ELBO loss • *Conditionality:* None
CTAB-GAN	Zhao et al. (2021)	• *Generator:* Convolutional neural network • *Discriminator:* Convolutional neural network • *Classifier:* Multi-layer perceptron • *Data preprocessing:* Continuous vs categorical vs mixed • *Loss function:* Cross-entropy, information and classification losses • *Conditionality:* On categorical variables

While some GANs have been developed for privacy preservation, TGAN (Xu and Veeramachaneni, 2018) focuses on learning the marginal distributions using recurrent neural networks. Since our focus is on creating representative synthetic data, we thus selected TGAN as the first model to be compared. It uses LSTM cells (Hochreiter and Schmidhuber, 1997) to generate each variable in the table. The LSTM cells are arranged linearly, following the order of the variables in the dataset. The authors make the difference between categorical and continuous variables. Both variable types are encoded differently: (i) continuous variables are encoded using Gaussian mixtures; (ii) categorical variables are one-hot encoded. Finally, TGAN is trained using the standard minimax loss function (Goodfellow et al., 2014), and it is compared to other data synthesizers such as Gaussian Copula and Bayesian Networks. The authors show that TGAN outperforms all these methods.

CTGAN (Xu et al., 2019) uses a fully-connected neural network for both the generator and the critic. Like TGAN, the variables are differentiated between categorical and continuous variables. CTGAN uses the same encoding procedure for both variable types with a slight difference for continuous variables: a Variational Gaussian Mixture (VGM) is used instead of standard Gaussian mixtures. The VGM uses a Dirichlet process to determine the number of modes in the distribution, while it is predefined for the standard Gaussian mixture. In addition, this model uses conditionality by adding a conditional vector on categorical variables. Finally, CTGAN is trained using the Wasserstein loss with gradient-penalty (Gulrajani et al., 2017).

TVAE (Xu et al., 2019) is an adaptation of a standard VAE by modifying the loss function and preprocessing the data. The variables are encoded using the same procedure as in CTGAN. TVAE uses the ELBO loss (Kingma and Welling, 2014). While the encoder is slightly updated compared to conventional VAEs, the decoder keeps a usual structure. The authors have compared TVAE, CTGAN, and other methods for synthesizing tabular data such as tableGAN. They show that both TVAE and CTGAN outperform other methods. On multiple metrics, TVAE performs better than CTGAN. However, as stated by the authors, CTGAN achieves differential privacy (Jordon et al., 2018) easier than TVAE since the generator never sees the original data.

Finally, the particularity of CTAB-GAN (Zhao et al., 2021) compared to the previous models is that it aims at fixing issues with skewed continuous distributions. Indeed, continuous distributions can take many forms, such as long-tailed, exponential, or mixed distributions. Therefore, this model implements multiple data preprocessing methods for different distributions. CTAB-GAN comprises three neural networks: a generator, a discriminator, and an additional classifier. The latter is used to learn the semantic integrity (data type) of the original data and predict the synthetic data classes. This helps produce more accurate labeled synthetic data. The generator and discriminator are convolutional neural networks, while the classifier is a multi-layer perceptron. In addition, CTAB-GAN uses conditionality to counter the imbalance in the training dataset to improve the learning process. Finally, CTAB-GAN is trained using the standard cross-entropy loss function with the addition of an information loss and a classification loss. The authors have tested their model against other state-of-the-art models such as tableGAN and CTGAN. They have shown that their model outperforms all the other models using ML efficacy and statistical similarity metrics.

3.2.4 Model evaluation

Model evaluation is intrinsically linked to the research axis for which a model was developed. Indeed, a generator developed to correct bias should not be tested on the same characteristics as a model developed for privacy preservation. Thus, researchers have devised different methods for assessing generated synthetic datasets. In this section, we discuss two types of methods that are primarily used to assess the representativity of a synthetic dataset compared to its original counterpart: statistical methods and ML methods.

Statistical assessments

Multiple statistical tests can be used to compare two distributions, such as the χ^2 test or the Student's t-test. While these tests can provide good information when comparing the distributions of each variable separately, it does not consider the correlation between the variables. Since this aspect is essential for creating representative synthetic populations, researchers in the transportation community have been developing new statistical tests to address this issue. The SRMSE (Müller and Axhausen, 2010) is used in most transportation articles working on population synthesis to assess the generated datasets. The test consists in selecting one or multiple variables in a dataset and creating a frequency list based on the appearance of each unique value. We can, then, apply SRMSE (see Equation 3.41) formulation on this frequency list. While this technique has been shown to work well compared to other statistical methods, it has two main flaws: (i) the frequency lists are computed by counting the unique values (or combinations of unique values). Therefore, it is preferably used on discrete values. (ii) The choice of variables (or combination of variables) is up to the researcher or modeler. Thus, articles using this methodology tend only to test a couple of combinations. In order to address the first flaws, we can transform the continuous values into discrete values by assigning them to specific buckets. This is a relatively simple fix, but if the discretization is done correctly, SRMSE should still provide valid results. However, the second flaw is more problematic. Indeed, when generating a synthetic dataset, we want to ensure that all the correlations are correctly generated. Therefore, it is required to update the methodology of SRMSE to do systematic testing on all the variables and their possible combinations.

ML assessments

In ML, many datasets are considered classification datasets. Thus, they are used with a ML model to predict future instances of a unique variable. Therefore, researchers developing generators to improve the efficacy of such models only test the synthetic datasets using the predictive power of ML models on a single variable. While this technique works well in this specific case, it does not provide enough information if one is trying to assess the representativity of a synthetic dataset compared to an original one. Indeed, there might be missing correlations between the other variables in the dataset that the ML models will overlook. It is, thus, possible to update this technique such that a ML model is used to predict each variable in the dataset instead of a single one. If there are issues with correlations between the variables, the efficacy of the ML models will drop while predicting the other variables, thus providing more information.

3.2.5 Opportunities and limitations

The role of the generator in a GAN is to produce batches of synthetic data, taking only noise as an input. As such, the structure of the generator network should be closely matched to the underlying structure of the data being replicated. For instance, in images, each variable represents a pixel whose meaning is image-specific and only defined relative to other pixels in the image. In other words, the meaning of a pixel in an image is dependent on its relative position and value, not its absolute position and value. In addition, the meaning of a single pixel in one image is (largely) independent of the meaning of the corresponding pixel in another image in the same dataset. Therefore, it is typical for generators used in image generation to use Convolutional Neural Networks (CNNs) (Radford et al., 2016; Isola et al., 2017; Zhu et al., 2017), which model the relative definitions of the pixel values learned over thousands or millions of images in a dataset.

Unlike images, the variables in tabular data typically have a specific meaning and can be understood by their absolute positions and values (within a single dataset). For example, a column representing an individual's age in socio-economic data defines the age of every instance (row) in the table. Furthermore, the age value of each row can be understood without needing to know the values of the other variables. While the variables in a dataset have a fixed position-specific meaning, their values depend on the other variables in the dataset. As such, the generator must capture the interdependencies between these variables.

There are several different approaches to this in the literature. The first approach mimics what is done with images. Indeed, several models are built using Fully Connected Neural Networks (FCNNs) (Xu et al., 2019) or CNNs (Park et al., 2018; Zhao et al., 2021). The generator has to learn the structure from the data during the learning process using backpropagation. This can be rather cumbersome for the generator since it never has access to the original data. Therefore, another approach is to fix the structure of the data. For example, TGAN (Xu and Veeramachaneni, 2018) uses a sequential order based on columns' order. The structure is implicitly learned using attention vectors used with each dataset's variable.

In both approaches, the generator learns the relationships between the variables from the available data via backpropagation of the discriminator loss. However, there are two primary limitations of this approach. Firstly, the generator, which needs to be highly flexible, can overfit the noise in the data and generalize to relationships between the columns which do not actually exist in unseen data. Secondly, the generator has to use the limited signal in the data to learn the core structure of the data, which is often already known to some degree by the modeler. Both can cause issues when the signal-to-noise ratio is high, as is often the case with socio-economic datasets of limited size.

Therefore, there is an opportunity to develop techniques that address these flaws and use the learning power of GANs. Indeed, by defining the relationships between the variables beforehand using expert knowledge, we can force the generator only to learn specific correlations. In addition, if the researcher or modeler provides these relationships, the model starts its learning process with more information than a fully connected network that has to learn all these connections. Therefore, we can overcome the issues with GANs while keeping their strengths.

3.3 Methodology

Following several previous literature works, our synthetic tabular data generation approach uses a GAN (Goodfellow et al., 2014) to generate synthetic data. Our primary contribution is to closely match the generator structure to the underlying structure of the data through a DAG specified by the modeler. According to their prior expert knowledge of the data structure, the DAG allows the modeler to define the structure of the correlations between variables in the dataset as a series of directed links between nodes in a graph. Each link in the DAG represents a causal link that the generator can capture. If no links (either direct or indirect) exist between two variables, then the generator treats those variables independently. This has two primary advantages over the existing approaches within the context of the limitations identified in the literature review. Firstly, by restricting the set of permissible links between the variables in the datasets, the DAG represents an expert regularisation of the model and restricts the ability of the GAN to overfit noise in the training sample. Secondly, giving the generator a headstart in knowing the underlying structure of the data allows the GAN to make more efficient use of the training sample when learning to generate data. These benefits could enable DATGAN to use limited available original data more efficiently when learning to produce realistic and representative synthetic data samples.

Figure 3.2 provides a high-level overview of DATGAN data generation process. As is typical with GANs, the generator in DATGAN (described in Section 3.3.1) never sees the original data. It generates data purely from a random noise input. The generator's structure is determined according to a DAG which specifies the structural relationships between the variables in the data, *i.e.* the expert knowledge. The DAG and the process linking it to the generator structure are presented in Section 3.3.1. At the same time, the discriminator (described in Section 3.3.2) is trained to classify/critique the original and synthetic data. Therefore, it can be considered a competitive game between two adversaries (the generator and the discriminator). The loss functions used to optimize both models are presented in Section 3.3.3. Since tabular data can contain attributes of different types (*e.g.* continuous, nominal, and ordinal), original and synthetic data must be processed before being used. We, thus, introduce several new data processing steps specific to DATGAN in Section 3.3.4. At the end of this section, we present the result assessment methods used to compare the synthetic datasets in Section 3.3.5. We conclude the methodology by providing some implementation notes in Section 3.3.6. Table B.1, in the appendix, provides a summary of the notations used in this methodology.

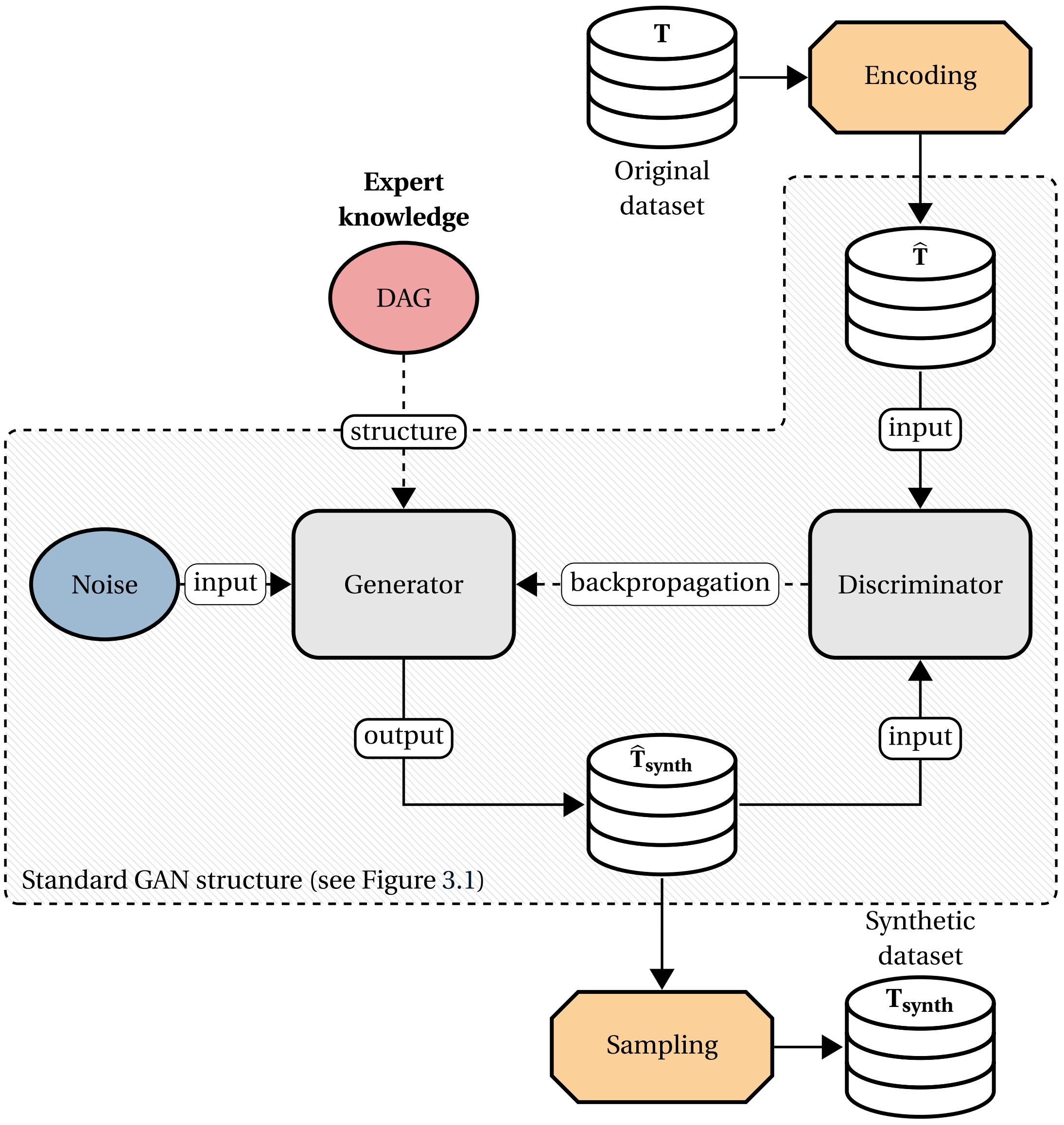


Figure 3.2: Global schematic representation of DATGAN. The different element in this figure are presented in the following sections: the Generator and the DAG are presented in Section 3.3.1, the Discriminator in Section 3.3.2, the Encoding and the Sampling processes in Section 3.3.4.

Formally, we consider a table $\mathbf{T}$ containing N_V columns. Each column in the table $\mathbf{T}$ is represented by $\boldsymbol{v}_t$ for $t = 1,\dots,N_V$. We, thus, have $\mathbf{T} = \left\{\boldsymbol{v}_{1:N_V}\right\}$. These columns have been drawn from an unknown joint distribution of random variables V_t, *i.e.* the values in $\mathbf{T}$ are drawn from $\mathbb{P}\left(V_{1:N_V}\right)$. We, thus, usally refer to the variables in $\mathbf{T}$ using V_t. We represent the rows of $\mathbf{T}$ by $\left\{v_{1:N_V,i}\right\}$ for $i = 1,\dots,N_{\text{rows}}$ where N_{rows} corresponds to the number of rows in $\mathbf{T}$. We assume that each row of $\mathbf{T}$ is sampled independently, *i.e.* it is cross-sectional and does not contain panel or sequential data. Our goal is to learn a generative model $G(\boldsymbol{z})$, where $\boldsymbol{z}$ is a tensor of random noise, such that the samples generated from G create a synthetic table

$\mathbf{T_{synth}}$. For neural networks, we work in standardized space consisting of values between -1 and 1. We, thus, denote processed datasets with the character $\widehat{\cdot}$, as shown in Figure 3.2. In the meantime, the discriminator D is trained to differentiate between original data $\widehat{\mathbf{T}}$ and synthetic data $\widehat{\mathbf{T}}_{\mathbf{synth}}$.

3.3.1 Generator

The role of the generator is to produce batches of synthetic data, taking only noise as an input. Within DATGAN, the generator structure is defined using a DAG, which specifies the interdependencies between each variable V_t in the original dataset $\mathbf{T}$. We first present the DAG, including how the modeler should construct it. We then demonstrate how the DAG is used to automatically create the generator network through the use of LSTM cells (Gers et al., 2000). This includes defining a new multi-input LSTM cell required to capture complex correlations specified in the DAG.

Directed Acyclic Graph (DAG)

The DAG $\mathscr{G}$ is specified by the modeler to define the correlations between the variables in the data. However, a DAG must represent causal links between variables similarly to Bayesian networks. Indeed, correlations do not have a direction, while causal links do. The main reason to use a directed graph instead of an undirected one is due to the nature of the representation of the variables in the generator. Each variable $\boldsymbol{v}_t$ in $\mathbf{T}$ is represented by a single LSTM cell. These cells communicate with each other in a directed manner, *i.e.* the previous cell sends information to the next one. Therefore, to better reflect this behavior, a directed graph is required.

The mathematical definition of a DAG is given by:

- The graph $\mathscr{G}$ must be directed, *i.e.* each edge in the graph has only direction.
- The graph $\mathscr{G}$ must not contain any cycle, *i.e.* the starting vertex of any given path cannot be the same as the ending vertex.

These two properties ensure that the DAG is a topological sorting. It means that we can extract a linear ordering of the vertices such that for every directed edge $\boldsymbol{v}_{t_1} \to \boldsymbol{v}_{t_2}$ from vertex $\boldsymbol{v}_{t_1}$ to vertex $\boldsymbol{v}_{t_2}$, $\boldsymbol{v}_{t_1}$ comes before $\boldsymbol{v}_{t_2}$ in the ordering.

With these rules in mind, the modeler can define the DAG for DATGAN. It is possible to get inspiration from Bayesian networks and how their respective DAG is created manually (Lucas et al., 2004). However, there are some slight differences between the two DAGs. Therefore, we provide some insights into the components of our DAG:

- Each variable $\boldsymbol{v}_t$ in the table $\mathbf{T}$ must be associated with a node in the graph $\mathscr{G}$.

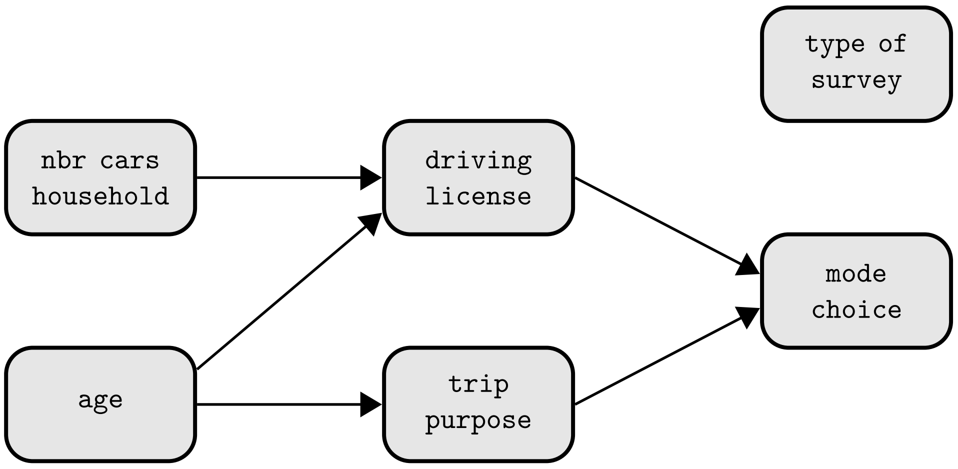
- A directed edge between two vertices, *i.e.* $\boldsymbol{v}_{t_1} \rightarrow \boldsymbol{v}_{t_2}$, means that the generation of the first variable $\boldsymbol{v}_{t_1}$ will influence the generation of the second variable $\boldsymbol{v}_{t_2}$. The direction of the edge is a matter of judgement and should not influence the final result.

- The absence of a link between two variables means that their correlation is not *directly* learned by the generator. However, it is possible to obtain some correlation in the final synthetic dataset if these two variables have a common ancestor in the graph $\mathscr{G}$. Therefore, two variables will not show any correlations in the synthetic dataset if they do not have any common ancestors or links in the DAG.

- The graph $\mathscr{G}$ can be composed of multiple DAGs as long as the first rule is respected. By creating multiple DAGs, the modeler ensures that the different parts of the dataset are not correlated. While this approach is not common, it could be used in a dataset containing variables about multiple unrelated topics.

age	driving license	trip purpose	type of survey	nbr cars household	mode choice
continuous 0-100	**boolean** True False	**nominal** Work Leisure ...	**nominal** Internet Phone ...	**ordinal** 0 1 2 ...	**nominal** Driving Soft Modes ...

(a) Example of a mock dataset



(b) Example of a DAG used to represent the structure of the variables

Figure 3.3: Example of tabular data structure. Figure 3.3a shows the structure of a table with six variables. Figure 3.3b shows one possible DAG used to represent the variables in Figure 3.3a.

As for Bayesian networks, there is no unique way to create a DAG for a given dataset. We present different possibilities using the example shown in Figure 3.3. The first one consists of following the variables' order in the datasets. This creates a simple ordered list of the variables.

For example, TGAN (Xu and Veeramachaneni, 2018) uses this specific list to link each of its LSTM cells. The advantage of such a DAG is its simplicity since no prior knowledge of the data is required. However, this DAG defines causal links based on an arbitrary order. Thus, it does not use expert knowledge and results in poorer results. Another possibility is to create a DAG centered around predicting a given variable. For example, in the case of Table 3.3a, one could want to predict the variable `mode choice`. Therefore, a possible structure for the DAG is to link all the other nodes in the table to a single sink node representing the variable we want to predict. However, while this DAG would capture all possible correlations between each variable and the one that needs to be predicted, it will not capture other correlations. Therefore, a dataset generated using this DAG would fail basic correlation tests.

While the two possibilities presented above allow creating a DAG without prior knowledge of the data, they will fail to deliver a synthetic dataset that correctly models the correlations between the variables in a table **T**. We recommend building a DAG containing as many links as possible. It is always possible to perform a transitive reduction of $\mathscr{G}$, *i.e.* removing paths such that for all vertices $\boldsymbol{v}_{t_1}$ and $\boldsymbol{v}_{t_2}$ there exists only a unique path that goes from $\boldsymbol{v}_{t_1}$ to $\boldsymbol{v}_{t_2}$, after its definition. There are no strict rules on whether one should add a causal link between two variables. It is a matter of judgment, and multiple trials and errors will be needed. However, we provide a set of instructions that can help the modeler define such a DAG:

- If the dataset is used to predict one variable, define this variable as a sink node in $\mathscr{G}$. For example, in Figure 3.3, the dataset can be used to predict the variable `mode choice`. It is, thus, the sink node of the graph. It is also possible to define multiple sink nodes.
- Datasets contain different categories of variables. For example, a travel survey dataset might contain trips, individuals, and household variables. It is, thus, generally easier to define the causal links between variables belonging to similar semantic groups.
- The next step consists in defining the source nodes. However, there are no specific rules for this. It is entirely up to the modeler.
- Finally, the modeler has to choose the direction of the causal links. Again, there are no rules for this. In the example of Figure 3.3b, one could decide that the variables `driving license` and `age` have an inverted causal link. This would slightly change the DAG but should not fundamentally change the results.

As shown in Figure 3.3, we present a mock dataset (see Figure 3.3a) and one possible DAG (see Figure 3.3b) representing the causal links between the variables. This dataset is a travel survey dataset. We thus define the mode choice as the sink node. We can define the following category of variables: (i) trip-related variables: `mode choice` and `trip purpose`; (ii) individual-related variables: `age` and `driving license`; (iii) household-related variables: `nbr cars household`; (iv) survey-related variables: `type of survey`. For each of these categories, we want to make sure that the variables are linked together. Since `mode choice` is the sink node, we can create an edge from `trip purpose` to `mode choice`. For the individual-related variables, the direction of the causal link can be either direction. For the source nodes, we set the variables `age` and `nbr cars household` as the source nodes. Finally, we can add some more links to complete the DAG. We decided, on purpose, to let the variable `type of survey` out of the DAG not to influence the data generation. One could argue that it could be linked to age since older individuals are less familiar with internet technologies. However, as stated earlier, the modeler has to make choices while constructing the DAG, requiring trials and errors.

Once the DAG $\mathscr{G}$ has been created, we can define several valuable sets. These sets are used later when representing the structure of the DAG in the generator.

- $\mathscr{A}(V_t)$: the set of ancestors of the variable V_t.
- $\mathscr{P}(V_t)$: the set of predecessors of the variable V_t.
- $\mathscr{S}(V_t)$: the set of sources nodes leading to the variable V_t.
- $\mathscr{E}(V_t)$: the set of in-edges of the variable V_t.

If we use the variable `mode choice` from Figure 3.3b as an example, we can define these different sets:

- $\mathscr{A}(\texttt{mode choice}) = \{\texttt{nbr cars households; age; driving license; trip purpose}\}$
- $\mathscr{P}(\texttt{mode choice}) = \{\texttt{driving license; trip purpose}\}$
- $\mathscr{S}(\texttt{mode choice}) = \{\texttt{nbr cars households; age}\}$
- $\mathscr{E}(\texttt{mode choice}) = \{\texttt{driving license} \rightarrow \texttt{mode choice; trip purpose} \rightarrow \texttt{mode choice}\}$

Representation of the DAG

As explained in this section's introduction, the DAG represents the causal links between the variables. Thus, we want to develop an architecture for the generator similar to the specified DAG. Tabular data cannot be considered sequential since the order of the variables in a dataset is random. However, the DAG allows us to have a sequence of variables with a specific order. Thus, we can use Neural Networks models that work well with this type of data. More specifically, we use LSTM cells (Hochreiter and Schmidhuber, 1997), a type of recurrent neural network, to generate synthetic values for each variable V_t. We denote the LSTM cell associated to the variable V_t by $\mathbf{LSTM_t}$. The advantage of using recurrent neural networks is that the previous output affects the current state of the neural network. Using the sequence defined by the DAG $\mathscr{G}$, we can, thus, use previous outputs, *i.e.* synthetic values of previous variables, to influence the generation process of a given variable V_t.

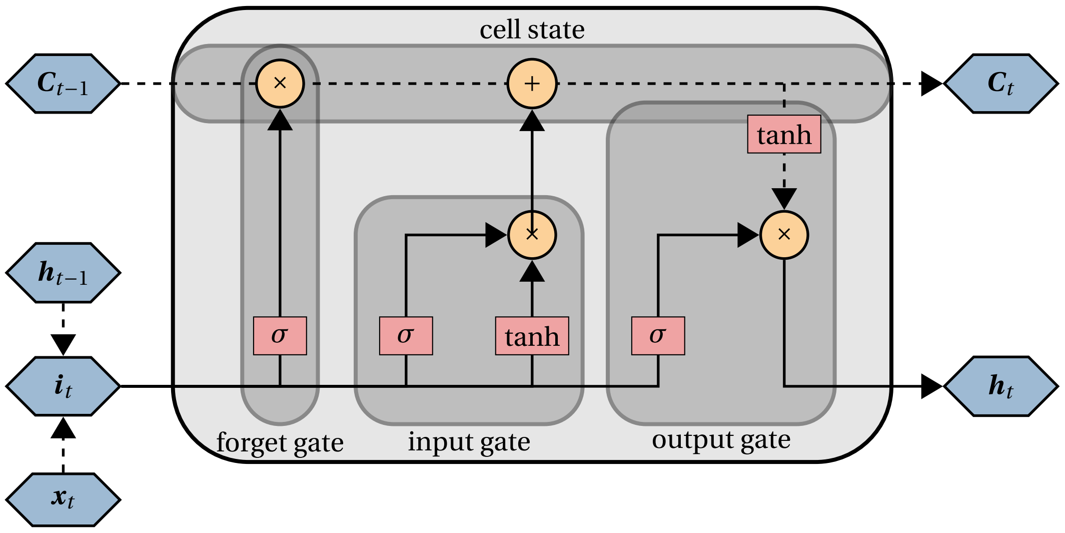


Figure 3.4: Main components of a LSTM cell following Gers et al. (2000). The blue hexagons represent variables, the red rectangles neural network layers, and the orange circles mathematical operations. The different gates used to transform the input and the cell state are shown in dark gray.

The key elements of an LSTM cell are the cell state and the multiple gates used to protect and control the cell state, as shown in Figure 3.4. For conciseness, we do not present a detailed overview of the mathematical operations of an LSTM cell. For a full description, we direct the reader to Gers et al. (2000). In Figure 3.4, the input cell state is characterized by $\boldsymbol{C}_{t-1}$ and the output cell state by $\boldsymbol{C}_t$. The cell will receive an input that is the concatenation between the output of the previous cell ($\boldsymbol{h}_{t-1}$) and an input vector ($\boldsymbol{x}_t$). It is thus given by:

$$\boldsymbol{i}_t = \boldsymbol{h}_{t-1} \oplus \boldsymbol{x}_t \tag{3.1}$$

This input vector will pass through three different gates to transform the cell state as it is necessary:

1. **the forget gate** is used to decide which old information is forgotten in the cell state.
2. **the input gate** is used to decide which new information is stored kept in the new cell state
3. **the output gate** is used to decide the output of the cell using information from both the input $\boldsymbol{i}_t$ and the new cell state C_t.

The modeler has to define the size of the hidden layers N_h in the LSTM cell and the batch size N_b. The first defines the size of the output vector $\boldsymbol{h}_t$ as well as the cell state $\boldsymbol{C}_t$. The latter corresponds to the number of data points fed into the network. Therefore, we define the output $\boldsymbol{h}_t$ and cell state $\boldsymbol{C}_t$ as tensors of size $N_h \times N_b$. The input x_t takes a different size and is thus characterized by a tensor of size $N_x \times N_b$ (the batch size has to remain the same between all the tensors).

In the case of DATGAN, we have to modify the inputs and outputs of the LSTM cell according to the principles of GANs. Figure 3.5 provides a schema of the LSTM structure in DATGAN. The insides of the LSTM cell $\mathbf{LSTM_t}$ are the same as the one shown in Figure 3.4. The first main modification concerns the inputs. Indeed, the generator in a GAN takes random noise as an input instead of an input vector such as $\boldsymbol{x}_t$. In addition, we add an attention vector to the input tensor $\boldsymbol{i}_t$. The idea behind the attention vector is to keep information from intermediate encoders and pass it to a new encoder. This mimics cognitive attention and, thus, helps with the long-term memory of the LSTM cells. Therefore, the input tensor $\boldsymbol{i}_t$ corresponds to the concatenation of three tensors:

- $\boldsymbol{z}_t$ is a tensor of Gaussian noise with dimension $N_z \times N_b$. For each source node in the DAG $\mathscr{G}$, we randomly draw values from $\mathcal{N}(0,1)$. For all the other variables V_t, the noise vector is a concatenation of the noise from the source nodes passed through a fully connected layer without any activation function, *i.e.*

$$\boldsymbol{z}_t = \texttt{FC}\left(\bigoplus_{k\in\mathscr{S}(V_t)} \boldsymbol{z}_k, N_z\right) \tag{3.2}$$

  If two different variables V_{t_1} and V_{t_2} have the same source nodes, *i.e.* $\mathscr{S}(V_{t_1}) = \mathscr{S}(V_{t_2})$, the noise tensor is the same for both variables, *i.e.* $\boldsymbol{z}_{t_1} = \boldsymbol{z}_{t_2}$. There are two reasons to apply such a rule: (i) it removes pointless computation by creating new variables; (ii) if two variables have a unique source node, they will receive the same noise as an input. We must, therefore, follow this rule if there is more than one source node.

- $\boldsymbol{f}_{t-1}$ can be compared to the previous output tensor $\boldsymbol{h}_{t-1}$ in Figure 3.4. However, one of the differences between DATGAN and a usual LSTM network is that we do not directly use the output tensor of the LSTM $\boldsymbol{h}_t$. Indeed, as shown in Figure 3.5, we transform it into the encoded synthetic variable $\widehat{\boldsymbol{v}}_t^{\mathbf{synth}}$. Since $\widehat{\boldsymbol{v}}_t^{\mathbf{synth}}$ do not have a standard size, we thus need to transform it in order to obtain the usable tensor $\boldsymbol{f}_t$ of dimension $N_h \times N_b$. We can thus say that $\boldsymbol{f}_t$ corresponds to the transformed output of the LSTM cell $\mathbf{LSTM_t}$. If the variable V_t is a source node, this tensor is randomly initialized and learned during the optimization process.

- $\boldsymbol{a}_t$ is an attention tensor that allows the cell to learn which previous outputs are relevant to the input. It is defined as a weighted average over all the LSTM outputs. If the current variable V_t has at least one ancestor, we learn an attention weight vector $\alpha_t \in \mathbb{R}^{|\mathscr{A}(t)|}$. The context vector is thus computed as:

$$\boldsymbol{a}_t = \sum_{k\in\mathscr{A}(t)\setminus\mathscr{P}(t)} \frac{\exp \alpha_t^{(k)}}{\sum_j \exp \alpha_t^{(j)}} \boldsymbol{f}_k \tag{3.3}$$

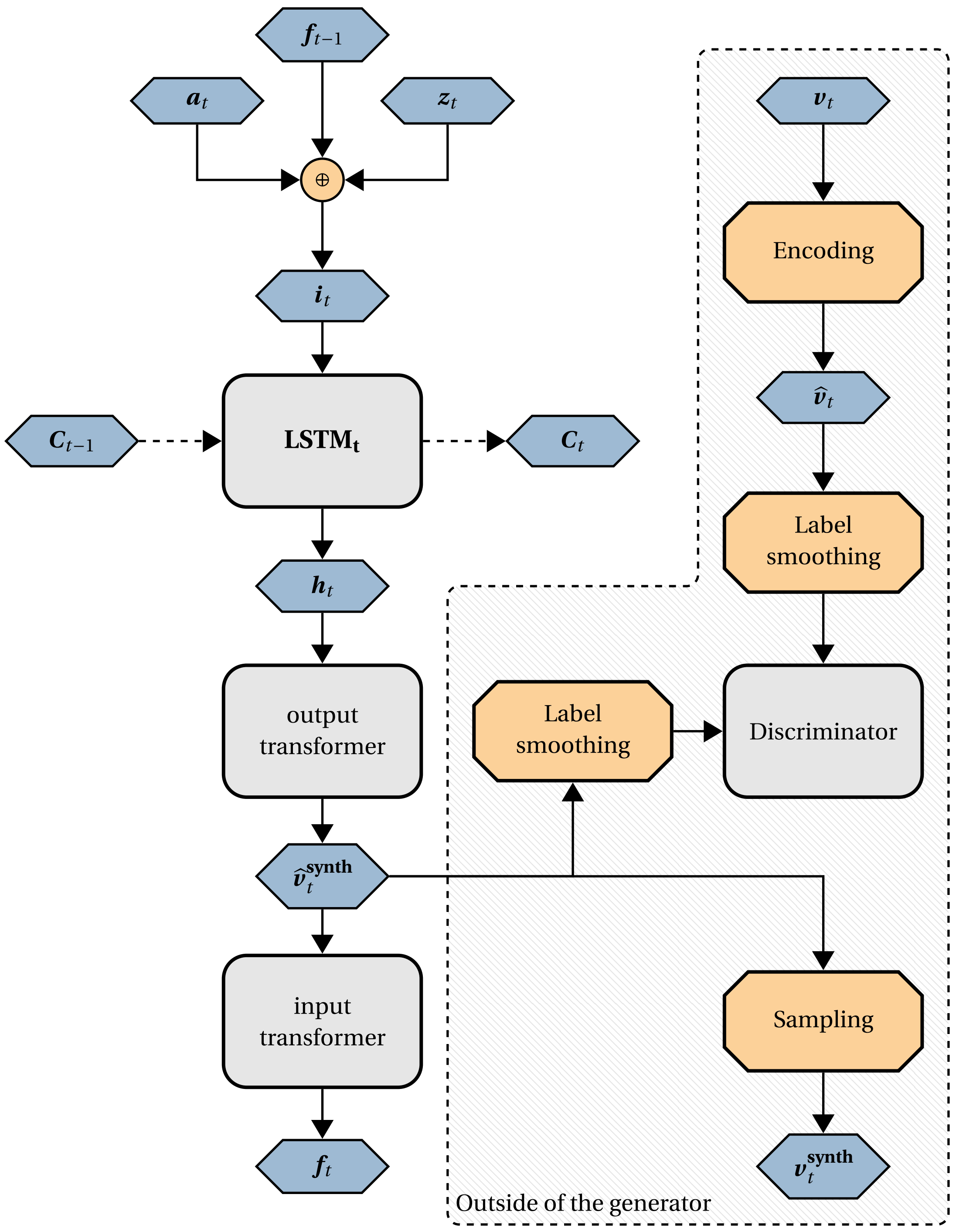


Figure 3.5: Schematic representation of how the LSTM cells are used within DATGAN to generate the synthetic variable $\hat{v}_t^{\text{synth}}$. In Section 3.3.1, we mainly discuss the left part of the diagram. The right part shows how the new generated variable is passed to the discriminator and sampled. These different elements are presented in the following sections: the Discriminator is presented in Section 3.3.2 and the Encoding, the Label smoothing, and the Sampling in Section 3.3.4.

This context tensor has a dimension $N_h \times N_b$. If variable V_t is a source node, we define $\boldsymbol{a}_t$ as a zero-vector of dimension $N_h \times N_b$.

LSTM cells are initially designed to work in sequence, *i.e.* each cell is linked to a unique following cell. However, as shown in the DAG in Figure 3.3b, some variables can have multiple direct ancestors. For example, the variable `mode choice` has two ancestors: `driving license` and `trip purpose`. We, thus, need a way to connect multiple LSTM cells. If one cell has multiple outputs, we send the output of the cell to the next cells, *e.g.* the inputs of the cells for the variables `driving license` and `trip purpose` coming from the variable `age` are the same. The main issue lies in having multiple cell inputs. The attention tensor $\boldsymbol{a}_t$ and the noise tensor $\boldsymbol{z}_t$ are defined based on all the ancestors of the current variable. Therefore, we do not have to change these definitions. On the other, the previous cell state $\boldsymbol{C}_{t-1}$ and the transformed output $\boldsymbol{f}_{t-1}$ are defined to work in sequence. Therefore, we define the multi-input cell state and multi-input transformed output by concatenating the cell states and transformed outputs from the direct ancestors and passing them through a fully connected layer to resize them:

$$\boldsymbol{C}_{t-1} = \texttt{FC}\left(\bigoplus_{k\in\mathscr{P}(V_t)} \boldsymbol{C}_k, N_h\right) \tag{3.4}$$

$$\boldsymbol{f}_{t-1} = \texttt{FC}\left(\bigoplus_{k\in\mathscr{P}(V_t)} \boldsymbol{f}_k, N_h\right) \tag{3.5}$$

During the training process, the two layers' weights must be learned. We can, thus, feed the LSTM cell with homogeneous inputs.

One final issue remains in constructing the structure of the generator. Indeed, we know how to generate each variable separately and connect them using the DAG and the multi-input LSTM cells. However, while building the generator's structure, we cannot start with any random variable in the DAG. Indeed, as per the definition of the inputs of the LSTM cell, the ancestors $\mathscr{A}(V_t)$ must have already been built first. For example, the attention tensor uses the outputs of all the ancestors' cells. Therefore, we need an algorithm that creates an ordered list based on the provided DAG $\mathscr{G}$. This algorithm is given in Algorithm 3.1. The goal of this algorithm is to take the DAG $\mathscr{G}$ and return an ordered list of the variables V_t such that all the ancestors of a given variable have a smaller index in the list, *i.e.* they appear first in the list. The algorithm is built recursively. The idea is to define two lists, one with untreated nodes `untreated`, containing all the nodes at the beginning of the algorithm, and one with treated nodes `treated`, empty at the beginning. We, then, start by selecting all source nodes and adding them to a list named `to_treat`. Then, while the list of untreated nodes is not empty, *i.e.* while there are still nodes to be added to the final list, we start by assigning all nodes in the list `to_treat` in the `treated` list. Then, we check each edge in the DAG $\mathscr{G}$ and check if all the ancestors of the out-vertex have been treated. If it is the case, we can add this node to the `to_treat` list and assign it later to the final list. The algorithm stops when all the nodes have been added to the `treated` list.

Algorithm 3.1 Ordering of the variables using a DAG

```
Inputs: DAG: $\mathscr{G}$
Output: ordered list of variables: treated
 1: Compute a dictionary in_edges with $V_t$ as the key and $\mathscr{E}(V_t)$ as the value for all $t = 1, \dots, N_V$.
 2: Initialize untreated as a set with all the variables names and treated an empty list
 3: Initialize to_treat as a list containing all the variables with 0 in-edges
 4: while |untreated| > 0 do
 5:     for all $n \in$ to_treat do
 6:         Remove $n$ from untreated and add it to treated
 7:     Set to_treat as an empty list
 8:     for all $e \in \mathscr{G}.E$ do    ▷ $e$ is an edge and it is a tuple with 2 values: the out-vertex and the in-vertex
 9:         Initialize boolean all_ancestors_treated to True
10:         for all $\ell \in$ in_edges[$e$[1]] do
11:             if $\ell \notin$ treated then
12:                 Set all_ancestors_treated to False
13:         if $e$[0] $\in$ treated and all_ancestors_treated is True and $e$[1] $\notin$ treated and $e$[1] $\notin$ to_treat then
14:             Add $e$[1] to the list to_treat
15: return treated
```

Now that every component has been defined for the generator, we can build it following the ordered list provided by Algorithm 3.1. Each time that we create a LSTM cell for the variable V_t, as in Figure 3.5, we check the predecessors $\mathscr{P}(V_t)$. We apply the multi-input technique to the LSTM cell if there is more than one direct ancestor. The generator is finished once one LSTM cell has been created for each variable in the ordered list.

3.3.2 Discriminator

As seen in Figure 3.2, the generator is used to create the synthetic dataset $\widehat{\mathbf{T}}_{\mathbf{synth}}$. The role of the discriminator is to compare this dataset with the encoded original dataset $\widehat{\mathbf{T}}$. We, thus, want to train the discriminator to be able to identify original and synthetic data. The generator must, then, produce better synthetic data to fool the discriminator.

Following Xu and Veeramachaneni (2018), we use a fully connected neural network with N_L-layers for the discriminator, where the internal layers, for $i = 1, \dots, N_L$, are given by:

$$\widehat{\boldsymbol{l}}_i = \texttt{FC}\left(\boldsymbol{l}_{i-1}, N_l\right) \tag{3.6}$$

$$\boldsymbol{l}_i = \texttt{LeakyReLU}\left(\texttt{BN}\left(\widehat{\boldsymbol{l}}_i \oplus \texttt{div}\left(\widehat{\boldsymbol{l}}_i\right)\right)\right) \tag{3.7}$$

where (i) $\texttt{div}(\cdot)$ represents the mini-batch discrimination vector presented by Salimans et al. (2016); (ii) $\texttt{BN}(\cdot)$ corresponds to the batch normalization; (iii) $\texttt{LeakyReLU}(\cdot)$ is the leaky reflect linear activation function. The output of the discriminator is computed using a fully connected layer with a size of 1 to return an unbounded scalar:

$$l_D = \texttt{FC}\left(\boldsymbol{l}_{N_L}, 1\right) \tag{3.8}$$

The input vector $\boldsymbol{l}_0$ of the discriminator is different depending on the data it is using:

- For the original dataset, $\boldsymbol{l}_0$corresponds to the concatenation of all the column vectors $\left\{\widehat{\boldsymbol{v}}_{1:N_V}\right\}$ after an encoding step, as shown in Figure 3.2.
- For the synthetic dataset, $\boldsymbol{l}_0$ corresponds to the concatenation of all the usable outputs $\left\{\widehat{\boldsymbol{v}}^{\mathbf{synth}}_{1:N_V}\right\}$ given by the generator, as shown in Figure 3.5.

However, as seen in Figure 3.5, a label smoothing step has to be performed before feeding these matrices to the discriminator. This step is discussed in Section 3.3.4.

3.3.3 Loss function

The loss function sets up the game between the discriminator and the generator. The discriminator D aims to maximize the loss function when comparing the synthetic data produced by the generator against the original data. Meanwhile, the generator G aims to minimize the same loss function by generating synthetic data, fooling the discriminator. The generator thus learns from the discriminator by backpropagating the discriminator loss.

Since the loss function drives the optimization process to obtain the best possible model, we argue that our model does not have to be characterized by a single loss function. Therefore, we systematically test three different loss functions. The first one is the standard cross-entropy loss defined by Goodfellow et al. (2014), the second one is the Wasserstein or Earth-Mover distance defined by Arjovsky et al. (2017), and the third one is the Wasserstein distance with Gradient-Penalty defined by Gulrajani et al. (2017).

Standard loss: The first loss function is the standard loss function used in the original GAN by Goodfellow et al. (2014). We name it $\mathscr{L}^{\texttt{SGAN}}$ with SGAN standing for Standard GAN. It is given by:

$$\min_G \max_D \mathscr{L}^{\texttt{SGAN}}(D,G) = \mathop{\mathbb{E}}_{\left\{\widehat{\boldsymbol{v}}_{1:N_V}\right\}\sim\mathbb{P}(\widehat{\mathbf{T}})} \log D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) + \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{N}(0,1)} \log\left(1 - D(G(\boldsymbol{z}))\right) \tag{3.9}$$

This loss function requires the discriminator to produce a probability for each data point to be either original or synthetic. However, as defined in Section 3.3.2, the discriminator outputs an unbounded scalar, not a probability. In order to use this discriminator with this loss function, we thus pass the output through an additional sigmoid layer to produce bounded $[0,1]$ probabilities.

As explained by Goodfellow et al. (2014), $\log(1 - D(G(\boldsymbol{z})))$ tends to saturate during the training process. Therefore, instead of training G to minimize the full loss function, we can instead train it to maximize $\log D(G(\boldsymbol{z}))$. We can thus define the loss function for both networks separately. The goal is to minimize both losses simultaneously during the training process. They are given by:

$$\mathscr{L}_G^{\texttt{SGAN}} = -\underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} \log D(G(\boldsymbol{z})) \tag{3.10}$$

$$\mathscr{L}_D^{\texttt{SGAN}} = -\underset{\{\widehat{\boldsymbol{v}}_{1:N_V}\}\sim\mathbb{P}(\widehat{\mathbf{T}})}{\mathbb{E}} \log D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) + \underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} \log D(G(\boldsymbol{z})) \tag{3.11}$$

As suggested by the authors, we train our models using the Adam optimizer (Kingma and Ba, 2014).

Wasserstein loss: The second loss function has been implemented in WGAN by Arjovsky et al. (2017). It is defined using the Earth-Mover distance:

$$\min_G \max_D \mathscr{L}^{\texttt{WGAN}}(D,G) = \underset{\{\widehat{\boldsymbol{v}}_{1:N_V}\}\sim\mathbb{P}(\widehat{\mathbf{T}})}{\mathbb{E}} D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) - \underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} D(G(\boldsymbol{z})) \tag{3.12}$$

There are multiple advantages to use this loss function instead of the standard loss: (i) the main advantage is the fact that the discriminator becomes a critic since it does not need to produce a 0-1 output anymore. Indeed, we can use the output of the discriminator l_D as it is defined. It thus results in less vanishing gradients and an easier learning process for the generator G; (ii) the loss function correlates with the quality of the sample, contrary to the SGAN loss. It is, thus, possible to determine when the GAN has converged.

We can, now, define the separate loss functions for both networks as:

$$\mathscr{L}_G^{\texttt{WGAN}} = -\underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} D(G(\boldsymbol{z})) \tag{3.13}$$

$$\mathscr{L}_D^{\texttt{WGAN}} = -\underset{\{\widehat{\boldsymbol{v}}_{1:N_V}\}\sim\mathbb{P}(\widehat{\mathbf{T}})}{\mathbb{E}} D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) + \underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} D(G(\boldsymbol{z})) \tag{3.14}$$

As suggested by the authors, we train our models using RMSProp (Tieleman and Hinton, 2012).

Wasserstein loss with gradient penalty: The loss for the WGAN-GP is the same as the Wasserstein loss with the addition of a gradient penalty (Gulrajani et al., 2017). It is given by:

$$\min_G \max_D \mathscr{L}^{\texttt{WGGP}}(D,G) = \underset{\{\widehat{\boldsymbol{v}}_{1:N_V}\}\sim\mathbb{P}(\widehat{\mathbf{T}})}{\mathbb{E}} D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) - \underset{\boldsymbol{z}\sim\mathcal{N}(0,1)}{\mathbb{E}} D(G(\boldsymbol{z})) + \lambda \underset{\widetilde{\boldsymbol{v}}\sim\mathbb{P}(\widehat{\mathbf{T}},G(\boldsymbol{z}))}{\mathbb{E}} \left(\|\nabla_{\widetilde{\boldsymbol{v}}} D(\widetilde{\boldsymbol{v}})\|_2 - 1\right)^2 \tag{3.15}$$

where λ is a parameter defined by the modeler. The main issue with WGAN is that it needs to enforce the Lipschitz constraint on the critic. It does that by clipping the weights of the critic. Gulrajani et al. (2017) show that it leads undesired behaviour in the generator samples. We can, thus, fix this issue by adding a gradient penalty on the critic. In Equation 3.15, the mid-value $\widetilde{\boldsymbol{v}}$ is sampled uniformely along straight lines between pair of points sampled from the original dataset **T** ($\widehat{v}$) and generated data $G(\boldsymbol{z})$ ($\widehat{v}^{\textbf{synth}}$).

The separate loss functions for each networks are, therefore, defined as:

$$\mathscr{L}_G^{\texttt{WGGP}} = - \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{N}(0,1)} D(G(\boldsymbol{z})) \tag{3.16}$$

$$\mathscr{L}_D^{\texttt{WGGP}} = - \mathop{\mathbb{E}}_{\{\widehat{\boldsymbol{v}}_{1:N_V}\}\sim\mathbb{P}(\widehat{\mathbf{T}})} D\left(\widehat{\boldsymbol{v}}_{1:N_V}\right) + \mathop{\mathbb{E}}_{\boldsymbol{z}\sim\mathcal{N}(0,1)} D(G(\boldsymbol{z})) + \lambda \mathop{\mathbb{E}}_{\widehat{\boldsymbol{v}}\sim\mathbb{P}(\widehat{\mathbf{T}},G(\boldsymbol{z}))} \left(\|\nabla_{\widehat{\boldsymbol{v}}} D(\widehat{\boldsymbol{v}})\|_2 - 1\right)^2 \tag{3.17}$$

Finally, following Gulrajani et al. (2017), we replace the batch normalization in the discriminator with a layer normalization (Ba et al., 2016), we set $\lambda = 10$, and we train both models using the Adam optimizer (Kingma and Ba, 2014).

Following Xu and Veeramachaneni (2018), we include a Kullback-Leibler (KL) divergence term to all the generator losses. For two discrete probability distributions P and Q defined on the same probability space $\mathscr{X}$, the KL divergence is given by:

$$\mathrm{KL}\,(P,Q) = \sum_{x\in\mathscr{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{3.18}$$

Therefore, we can use this divergence for any discrete probability distributions in the original and synthetic datasets. The use of this divergence has two main consequences: (i) it gives a boost when starting the training of the generator since it is trying to make discrete probability distributions as close as possible; (ii) it makes the model more stable under training. We discuss which variables are concerned by this divergence in Section 3.3.4.

3.3.4 Data processing

Tabular data are generally composed of multiple data types, as seen in Figure 3.3a. In the context of this chapter, we consider two different variable types:

Continuous data corresponds to a random variable following a continuous distribution (*e.g.* the distance to travel to a destination). The variable can then be rounded to obtain discrete values, *e.g.* individuals' age.

Categorical data corresponds to all other types of data such as:
- binary random variables, *e.g.* whether someone is retired or not.
- nominal random variables, *i.e.* discrete random variable with three or more possible values, where there is no order nor notion of distance between the values (*e.g.* a color).
- ordinal random variables, *i.e.* discrete random variables with three or more possible values with a defined order (and possibly also distance) between each possible value (*e.g.* education level). Contrary to nominal data, we can define an order (and possibly a distance) between the different categories.

In ML, neural networks typically work with data ranging from -1 to 1 or 0 to 1. However, these four data types are not designed in such a way. We thus need to encode the original dataset $\mathbf{T}$ in a dataset $\widehat{\mathbf{T}}$, as shown in Figure 3.2, that transforms the different data types into more homogeneous types.

The table $\mathbf{T}$ contains N_C continuous random variables $\{C_1, \ldots, C_{N_C}\}$ and N_D categorical random variables $\{D_1, \ldots, D_{N_D}\}$ such that $N_C + N_D = N_V$. We can thus define the table $\mathbf{T}$ using vectors of continuous and categorical variables, *i.e.* $\mathbf{T} = \{\boldsymbol{c}_{1:N_C}, \boldsymbol{d}_{1:N_D}\}$. Similarly, the synthetic dataset is defined as $\mathbf{T}_{\mathbf{synth}} = \left\{\boldsymbol{c}^{\mathbf{synth}}_{1:N_C}, \boldsymbol{d}^{\mathbf{synth}}_{1:N_D}\right\}$.

Since we are considering two different data types in the table $\mathbf{T}$, we cannot process them the same way. Thus, next section explains how the encoding is done for each type. Then, we explain how these types of data are generated. Thirdly, we discuss how the synthetic and original data are passed to the discriminator. Finally, we show how the data is sampled from the generator's output to create the final synthetic dataset.

Encoding

DATGAN only takes as input $[-1,1]$ or $[0,1]$ bounded vectors. Therefore, we need to encode unbounded continuous and categorical variables to be processed by the GAN. Continuous data tend to follow multimodal distributions. We thus build on the previous methodology of Xu et al. (2019) who apply a VGM model (Bishop, 2006) to cluster continuous values into a discrete number of Gaussian mixtures. In this work, we develop this further by automatically determining the number of components from the data.

For each continuous variable C_t in the dataset, we first train a VGM on a random subset of the data with a high number of components ($N_{m,t} = 10$). We then determine how many components are needed to capture the distribution by comparing the component weights against a threshold and the number of predicted components N_{pred} with the original number of components $N_{m,t}$. We repeat this process until convergence. Finally, we retrain the model on the entire column vector $\boldsymbol{c}_t$ using only the number of significant components. From this trained model, we extract the means $\boldsymbol{\eta}_t$ and standard deviations $\boldsymbol{\sigma}_t$ from the VGM. We can then normalize the values $c_{t,j}$ using the following formula:

$$w^{(k)}_{t,j} = \frac{c_{t,j} - \eta^{(k)}_t}{\delta\sigma^{(k)}_t} \quad \text{for } k = 1, \ldots, N_{m,t}, \tag{3.19}$$

where δ is a parameter specified by the modeller. Following Xu and Veeramachaneni (2018), we use a value of $\delta = 2$ and clip the values of $w^{(k)}_{t,j}$ between -0.99 and 0.99. At the same time, we compute the posterior probability vectors $\boldsymbol{p}_{t,j}$ that the value $c_{t,j}$ belongs to each of the $N_{m,t}$ mixtures. Thus, each value $c_{t,j}$ in $\boldsymbol{c}_t$ are represented by the vector of probabilities $\boldsymbol{p}_{t,j} \in [0,1]^{N_{m,t}}$ and the vector of values $\boldsymbol{w}_{t,j} \in [-0.99, 0.99]^{N_{m,t}}$. Algorithm 3.2 shows a summary of the procedure used to preprocess continuous variables.

Algorithm 3.2 Continuous variables preprocessing

```
Inputs: List of float values ($\boldsymbol{c}_t$)
Output: Matrix of probabilities ($\boldsymbol{p}_t$) and values ($\boldsymbol{w}_t$)
 1: Set the initial number of modes $N_{m,t} = 10$
 2: while True do
 3:     Sample $\boldsymbol{s}_t$ from $\boldsymbol{c}_t$ and fit the BayesianGaussianMixture to $\boldsymbol{s}_t$ with $N_{m,t}$ modes
 4:     Predict the class on $\boldsymbol{s}_t$ and compute $N_{\text{pred}}$ the number of unique classes predicted by the VGM
 5:     Define $N_{\text{weights}}$, the number of weights of the model above a threshold $\varepsilon_w = 0.01$
 6:     if $N_{\text{pred}} < N_{m,t}$ or $N_{\text{weights}} < N_{m,t}$ then
 7:         $N_{m,t} = \min\left(N_{\text{pred}}, N_{\text{weights}}\right)$
 8:     else
 9:         break
10: Fit the BayesianGaussianMixture to $\boldsymbol{c}_t$ with $N_{m,t}$ modes.
11: Means and standard deviations of the $N_{m,t}$ Gaussian mixtures are given by
        $$\boldsymbol{\eta}_t = \left(\eta_t^{(1)}, \ldots, \eta_t^{(N_{m,t})}\right) \text{ and } \boldsymbol{\sigma}_t = \left(\sigma_t^{(1)}, \ldots, \sigma_t^{(N_{m,i})}\right)$$
12: Compute the posterior probability of $c_{t,j}$ coming from each of the $N_{m,t}$ mixtures as a vector $\boldsymbol{p}_{t,j} = \left(p_{t,j}^{(1)}, \ldots, p_{t,j}^{(N_{m,t})}\right)$. It corresponds to a normalized probability distributions over the $n_{m,i}$ Gaussian distributions.
13: Normalize $c_{t,j}$ for each Gaussian mixture using Equation 3.19.
14: Clip each value $w_{t,j}^{(k)}$ between -0.99 and 0.99 and set $\boldsymbol{w}_{t,j} = \left(w_{t,j}^{(1)}, \ldots, w_{t,j}^{(N_{m,t})}\right)$.
15: return $\boldsymbol{p}_t$ and $\boldsymbol{w}_t$
```

For categorical variables[1], we transform them using one-hot encoding. We consider $\boldsymbol{d}_t$ the realizations of the random variable D_t. $\boldsymbol{d}_t$ is transformed using $|D_t|$-dimensional one-hot vector $\boldsymbol{o}_t$ where $|D_t|$ corresponds to the number of unique categories in D_t.

We can thus convert the initial table $\mathbf{T}$ into an intermediate table $\widehat{\mathbf{T}} = \left\{\boldsymbol{w}_1, \boldsymbol{p}_1, \ldots, \boldsymbol{w}_{N_C}, \boldsymbol{p}_{N_C}, \boldsymbol{o}_1, \ldots, \boldsymbol{o}_{N_D}\right\}$. As a simplification, we write $\widehat{\mathbf{T}} = \left\{\boldsymbol{w}_{1:N_C}, \boldsymbol{p}_{1:N_C}, \boldsymbol{o}_{1:N_D}\right\}$. The dimension of this new table is given by $\sum_{t=1}^{N_C} 2N_{m,t} + \sum_{t=1}^{N_D} |D_t|$. While this new encoded table is larger than the original table, we can now use define the KL divergence on multiple variables. Indeed, the vector $\boldsymbol{p}_{t,j}$ corresponds to a discrete probability distribution for a given row. We can, thus, apply the KL divergence on this term. In addition, the one-hot encoded vector $\boldsymbol{o}_{t,j}$ also corresponds to a discrete probability distributions. The only difference is that this distribution is composed of only 1s and 0s. Nevertheless, the KL divergence is also applicable on this variable.

Generator output

In Figure 3.5, we show how the LSTM cell $\mathbf{LSTM_t}$ produces an output $\boldsymbol{h}_t$ before it is transformed into the encoded synthetic variable $\widehat{\boldsymbol{v}}_t^{\mathbf{synth}}$. However, in Section 3.3.4, we show that

[1]The same treatment is applied to categorical and boolean variables due to the label smoothing.

we distinguish between two variable types. Thus, the output transformer in Figure 3.5 differs depending on if we are working with a continuous or a categorical variable. Nevertheless, the first step for the output transformer is similar in both cases. Indeed, the goal is to use some semblance of convolution on the LSTM output $\boldsymbol{h}_t$ to improve the results. We, thus, transform this output through a hidden layer:

$$\boldsymbol{h}'_t = \texttt{tanh}(\boldsymbol{h}_t, N_{\text{conv}}) \tag{3.20}$$

The final transformation into the synthetic encoded variable depends on the variable type. For continuous variables, we thus pass the reduced output $\boldsymbol{h}'_t$ through two different fully connected layers to extract both the vector of probabilities $\boldsymbol{p}_t^{\textbf{synth}}$ and the vector of corresponding values $\boldsymbol{w}_t^{\textbf{synth}}$:

$$\boldsymbol{w}_t^{\textbf{synth}} = \texttt{tanh}(\boldsymbol{h}'_t, N_{m,t}) \tag{3.21}$$

$$\boldsymbol{p}_t^{\textbf{synth}} = \texttt{softmax}(\boldsymbol{h}'_t, N_{m,t}) \tag{3.22}$$

For the categorical variables, we pass the reduced output $\boldsymbol{h}'_t$ through a single fully connected layer to extract the output probabilities $\boldsymbol{o}_t^{\textbf{synth}}$ belonging to each class:

$$\boldsymbol{o}_t^{\textbf{synth}} = \texttt{softmax}(\boldsymbol{h}'_t, |D_t|) \tag{3.23}$$

Both matrices of discrete probabilities $\boldsymbol{p}_t^{\textbf{synth}}$ and $\boldsymbol{o}_t^{\textbf{synth}}$ are using a softmax activation function in order to ensure that the sum along the rows is equal to one. This ensures that the rows of these matrices correspond to discrete probability vectors. The matrix $\boldsymbol{w}_t^{\textbf{synth}}$ uses a tanh activation function since we allow this matrix to take values between -1 and 1.

Since these encoded synthetic variables do not have homogeneous sizes, we cannot use them directly as the input of the next LSTM cell. This, thus, explains why we are passing the encoded synthetic values $\widehat{\boldsymbol{v}}_t^{\textbf{synth}}$ through an input transformer. The goal of this transformer is to take $\widehat{\boldsymbol{v}}_t^{\textbf{synth}}$ and transform it back to the same tensor for all the different variables V_t. Therefore, we have to distinguish between continuous and categorical variables again. For the continuous variables, we concatenate the transformed synthetic variables together and pass them through a fully connected layer to obtain $\boldsymbol{f}_t$:

$$\boldsymbol{f}_t = \texttt{FC}(\boldsymbol{w}_t^{\textbf{synth}} \oplus \boldsymbol{p}_t^{\textbf{synth}}, N_h) \tag{3.24}$$

For the categorical variables, we just pass $\boldsymbol{o}_t^{\textbf{synth}}$ through a fully connected layer:

$$\boldsymbol{f}_t = \texttt{FC}(\boldsymbol{o}_t^{\textbf{synth}}, N_h) \tag{3.25}$$

Finally, the tensors $\boldsymbol{w}_t^{\textbf{synth}}$, $\boldsymbol{p}_t^{\textbf{synth}}$, and $\boldsymbol{o}_t^{\textbf{synth}}$ are combined to form the encoded synthetic table $\widehat{\mathbf{T}}_{\textbf{synth}} = \left\{\boldsymbol{w}_{1:n_C}^{\textbf{synth}}, \boldsymbol{p}_{1:n_C}^{\textbf{synth}}, \boldsymbol{o}_{1:n_D}^{\textbf{synth}}\right\}$. This synthetic table is passed to the discriminator as an input for the optimization process. We thus directly compare it to the encoded table $\widehat{\mathbf{T}}$.

Discriminator input

As explained in Section 3.3.2, the input tensor $\boldsymbol{l}_0$ corresponds to the concatenation of all the variables in $\widehat{\mathbf{T}}$ for the original data or $\widehat{\mathbf{T}}_{\mathbf{synth}}$ for the synthetic data.

For categorical variables, where the original data are one-hot encoded, it would be trivial for a discriminator to differentiate between the original and synthetic values (as the generator produces probabilities over each class, which will not be $\{0,1\}$ vectors). In addition, as explained by Goodfellow (2017), deep networks tend to produce overconfident results when adversarially constructed. The author thus suggests using one-sided label smoothing for the standard loss function, as defined in Section 3.3.3. It means that we perturb the $\{0,1\}$ vectors with additive uniform noise and rescale them to produce $[0,1]$ bounded vectors. Formally, label smoothing is defined as follows:

$$
\begin{aligned}
\widetilde{o}_{t,j}^{(k)} &= o_{t,j}^{(k)} + \mathscr{U}_{[0,\gamma]}, \quad k = 0,\ldots,|D_t| \\
\widetilde{\boldsymbol{o}}_t &= \widetilde{\boldsymbol{o}}_t / ||\widetilde{\boldsymbol{o}}_t||
\end{aligned}
\tag{3.26}
$$

where γ is a parameter defined by the modeler. $\widetilde{\boldsymbol{o}}_t$ now corresponds to a noisy version of the original one-hot encoded vector $\boldsymbol{o}_t$.

An issue with applying label smoothing is that the generator output tries to match the distorted representation of the data, and so the generator probability outputs will be biased towards low proability values. To address this, we propose here to apply equivalent smoothing to the generator output before passing it to the discriminator:

$$
\begin{aligned}
\widetilde{o}_{t,j}^{\mathbf{synth},(k)} &= o_{t,j}^{\mathbf{synth},(k)} + \mathscr{U}_{[0,\gamma]}, \quad k = 0,\ldots,|D_t| \\
\widetilde{\boldsymbol{o}}_t^{\mathbf{synth}} &= \widetilde{\boldsymbol{o}}_t^{\mathbf{synth}} / ||\widetilde{\boldsymbol{o}}_t^{\mathbf{synth}}||
\end{aligned}
\tag{3.27}
$$

Where γ should match the parameter used for the input smoothing. It removes the bias in the generator output and, thus, it is effectively trying to learn the original $[0,1]$ representations. Therefore, it produces unbiased probabilities. We refer to this as two-sided label smoothing.

To investigate the benefits of label smoothing, we systematically investigate three possible strategies for categorical variables:

$$
\text{no label smoothing:} \quad \boldsymbol{l}_0^{\texttt{NO}} = \begin{cases} \boldsymbol{w}_{1:N_C} \oplus \boldsymbol{p}_{1:N_C} \oplus \boldsymbol{o}_{1:N_D} & \text{for original data} \\ \boldsymbol{w}_{1:N_C}^{\mathbf{synth}} \oplus \boldsymbol{p}_{1:N_C}^{\mathbf{synth}} \oplus \boldsymbol{o}_{1:N_D}^{\mathbf{synth}} & \text{for synthetic data} \end{cases} \tag{3.28}
$$

$$
\text{one-sided label smoothing:} \quad \boldsymbol{l}_0^{\texttt{OS}} = \begin{cases} \boldsymbol{w}_{1:N_C} \oplus \boldsymbol{p}_{1:N_C} \oplus \widetilde{\boldsymbol{o}}_{1:N_D} & \text{for original data} \\ \boldsymbol{w}_{1:N_C}^{\mathbf{synth}} \oplus \boldsymbol{p}_{1:N_C}^{\mathbf{synth}} \oplus \boldsymbol{o}_{1:N_D}^{\mathbf{synth}} & \text{for synthetic data} \end{cases} \tag{3.29}
$$

$$\text{two-sided label smoothing:} \qquad \boldsymbol{l}_0^{\texttt{TS}} = \begin{cases} \boldsymbol{w}_{1:N_C} \oplus \boldsymbol{p}_{1:N_C} \oplus \widetilde{\boldsymbol{o}}_{1:N_D} & \text{for original data} \\ \boldsymbol{w}^{\textbf{synth}}_{1:N_C} \oplus \boldsymbol{p}^{\textbf{synth}}_{1:N_C} \oplus \widetilde{\boldsymbol{o}}^{\textbf{synth}}_{1:N_D} & \text{for synthetic data} \end{cases} \tag{3.30}$$

Sampling

Once the generator has been trained against the discriminator, we need to be able to generate the final synthetic dataset $\mathbf{T}_{\textbf{synth}}$. However, the dataset created by the generator $\widehat{\mathbf{T}}_{\textbf{synth}}$ corresponds to the encoded dataset $\widehat{\mathbf{T}}$. Therefore, we need to decode $\widehat{\mathbf{T}}_{\textbf{synth}}$ to obtain the final synthetic dataset.

In previous works (Xu and Veeramachaneni, 2018; Xu et al., 2019), the synthetic value is sampled from the probability distribution by simply assigning the value to the highest probability class (*i.e.* arg max assignment). However, this approach does not result in representative mode shares. Instead, as is typical in choice modeling scenarios (Ben-Akiva and Lerman, 1985), we propose to sample the synthetic value through simulation, *i.e.* drawing according to the output probability values (without any label smoothing applied). Thus, for categorical variables, we have two different ways to obtain the final value:

$$d_{t,j}^{\textbf{synth},\arg\max} = \arg\max \boldsymbol{o}_{t,j}^{\textbf{synth}} \tag{3.31}$$

$$d_{t,j}^{\textbf{synth},\texttt{simul}} = \texttt{simulation}\left[\boldsymbol{o}_{t,j}^{\textbf{synth}}\right] \tag{3.32}$$

Similary, by inverting Equation 3.19, we have for continuous variables:

$$c_{t,j}^{\textbf{synth},\arg\max} = \delta\, w_{t,j}^{\textbf{synth},(k)} \sigma_t^{(k)} + \eta_t^{(k)} \qquad \text{where } k = \arg\max \boldsymbol{p}_{t,j}^{\textbf{synth}} \tag{3.33}$$

$$c_{t,j}^{\textbf{synth},\texttt{simul}} = \delta\, w_{t,j}^{\textbf{synth},(k)} \sigma_t^{(k)} + \eta_t^{(k)} \qquad \text{where } k = \texttt{simulation}\left[\boldsymbol{p}_{t,j}^{\textbf{synth}}\right] \tag{3.34}$$

where δ corresponds to the same values used to encode the continuous variables $\boldsymbol{c}_t$.

In order to test the impacts of simulation versus maximum probability assignment for the categorical and continuous variables, we systematically test four different sampling strategies:

$$\arg\max \text{ for cont. and cat.:} \quad \mathbf{T}_{\textbf{synth}}^{\texttt{AA}} = \left\{\boldsymbol{c}_{1:N_C}^{\textbf{synth},\arg\max}, \boldsymbol{d}_{1:N_D}^{\textbf{synth},\arg\max}\right\} \tag{3.35}$$

$$\texttt{simulation} \text{ for cont. and } \arg\max \text{ for cat.:} \quad \mathbf{T}_{\textbf{synth}}^{\texttt{SA}} = \left\{\boldsymbol{c}_{1:N_C}^{\textbf{synth},\texttt{simul}}, \boldsymbol{d}_{1:N_D}^{\textbf{synth},\arg\max}\right\} \tag{3.36}$$

$$\arg\max \text{ for cont. and } \texttt{simulation} \text{ for cat.:} \quad \mathbf{T}_{\textbf{synth}}^{\texttt{AS}} = \left\{\boldsymbol{c}_{1:N_C}^{\textbf{synth},\arg\max}, \boldsymbol{d}_{1:N_D}^{\textbf{synth},\texttt{simul}}\right\} \tag{3.37}$$

$$\texttt{simulation} \text{ for cont. and cat.:} \quad \mathbf{T}_{\textbf{synth}}^{\texttt{SS}} = \left\{\boldsymbol{c}_{1:N_C}^{\textbf{synth},\texttt{simul}}, \boldsymbol{d}_{1:N_D}^{\textbf{synth},\texttt{simul}}\right\} \tag{3.38}$$

As a side note, we would like to add that the sampling process is entirely independent of the optimization process of both the generator and the discriminator. It is, therefore, possible to train a single model and test the different sampling methods afterward.

Summary

Table 3.3 provides a summary of the possible DATGAN versions using the proposed loss functions, label smoothing strategies, and sampling strategies. In Section 3.5, we compare these versions against each other to select the best-performing.

Table 3.3: Summary of all the DATGAN versions.

Name	Loss function	Label smoothing	Sampling
`SGAN_NO_AA`	SGAN (Eq. 3.9)	none	$\arg\max$ for C_t and D_t
`SGAN_NO_SA`			`simulation` for C_t, $\arg\max$ for D_t
`SGAN_NO_AS`			$\arg\max$ for C_t, `simulation` for D_t
`SGAN_NO_SS`			`simulation` for C_t and D_t
`SGAN_OS_AA`		one-sided	$\arg\max$ for C_t and D_t
`SGAN_OS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`SGAN_OS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`SGAN_OS_SS`			`simulation` for C_t and D_t
`SGAN_TS_AA`		two-sided	$\arg\max$ for C_t and D_t
`SGAN_TS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`SGAN_TS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`SGAN_TS_SS`			`simulation` for C_t and D_t
`WGAN_NO_AA`	WGAN (Eq. 3.12)	none	$\arg\max$ for C_t and D_t
`WGAN_NO_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGAN_NO_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGAN_NO_SS`			`simulation` for C_t and D_t
`WGAN_OS_AA`		one-sided	$\arg\max$ for C_t and D_t
`WGAN_OS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGAN_OS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGAN_OS_SS`			`simulation` for C_t and D_t
`WGAN_TS_AA`		two-sided	$\arg\max$ for C_t and D_t
`WGAN_TS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGAN_TS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGAN_TS_SS`			`simulation` for C_t and D_t
`WGGP_NO_AA`	WGGP (Eq. 3.15)	none	$\arg\max$ for C_t and D_t
`WGGP_NO_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGGP_NO_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGGP_NO_SS`			`simulation` for C_t and D_t
`WGGP_OS_AA`		one-sided	$\arg\max$ for C_t and D_t
`WGGP_OS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGGP_OS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGGP_OS_SS`			`simulation` for C_t and D_t
`WGGP_TS_AA`		two-sided	$\arg\max$ for C_t and D_t
`WGGP_TS_SA`			`simulation` for C_t, $\arg\max$ for D_t
`WGGP_TS_AS`			$\arg\max$ for C_t, `simulation` for D_t
`WGGP_TS_SS`			`simulation` for C_t and D_t

3.3.5 Result assessments

For assessing the quality of synthetic datasets compared to the original datasets, we use two main methods: (i) a statistical method; (ii) a ML-based method. The goal of the first method is to verify that the synthetic datasets display the same statistical properties compared to the original dataset. In order to do this, we compare the distributions of each column individually between the synthetic and original datasets. We then test combinations of multiple columns to study if the models can grasp more complex correlations between the variables. The second method is closer to a real-world problem one can face. Indeed, the goal is to study if the synthetic datasets can be used in the classification/regression context of ML.

Statistical tests

For the statistical tests, we build on existing approaches in the transportation literature (Garrido et al., 2019; Borysov et al., 2019; Badu-Marfo et al., 2020). The idea is to compute frequency lists, *i.e.* frequency count of each unique value, for each column on both the original dataset $\boldsymbol{\pi}$ and the synthetic dataset $\boldsymbol{\pi}^{\textbf{synth}}$. In the literature, authors typically only calculate the frequency lists for single columns, *i.e.* marginal distributions, for a few relevant variables and test them against each other. In this chapter, we build on this in two ways:(i) calculating joint frequency lists for n columns simultaneously (therefore assessing joint distributions of order n) and (ii) systematically testing all possible combinations of columns at each aggregation level.

If we only compute the frequency lists for single variables, we will only assess the marginal distribution of each variable independently. Indeed, it verifies whether each column of the synthetic data matches the distribution of the corresponding column in the original data. However, it does not provide any information on the correlations between the columns in either the synthetic or original data, *i.e.* it assesses each column independently of all other columns. Therefore, to assess whether relationships between variables in the synthetic data match that of the original data, it is necessary to investigate the joint distributions of multiple columns simultaneously. To address this, we simultaneously calculate the joint frequency lists for multiple columns. Furthermore, at each aggregation level (*i.e.* number of columns), we calculate the frequency lists for all possible combinations of columns.

Since the number of possible combinations at each aggregation level increases factorially, we limit the level of aggregation to one, two, or three columns as follows:

First order: Columns are compared to each other, giving N_V different aggregated lists.

Second order: Columns are aggregated two-by-two, giving $\binom{N_V}{2}$ different aggregated lists.

Third order: Columns are aggregated three-by-three, giving $\binom{N_V}{3}$ different aggregated lists.

Note that continuous variables must be binned to calculate frequency lists. We arbitrarily set the number of bins for each continuous column to 10, such that the second and third order frequency lists of continuous columns will have 100 and 1000 unique values, respectively.

Once these frequency lists have been computed, we can compare them using standard statistic metrics defined in the literature. We select five different metrics:

- Mean Absolute Error (MAE):

$$\mathrm{MAE}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right) = \frac{\sum_{i=1}^{N_{\mathrm{cnt}}} |\pi_i^{\mathbf{synth}} - \pi_i|}{N_{\boldsymbol{pi}}} \tag{3.39}$$

  where $N_{\boldsymbol{\pi}}$ corresponds to the size of the frequency list $\boldsymbol{\pi}$.

- Root Mean Squared Error (RMSE):

$$\mathrm{RMSE}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right) = \left(\frac{\sum_{i=1}^{N_{\mathrm{cnt}}} \left(\pi_i^{\mathbf{synth}} - \pi_i\right)^2}{N_{\mathrm{cnt}}}\right)^{1/2} \tag{3.40}$$

- SRMSE (Müller and Axhausen, 2011):

$$\mathrm{SRMSE}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right) = \frac{\mathrm{RMSE}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right)}{\overline{\boldsymbol{\pi}}} \tag{3.41}$$

  where $\overline{\boldsymbol{\pi}}$ corresponds to the average value of $\boldsymbol{\pi}$.

- Coefficient of determination:

$$R^2\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right) = 1 - \frac{\sum_{i=1}^{N_{\mathrm{cnt}}} \left(\pi_i^{\mathbf{synth}} - \pi_i\right)^2}{\sum_{i=1}^{N_{\mathrm{cnt}}} \left(\overline{\pi}_i - \pi_i\right)^2} \tag{3.42}$$

- Pearson's correlation:

$$\rho_{\mathrm{Pearson}}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right) = \frac{\mathrm{cov}\left(\boldsymbol{\pi}^{\mathbf{synth}}, \boldsymbol{\pi}\right)}{\sigma_{\boldsymbol{\pi}} \sigma_{\boldsymbol{\pi}^{\mathbf{synth}}}} \tag{3.43}$$

  where $\mathrm{cov}(\cdot, \cdot)$ corresponds to the covariance matrix and $\sigma_{\boldsymbol{X}}$ the standard deviation of a given vector $\boldsymbol{X}$.

Finally, the results can be averaged over all combinations to obtain a single number per synthetic dataset, statistic, and aggregation level.

Supervised learning-based validation

We propose a new supervised learning-based validation method for synthetic data that uses supervised classification and regression models to approximate the complete conditional distributions of each variable, given all other variables in the dataset. The general approach is to estimate two regression or classification models for each variable $\boldsymbol{v}_t$. The first model (m_t) is estimated on a training portion of the original data, and the second ($m_t^{\textbf{synth}}$) is estimated on a corresponding training portion of the synthetic data. In each case, the model tries to predict the values in the corresponding column conditional on all other columns in the dataset.

Each model is then validated on the same test portion of the original data, providing two loss scores. The expected value of the loss of the model estimated on the synthetic data (which approximates the conditionals in the synthetic dataset) should be greater than or equal to the expected loss of the model estimated on the original data (which approximates the true conditionals in the original data). The closer the loss scores of the models estimated on the synthetic and original data, the more closely the synthetic data has captured the conditional distributions of the original data. The approach is detailed in Algorithm 3.3.

Algorithm 3.3 Supervised learning-based validation

```
Inputs: Original data $\mathbf{T}$, synthetic data $\mathbf{T}_{\textbf{synth}}$
Output: Similarity score for each variable $\boldsymbol{v}_t \in \mathbf{T}$
 1: for all $\boldsymbol{v}_t \in \mathbf{T}$ do
 2:     $\boldsymbol{y}_t = \boldsymbol{v}_t$
 3:     $\boldsymbol{X}_t = \mathbf{T} \setminus \boldsymbol{v}_t$
 4:     Divide $\boldsymbol{y}_t$ and $\boldsymbol{X}_t$ into training set $(\boldsymbol{y}_{t,\text{train}}, \boldsymbol{X}_{t,\text{train}})$ and test set $(\boldsymbol{y}_{t,\text{test}}, \boldsymbol{X}_{t,\text{test}})$
 5:     $\boldsymbol{y}_t^{\textbf{synth}} = \boldsymbol{v}_t^{\textbf{synth}}$
 6:     $\boldsymbol{X}_t^{\textbf{synth}} = \mathbf{T}_{\textbf{synth}} \setminus \boldsymbol{v}_t^{\textbf{synth}}$
 7:     Sample training set $(\boldsymbol{y}_{t,\text{train}}^{\textbf{synth}}, \boldsymbol{X}_{t,\text{train}}^{\textbf{synth}})$ from $\boldsymbol{y}_t^{\textbf{synth}}$ and $\boldsymbol{X}_t^{\textbf{synth}}$, with the same dimensions
      as $(\boldsymbol{y}_{t,\text{train}}, \boldsymbol{X}_{t,\text{train}})$.
 8:     if $\boldsymbol{v}_t \in \boldsymbol{c}_{1:N_C}$ then
 9:         Estimate regression model $m_{t,\text{reg}}$ on $(\boldsymbol{y}_{t,\text{train}}, \boldsymbol{X}_{t,\text{train}})$
10:         Estimate regression model $m_{t,\text{reg}}^{\textbf{synth}}$ on $(\boldsymbol{y}_{t,\text{train}}^{\textbf{synth}}, \boldsymbol{X}_{t,\text{train}}^{\textbf{synth}})$
11:         $g_t^{\text{reg}} = \mathscr{L}_{\text{MSE}}(\boldsymbol{y}_{t,\text{test}}, m_{t,\text{reg}}^{\textbf{synth}}(\boldsymbol{X}_{t,\text{test}})) / \mathscr{L}_{\text{MSE}}(\boldsymbol{y}_{t,\text{test}}^{\textbf{synth}}, m_{t,\text{reg}}(\boldsymbol{X}_{t,\text{test}}^{\textbf{synth}}))$
12:         return $g_t^{\text{reg}}$
13:     else
14:         Estimate probabilistic classification model $m_{t,\text{class}}$ on $(\boldsymbol{y}_{t,\text{train}}, \boldsymbol{X}_{t,\text{train}})$
15:         Estimate probabilistic classification model $m_{t,\text{class}}^{\textbf{synth}}$ on $(\boldsymbol{y}_{t,\text{train}}^{\textbf{synth}}, \boldsymbol{X}_{t,\text{train}}^{\textbf{synth}})$
16:         $g_t^{\text{class}} = \mathscr{L}_{\text{log-loss}}(\boldsymbol{y}_{t,\text{test}}, m_{t,\text{class}}^{\textbf{synth}}(\boldsymbol{X}_{t,\text{test}})) - \mathscr{L}_{\text{log-loss}}(\boldsymbol{y}_{t,\text{test}}^{\textbf{synth}}, m_{t,\text{class}}(\boldsymbol{X}_{t,\text{test}}^{\textbf{synth}}))$
17:         return $g_t^{\text{class}}$
```

We make use of gradient boosting ensembles of decision trees for both $m_{\text{reg},t}$ and $m_{\text{class},t}$ as (i) they can be easily applied to both regression and probabilistic classification problems; (ii) are computationally efficient to estimate; (iii) have been shown to have high predictive performance on a wide variety of supervised learning tasks; (iv) can determine appropriate regularisation automatically using early stopping. We specifically make use of the LightGBM library (Ke et al., 2017) which inherently handles categorical input features, thus avoiding the need for one-hot encoding of categorical variables.

For continuous variables, the score g_t^{reg} is the ratio of the mean squared error of the model estimated on the synthetic dataset to the mean squared error of the model estimated on the original dataset, with a score of 1 indicating a perfect match, and a higher score representing a worse fit. For categorical variables, the score g_t^{class} is the absolute difference between the normalized log-loss of the model estimated on the synthetic dataset and the normalized log-loss of the model estimated on the original dataset, with a score of 0 indicating a perfect match, and a higher score representing a worse fit. The scores can be summed over all columns to give aggregate scores for all continuous and categorical variables.

While Algorithm 3.3 describes a single train-test split, the same algorithm can be used with k-fold cross-validation to obtain more accurate estimates of the model losses. We use 5-fold cross-validation and stratified sampling to select training folds for the categorical data.

3.3.6 Implementation notes

The code for DATGAN has been implemented using Python 3.9. We use the library `tensorflow` (v2.8) (Abadi et al., 2016) for the main components of the neural networks. In addition, we use the library `networkx` (v2.5) (Hagberg et al., 2008) for specifying the DAG $\mathcal{G}$ discussed in Section 3.3.1. This library already has built-in functions to verify that a user-specified graph is a DAG.

For the optimization process using the different loss functions presented in Section 3.3.3, we follow the authors' instructions of the different articles for the hyperparameters. During the initial tests, we investigated different values of the learning rate and decided on the following learning rates for each loss, which appeared to work best in these initial tests:

Standard loss learning rate of $1 \cdot 10^{-3}$

Wasserstein loss learning rate of $2 \cdot 10^{-4}$

Wasserstein loss with gradient-penalty learning rate of $1 \cdot 10^{-4}$

We do not provide any specific results for this hyperparameter since it is not the main focus of this work.

The complete code for this project, including the different versions of DATGAN, the case studies, and the results, can be found on Github at https://github.com/gledерrey/SynthPop. In addition, a Python library has been created for the DATGAN model on Pypi, available at https://pypi.org/project/datgan/. The code for the library is available on Github at https://github.com/glederrey/DATGAN.

3.4 Case studies

In this section, we present the case studies for this chapter. First, we introduce the datasets in Section 3.4.1. We provide a short description of each dataset. Then, Section 3.4.2 gives a detailed overview of the training method used with all the models to bring as much fairness as possible in comparing the models.

3.4.1 Datasets

The first dataset is a household travel survey of the Chicago metropolitan area, conducted from January 2007 to February 2008. It is named after the agency that collected it: the Chicago Metropolitan Agency for Planning (CMAP) dataset. The trips are given as one and two-day travel diaries, provided by all the members of the households. The data is therefore hierarchical. The dataset has first been cleaned to remove incomplete entries. Then, we selected one unique trip per individual per household for the final dataset to remove data leakage. It thus contains a total of 8’929 trips with 15 columns. The appendix gives a complete description of this dataset in Table C.1. The DAG used for DATGAN with this dataset can also be found in the appendix; see Figure C.1.

The second dataset is the London Passenger Mode Choice (LPMC) dataset (Hillel et al., 2018). It combines the London Travel Diary Survey (LTDS) records with matched trip trajectories and corresponding mode alternatives. The LTDS was conducted between April 2012 and March 2015 and records trips made by individuals residing within Greater London. The trip trajectories are extrapolated from Google Maps API. The final dataset has been processed to not lead to data leakage. Similar to the CMAP dataset, we selected only one trip per household. The final dataset contains a total of 17’616 trips with 27 columns. The appendix gives a complete description of this dataset in Table C.2. The DAG used for DATGAN with this dataset can also be found in the appendix; see Figure C.2. In addition, we created a smaller version of the LPMC dataset by randomly selecting 50% of the rows for testing the DATGAN versions. This dataset is conveniently named LPMC_half. We mainly use it to understand the effect of the number of rows on the performance of the models.

Table 3.4: Summary of the datasets used in the case studies. Full description of the datasets can be found in the Appendix.

Name	#columns	#continuous	#categorical	#rows
CMAP	15	3	12	8’929
LPMC	27	13	14	17’616
LPMC_half	27	13	14	8’808
ADULT	14	4	10	45’222

The third and final dataset is the ADULT dataset (Kohavi, 1996), also known as the Census-Income dataset. This dataset contains socio-economic variables on multiple individuals to predict if their income is below or above $50k/yr. From the original dataset, we removed all the rows with unknown values. The appendix gives a complete description of this dataset in Table C.3. The DAG used for DATGAN with this dataset can also be found in the appendix; see Figure C.3. Due to its larger size than the other datasets, the ADULT dataset is only used when comparing DATGAN with state-of-the-art models.

3.4.2 Training process

This chapter aims to propose a new way to generate synthetic data. However, we need to test all these generative models on a similar playground for fairness toward state-of-the-art methods in the literature. We, thus, decide to train every model on 1'000 epochs with a batch size N_b of 500, even if the optimization process could be stopped earlier. In addition, we decided to keep the original hyperparameters provided in the articles. While optimizing these parameters would likely lead to better results, most users would use the models as the authors provide them. In addition, each model is trained five times on each dataset, and each of the five models generates five synthetic datasets with the same number of rows as the original dataset. This means that each test is performed on 25 synthetic datasets. Thus, the results provided in Section 3.5 correspond to the test's average value.

The training process is slightly different for the DATGAN versions. Indeed, the sampling can be done independently after the training process. Therefore, we only have to train nine different models (combinations of loss functions and label smoothing). Each of these models is trained five times. Then, for each of these 45 models, we use the four different sampling methods to produce five synthetic datasets for each method. Therefore, we get a total of 900 synthetic datasets to be compared for each dataset, *i.e.* 25 synthetic datasets for each of the 36 models presented in Table 3.3

3.5 Results

In this section, we present the results obtained using the assessment methods presented in Section 3.3.5 on the different case studies presented in Section 3.4. Section 3.5.1 compares the 36 different versions of the DATGAN model to find the best performing. Then, Section 3.5.2 compares DATGAN against state-of-the-art models presented in Section 3.2.3. Finally, Section 3.5.3 performs a sensitivity analysis on the DAG used for DATGAN to understand its effect on the performance of the model.

3.5.1 Comparison of DATGAN versions

For the CMAP case study, the complete table of results are provided in the appendix, Section F.1. We see that DATGAN with the `WGGP` loss provides poorer results compared to DATGAN with the `SGAN` or `WGAN` loss. It is the case with both assessment methods. This is quite unexpected since the `WGGP` loss is the most recent loss function available amongst the three. We do not clearly explain why the model fails with this loss on this particular dataset. Secondly, we see that all the models using the one-sided label smoothing (`OS`) tend to perform worse than their counterparts. This result was expected, especially when using simulation while sampling the synthetic variables. Indeed, if the model only uses one-sided label smoothing, the generator will learn the noisy version of the categorical variables. If we use $\arg\max$ for selecting the categorical variables, it does not have a large effect since the noise is quite small. However, the final synthetic probability distribution does not correspond to the original one when using simulation. Therefore, the model will sample the values wrongly and thus lead to poor results. When we compare two-sided label smoothing (`TS`) against no label smoothing (`NO`), we see that the results are different for the `SGAN` and the `WGAN` loss functions. Indeed, it seems that it is better to use the two-sided label smoothing with the `WGAN` loss, and it is better to avoid label smoothing with the `SGAN` loss. Table 3.5 shows the ten best models in terms of average ranking on the two assessment methods. We see that both loss functions are viable. However, the `WGAN` loss seems to be slightly better on average. The sampling method seems to have less influence than the label smoothing and the loss function on the final results.

Table 3.5: Average rankings for the ten best DATGAN models on the CMAP dataset.

Name	Avg. rank stats	Avg. rank ML	rank
`WGAN_TS_AS`	$\boxed{\textbf{2.88}}$	8.0	$\boxed{\textbf{5.44}}$
`WGAN_TS_AA`	8.24	4.0	6.12
`WGAN_TS_SS`	4.04	9.5	6.77
`SGAN_NO_SA`	10.08	3.5	6.79
`SGAN_NO_SS`	8.56	6.0	7.28
`WGAN_TS_SA`	6.84	8.0	7.42
`SGAN_NO_AS`	12.68	4.0	8.34
`SGAN_NO_AA`	15.96	$\boxed{\textbf{2.0}}$	8.98
`SGAN_TS_AS`	8.68	9.5	9.09
`SGAN_TS_SA`	7.88	10.5	9.19

For the LPMC dataset, we have quite unexpected results compared to the CMAP case study. Numerical results are provided in the appendix, Section F.6. The conclusions on the label smoothing remain the same. However, the best loss function to use in this case is the `WGGP` loss. This is especially visible when using the ML efficacy method to assess the results. All the models with the `WGGP` loss consistently outperforms the models with the other losses. The

WGAN loss performs slightly worse than the SGAN loss. Table 3.6 shows the ten best models in terms of average on both assessment methods. We see that both models using simulation for the categorical variables and two-sided label smoothing outperform the other models. It is interesting to note that, in this case, the sampling method for continuous variables does not affect much the final results. This result can be explained thanks to the encoding we use for the continuous variables. Indeed, all the encoded values $w_{t,j}$ are computed such that they all correspond to the continuous value $c_{t,j}$ once drawn from their respective Gaussian mixture. Therefore, if the algorithm fails to choose the right mixture, the sampling will return the same continuous synthetic value.

Table 3.6: Average rankings for the ten best DATGAN models on the LPMC dataset.

Name	**Avg. rank stats**	**Avg. rank ML**	**rank**
WGGP_TS_SS	$\boxed{\mathbf{2.96}}$	4.0	$\boxed{\mathbf{3.48}}$
WGGP_TS_AS	5.16	3.0	4.08
WGGP_NO_AS	9.52	8.0	8.76
WGGP_NO_SS	9.76	8.5	9.13
WGGP_OS_SA	15.60	4.0	9.80
WGGP_OS_AA	18.36	$\boxed{\mathbf{2.5}}$	10.43
WGGP_TS_SA	12.36	9.5	10.93
WGGP_NO_AA	12.72	9.5	11.11
WGGP_TS_AA	14.80	7.5	11.15
WGGP_NO_SA	13.04	10.0	11.52

If we compare the two previous case studies, there are two hypotheses as to why the WGAN loss works better with the CMAP dataset and the WGGP loss with the LPMC dataset:(i) the number of data points in the dataset influences the optimization process with a given loss function; (ii) the ratio of continuous/categorical variables influences the choice of the loss function. The first hypothesis is easier to test since we can use the LPMC_half case study to compare with the LPMC case study. The results in the appendix, see Section F.12, show the same behavior with the smaller LPMC dataset than with the complete one. Indeed, the WGGP loss leads to significantly better results across both assessments methods compared to the other loss functions. Table 3.7 shows the ten best models in terms of average on both assessment methods. Once again, the version with the WGGP loss, two-sided label smoothing, and sampling using simulation for both continuous and categorical variables appears to be the best model. However, it is interesting to note that the versions with the WGAN loss function shows the best statistics when only looking at the first aggregation level on categorical variables. This hints at the fact that the WGAN loss is the most appropriate loss to use when a dataset contains a majority of categorical variables.

For further analysis, we settle on one particular version of the DATGAN. As seen across all the

Table 3.7: Average rankings for the ten best DATGAN models on the LPMC_half dataset.

Name	Avg. rank stats	Avg. rank ML	rank
WGGP_TS_SS	$\boxed{3.12}$	4.5	$\boxed{3.81}$
WGGP_TS_AS	4.84	4.0	4.42
WGGP_NO_SS	8.20	8.5	8.35
WGGP_OS_SA	14.36	4.0	9.18
WGGP_NO_AS	11.88	7.0	9.44
WGGP_OS_AA	16.00	$\boxed{3.5}$	9.75
WGGP_OS_SS	18.04	5.5	11.77
WGGP_OS_AS	20.24	5.0	12.62
SGAN_TS_SS	9.52	16.0	12.76
SGAN_NO_AS	10.96	16.0	13.48

case studies, two-sided label smoothing is the best method for the discriminator. In addition, using simulation on both types of variables for the sampling leads to better results for the two LPMC case studies. For the CMAP case study, it is amongst the best models. We therefore choose to use this sampling method to stay consistent across both variable types. Finally, we can not recommend the same loss function for all the datasets. Indeed, the WGAN loss performs best when there are more categorical variables than continuous in the dataset. Therefore, we use the WGAN loss function for the CMAP case study and the WGGP loss function for the LPMC case studies.

3.5.2 Comparison with state-of-the-art models

At this point of the chapter, we have presented our new model for generating synthetic datasets and have selected the best version depending on the type of case study. However, we now want to compare our model against state-of-the-art models presented in the literature. We, thus, test our DATGAN model against the four models presented in the literature (see Section 3.2.3) on the four different case studies. We use the same assessment methods as previously to compare all the models. The detailed results are provided in the appendix, Section F.2. The summarized results are given in Table 3.8. It shows the average rankings on both assessment methods for the four different case studies. We see that DATGAN outperforms all the other models in the first three case studies. For example, it is consistently the best model when using the ML efficacy method. For the ADULT case study, we decided to test DATGAN with the WGAN and the WGGP loss functions. Since the ADULT dataset contains more categorical variables than continuous, we expect the WGAN loss to perform better, as it is shown in Table 3.8. It performs the best on the statistical assessments. However, it seems to struggle with the ML efficacy method. While the results are quite close on the continuous variables, the two

Table 3.8: Average rankings of the state-of-the-art models against DATGAN on the four case studies.

Name	Avg. rank stats	Avg. rank ML	rank
CMAP case study			
DATGAN (`WGAN`)	$\boxed{\textbf{1.00}}$	$\boxed{\textbf{1.0}}$	$\boxed{\textbf{1.00}}$
CTAB-GAN	2.92	2.5	2.71
TGAN	2.60	3.0	2.80
CTGAN	4.24	4.0	4.12
TVAE	4.24	4.5	4.37
LPMC case study			
DATGAN (`WGGP`)	$\boxed{\textbf{1.00}}$	$\boxed{\textbf{1.0}}$	$\boxed{\textbf{1.00}}$
TGAN	2.36	2.5	2.43
CTGAN	3.04	3.0	3.02
CTAB-GAN	4.08	3.5	3.79
TVAE	4.52	5.0	4.76
LPMC_half case study			
DATGAN (`WGGP`)	$\boxed{\textbf{1.08}}$	$\boxed{\textbf{1.0}}$	$\boxed{\textbf{1.04}}$
TGAN	2.60	2.5	2.55
CTAB-GAN	3.04	4.0	3.52
CTGAN	3.64	3.5	3.57
TVAE	4.64	4.0	4.32
ADULT case study			
DATGAN (`WGAN`)	$\boxed{\textbf{1.08}}$	$\boxed{\textbf{3.0}}$	$\boxed{\textbf{2.04}}$
TGAN	1.92	3.5	2.71
DATGAN (`WGGP`)	3.32	3.5	3.41
TVAE	3.80	3.5	3.65
CTGAN	4.88	$\boxed{\textbf{3.0}}$	3.94
CTAB-GAN	6.00	4.5	5.25

DATGAN models are the only models that do not fail the test on the categorical variables. Indeed, it seems that the other models tend not to produce enough of the low probability categories. Thus, these models tend to oversimplify the generated synthetic data compared to the DATGAN models. One of the reasons why such a thing happens might come from the sampling process. Indeed, using simulation to get the final categories allows for more representation of the low probability values than using the maximum probability estimator.

Since we compared models across multiple articles, it is interesting to look at their conclusions. For example, Xu et al. (2019) show that their CTGAN model is consistently outperforming TVAE. We see the same ranking between the two models except for the ADULT case study. Therefore, we can draw the same conclusion on these two models as the authors. However, Zhao et al. (2021) claim that CTAB-GAN outperforms CTGAN across all their assessments. In our case, we see that CTAB-GAN outperforms CTGAN only when the case studies contain a small number of data points. While we have used both models as intended to be used by their authors, we only changed the final number of epochs. In their article, Zhao et al. (2021) have trained both models on 150 epochs. Therefore, CTAB-GAN may be providing a better early optimization process than CTGAN. However, further work would be required to analyze this result, and it is out of the scope of this work.

In addition, we have tested both models presented by Garrido et al. (2019): a WGAN and a VAE. Unfortunately, results are not shown in this chapter because both models failed to produce adequate continuous variables. Indeed, the encoding of continuous variables is done such that they are binned based on their original distributions, thus treating them as categorical variables. This, therefore, lead to especially poor results when comparing continuous distributions.

3.5.3 Sensitivity analysis of the DAG

In this section, we want to analyze how the DAG can affect the performances of DATGAN and how it can be used to modify the generation of synthetic datasets. We first perform the analysis on the DAG using different versions of the latter. Then, we show how we can alter the DAG to generate hypothetical synthetic datasets.

Structure of the DAG

In the previous section, we have analyzed all the different versions of DATGAN to select the best one. However, in this section, we want to investigate how the DAG will influence the results. Thus, we only use the best possible model for each case study. The idea is to start with the DAG presented in the appendix for each case study and make variations of it to study the generated datasets. Therefore, we created five different DAGs for each case study:

- `full`: Complete DAG presented in the appendix for each case study.

- `trans. red.`: Transitive reduction of the `full` DAG. The transitive reduction consists in removing as many edges as possible in a DAG such that there exists only one path between two vertices in the graph.

- `linear`: This DAG consists in taking the variables in the order provided by the dataset and linking them to each other linearly. Thus, there are no multi-inputs within the DAG. This is similar to the technique used in TGAN (Xu and Veeramachaneni, 2018).

- `prediction`: This DAG consists of only one sink node. All the other nodes are considered source nodes linked to the sink node. The source nodes are not linked to each other. The sink nodes are the `choice` for the CMAP case study and the `travel_mode` for the LPMC case studies.

- `no links`: This DAG consists of only nodes without any edges. This, thus, cuts all the links between the variables.

At first glance, we can already predict that the last two DAGs should perform badly since the connections between them are either badly implemented or absent. Thus, we are mostly interested in the first three DAGs. Indeed, if the DAG can help the model to generate more representative synthetic data, the `full` DAG or the `trans. red.` DAG should outperform the `linear` DAG. As for the previous section, the details of the results are given in the appendix, Section F.3. Table 3.9 shows the rankings of the DAGs on the CMAP case study. As expected, the best two DAGs are the two complete ones. It is interesting to note that the `full` DAG provides better results on the ML efficacy assessment while the `trans. red.` provides better results on the statistical assessments. Since the model with the `full` DAG contains more edges than the other DAGs, it is, therefore, larger and more complex to train. Thus, this can hurt the LSTM cells' performance when creating the synthetic variables. However, the correlations between the variables are better with the complete DAG since it always performs best compared to the other DAGs.

Table 3.9: Average rankings of the different DAGs on the CMAP dataset

Name	**Avg. rank stats**	**Avg. rank ML**	**rank**
`trans. red.`	$\boxed{\textbf{1.96}}$	2.0	$\boxed{\textbf{1.98}}$
`full`	3.04	$\boxed{\textbf{1.0}}$	2.02
`linear`	2.80	3.0	2.90
`prediction`	3.80	4.0	3.90
`no links`	3.40	5.0	4.20

The CMAP dataset is relatively small. Therefore, learning the correlations between the variables can be pretty difficult. The LPMC dataset, on the other hand, is twice as big. It is thus interesting if the DAG can still provide the same kind of help on a larger dataset. Table 3.10

shows the rankings of the DAG on the LPMC case study. We see that this time, the `linear` DAG is the best one, closely followed by the two complete DAGs. While the `full` DAG has some issues with the statistical assessments, it remains the best on the ML efficacy method. Therefore, it seems that if one wants to generate a synthetic dataset with column data as close as possible to the original dataset, a simpler DAG leads to better results. However, if one wants to keep as much correlation as possible, one should opt for a complete DAG.

Table 3.10: Average rankings of the different DAGs on the LPMC dataset

Name	Avg. rank stats	Avg. rank ML	rank
`linear`	$\boxed{\textbf{2.08}}$	2.0	$\boxed{\textbf{2.04}}$
`full`	2.80	$\boxed{\textbf{1.5}}$	2.15
`trans. red.`	2.16	2.5	2.33
`prediction`	4.00	4.0	4.00
`no links`	3.96	5.0	4.48

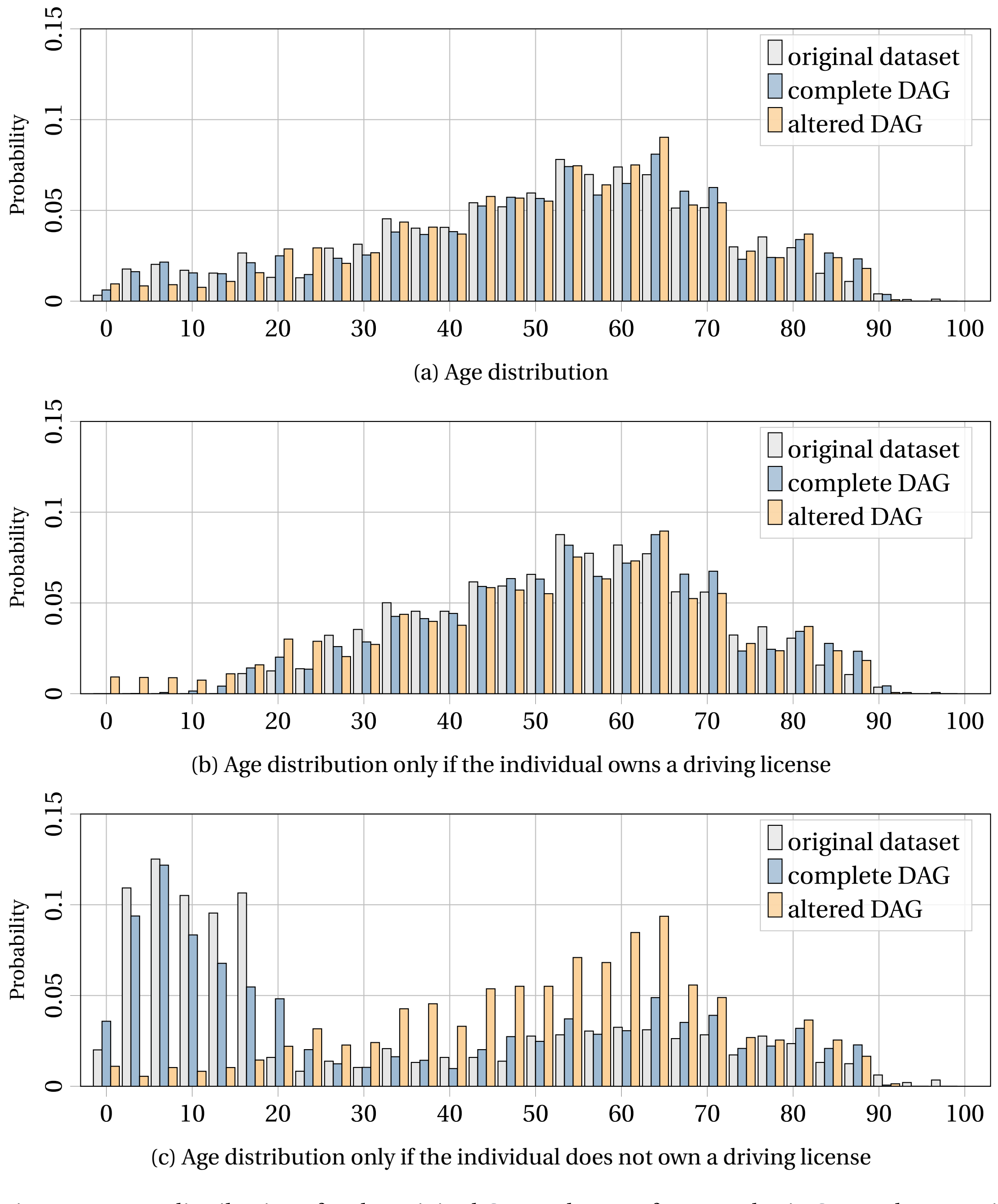
Finally, we want to confirm our hypothesis that the completeness of the DAG is less important with more data points. We, thus, make the same tests on the smaller LPMC case study. Table 3.11 shows the rankings of the DAG on the LPMC_half case study. Results show that the `linear` DAG performs better than the `full` DAG on both metrics. At first glance, this result can be quite surprising. However, when we look at the DAG for both the CMAP and the LPMC case study, we see that the LPMC DAG is much more complex than the CMAP DAG. Indeed, the CMAP DAG contains a total of 25 edges for 15 nodes for an average of $1.\bar{6}$ edges per node. On the other hand, the LPMC DAG contains a total of 63 edges for 27 nodes for an average of $2.\bar{3}$ edges per node. It is, thus, possible that the model struggles to train correctly with a more complex DAG. However, we see that the `trans. red.` version of the LPMC DAG leads to even worse results than the `full` DAG. Therefore, it might also be possible that this DAG is not well constructed for this particular case study. It, thus, requires further investigation to fully understand how the DAG affects the results of this case study.

Table 3.11: Average rankings of the different DAGs on the LPMC_half dataset

Name	Avg. rank stats	Avg. rank ML	rank
`linear`	$\boxed{\textbf{1.84}}$	$\boxed{\textbf{1.0}}$	$\boxed{\textbf{1.42}}$
`full`	2.04	2.0	2.02
`trans. red.`	3.76	3.0	3.38
`prediction`	4.08	4.0	4.04
`no links`	3.28	5.0	4.14

Effect of the DAG on the synthetic dataset

The final section of the results shows what can be achieved with the DAG depending on the desire of the modeler, *i.e.* how the modeler can introduce structural zeros in the data. Indeed, we have shown that using a complete DAG allows generating the best possible synthetic datasets compared to state-of-the-art models. However, the DAG can also be used to create



(a) Age distribution

(b) Age distribution only if the individual owns a driving license

(c) Age distribution only if the individual does not own a driving license

Figure 3.6: Age distributions for the original CMAP dataset, for a synthetic CMAP dataset with a complete DAG, and for a synthetic CMAP dataset with an altered DAG.

hypothetical synthetic datasets. Since the DAG controls the causal links between the variables, removing any relationships between two or more variables is simple. For example, in the CMAP case study, we could imagine a hypothetical population with no minimum age requirement to get a driving license. In order to achieve this, we can simply remove the link between the variables `age` and `license` in the DAG presented in Figure C.1. If the modeler wants to ensure that two variables do not interact, these variables should be defined as source nodes in the DAG. Figure 3.6 shows the age distributions for the original CMAP dataset, for a synthetic dataset generated with a complete DAG, and for a synthetic dataset generated with the altered DAG. Figure 3.6a shows the distribution of the variable `age` in all of the datasets. As we can see, the synthetic probability distributions are quite similar to the original probability distributions. However, if we look at the age distribution when the individual owns a driving license (Figure 3.6b), we see that the synthetic dataset with the altered DAG still produces individuals of age lower than 18 owning a driving license. It is also the case with the synthetic dataset generated from a complete DAG. However, the number of minors with a driving license is marginally less than the data generated with the altered DAG. The effect of the altered DAG can be seen even better when looking at the age distribution of individuals not owning a driving license (Figure 3.6c). Indeed, both the original and synthetic datasets with a complete DAG show most young individuals who do not own a driving license. However, the altered DAG produces the same type of distributions compared to the previous two. This, thus, shows that we eliminated the correlation between age and owning a driving license with the altered DAG.

The results presented in this last section show that the modeler can easily introduce structural zeros, *i.e* forcing correlations between variables to be null. However, the current methodology cannot generate unexisting correlations in the data. In addition, methodologies focusing on structural zeros often compete with methodologies focusing on sampling zeros, *i.e.* unseen data. Thus, one could think that it is also the case with DATGAN. However, as shown in Figure 3.6a, the age distribution does not suffer from adding structural zeros. Indeed, since the DAG is specified beforehand, both processes are not competing against each other. Therefore, the learning phase of DATGAN only takes care of sampling zeros and produces a robust model. Additionally, one could use the $\arg\max$ sampling process to improve structural zeros at the cost of slightly lower performance on the sampling zeros.

3.6 Summary

This chapter presents a novel GAN architecture, the DATGAN, that integrates expert knowledge to control the causal links between the variables. The methodology shows how a DAG can model the generator's structure and how synthetic variables are generated using LSTM cells. In addition, we provide an efficient way to encode categorical and continuous variables. While the core mechanics of DATGAN remain the same, we explore different loss functions for training DATGAN, using label smoothing on categorical variables and multiple sampling methods. To compare the results as fairly as possible, we provide two new systematic assessment methods for comparing synthetic datasets: a statistical method and a ML efficacy method. We use these two methods to show that two-sided label smoothing and simulation to sample the final synthetic variables lead to the best performances. The most optimal loss function, on the other hand, depends on the ratio of continuous and categorical variables. Indeed, if a dataset contains primarily categorical variables, we recommend using a `WGAN` loss function. On the contrary, if the dataset contains more continuous variables than categorical variables, we recommend using a `WGGP` loss function. We then show that a complete DAG leads to better correlations in the final synthetic dataset than a stripped one. The optimal DATGAN models are then compared against state-of-the-art models. We show that DATGAN outperform all the other models in all the case studies using both assessment methods. Finally, we show how DATGAN can create hypothetical synthetic populations.

The DATGAN architecture has been developed to improve the representativity of its generated synthetic data. Such datasets can then be used in simulations. They might improve the latter's results since it has shown better results than other synthetic datasets from state-of-the-art models found in the literature. However, the DATGAN methodology can be improved on multiple aspects. The first one consists in removing the directionality aspect in the DAG. Indeed, due to the linear nature of LSTM cells, DATGAN requires the use of a directed graph. However, the causality between variables is usually unclear. Thus, an undirected graph would better represent the correlations between the variables instead of the causality. To achieve this, the architecture of the generator should be upgraded. For example, Graph Neural Networks are built based on an undirected graph. Such an architecture would fit well the essence of the DATGAN methodology. In addition, DATGAN, as every other GAN, can achieve differential privacy (Jordon et al., 2018) by the addition of Laplacian noise. Therefore, privacy preservation is generally not a concern for GANs. For the ML efficacy research axis, DATGAN is already showing improvements thanks to the ML evaluation metric results. However, DATGAN does not consider bias in the original data. Therefore, it also generates biased data. Thus, improving the bias correction of DATGAN will also improve the ML efficacy. A simple fix that already exists in the literature is the use of conditionality on GANs. We could, thus, update DATGAN such that it can conditionally generate synthetic data to reduce the bias in the data. Furthermore, one of the difficult tasks with DATGAN is to create a good DAG that does not hinder the models' performances. While the relationship between some variables might be easy to find, it is more subtle for others. Therefore, it would be interesting to add a feature to combine multiple variables in a cluster such that they all influence each other without having to decide in which specific order. However, such an improvement might not work with the current design of DATGAN using LSTM cells since each cell is assigned to a single variable. Therefore, a complete redesign of DATGAN might be needed to implement such an improvement. Thus far, we have discussed four of the five research axes in the literature review. The final axis is transfer learning. Researchers have already been working on this topic with synthetic image generators. Therefore, one of the future steps for synthetic tabular data generators is to follow this trend and start implementing models that can transfer knowledge between multiple datasets.

4 Generation of detailed synthetic populations using deep learning

This chapter is based upon the following technical report:

> Lederrey, Gael, Hillel, Tim and Bierlaire, Michel. ciDATGAN: Conditional Inputs for Tabular GANs. *arXiv:2210.02404 [cs]*, October 2022. https://arxiv.org/abs/2210.02404. arXiv: 2210.02404.

4.1 Introduction

Synthetic data presents new opportunities for modeling complex systems with recent advances in deep generative learning methodologies. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are considered the most advanced state-of-the-art models to generate such synthetic data. While there is a large body of work on generating synthetic images (Shin, 2022; Hindupur, 2022), there is still a gap in the research for generating tabular data. Multiple iterations of GANs for tabular data (Xu and Veeramachaneni, 2018; Park et al., 2018; Lederrey et al., 2022) have been proposed. However, only a few examples of conditional GANs for tabular data exists (Xu et al., 2019; Zhao et al., 2021), and they lack the advances made for GANs specialized in images. Conditionality in GANs consists in feeding latent information about the desired synthetic data to control the generation process. For example, this can be used when one wants to generate images of a specific subject or generate images matching a descriptive sentence.

However, the concept of latent information in tabular data is less obvious. For example, for socio-economic data, much of the information that would be considered latent for an image (such as the individual's age, gender, or nationality) is typically included in the data as manifest variables. In this context, the conditionality of tabular data based on high-level descriptive details can be considered more similar to tasks such as image completion, where GANs attempt to recover the masked part of an image using the information from the neighboring pixels. In this chapter, we propose a new type of conditionality based on "*conditional inputs*" that is a mix between latent conditionality and data completion. The idea is to feed some original data variables to the generator as conditional inputs and let the GAN learn the correlations with the other variables during the training process. Then, we can feed it some unknown values for the conditional inputs for the sampling. This does not break the principle of GANs since the generator never sees any original data it is trying to generate. This type of conditionality, thus, has two main use-cases:

1. Removing the bias in a dataset, *e.g.* correcting a non-representative sample in a survey data (*e.g.* older individuals in a web-based survey)
2. Combining information from two datasets, *e.g.* combining representative aggregate data (*e.g.* aggregate control totals from census) with detailed small-sample data (*e.g.* household travel surveys) to create representative and fully disaggregate synthetic populations.

The conditional inputs methodology is directly implemented in the Directed Acyclic Tabular GAN (DATGAN) (Lederrey et al., 2022). For ease of comprehension, we name this model conditional inputs DATGAN (ciDATGAN). The code for the model itself is available on Github at https://github.com/glederrey/DATGAN. In addition, a PyPI library is also available at https://pypi.org/project/datgan/. Finally, the code that has been used to generate the results for this chapter is also available on Github at https://github.com/glederrey/ciDATGAN.

4.2 Literature review

GANs (Goodfellow et al., 2014) are considered the state-of-the-art generative models for creating synthetic datasets. While the fundamental concepts remain unchanged, many different approaches have been used depending on the research context and application. For this literature review, we investigate three types of GANs: conditional GANs (Section 4.2.1), image completion GANs (Section 4.2.2), and tabular GANs (Section 4.2.3).

4.2.1 Conditional GANs

GANs are built to generate highly representative synthetic data. However, it is usually impossible to control the generation process once the model has been trained. Therefore, the concept of conditionality has been introduced to overcome this limitation. It consists in feeding latent information about the desired synthetic data, such as the label of an image or a description, to both the generator and the discriminator during the training process. Thus, once the model has been trained, this information can be fed to the generator to control its sampling procedure. In the literature, we find four main types of conditionality: (i) **Conditionality by concatenation** (Mirza and Osindero, 2014; Reed et al., 2016) which consists of concatenating a vector of conditionals with the noise input for the generator and with the input values for the discriminator; (ii) **Conditionality using an auxiliary classifier** (Odena et al., 2017) which consists of using an auxiliary neural network to classify the synthetic data and add the results to the loss function; (iii) **Conditionality with projection** (Miyato and Koyama, 2018) which consists of including a projection between the conditionality and the features extracted from the images in the discriminator, thus allowing the discriminator to compute the similarity between the conditionality and the image; and finally (iv) **Conditional Batch Normalization (CBN)** (Dumoulin et al., 2017; de Vries et al., 2017) which consists of using the conditional vectors to modulate the activation functions in the generator. Zhang et al. (2019b); Brock et al. (2019) used this idea in recent GAN architecture by replacing the Batch Normalization layers in the generator with CBN layers.

While all these methods have shown different levels of success until now, they have always been developed around the generation of synthetic images. The conditionality is defined relative to either: (i) the subject of the image; (ii) a sentence describing an image; or (iii) another image in a different style. Thus, conditionality in GANs corresponds to latent conditionality, *i.e.* it is not directly part of an image. The complement to latent conditionality in GANs is image completion.

4.2.2 Image completion GANs

Image completion GANs consist in generating part of an image that has been hidden with specific or random masks. Yeh et al. (2017) proposed a DCGAN for semantic inpainting (Pathak et al., 2016). This model aims at inferring large missing parts of an image based on its semantics. Building on this work, Nazeri et al. (2019) propose the EdgeConnect GAN that uses information about edges in the image to complete it. Zheng et al. (2019) extended the methodology to produce multiple possible completion for incomplete images. Chen et al. (2020) proposed the iGPT model based on the well-known GPT3 (Brown et al., 2020) model for language. Their model translates the idea of semantics for text to images.

4.2.3 Tabular GANs

Since tabular data fundamentally differs from images, specific GANs have been developed to generate such data. Multiple architectures have been used for tabular GANs, including those based on: (i) Long Short-Term Memory (LSTM) cells (Xu and Veeramachaneni, 2018; Lederrey et al., 2022); (ii) Fully Connected Neural Networks (FCNNs) (Xu et al., 2019); or (iii) Convolutional Neural Networks (CNNs) (Zhao et al., 2021). While these different architectures provide different strengths for each model, Lederrey et al. (2022) shows that their model, the DATGAN, outperforms the state-of-the-art models for generating representative synthetic data (without conditionality).

Conditional Tabular GAN (CTGAN) (Xu et al., 2019) introduces conditionality for tabular GANs using concatenation. While they show that the conditional vector is critical for imbalanced datasets, they do not provide specific results on the conditionality. Zhao et al. (2021) also implement conditionality in their Conditional TABular GAN (CTAB-GAN). However, they use conditionality with an auxiliary classifier and, thus, have to restrict the conditionality on a single variable. Nonetheless, they show that in the context of Machine Learning (ML) classification, such conditionality improves the performance of resulting models by a significant margin.

Both methods for conditionality in tabular GANs cited here implement conditionality based on latent variables, which has been applied in image generation. As such, they either highly restrict the use of the model (conditionality using an auxiliary classifier) or fail to generate accurate synthetic data (conditionality by concatenation). Typically for tabular data, the types of variables that are latent in images are instead included explicitly in a dataset as manifest variables. For instance, a survey will typically contain the respondent's age, gender, and socio-demographic details. As such, we identify a gap in the literature for conditionality of tabular data generation based on manifest variables, which could be used in applications such as synthetic populations and data combinations. For example, to create a large and complete synthetic population, one would need to combine information from multiple datasets, *e.g.* census data, activity timelines, and/or travel diaries. In this chapter, we address this gap by creating a methodology that allows the user to be flexible on its use of conditionality and the generation process of the model by taking inspiration from image completion methods.

4.3 Methodology

This section describes our proposed methodology for generating synthetic tabular data based on conditional inputs. First, we introduce the concept of *conditional inputs*. It takes inspiration from image completion, in contrast to latent conditionality. The overall approach is to train a model using complete data to generate synthetic observations conditional on one or more manifest variables. This type of conditionality can, thus, be used for either removing the bias in a dataset using unbiased variables as conditional inputs or combining information from two datasets using variables from a second dataset as conditional inputs for the sampling phase.

For example, datasets may be either high-detail small samples (with many variables and few rows) or low-detail large samples (with few rows but many variables). For example, in the case of population synthesis, one could have access to census data covering the whole population with high-level socio-economic characteristics and a more detailed dataset describing detailed mobility characteristics for a much smaller population sample. Thus, a model which learns how to generate the detailed mobility variables conditional on the high-level socio-economic census variables could be used to enrich the census population with these variables, thus creating a detailed synthetic dataset covering the whole of the population. We illustrate this example in Figure 4.1. The dataset with many variables and fewer rows is named the feeder data ($\mathbf{T_F}$), and the dataset with low details but a large number of rows is named the distributor dataset ($\mathbf{T_D}$). As shown in Figure 4.1, this methodology aims to learn part of the feeder dataset and generate synthetic data to complete the distributor dataset. It is done using common variables between the two datasets that we call conditional inputs.

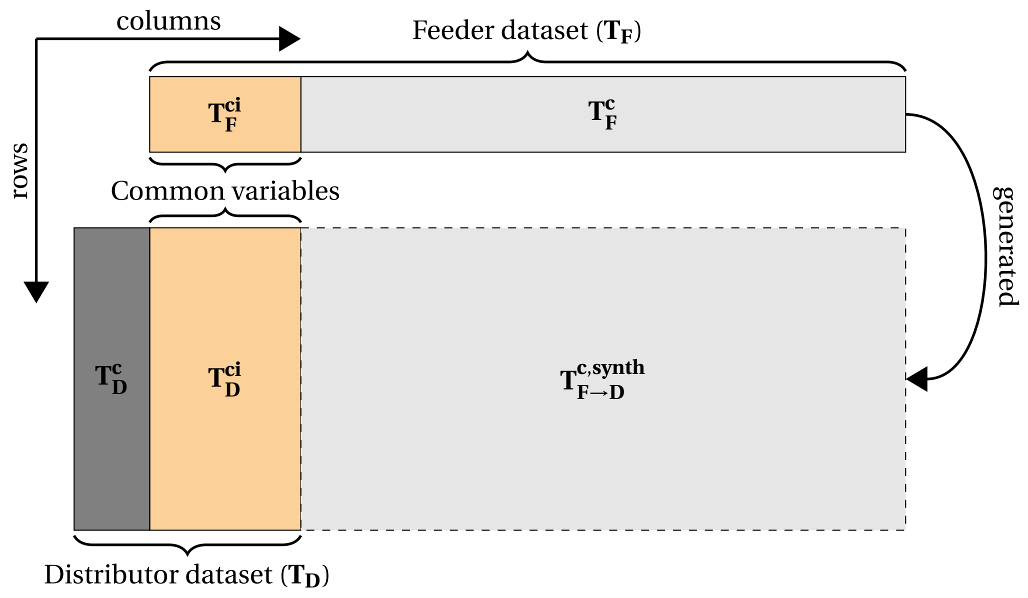


Figure 4.1: Representation of dataset completion in tabular data.

Formally, the feeder dataset $\mathbf{T_F}$ contains $N_\mathbf{F}$ variables ($\boldsymbol{v}_i^\mathbf{F}$ for $i = 1,\dots,N_\mathbf{F}$) and the distributor dataset $\mathbf{T_D}$ contains $N_\mathbf{D}$ columns with $N_\mathbf{F} > N_\mathbf{D}$. To complete the distributor dataset with the information from the feeder dataset, we ensure that they contain $N_c \leq N_\mathbf{F}$ common variables. We designate these subsets as $\mathbf{T_F^{ci}}$ for the feeder dataset and $\mathbf{T_D^{ci}}$ for the distributor dataset. These common variables are the variables used as conditional inputs. The model has to learn the logic to generate the complementary variables in the feeder dataset $\mathbf{T_F^{c}} = \mathbf{T_F} \setminus \mathbf{T_F^{ci}}$. In the sampling phase, it can complete the distributor dataset by generating the completementary variables of the feeder dataset $\mathbf{T_{F\rightarrow D}^{c,synth}}$ using the values of the common variables in the distributor dataset $\mathbf{T_D^{ci}}$ as conditional inputs. The final dataset is thus comprised of the distributor dataset $\mathbf{T_D}$ and the generated data $\mathbf{T_{F\rightarrow D}^{c,synth}}$.

4.3.1 ciDATGAN

ciDATGAN is a direct extension of DATGAN (Lederrey et al., 2022). The core methodology is the same for both models. Therefore, we give here a summary. The reader will find all the details in Chapter 3. Table B.1, in the appendix, provides a summary of the notations used in this methodology.

ciDATGAN uses LSTM cells (Hochreiter and Schmidhuber, 1997) in the generator to generate each variable, with a network structure based on a Directed Acyclic Graph (DAG) provided by the user which specifies the causal links between variables in the feeder dataset. The variables in the data are encoded at discriminator input (and generator input for the conditional inputs) and decoded at generator output, with different encoding/decoding used for continuous and categorical variables.

Figure 4.2 shows how the different elements are combined to train ciDATGAN. First, the DAG is modified based on the selected variables in $\mathbf{T}_\mathbf{F}^\mathbf{ci}$. This modified DAG is then used to form the generator structure in an automated process. As for any GAN, the generator uses Gaussian noise as input. However, in ciDATGAN, we also include as input the conditional inputs $\mathbf{T}_\mathbf{F}^\mathbf{ci}$. Ultimately, the complementary variables $\mathbf{T}_\mathbf{F}^\mathbf{c}$ are used to train the discriminator against the generated data $\mathbf{T}_\mathbf{F}^\mathbf{c,synth}$. Thus, the training process remains similar to any existing GAN.

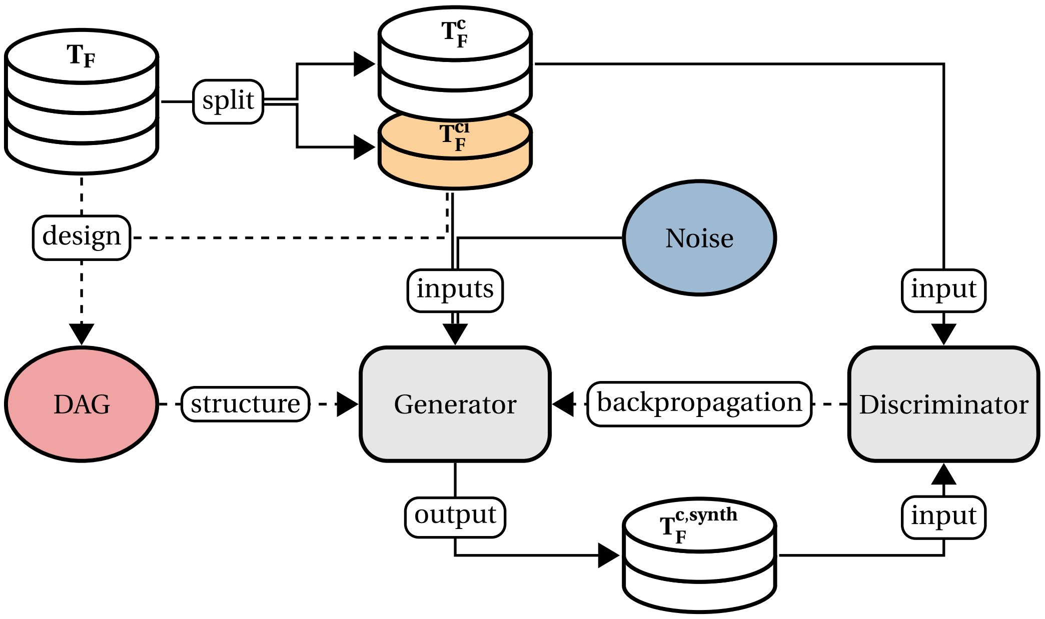


Figure 4.2: Schema of the training process of ciDATGAN.

Between DATGAN and ciDATGAN, there are two primary modifications to integrate the conditional inputs. First, the variables selected as conditional inputs must be considered as the generator's inputs. We, thus, impose each variable in $\mathbf{T}_\mathbf{F}^\mathbf{ci}$ to be a source node in the DAG. This modified DAG can be obtained through automated modification of an existing DAG.

Secondly, the conditional inputs can not be treated the same way as the generator's variables in $\mathbf{T}_\mathbf{F}^\mathbf{c}$. Indeed, these variables are generated using LSTM cells, similarly to DATGAN. We give here a summary of how these variables are generated. The details are given in the previous chapter, Section 3.3.1. The LSTM cells follow an order provided by the linearization of the DAG. The cell LSTM$_t$ is associated with the variable $\boldsymbol{v}_t^\mathbf{F}$, in which the indices are ordered based on the DAG. Each cell takes as input the cell state of the previous variable in the DAG $\boldsymbol{C}_{t-1}$ and the input tensor $\boldsymbol{i}_t$. The input tensor is a concatenation of the following tensors:

- $\boldsymbol{z}_t$: a tensor of Gaussian noise

- $\boldsymbol{f}_{t-1}$: the transformed output of the previous LSTM cell in the DAG.

- $\boldsymbol{a}_t$: the attention vector used to retain information from previous ancestors that are not directly linked to the current cell in the DAG. It is defined as:

$$\boldsymbol{a}_t = \sum_{k \in \mathscr{A}(t) \setminus \mathscr{P}(t)} \frac{\exp \alpha_k^{(t)}}{\sum_{j=1}^{|\alpha^{(t)}|} \exp \alpha_j^{(t)}} \boldsymbol{f}_k \tag{4.1}$$

  where $\mathscr{A}(t) \setminus \mathscr{P}(t)$ is the set of ancestors of the variable $\boldsymbol{v}_t^{\mathbf{F}}$ in the DAG in which we removed the direct predecessor(s), $\alpha^{(t)}$ is a learned attention weight vector, and $\boldsymbol{f}_k$ is the final output of the LSTM cell $\mathbf{LSTM_k}$.

Each cell outputs two tensors: the new cell state $\boldsymbol{C}_t$ and the output of the cell $\boldsymbol{h}_t$. This output is then passed through a set of fully connected layers to get the synthetic values $\boldsymbol{v}_t^{\mathbf{F,synth}}$. Finally, since this synthetic tensor can have different dimensions depending on the encoding of the original columns $\boldsymbol{v}_t^{\mathbf{F}}$, it is passed through an input transformer to resize it to a common size between all variables. Figure 4.3 shows the schematic representation of the generation of these synthetic variables.

We only have access to the values themselves for the variables in $\mathbf{T_F^{ci}}$. However, the generator needs to have the two following values for each variable: the transformed output $\boldsymbol{f}_{t-1}$ of the direct ancestor and the direct output $\boldsymbol{h}_k$ of all the ancestors. Therefore, the variables in $\mathbf{T_F^{ci}}$ must be transformed accordingly. First, we use the same type of input transformer to get $\boldsymbol{f}_t$. However, we cannot get access to the LSTM output $\boldsymbol{h}_t$ for the variables in the conditional inputs $\mathbf{T_F^{ci}}$ since we do not use LSTM cells for these variables. Therefore, we transform the original value $\boldsymbol{v}_t$ using a Dense layer. The parameters in this additional dense layer are learned during the training process. This allows the model to use the conditional inputs in the attention vector. Figure 4.3 shows how the variables $\boldsymbol{v}_t$ are transformed when they are considered as conditional inputs.

The model needs to receive two inputs for the sampling phase: the Gaussian noise and the conditional inputs. The conditional inputs can either come from the feeder dataset $\mathbf{T_F^{ci}}$ or from a distributor dataset $\mathbf{T_D^{ci}}$. The generator will, thus, sample the complementary set of variables $\mathbf{T_{F \to D}^{c,synth}}$ based on the conditional inputs. Finally, it combines the conditional inputs and the synthetic data to deliver the final dataset. Figure 4.4 shows a schematic representation of the sampling process.

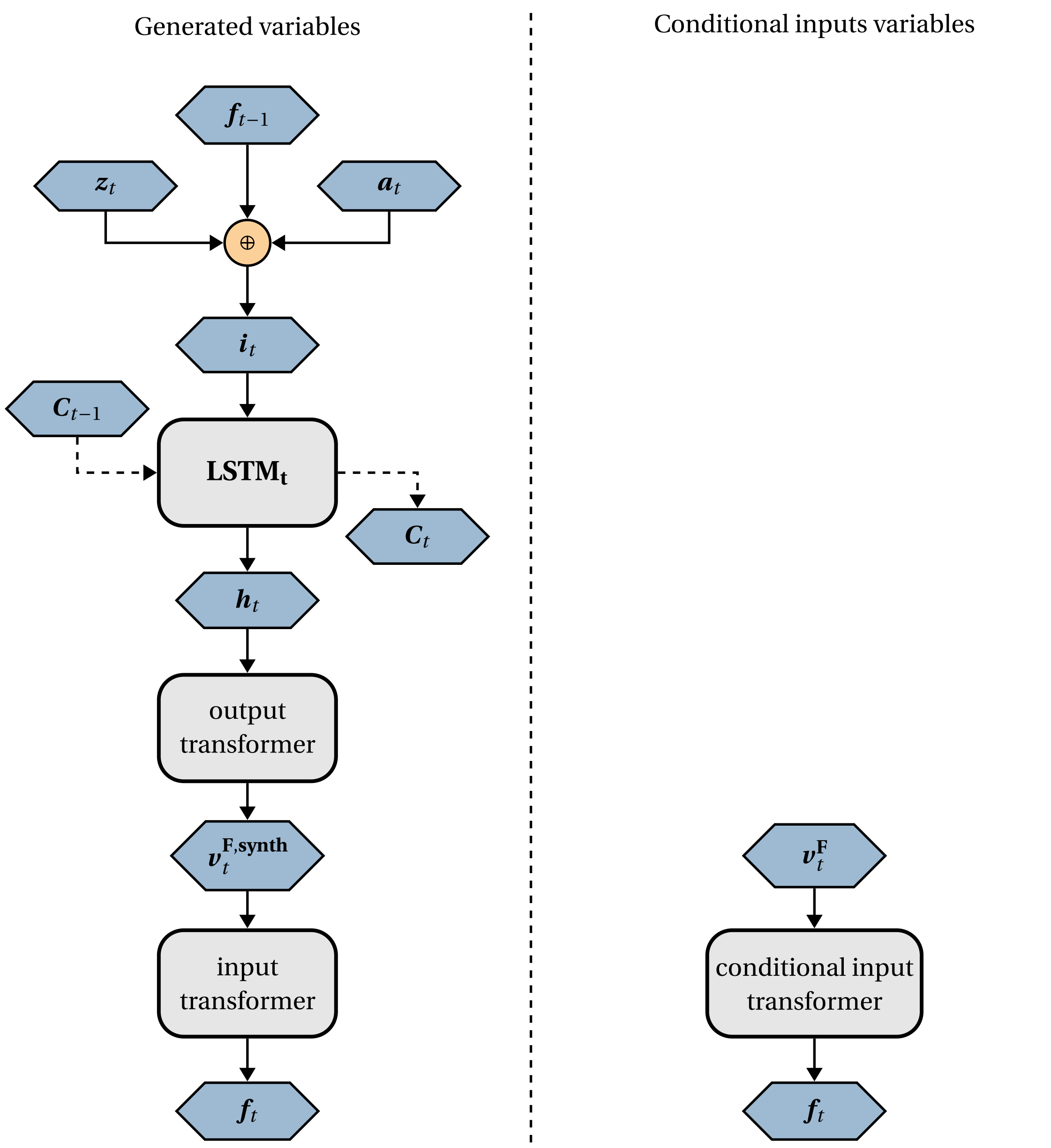


Figure 4.3: Schematic representation of the variables in the generator. On the left, we show how the variables in $\mathbf{T}^{\mathbf{c}}$ are generated using LSTM cells, as shown in Figure 3.5. On the right, we show how the variables in $\mathbf{T}^{\mathbf{ci}}$ are transformed to match the elements on the left.

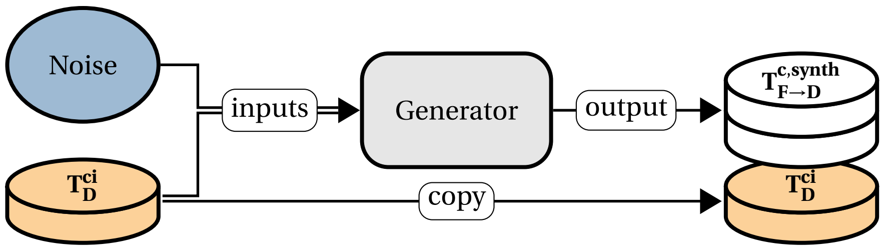


Figure 4.4: Schema of the sampling process of ciDATGAN.

4.4 Results

In this section, we investigate three questions:

- Does ciDATGAN generate data well?, *i.e.* are the generated data representative, and do they correspond to the original data?
- Can ciDATGAN correct bias?, *i.e.* can the model use unbiased conditional inputs to correct the bias in the original data?
- Can ciDATGAN be used to generate large synthetic populations?, *i.e.* can the model learn from a feeder dataset and efficiently generate a population using a large sample for the conditional inputs?

The first two questions are answered in Section 4.4.1, the third question in Section 4.4.2.

4.4.1 ciDATGAN vs DATGAN

This section answers the first two questions by comparing DATGAN and ciDATGAN. We first present the case study and the results assessments in the next section. Then, we answer the first two questions in order.

Case study and results assessments

To evaluate the performance of ciDATGAN, we use a modified version of the London Passenger Mode Choice (LPMC) dataset (Hillel et al., 2018) as the feeder dataset. A description of the variables included in this modified dataset is given in the appendix; see Table D.1. It consists of 18 variables and 16'904 trips. In the appendix, we also provide the DAG used in ciDATGAN for this dataset in Figure D.1. The conditional inputs are the variables `age`, `female`, and `hh_region`. We trained both ciDATGAN and DATGAN on the LPMC dataset five times. Then, we generated five different synthetic datasets for each of these models, thus giving 25 synthetic datasets per model. To assess the results, we use the same methodology as Lederrey et al. (2022), namely:

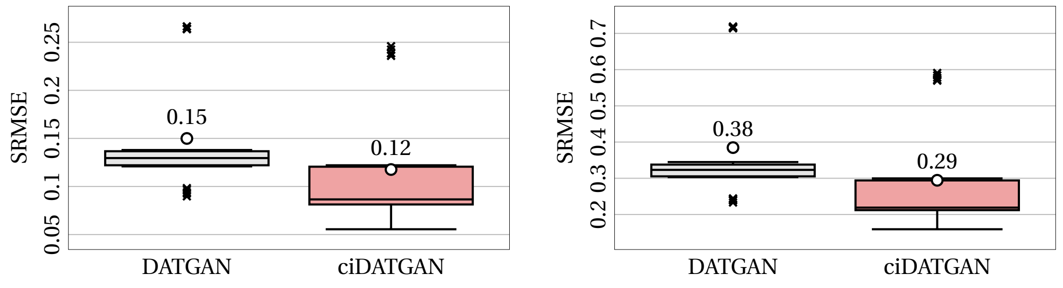
- We compute frequency lists for each variable in the synthetic dataset and compare it to similar lists computed on the original dataset.
- We compute and compare frequency lists for each combination of two variables for the synthetic and original datasets.
- Finally, we train a LightGBM (Ke et al., 2017) model on all but one column and use the resulting model to predict the corresponding column in the original data. We do this for each column. We can then compare the results between the models trained on the synthetic datasets and those trained on the original dataset.

Details of this methodology are given in the previous chapter, Section 3.3.5. For conciseness, in this chapter, we present only the results for the Standardized Root Mean Squared Error (SRMSE) (Müller and Axhausen, 2011). This metric has been used thoroughly while investigating methods to generate synthetic populations. It consists in computing the Root Mean Squared Error (RMSE) on two frequency lists and dividing the value by the average value of the frequency list for the original data. Then, for each of the 25 datasets, we compute the average value for the given metric, and we can use boxplots to show the results over multiple generated datasets.

Data generation

As shown by Lederrey et al. (2022), DATGAN achieves state-of-the-art performance when generating synthetic tabular data regarding the generated data's quality and representativity. As such, if ciDATGAN can match or better DATGAN's performance, we demonstrate a positive answer to the first research question. To test this, we apply the results assessments on all the generated variables, *i.e.* we discard the conditional inputs in the assessments since ciDATGAN uses the original values directly and would have an unfair advantage.

Results for the statistical assessments are given in Figure 4.5. Figure 4.5a shows the results when comparing each variable independently, and Figure 4.5b shows the results when comparing each combination of two variables. We can see that, in both cases, ciDATGAN performs better than DATGAN. This difference comes from two different factors: (i) ciDATGAN has fewer variables to generate, thus, making it smaller and simpler to train; and (ii) ciDATGAN has direct access to true values from the datasets, thus, making the learning process easier.

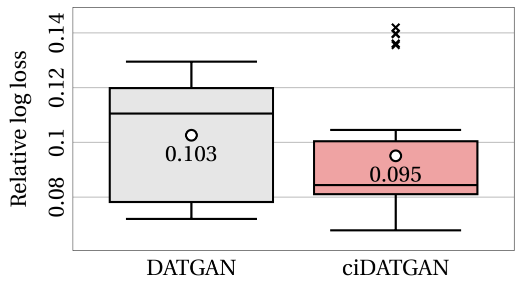


(a) Individual variables

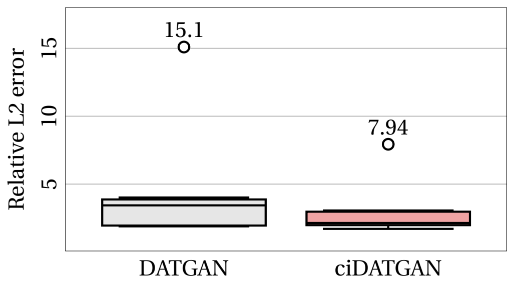
(b) Combinations of two variables

Figure 4.5: Boxplots of SRMSE values for all the generated datasets from DATGAN and ciDATGAN. The white dot corresponds to the average value. Lower is better.

Results for the ML efficacy are given in Figure 4.6. For categorical and continuous variables, ciDATGAN performs slightly better than DATGAN. This confirms the results obtained on the statistical metrics. Therefore, we can conclude that ciDATGAN better DATGAN's performances on similar datasets.



(a) Categorical variables

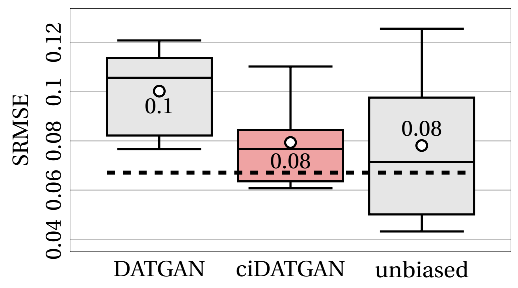


(b) Continuous variables

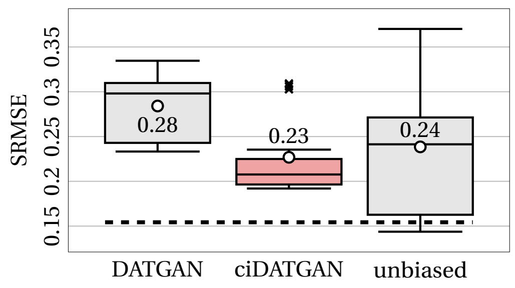
Figure 4.6: Boxplot of the ML efficacy metrics for all the generated datasets from DATGAN and ciDATGAN. The white dot corresponds to the average value. Lower is better.

Correcting bias

We have shown that ciDATGAN can generate data of equivalent or better quality to that generated by DATGAN. Thus, adding conditional inputs does not hinder the model's performance. This section investigates the use of ciDATGAN to remove bias from datasets using unbiased conditional inputs. Thus, we use the available LPMC dataset and bias it manually by removing: (i) 70% of males; (ii) 70% of individuals 20 years old and younger; and (iii) 70% of the households in the region "Central London". This gives us a biased LPMC dataset of 8'437 individuals. The goal is to train both DATGAN and ciDATGAN on this biased dataset. Then, we generate data from these models such that the final dataset has the same size as the unbiased LPMC dataset. For ciDATGAN, we use the values from the unbiased LPMC dataset as the conditional inputs for the sampling phase. Finally, we test the generated data against the unbiased LPMC dataset. The assessments are done on a subset of the LPMC variables with higher correlations to the age, gender, and household region variables.



(a) Individual variables

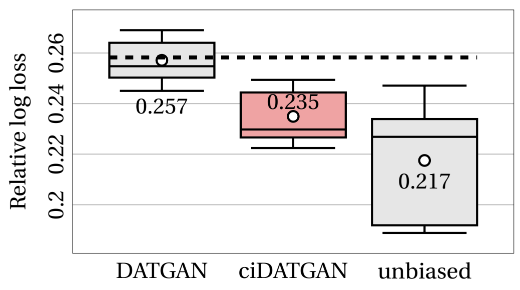


(b) Combinations of two variables

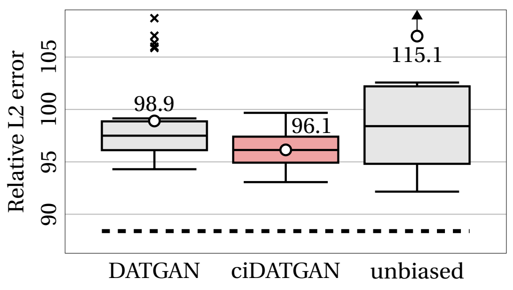
Figure 4.7: Boxplots of SRMSE values for all the generated datasets from DATGAN and ciDATGAN. The two models on the left have been trained on the biased LPMC dataset. The model on the right has been trained on the unbiased LPMC dataset. The black dashed line corresponds to the comparison between the biased and the unbiased LPMC datasets. The white dot corresponds to the average value. Lower is better.

Results for the statistical assessments are given in Figure 4.7. On average, we see that the data generated by ciDATGAN is equivalent to that generated by DATGAN trained on the unbiased LPMC dataset. Both models outperform DATGAN trained on the biased LPMC dataset as the latter cannot correct for the bias introduced in the dataset.

Results for the ML efficacy are given in Figure 4.8. For the categorical variables, we observed that ciDATGAN and DATGAN trained on the unbiased LPMC dataset are performing well (Figure 4.8a). However, this is not the case for the continuous variables (Figure 4.8b). This result is explained because it is more difficult to generate representative continuous variables.



(a) Categorical variables



(b) Continuous variables

Figure 4.8: Boxplot of the ML efficacy metrics for all the generated datasets from DATGAN and ciDATGAN. The two models on the left have been trained on the biased LPMC dataset. The model on the right has been trained on the unbiased LPMC dataset. The black dashed line corresponds to the comparison between the biased and the unbiased LPMC datasets. The white dot corresponds to the average value. Lower is better.

4.4.2 Population synthesis

We need to compare the generated synthetic data with real data to answer the final question: *Can ciDATGAN be used to generate large synthetic populations?* However, getting access to real micro-data is difficult due to privacy issues. Thus, we use aggregated data from the UK Census 2011, available on the nomis website[1] as the ground truth. Nomis is a service provided by the Office for National Statistics (ONS), UK's largest independent producer of official statistics. The feeder dataset is a modified version of the London Travel Diary Survey (LTDS) dataset (Hillel et al., 2018) that only consists of variables for the individuals and their households. We selected a total of 10 (out of the 33) boroughs such that it gives a good representation of the London population. The feeder dataset, thus, consists of 8 variables and 29'158 individuals. The complete description of the dataset is given in Table D.2 in the appendix. In addition, the DAG used for this dataset is provided in Figure D.2, in the appendix.

The distributor data has been created by drawing samples from the aggregate breakdowns of age and gender at the level of London boroughs from the nomis data. It contains a total of 2.7 million individuals for ten boroughs. Once the final synthetic dataset is generated, we aggregate the results and compare them against four overlapping variables, contained in both

[1] https://www.nomisweb.co.uk/census/2011/data_finder

the LTDS and nomis control totals at the borough level: family composition, number of people per household, number of cars/vans per household, and ethnicity.

Both the feeder and distributor datasets are defined on an individual level, *i.e.* each row of the dataset corresponds to an individual. However, three of the four selected control totals are at the household level. Thus, directly comparing the synthetic population with the aggregated distributions is impossible. In the generated population, each individual is generated with a corresponding household size and several household level statistics, *e.g.* number of vehicles in the household. To obtain unbiased aggregate household level statistics, we divide the household values by the number of household members. For example, to calculate the total number of vehicles in a borough, we would divide the number of vehicles in the household for each individual by the corresponding household size and sum over all individuals.

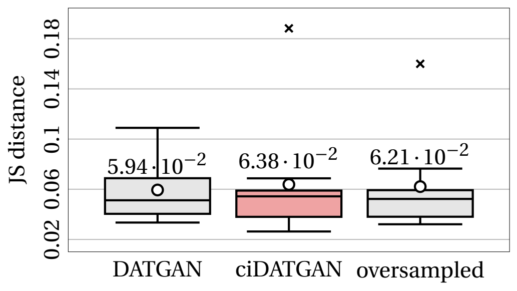
To validate the results, we compare ciDATGAN with DATGAN and an oversampled version of the LTDS datase. The latter is generated by oversampling the original LTDS dataset. For each selected borough, we oversample with replacement individuals from the same borough in the LTDS until the sample size is equal to the borough population, as recorded in the nomis data. We do not recommend doing this to augment the data since it introduces much redundancy. However, it will help understand if the synthetic data is on the same level of details and representativity as the original data in the feeder dataset. DATGAN is trained on the same feeder dataset as ciDATGAN. During the sampling phase, we randomly generate individuals until the sample size equals the borough population found in the nomis dataset. In the end, it gives us a total of three synthetic populations generated using three different methodologies such that each population reflects the nomis population for each borough.

We proposed systematic methods based on statistical assessments and ML efficacy methods to compare synthetic data in the previous chapter. However, we compare generated tabular data with aggregate data in this section. Thus, it is not possible to use the same assessment methods. However, all the overlapping variables are categorical, see Table D.2. Thus, the aggregate distributions are all discrete, and we can use the Jensen-Shannon (JS) distance (Lin, 1991) to measure the distance between these aggregate distributions. We define the JS distance, the square root of the JS divergence, between two probability arrays P and Q by:

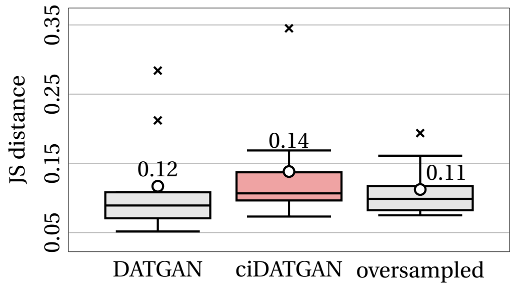
$$JS(P,Q) = \sqrt{\frac{KL(P,M) + KL(Q,M)}{2}} \tag{4.2}$$

where M is the pointwise mean of P and Q, and $KL(\cdot,\cdot)$ is the Kullback-Leibler (KL) divergence defined in Equation 3.18. The results are given in Figure 4.9.

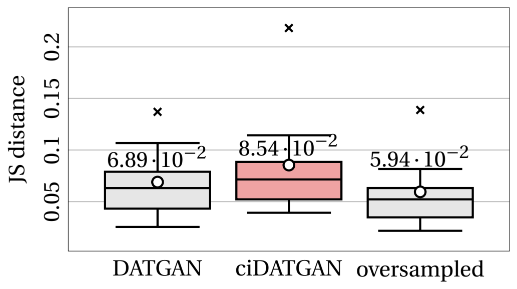
Figure 4.9 show that the three methodologies generate roughly equivalent synthetic populations compared to the aggregate control totals in the nomis data. Thus, it shows that the addition of the conditional inputs in ciDATGAN does not hinder its performance compared to DATGAN. However, it is interesting to note that the oversampled LTDS dataset generates slightly more representative data for the family composition and ethnicity.



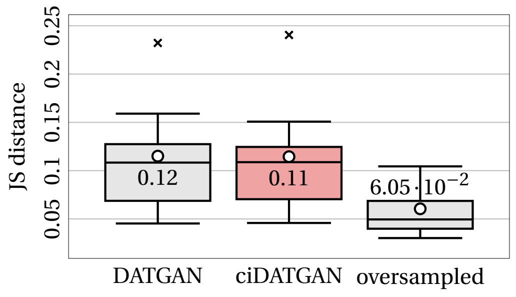
(a) Number of individuals per household



(b) Number of cars/vans per household



(c) Family composition per household



(d) Ethnicity

Figure 4.9: Boxplot of the JS distance comparing the three synthetic data methodologies against the nomis data on the four selected variables. The white dot corresponds to the average value. Lower is better.

This behavior is not surprising due to the collection process of the LTDS dataset. Indeed, it is intended to use as representative a sample as possible. Thus, LTDS is a good sample of the true population, as it is given in the nomis data. In addition, since the feeder dataset does not contain any bias, the distributor dataset follows the same distributions. Thus, it does not improve the final synthetic population generated by ciDATGAN. We show this similarity between the aggregated nomis data and the LTDS data in Figures 4.10 and 4.11.

Figure 4.10 shows the ethnicity distribution for these two datasets on an aggregated level for each selected borough. The same trends are found in both datasets. For example, we can see that Brent has fewer White people and more Asian people than the other boroughs in both datasets. Figure 4.11 shows similar distributions for the number of cars/vans per household. We see that, in both datasets, the boroughs close to the center of London (Camden and Westminster) have fewer cars in their household than the others. However, we can see that the LTDS data tends to have more households with one vehicle and fewer without vehicles. This will skew the results of the synthetic data since the models tend to generate more households with one vehicle instead of zero. However, since we are comparing the generated data against the LTDS data, both data should suffer equally from this. However, DATGAN has already shown its capabilities of generating representative synthetic data. Thus, we want to investigate how the conditionality of ciDATGAN can improve the generation process when the model is dealing with poorly designed data.

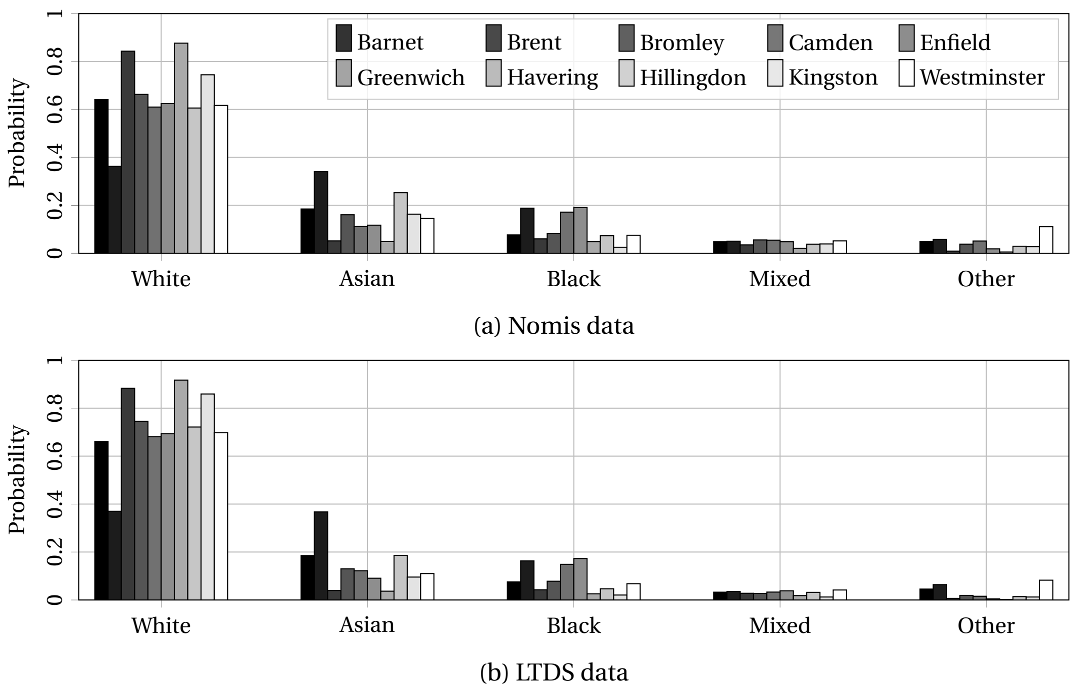


Figure 4.10: Distributions of the individuals' ethnicity for each borough.

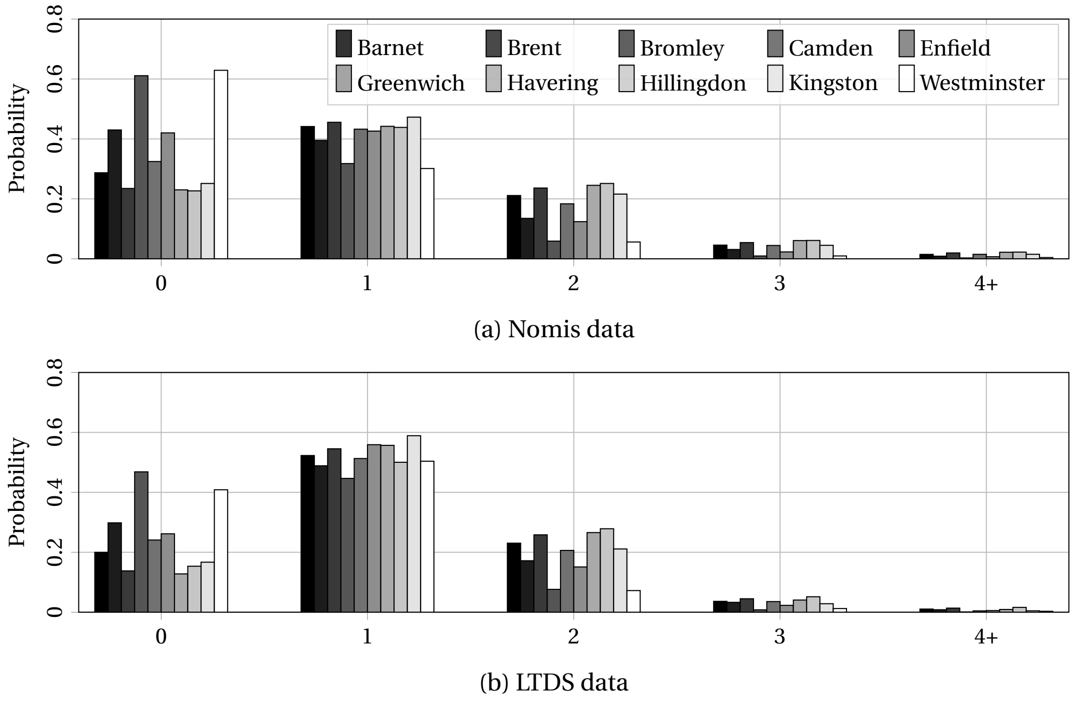
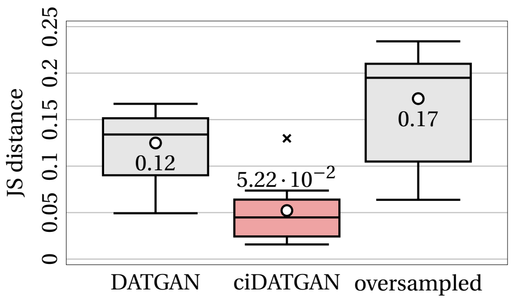
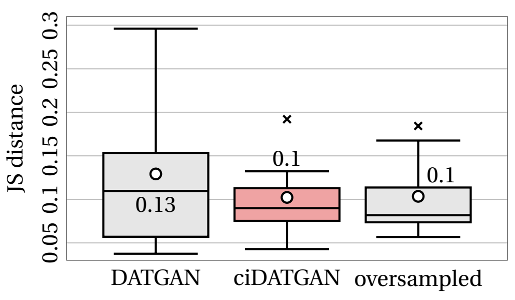


Figure 4.11: Distributions of the number of cars/vans per household for each borough.

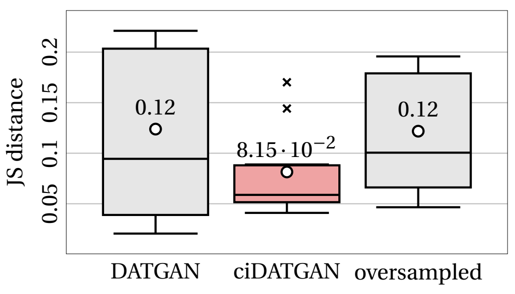
In Section 4.4.1, we have shown that ciDATGAN outperforms DATGAN when correcting for bias. Thus, we investigate if ciDATGAN provides similar results on a larger synthetic population. First, we need to bias the LTDS dataset. However, since we compare the synthetic population aggregated at the borough level, we do not bias the sample by boroughs alone. In addition, gender has a low correlation with the other variables, as shown in Figure 4.13. Thus, we bias the dataset on age. For each borough, we randomly select one of the three age categories: young (below 25 y.o.), middle (between 25 and 55 y.o.), and old (above 55 y.o.). We remove 95% of the individuals from the other age categories. The final biased LTDS dataset contains a total of 10'009 individuals. Similar to the previous experiment, we train DATGAN and ciDATGAN on this biased dataset without changing the DAG in Figure D.2 in the appendix. ciDATGAN is generating the new synthetic population using the previously defined distributor dataset, while the other two methodologies use the same sampling process. Results are given in Figure 4.12.



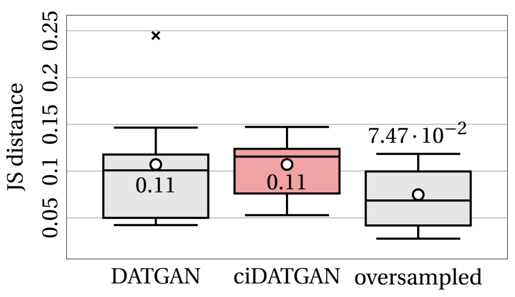
(a) Number of individuals per household



(b) Number of car/van per household



(c) Family composition per household



(d) Ethnicity

Figure 4.12: Boxplot of the JS distance comparing the three synthetic data methodologies, using the biased LTDS dataset, against the nomis data on the four selected variables. The white dot corresponds to the average value. Lower is better.

Results show that the synthetic population generated using ciDATGAN is generally more representative than the one generated using DATGAN. For the number of individuals per household and the household composition, ciDATGAN performs the best. This can be explained by the correlation between the individual's age and these two variables. Indeed, Figure 4.13 shows a high negative correlation between age and the number of individuals per household. Thus, ciDATGAN learns this stronger correlation. Then, when we use the actual age distribution for the sampling phase, ciDATGAN can generate more representative individuals. For the other two variables, ciDATGAN performs similarly to the other two methodologies. Ethnicity is not linked to age in the DAG. Thus, correcting for the age, in this case, does not influence the ethnicity. Therefore, it explains that the results for the ethnicity did not improve between DATGAN and ciDATGAN. This link has been omitted deliberately as a control correlation to study the effect of the DAG in this process. Indeed, as shown in Figure 4.13, there is a positive correlation between age and ethnicity. It makes sense since migrant populations are more likely to be of working age. Thus, adding this link in the DAG, in Figure D.2, should improve the results as shown with the other variables. Finally, we cannot see any improvements in the number of cars/vans per household. It is expected since the correlation between age and the number of cars/vans per household is weak. It has, thus, been omitted in the DAG. Since the oversampled LTDS synthetic population shows similar results, adding this link would not improve the results of ciDATGAN.

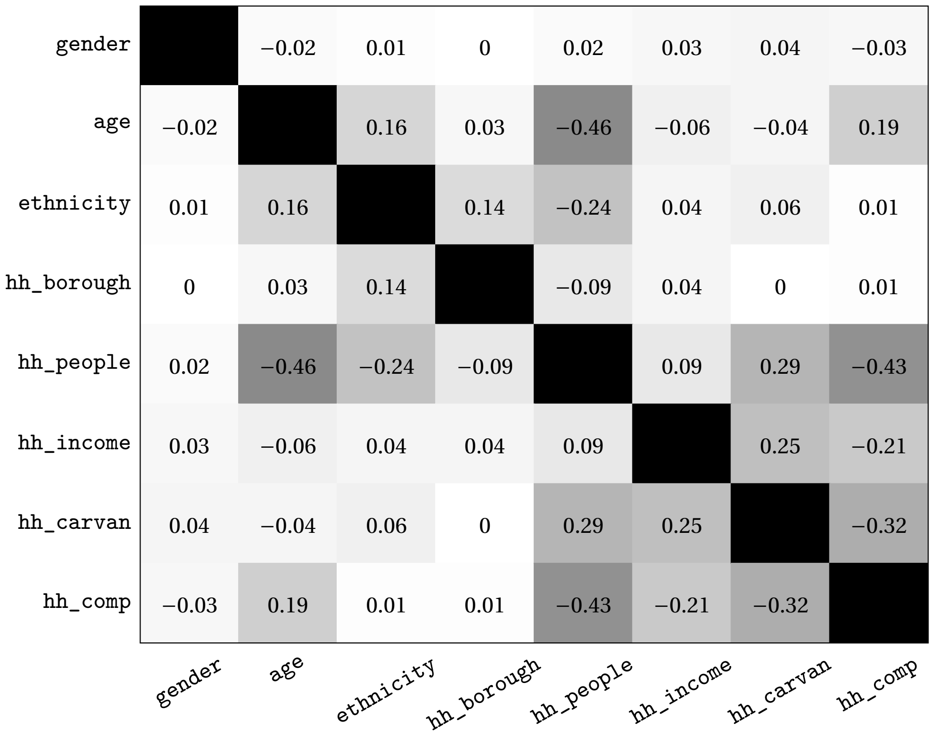
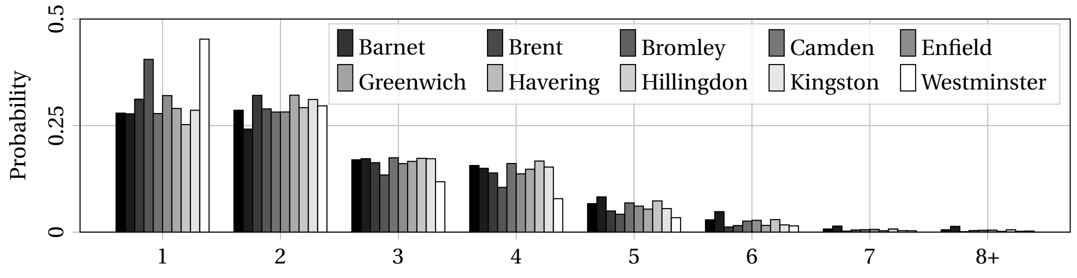
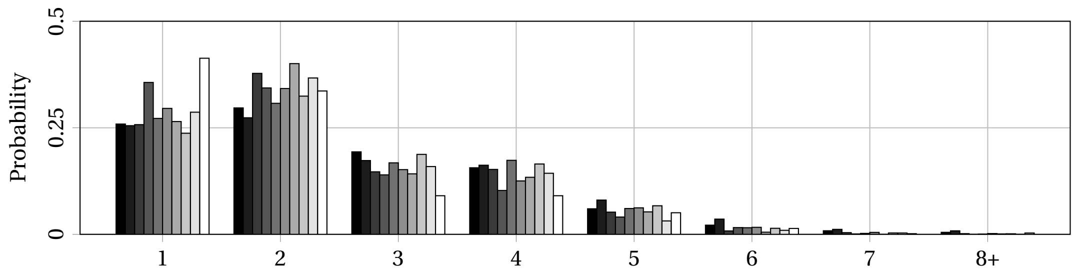


Figure 4.13: Pearson's correlation matrix of the LTDS dataset.

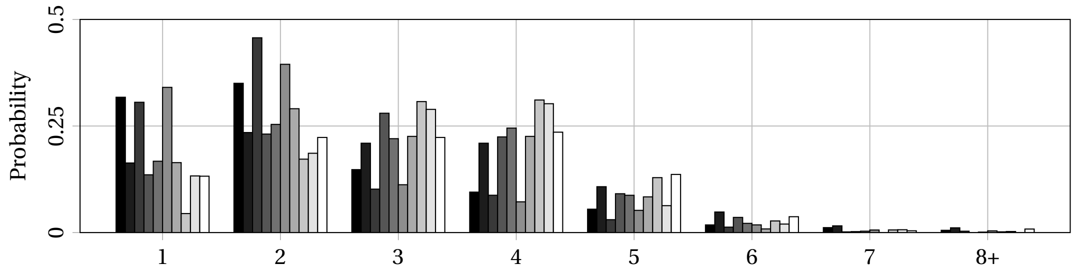
Lastly, we examine some distributions in more detail. Figure 4.14 shows the distribution of individuals per household for each dataset. We see that the LTDS data (Figure 4.14b) shows similar trends to the nomis data (Figure 4.14a). This similarity was already shown in Figures 4.10 and 4.11. On the other hand, the biased LTDS data (Figure 4.14c) shows different distributions with fewer households with a single individual. DATGAN can correct some boroughs as shown in Figure 4.14d. However, depending on the borough, it mainly assigns one or two individuals per household. Finally, Figure 4.14e shows that ciDATGAN is the best



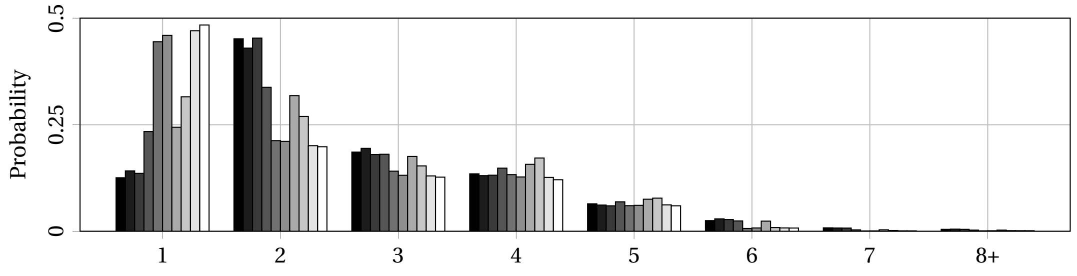
(a) Nomis data



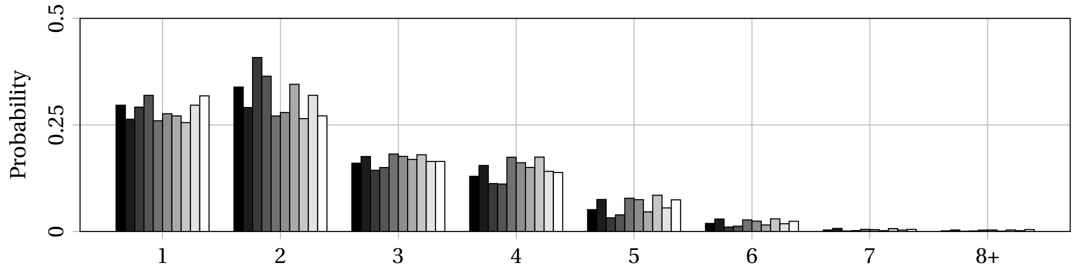
(b) LTDS data



(c) biased LTDS data



(d) DATGAN, trained on biased LTDS data



(e) ciDATGAN, trained on biased LTDS data

Figure 4.14: Distributions of the number of individuals per household for each borough.

model for correcting the bias. Despite these results, we see that ciDATGAN cannot retrieve some of the trends, *e.g.* boroughs near the center of London (Camden and Westminster) have more households with a single individual. This shows one of the limitations of ciDATGAN, *i.e.* the model can only learn the logic in the original data. Thus, it will not be able to generate data that was never seen. Nonetheless, these results indicate that ciDATGAN has been able to address some of the bias in the LTDS dataset by using aggregate borough level control totals from the UK census.

This final section shows that both DATGAN and ciDATGAN can generate large disaggregate synthetic populations. On an RTX 2080, both models were trained on the original LTDS dataset in around 25 minutes (1'000 epochs), and the sampling process took less than 10 minutes. This means that these models can generate a sizeable synthetic population in a short time. As shown in Figure 4.14, ciDATGAN can correct for possible biases in the feeder dataset using an unbiased distributor dataset. As the census data is fully aggregated at the borough level, there is no direct one-to-one correspondence between any individual in the synthetic population and the actual London population, maintaining privacy. However, the synthetic population, on an aggregate level, is representative of the London population.

Creating synthetic populations for agent-based simulations can be laborious and imprecise, limiting their practical applications. However, ciDATGAN shows that one can create closely tailored and representative populations with limited human input in a short time.

4.5 Summary

This chapter presents the ciDATGAN model, an evolution of DATGAN that uses a novel type of conditionality for tabular data inspired by image completion. It consists of completing a large dataset with few variables (distributor dataset) using synthetic data learned from a smaller dataset with more variables (feeder dataset). We test this model on a trip-based dataset against DATGAN. First, we show that ciDATGAN provides equal or better quality data than the state-of-the-art DATGAN model. Then, we show that ciDATGAN, trained on a highly biased dataset, can perform as well as DATGAN trained on an unbiased dataset. It, thus, shows that ciDATGAN can be used to unbias a dataset if one has access to unbiased values for the conditional inputs. Finally, we show that ciDATGAN can learn the logic of the feeder dataset and can, therefore, be used to generate a large, detailed, and representative dataset using an external distributor dataset.

While ciDATGAN can generate unbiased and unknown datasets if good conditional inputs are provided, many research directions are still available. For example, ciDATGAN can only be used to generate independent rows, *i.e.* cross-sectional data. A natural progression is, therefore, to allow for hierarchical data structures as in Aemmer and MacKenzie (2022), such as several individuals with correlated individual level attributes belonging to a single household with matching household level attributes. Complementary to generating hierarchical data, ciDATGAN could be upgraded to generate sequential data, such as activity patterns. However, the current architecture of the model cannot generate such data. Thus, one could combine ciDATGAN with another GAN designed to generate sequential data. For example, Badu-Marfo et al. (2020) developed a composite GAN that is composed of two different models: the first GAN generates the socioeconomic characteristics of the individuals and the second one generates sequential mobility data. However, in this case, the challenge is combining the methodology using the DAG with a new architecture that generates activity patterns.

5 Conclusion

This thesis' introduction promised disruptive and breakthrough innovations, mixing data-driven and model-driven methodologies in the era of Big Data. Thus, the three previous chapters provided such innovations: efficient estimation of complex Discrete Choice Models (DCMs) on large datasets, an efficient generative model for synthetic tabular data, and a new framework allowing for much control for the generation of synthetic population. These methodologies allow modelers to be more efficient while using Big Data technologies. In addition, they provide tools that deliver more accurate results compared to state-of-the-art methodologies. Thus, it paves the way for new and exciting research axes combining the strengths of both data-driven and model-driven methodologies.

The remainder of this conclusion, first, outlines the main findings in Section 5.1. Then, it discusses future research directions in Section 5.2. Two different topics have been selected, showing potential applications using the frameworks developed in this thesis. Finally, this thesis is concluded in Section 5.3 with some final remarks.

5.1 Main findings

The work presented in this thesis aims at bridging the gap between model-driven and data-driven methodologies in the era of Big Data. Thus, this section summarizes the obtained results in each chapter of this thesis. In addition, a small discussion is provided to discuss how each chapter contributes to bridging this gap.

Efficient estimation of complex choice models on large datasets (Chapter 2): This chapter proposes a novel algorithmic framework for efficiently estimating DCMs. Multiple algorithms, ranging from standard non-linear optimization methods to stochastic methods with adaptive batch size, have been examined in multiple use cases. The most efficient algorithm is the Hybrid Adaptive Moving Average Batch Size (HAMABS) algorithm. It consists of three distinct principles: (i) the use of second-order stochastic algorithms, (ii) the modification of the batch size when the algorithm is not improving enough, and (iii) the hybridization between

algorithms during the optimization process. Results show that the HAMABS algorithm can be up to 23 times faster compared to the current state-of-the-art software Biogeme when estimating complex DCMs on large datasets. This algorithm, thus, outperforms Biogeme manifolds without compromising the precision of the parameter values.
This chapter aims to bridge the gap between Machine Learning (ML) and choice modeling by using ML-inspired optimization method in choice modeling. The main challenge is that DCMs require high precision since the parameter values are used to understand individuals' behavior. In addition, the objective function of DCMs tends to be flat around the optimum as shown by Lederrey et al. (2018b). Thus, standard stochastic optimization algorithms used in ML fail to estimate DCMs to convergence. However, this chapter shows that data-driven methodologies can be adapted to the field's requirements, *i.e.* high precision on the parameter values for choice modeling.

Generating synthetic data from deep learning with expert knowledge (Chapter 3): This chapter presents a new Generative Adversarial Network (GAN) architecture integrating expert knowledge to generate synthetic tabular data. GANs have initially been developed to augment datasets comprised of images. However, they have quickly evolved to generate more data types. This work concentrates on tabular data since it is the most common dataset type in transportation research, *e.g.* travel survey or census microdata. In addition, the Directed Acyclic Tabular GAN (DATGAN) uses a Directed Acyclic Graph (DAG) that defines that structure of the generator. This DAG defines the causal links between the variables in the original dataset and is provided by the user. Thus, the user has to use his knowledge to define the generator's structure. DATGAN has been tested against multiple state-of-the-art synthetic tabular data generators in multiple case studies. Results show that DATGAN generates more representative synthetic data than the other selected models. In addition, using the DAG allows modelers to generate hypothetical synthetic data that would not exist in the real world to test their hypotheses.
This work is a prime example of combining the strength of data-driven and model-driven methodologies. Indeed, GANs are the current state-of-the-art deep learning methodologies to generate synthetic data. These methods have been shown to outperform standard statistical techniques, *e.g.* Iterative Proportional Fitting (IPF) or Gibbs sampling, when generating synthetic data. Indeed, they are both faster and provide more representative data. However, the main issue with deep learning methodologies is their "black-box" aspect. Usually, users only have to provide the input data to the model, and it will do the rest, removing the control over the generation process. On the other hand, this control is the strength of the statistical generative methods, as explained by Kukić and Bierlaire (2022). Thus, by allowing the user to define the causal links between the variables directly, DATGAN gives back some control to the user while preserving the strengths of GANs.

Generation of detailed synthetic populations using deep learning (Chapter 4): This last chapter extends the DATGAN methodology to provide more controls during the sampling phase of the generative model. Indeed, GANs are trained on an original dataset and then sample the new synthetic data. Conditionality is the common methodology to add such

control during the sampling phase. There are multiple ways to achieve conditionality in GANs. However, they use latent information in the data, *e.g.* label of an image, rather than explicit one, *e.g.* the dataset's variables. Thus, the conditional inputs DATGAN (ciDATGAN) takes inspiration from image completion GANs and translate it to tabular data. The idea is to train the model on a low sample, highly detailed feeder dataset. Then, the user provides a second dataset with high sample and low details, named distributor, to ciDATGAN during the sampling phase. ciDATGAN completes the distributor dataset with synthetic variables learned from the feeder dataset. This methodology only works if both datasets contain a set of common variables, named conditional inputs in this case. Results show that ciDATGAN performs similarly to DATGAN when generating representative synthetic data. However, the strength of this new architecture is to use the distributor dataset to correct for possible bias in the feeder dataset, as demonstrated in the results. Thus, ciDATGAN learn correlations in the feeder dataset and can generate any population based on the distributor dataset.
This final project offers a highly efficient tool for modelers to generate synthetic populations. Indeed, a complex synthetic population composed of millions of individuals can be generated in minutes, including the training time of ciDATGAN. This model is especially effective at generating synthetic population since the modeler can provide any distributor dataset. For example, one could train ciDATGAN on a given feeder dataset and create multiple synthetic populations with different distributor datasets. It allows modelers to generate representative synthetic populations with a high degree of control over the final results. Both model-driven and data-driven approaches are required to achieve this result. Indeed, the processing power of GANs efficiently trains the model on a large dataset. At the same time, modeling is used both to improve the representativity of the synthetic data, as done with DATGAN, and to control the sampling process.

To summarize, this thesis provides three algorithmic frameworks that combine model-driven and data-driven approaches. The different chapters show that combining both approaches leads to increased performances as long as the application requirements are respected, *e.g.* precision of the parameter values in the estimation of DCMs. Ultimately, it provides new tools to modelers, leading to new and exciting opportunities for unexplored research directions.

5.2 Future research directions

This section provides potential future research directions based on the innovations presented in this thesis. The aim is to show that the frameworks developed in this thesis can be used in different contexts and provide exciting new opportunities for researchers. This section focuses on two possible applications: assisted specification for DCMs with deep learning and transfer knowledge for synthetic populations.

Assisted Utility Specification (AUS) with deep learning consists in helping the modeler to build the utility specification of DCMs. As stated in the introduction, Ortelli et al. (2021) transformed this process into a multi-objective combinatorial optimization problem. In addition, the authors use a Variable Neighborhood Search (VNS) algorithm to generate promising sets of model specifications. This methodology requires repeatedly training a different choice model on the same dataset before converging to a suitable utility specification. A more efficient DCMs estimation would naturally benefit such work. However, this kind of methodology is limited by the sets of model specifications by the VNS algorithm. Data-driven approaches, such as deep learning, have shown great results when learning correlations in a dataset. Thus, data-driven approaches might be more efficient for AUS by generating more accurate specifications. Indeed, a suitable deep learning architecture would not have to be limited by a neighboring search space. It could analyze the current specification and the dataset to provide the best possible addition. In addition, it might even be able to provide a starting model with multiple parameters instead of a generic utility specification. However, the main issue with deep learning methodologies is their tendency to overfit. Indeed, feeding it with the same data over and over would result in a specific model only tailored for the given data. However, in choice modeling, the modelers build specific models to understand individuals' behavior while staying as generic as possible. Indeed, the goal is to understand the main trends. One can use synthetic data to train the DCMs to avoid this issue. For example, DATGAN could be used to generate representative synthetic data parallel to the AUS process. This way, the deep learning model would always receive unseen data at each iteration, reducing its tendency to overfit. Using data-driven methodologies for AUS would help modelers to define generic yet realistic choice models. For more specific applications, modelers can always use the generated choice model and improve it based on their particular needs. However, the main hurdle with this methodology is controlling that the generated utility function follows behavioral realism. Indeed, modelers follow specific rules to define the utility of DCMs. Thus, the AUS methodology has to follow the same rules. It would require additional control in the deep learning methodology, based on expert knowledge, to ensure that the generated utility functions are consistent with the theory.

Transfer knowledge for synthetic populations can be separated into two different methodologies: temporal and spatial. Temporal transfer consists in generating future populations using currently available data. This type of transfer has already been discussed in the literature. However, spatial transfer for synthetic populations can lead to new and exciting research. It consists of transferring knowledge from one population and applying it to another, *e.g.* in poorer regions of the world. Indeed, developed countries have easier access to data about their population. On the other hand, underdeveloped countries do not allocate many resources to collect such data. Thus, research is mainly done on data originating from these richer countries. The idea is to train a model on available data and transfer the model's knowledge to generate synthetic populations for these poorer countries. For example, one could train ciDATGAN on a travel survey dataset from the UK and complete a dataset from a poorer country to obtain such a synthetic population. However, this would lead to poor results since ciDATGAN would learn how UK citizens travel. For instance, wealthier countries tend to have a more developed public transportation infrastructure compared to poorer countries. Therefore, ciDATGAN would not be able to determine this from the provided data. One possible workaround would be integrating user-specified rules during the training process of ciDATGAN to enforce these differences, as done with Gibbs sampling. Another possibility is to use inspiration from the ML community. Indeed, this research problem is known as transfer learning. For example, Karimpanal and Bouffanais (2019) have developed a multi-task reinforcement learning agent to learn new tasks based on previously seen ones. This methodology has shown promising results on similar types of tasks. Thus, a reinforcement learning or a meta-learning version of ciDATGAN could use datasets from wealthier countries as the primary training material. Then, it could finalize its training process using modeling rules and smaller datasets from poorer countries. In the end, the generated synthetic population should represent the final dataset since it uses cumulative rewards. Similar approaches have already been applied to synthetic data in the field of medicine (Gu and Duan, 2022). Thus, transfer learning applied to synthetic populations is a promising and exciting research direction.

5.3 Final remarks

The thesis promised to deliver disruptive and breakthrough innovations through its different chapters. The HAMABS methodology can be considered a disruptive innovation since it is a new hybrid stochastic method used in a known environment, *i.e.* choice modeling. This innovation helps modelers become more efficient since it allows them to estimate DCMs in a shorter amount of time. However, its main benefit is to open the path for new research directions where one would need to estimate a large number of DCMs, as explained in the previous section. On the other hand, transfer learning for synthetic populations can be considered a breakthrough innovation. Indeed, by taking inspiration from both the methodologies presented in this thesis and transfer learning methodologies, one could make a scientific breakthrough in generating synthetic populations, leaning towards accessible and open data worldwide. DATGAN and ciDATGAN are the first steps in this direction using the strengths of both data-driven approaches, with the deep learning architecture, and model-driven approaches, with the addition of expert knowledge and control of the sampling process. Indeed, these new architectures for GAN have been developed to help modelers generate synthetic data and populations in an accessible and efficient manner.

In addition, the codebase developed in this thesis is openly available as ready-to-use tools, *e.g.* a Pypi library is available for DATGAN and ciDATGAN. It is hoped that this will enable researchers to explore new and exciting directions, such as assisted utility specification or transfer learning for synthetic populations.

Appendix

A Table of notations (Chapters 2)

Table A.1: Notations used in the methodology of Chapter 2.

Notation	Name	Description
Maximum likelihood estimation		
x_n	Explanatory variables	Vector of explanatory variables for a DCM.
θ	Model parameters	Vector of model parameter to be estimated. (Size: K)
$P_n(i\|x_n;\theta)$	Probability of choice	Probability that individual n chooses alternative i specified by x_n and θ.
$\mathscr{L}$	log likelihood	Objective function for DCMs, defined in Equation 2.3.
$\max \mathscr{L}(\theta)$	Maximum likelihood estimation	Objective function used to estimate θ, the parameters of the model.
$\nabla\mathscr{L}$	Gradient of the log likelihood	Gradient of the log likelihood function, defined in Equation 2.4.
$\nabla^2\mathscr{L}$	Hessian of the log likelihood	Hessian of the log likelihood function, sothcastic version defined in Equation 2.13.
Line search methods		
θ_k	Iterate k of the parameters	Iteration k of the parameters of the DCM.
α_k	Step size	Step size for the current iteration.
d_k	Descent direction	Descent direction obtained by preconditioning $\nabla\mathscr{L}$.
D_k	Preconditioning matrix	Positive definite matrix used to precondition $\nabla\mathscr{L}$.

Continues on next page...

Table A.1 – continued from previous page

Notation	Name	Description
B_k	Approximation of the Hessian	Approximation of the Hessian $\nabla^2 \mathscr{L}$ using the BFGS algorithm.
	Trust-region methods	
m_k	Quadratic approximation	Quadratic approximation of the objective function.
ρ_k	Function reduction	Function reduction ratio used to update the trust region.
Δ_k	trust region	Size of the trust region, changed in function of ρ_k.
	HAMABS	
$\text{WMA}_{k,W}$	WMA	WMA computed at the k-th iteration with a window size of W.
I_k	Progress	Progress of the optimization method computed using WMA.
$\nabla_{rel}\mathscr{L}(\theta)$	Relative gradient	Relative gradient used to compute the stopping criterion of HAMABS.
	HAMABS parameters	
W	Window size	Size of the window for the WMA.
Δ	Threshold for successful iterations	Threshold value to update the batch size when the algorithm is stalling.
C	Count	Maximum number of unsuccessful iterations with the same batch size.
τ	Expansion factor	Expansion factor when updating the batch size.
Δ_H	Threshold for hybridization	Threshold value deciding when hybridization has to happen.
ε	Stopping criterion	Threshold value for the stopping criterion.
N'_{init}	Initial batch size	Initial batch size value.
	Performance profile	
$\mathscr{A}$	Algorithms	Set of optimization algorithms.
$\mathscr{P}$	Problems	Set of optimization problems.

Continues on next page...

Table A.1 – continued from previous page

Notation	Name	Description
$t_{p,a}$	Performance measure	Performance measure (time or epochs) for algorithm a on problem p.
$\mathscr{C}_{p,a}$	Convergence test	Test of convergence for algorithm a on problem p.
$r_{p,a}$	Performance ratio	Performance ratio defined using $\mathscr{C}_{p,a}$ for algorithm a on problem p.
$\rho_a(\pi)$	Performance profile	Performance profile of algorithm a based on the factor π of the best performance ratio.
$\mathscr{R}$	Ratio's upper bound	Upper bound of the ratio $r_{p,a}$, used to constrain the performance profile values.

B Table of notations (Chapters 3 and 4)

Table B.1: Notations used in the methodology of Chapters 3 and 4.

Notation	Name	Description
	Main elements	
D	Discriminator	Neural network model used to discriminate/critic the original and synthetic data.
G	Generator	Neural network model used to generate synthetic data from random noise.
$\mathbf{T}$	Original dataset	Original dataset provided by the modeler. (size: $N_V \times N_{\text{rows}}$)
$\mathbb{P}(\mathbf{T})$	Distributions	Unknown joint distribution that the random variables V_t follow.
$\mathbf{T}_{\mathbf{synth}}$	Synthetic dataset	Final synthetic dataset after the sampling procedure; it corresponds to the final output of the generator (size: $N_V \times N_{\text{rows}}$).
$\widehat{\mathbf{T}}$	Encoded original dataset	Original dataset that has been encoded. (size: $\left(\sum_{t=1}^{N_C} 2N_{m,t} + \sum_{t=1}^{N_D} \lvert D_t \rvert\right) \times N_{\text{rows}}$)

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
$\widehat{\mathbf{T}}_{\mathbf{synth}}$	Encoded synthetic dataset	Synthetic dataset that is directly returned by the generator before the sampling step. (size: $\left(\sum_{t=1}^{N_C} 2N_{m,t} + \sum_{t=1}^{N_D} \lvert D_t \rvert\right) \times N_{\text{rows}}$)
	Main variables	
V_t	Random variable	t-th random variable.
$\boldsymbol{v}_t$	Column data in **T**	Column-vectors of data that define the table **T**. (size: N_{rows})
$v_{t,j}$	Single value in **T**	Scalar value in a column-vector.
$\boldsymbol{z}$	Noise	Random noise tensor used as an input for the Generator G. (size: $N_V \times N_{\text{rows}} \times N_{\text{sources}}$)
$\boldsymbol{v}_t^{\mathbf{synth}}$	Column data in $\mathbf{T}_{\mathbf{synth}}$	Column-vectors of data that define the table $\mathbf{T}_{\mathbf{synth}}$. (size: N_{rows})
$v_{t,j}^{\mathbf{synth}}$	Single value in $\mathbf{T}_{\mathbf{synth}}$	Scalar value in a column-vector.
$\widehat{\boldsymbol{v}}_t$	Encoded column data in **T**	Encoded version of $\boldsymbol{v}_t$.
$\widehat{v}_{t,j}$	Encoded single value in **T**	Encoded version of $v_{t,j}$.
$\widehat{\boldsymbol{v}}_t^{\mathbf{synth}}$	Encoded column data in $\mathbf{T}_{\mathbf{synth}}$	Encoded version of $\boldsymbol{v}_t^{\mathbf{synth}}$.
$\widehat{v}_{t,j}^{\mathbf{synth}}$	Encoded single value in $\mathbf{T}_{\mathbf{synth}}$	Encoded version of $v_{t,j}^{\mathbf{synth}}$.
	Directed Acyclic Graph (DAG)	
$\mathscr{G}$	DAG	DAG used to represent the interdependencies between the variables.
$\mathscr{A}(V_t)$	Ancestors	Set of all the ancestors of the variable V_t in the DAG $\mathscr{G}$.
$\mathscr{D}(V_t)$	Direct ancestors	Set of all the direct ancestors of the variable V_t in the DAG $\mathscr{G}$.
$\mathscr{S}(V_t)$	Source nodes	Set of all the source nodes leading to variable V_t in the DAG $\mathscr{G}$.

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
$\mathscr{E}(V_t)$	In-edges	Set of all in-edges of the variable V_t in the DAG $\mathscr{G}$.
	LSTM cells	
$\mathbf{LSTM_t}$	LSTM cell	LSTM cell used to generate synthetic values for the variable V_t.
$\boldsymbol{C}_t$	Cell state	Output cell state of the LSTM cell $\mathbf{LSTM_t}$. (size: $N_h \times N_b$)
$\boldsymbol{h}_t$	Output	Output tensor of the LSTM cell $\mathbf{LSTM_t}$. (size: $N_h \times N_b$)
$\boldsymbol{x}_t$	Generic input	Generic input tensor for a LSTM cell. (size: $N_x \times N_b$)
$\boldsymbol{i}_t$	Input	Input tensor of the LSTM cell $\mathbf{LSTM_t}$. (generic size: $(N_h + N_x) \times N_b$; DATGAN size: $(2N_h + N_z) \times N_b$)
$\boldsymbol{z}_t$	Noise	Random noise tensor used as an input for the LSTM cell $\mathbf{LSTM_t}$. (size: $N_z \times N_b$)
$\boldsymbol{f}_t$	Transformed output	Transformed output of the variable $\widehat{\boldsymbol{v}}_t^{\mathbf{synth}}$ to resize it according to the LSTM cell. (size: $N_h \times N_b$)
$\boldsymbol{a}_t$	Attention	Attention tensor used in the input of the LSTM cell $\mathbf{LSTM_t}$. (size: $N_h \times N_b$)
$\boldsymbol{\alpha}_t$	Attention weights	Attention weight vector used to compute a_t. (size: $\lvert\mathscr{A}(V(t))\rvert$)
$\boldsymbol{h}'_t$	Reduced output	Reduced output tensor of the LSTM cell $\mathbf{LSTM_t}$ after using a convolutional layer. (size: $N_{\text{conv}} \times N_b$)
	Discriminator	
$\boldsymbol{l}_i$	Internal layer	Internal layer used in the discriminator. (undefined size)
$\widehat{\boldsymbol{l}}_i$	Standardized internal layer	Internal layer after passing it through a fully connected layer to resize it (size: $N_L \times N_b$)
l_D	Discriminator output	Final unbounded scalar result used as an output of the discriminator.
$\boldsymbol{l}_0$	Discriminator input	Input layer used for the discriminator.
	Loss function	

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
$\mathscr{L}(D,G)$	Loss function	Generic loss function. The generator is trying to minimize it while the discriminator is trying to maximize it at the same time.
$\mathscr{L}_G$	Generator loss function	Loss function applying only to the generator, which is trying to minimize it.
$\mathscr{L}_D$	Discriminator loss function	Loss function applying only to the discriminator, which is trying to minimize it.
KL	KL divergence	KL divergence defined in Equation 3.18.
	Data processing	
$\boldsymbol{c}_t$	Continuous column data in $\mathbf{T}$	Column-vectors of continuous data that define the table $\mathbf{T}$. (size: N_{rows})
$\boldsymbol{d}_t$	Categorical column data in $\mathbf{T}$	Column-vectors of continuous data that define the table $\mathbf{T}$. (size: N_{rows})
$\boldsymbol{c}_t^{\textbf{synth}}$	Continuous column data in $\mathbf{T}_{\textbf{synth}}$	Column-vectors of continuous data that define the table $\mathbf{T}_{\textbf{synth}}$. (size: N_{rows})
$\boldsymbol{d}_t^{\textbf{synth}}$	Categorical column data in $\mathbf{T}_{\textbf{synth}}$	Column-vectors of continuous data that define the table $\mathbf{T}_{\textbf{synth}}$. (size: N_{rows})
$\boldsymbol{\eta}_t$	Means of VGM	Vector containing the mean of each component in the VGM for the continuous variable C_t. (size: $N_{m,t}$)
$\boldsymbol{\sigma}_t$	Variances of VGM	Vector containing the variance of each component in the VGM for the continuous variable C_t. (size: $N_{m,t}$)
$\boldsymbol{w}_t$	Values of encoded continuous variable	Matrix containing the values of the encoded variable $\boldsymbol{c}_t$. (size: $N_{m,t} \times N_{\text{rows}}$)
$\boldsymbol{p}_t$	Probabilities of encoded continuous variable	Matrix containing the probabilities of the encoded variable $\boldsymbol{c}_t$. (size: $N_{m,t} \times N_{\text{rows}}$)
$\boldsymbol{o}_t$	One-hot encoding of categorical variable	Matrix containing the one-hot encoding of the categorical variable $\boldsymbol{d}_t$. (size: $\lvert D_t \rvert \times N_{\text{rows}}$)
$\boldsymbol{w}_t^{\textbf{synth}}$	Generated matrix similar to $\boldsymbol{w}_t$	Output of the generator for the values of the encoded continuous variables. (size: $N_{m,t} \times N_{\text{rows}}$)

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
$\boldsymbol{p}_t^{\textbf{synth}}$	Generated matrix similar to $\boldsymbol{p}_t$	Output of the generator for the probabilities of the encoded continuous variables. (size: $N_{m,t} \times N_{\text{rows}}$)
$\boldsymbol{o}_t^{\textbf{synth}}$	Generated matrix similar to $\boldsymbol{o}_t$	Output of the generator for the one-hot encoded categorical variables. (size: $\|D_t\| \times N_{\text{rows}}$)
$\widetilde{\boldsymbol{o}}_t$	Noisy version of $\boldsymbol{o}_t$	Corresponds to $\boldsymbol{o}_t$ after adding uniform noise for the label smoothing. (size: $\|D_t\| \times N_{\text{rows}}$)
$\widetilde{\boldsymbol{o}}_t^{\textbf{synth}}$	Noisy version of $\boldsymbol{o}_t^{\textbf{synth}}$	Corresponds to $\boldsymbol{o}_t^{\textbf{synth}}$ after adding uniform noise for the label smoothing. (size: $\|D_t\| \times N_{\text{rows}}$)
Results assessments		
$\boldsymbol{\pi}$	Frequencies of original data	Frequency list computed on the original dataset $\mathbf{T}$.
$\boldsymbol{\pi}^{\textbf{synth}}$	Frequencies of synthetic data	Frequency list computed on the synthetic dataset $\mathbf{T}^{\textbf{synth}}$.
MAE	Mean Absolute Error	Mean Absolute Error computed between frequencies lists on original and synthetic datasets.
RMSE	Root Mean Square Error	Root Mean Square Error computed between frequencies lists on original and synthetic datasets.
SRMSE	Standardized Root Mean Square Error	Standardized Root Mean Square Error computed between frequencies lists on original and synthetic datasets.
R^2	Coefficient of determination	Coefficient of determination computed between frequencies lists on original and synthetic datasets.
ρ_{Pearson}	Pearson's correlation	Pearson's correlation computed between frequencies lists on original and synthetic datasets.
$\mathscr{L}_{\text{MSE}}$	Mean Square Error	Mean Square error loss used in the supervised learning validation on continuous variables.
$\mathscr{L}_{\text{log-loss}}$	Log loss error	Log loss error used in the supervised learning validation on categorical variables.
m_t	LightGBM model	LightGBM model trained on variable $\boldsymbol{v}_t$. It corresponds to a classification model ($m_{\text{class},t}$) for categorical variables, and to a regression model ($m_{\text{reg},t}$) for continuous variables.

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
g_t^{reg}	Score for the continuous variables	Final score of the supervised learning-based validation for continuous variables based on a regression model.
g_t^{class}	Score for the categorical variables	Final score of the supervised learning-based validation for categorical variables based on a classification model.
DATGAN versions		
`SGAN`	SGAN loss function	Standard two-player minimax loss function is used while optimising the DATGAN, see Equation 3.9.
`WGAN`	WGAN loss function	Wasserstein loss function is used while optimising the DATGAN, see Equation 3.12.
`WGGP`	WGGP loss function	Wasserstein loss function with a gradient penalty is used while optimising the DATGAN, see Equation 3.15.
`NO`	No label smoothing	No label smoothing applied for the discriminator input, see Equation 3.28.
`OS`	One-sided label smoothing	One-sided smoothing applied on the original data for the discriminator input, see Equation 3.29.
`TS`	Two-sided label smoothing	Two-sided smoothing applied on the original and synthetic data for the discriminator input, see Equation 3.30.
`AA`	$\arg\max$ sampling for continuous and categorical	Maxmimum probability assignment used to sample both final continuous and categorical synthetic column data.
`SA`	`simulation` sampling for continuous and $\arg\max$ sampling for categorical	Probability assignment for continuous synthetic column data and maximum probability assignment for categorical synthetic column data.

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
`AS`	arg max sampling for continuous and `simulation` sampling for categorical	Maximum probability assignment for continuous synthetic column data and probability assignment for categorical synthetic column data.
`SS`	`simulation` sampling for continuous and categorical	Probability assignement used to sample both continuous and categorical final synthetic column data.
Sizes		
N_V	#variables	Number of random variables in the original dataset **T**.
N_{rows}	#rows	Number of rows in the original dataset **T**.
N_{sources}	#sources	Number of source nodes in the DAG $\mathscr{G}$.
N_h	Size of hidden layer	Size of the hidden layers used in the LSTM cells. $(N_h = 100)$
N_x	Size of input	Size of the generic input of the LSTM cell.
N_z	Size of noise tensor	Number of elements in the noise vector. $(N_z = 200)$
N_b	Batch size	Batch size. $(N_b = 500)$
N_L	#layers in discriminator	Number of layers used in the discriminator. $(N_L = 1)$
N_l	Size of discriminator	Size of the hidden layers inside the discriminator. $(N_l = 100)$
N_C	#continuous variables	Number of continuous variables in the original dataset **T**.
N_D	#categorical variables	Number of categorical variables in the original dataset **T**.
$N_{m,t}$	#modes	Number of modes used to encode the continuous variable C_t with a VGM.
$\lvert D_t \rvert$	#categories	Number of unique categories in the categorical variable D_t.

Continues on next page...

Table B.1 – continued from previous page

Notation	Name	Description
N_{conv}	Size of convolutional layer	Size of the layer used in the output transformer to act as a convolution. Ideally, we want to have $N_{\text{conv}} < N_h$. ($N_{\text{conv}} = 50$)
	Layers in Neural Networks	
`FC`	Fully Connected	Fully connected layer without any activation function.
`LeakyReLU`	LeakyReLU	Layer with a leaky reflect linear activation function.
`BN`	Batch Normalization	Batch Normalization.
`LN`	Layer Normalization	Layer Normalization as described by Ba et al. (2016).
`div`	Mini-batch discrimination	Mini-batch discriminator vector as described by Salimans et al. (2016).
`tanh`	tanh `FC`	Fully connected layer `FC` with a tanh activation function.
`softmax`	softmax `FC`	Fully connected layer `FC` with a softmax activation function.

C Case studies (Chapter 3)

In this section, we present the case studies used in Chapter 3. For each case study, we provide a summary of the dataset, the list of variables with a description as well as the DAG used with the DATGAN model.

C.1 CMAP

The Chicago Metropolitan Agency for Planning (CMAP) dataset is a household travel survey of the Chicago metropolitan area. It was conducted between January 2007 and February 2008. The trips are given as one and two-day travel diaries, provided by all the members of the households. The original dataset has been cleaned. Then, it has been processed to contain a single trip per individual per household to remove data leakage. Thus, it contains a total of 8'929 trips with 15 variables. This dataset can be downloaded on Github at https://github.com/glederrey/DATGAN/blob/master/example/data/CMAP.csv. It is used as an example for DATGAN. The description of the variables is given in Table C.1 and the associated DAG is given in Figure C.1.

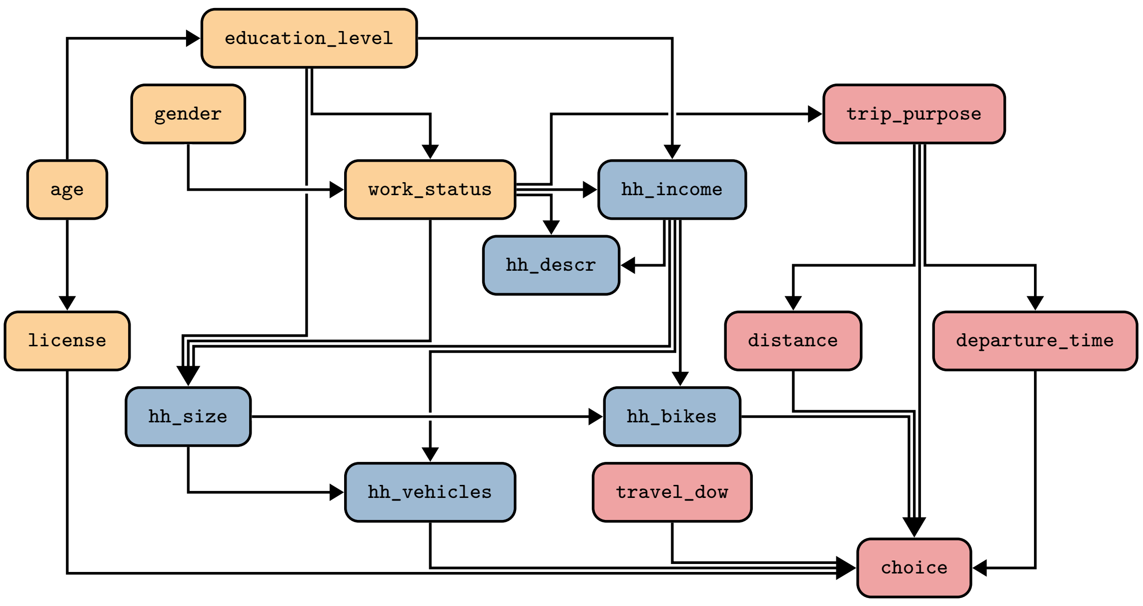


Figure C.1: DAG for the CMAP dataset. Colors correspond to the category of variables: blue for households, orange for individuals, and red for trips.

Table C.1: Details of the variables in the CMAP dataset. Colors correspond to the category of variables (in order): blue for households, orange for individuals, and red for trips.

Variables	Type	Description
hh_income	Categorical	Income of the household (7 categories)
hh_descr	Categorical	Household type (3 values)
hh_bikes	Categorical	Number of bikes in the household (8 regions)
hh_vehicles	Categorical	Number of vehicles in the household (9 values)
hh_size	Categorical	Number of individuals in the household (8 values)
age	Continuous	Age of individual in years
gender	Categorical	Gender of the individual (0=female, 1=male)
license	Categorical	Whether the traveller has a driving licence (0=no, 1=yes)
work_status	Categorical	Working status (8 categories)
education_level	Categorical	Highest level of education achieved (6 categories)
departure_time	Continuous	Departure time of trip (in decimal hours)

Continues on next page...

Table C.1 – continued from previous page

Variables	Type	Description
travel_dow	Categorical	Day of the week of travel (7 days)
distance	Continuous	Straight line trip distance
trip_purpose	Categorical	Journey purpose for trip (7 categories)
choice	Categorical	Mode of travel chosen (5 categories)

C.2 LPMC

The London Passenger Mode Choice (LPMC) (Hillel et al., 2018) dataset combines the London Travel Diary Survey (LTDS) records with matched trip trajectories and corresponding mode alternatives. The survey way conducted between April 2012 and March 2015 and records trips made by individuals residing and traveling within Greater London. The trip trajectories are extrapolated from Google Maps API. Similarly to the CMAP dataset, we selected a single trip per individual per household to avoid data leakage. The final dataset contains a total of 17'616 trips with 27 variables. The reader can contact the authors of (Hillel et al., 2018) to get access to this dataset. The description of the variables is given in Table C.2 and the associated DAG is given in Figure C.2.

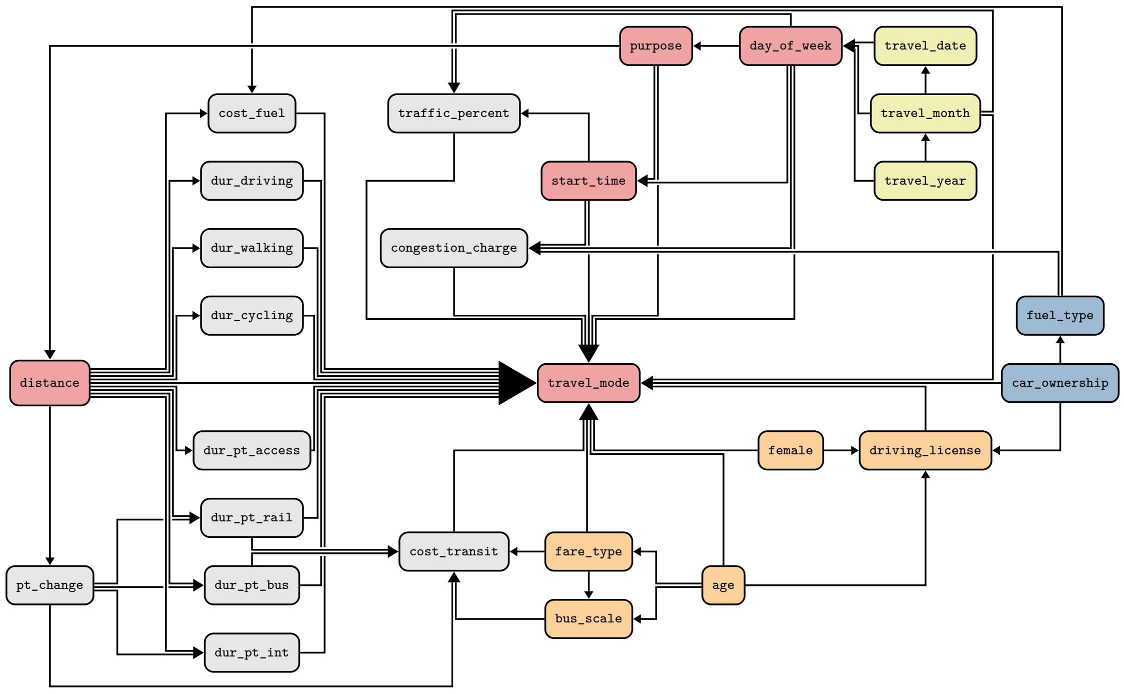


Figure C.2: DAG of the LPMC dataset. Colors correspond to the category of variables: blue for households, orange for individuals, gray for alternatives, red for trips, and yellow for survey.

Table C.2: Details of the variables in the LPMC dataset. Colors correspond to the category of variables (in order): blue for households, orange for individuals, gray for alternatives, red for trips, and yellow for survey. More details about these variables are given in Hillel et al. (2018).

Variables	Type	Description
fueltype	Categorical	Fuel type of passenger's vehicle (6 categories)
car_ownership	Categorical	Car ownership of household (3 categories)
age	Continuous	Age of individual in years
female	Categorical	Gender of the individual (0=male, 1=female)
driving_license	Categorical	Driving license ownership (0=no, 1=yes)
fare_type	Categorical	Public transport fare type of individual (5 categories)
bus_scale	Categorical	Percentage of the full bus fare paid (3 values)
dur_walking	Continuous	Duration of walking route
dur_cycling	Continuous	Duration of cycling route
dur_driving	Continuous	Duration of driving route
dur_pt_access	Continuous	Walking duration to/from first/last stop on PT route
dur_pt_bus	Continuous	Duration spent on bus services on PT route
dur_pt_rail	Continuous	Duration spent on rail services on PT route
dur_pt_int	Continuous	Total duration of public transport interchanges
pt_change	Categorical	Number of public transport interchanges (5 values)
cost_transit	Continuous	Cost of public transport route
cost_fuel	Continuous	Vehicle operation costs of driving route
congestion_charge	Categorical	Congestion charge for driving route (2 values)
traffic_percent	Continuous	Traffic variability in percentage
start_time	Continuous	Start time of trip (in decimal hours)
day_of_week	Categorical	Day of the week of travel (7 days)
distance	Continuous	Straight line trip distance
purpose	Categorical	Journey purpose for trip (5 categories)
travel_mode	Categorical	Mode of travel chosen by LTDS trip (4 categories)

Continues on next page...

Table C.2 – continued from previous page

Variables	Type	Description
travel_date	Categorical	Day of month of travel (31 values)
travel_month	Categorical	Month of travel (12 values)
travel_year	Categorical	Year of travel (4 values)

C.3 ADULT

The ADULT dataset (Kohavi, 1996), also known as the Census-Income dataset, is a well-known dataset in the Machine Learning (ML) community. It contains socio-economic variables on multiple individuals to predict if their income is below or above $50k/yr. The original dataset can be downloaded on the UCI archives at https://archive.ics.uci.edu/ml/datasets/adult. We removed all the rows with unknown values in the original dataset to obtain a total of 45'222 individuals with 14 variables. The description of the variables is given in Table C.3 and the associated DAG is given in Figure C.3.

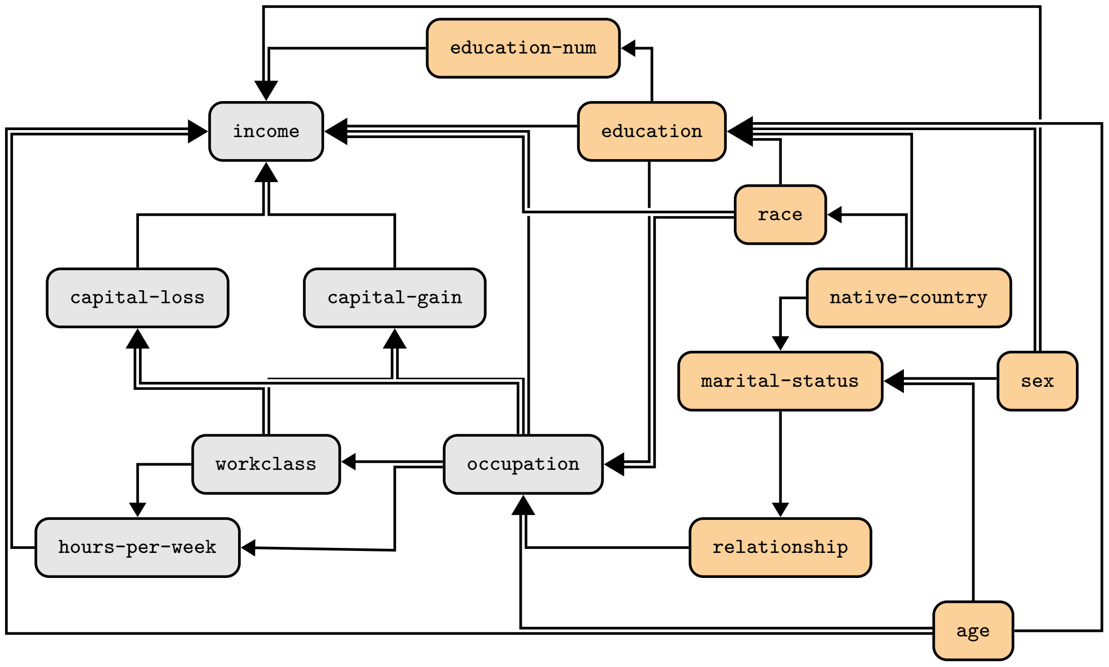


Figure C.3: DAG of the ADULT dataset. Colors correspond to the category of variables: orange for individuals and gray for occupation.

Table C.3: Details of the variables in the ADULT dataset. Colors correspond to the category of variables (in order): orange for individuals and gray for occupation.

Variables	Type	Description
`age`	Continuous	Age of individual in years
`gender`	Categorical	Gender of the individual (0=male, 1=female)
`race`	Categorical	Race of the individual (5 categories)
`native-country`	Categorical	Native country of the individual (41 categories)
`education`	Categorical	Education of the individual (16 categories)
`education-num`	Categorical	Numerical value for `education` (16 categories)
`marital-status`	Categorical	Marital status of the individual (7 categories)
`relationship`	Categorical	Relationship with a partner (6 categories)
`occupation`	Categorical	Occupation of the individual (14 categories)
`workclass`	Categorical	Work status of the individual (9 categories)
`hours-per-week`	Categorical	Hours worked per week (96 categories)
`capital-gain`	Continuous	Capital gains
`capital-loss`	Continuous	Capital losses
`income`	Categorical	Income greater or equal to 50k per year (2 categories)

D Case studies (Chapter 4)

In this section, we present the case studies used in Chapter 4. For each case study, we provide a summary of the dataset, the list of variables with a description as well as the DAG used with the conditional inputs DATGAN (ciDATGAN) model.

D.1 LPMC

The LPMC (Hillel et al., 2018) dataset used for this chapter is the same as the one presented in Section C.2 with less variables. Similarly, we selected a single trip per individual per household. Households located outside of Greater London have been removed from the dataset. It contains a total of 16'904 trips and 18 variables. The reader can contact the authors of (Hillel et al., 2018) to get access to the original LPMC dataset. The description of the variables is given in Table D.1 and the associated DAG is given in Figure D.1.

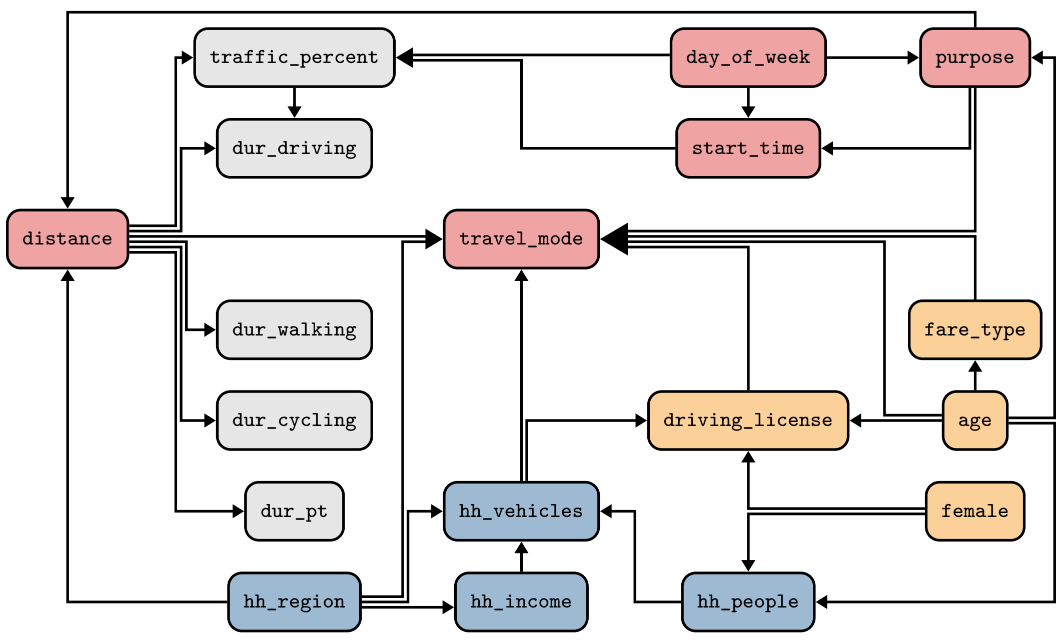


Figure D.1: DAG used for the LPMC case study. Colors correspond to the category of variables: blue for households, orange for individuals, gray for alternatives, and red for trips. The conditional inputs are the variables `age`, `gender`, and `hh_region`.

Table D.1: Details of the variables in the LPMC dataset. Colors correspond to the category of variables (in order): blue for households, orange for individuals, gray for alternatives, and red for trips. More details about these variables are given in Hillel et al. (2018).

Variables	**Type**	**Description**
`hh_income`	Categorical	Income of the household (10 categories)
`hh_people`	Categorical	Number of people in the household (11 values)
`hh_region`	Categorical	London region associated to the household (5 regions)
`hh_vehicles`	Categorical	Number of vehicles in the household (9 values)
`age`	Continuous	Age of individual in years
`female`	Categorical	Gender of the individual (0=male, 1=female)
`driving_license`	Categorical	Whether the traveller has a driving licence (0=no, 1=yes)
`fare_type`	Categorical	Public transport fare type of individual (5 categories)
`dur_walking`	Continuous	Duration of walking route
`dur_cycling`	Continuous	Duration of cycling route

Continues on next page...

Table D.1 – continued from previous page

Variables	Type	Description
dur_pt	Continuous	Duration of public transport route
dur_driving	Continuous	Duration of driving route
traffic_percent	Continuous	Traffic variability
start_time	Continuous	Start time of trip (in decimal hours)
day_of_week	Categorical	Day of the week of travel (7 days)
distance	Continuous	Straight line trip distance
purpose	Categorical	Journey purpose for trip (5 categories)
travel_mode	Categorical	Mode of travel chosen by LTDS trip (4 categories)

D.2 LTDS

The LTDS dataset contains unprocessed variables from the LPMC dataset presented in Section C.2. We only selected variables related to individuals and households. Some variables are available in the original LPMC dataset while other have been added from the collected surveys. The final dataset contains a total of 29'158 individuals and 8 variables. The description of the variables is given in Table D.2 and the associated DAG is given in Figure D.2.

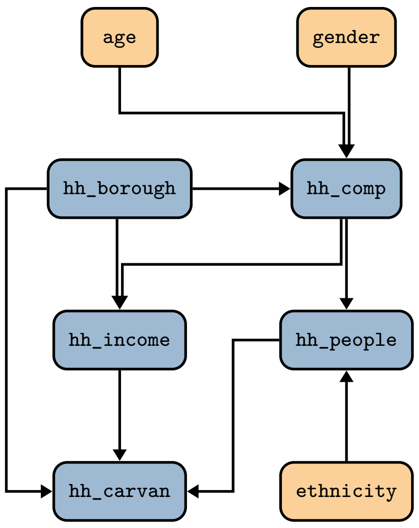


Figure D.2: DAG used for the LTDS case study. Colors correspond to the category of variables: blue for households and orange for individuals. The conditional inputs are the variables age, gender, and hh_borough.

Table D.2: Details of the variables in the LTDS dataset. Colors correspond to the category of variables (in order): blue for households and orange for individuals.

Variables	Type	Description
hh_income	Categorical	Income of the household (10 categories)
hh_people	Categorical	Number of people in the household (11 values)
hh_borough	Categorical	London borough associated to the household (10 boroughs)
hh_carvan	Categorical	Number of cars/vans in the household (8 values)
hh_comp	Categorical	Family composition in the household (4 categories)
age	Continuous	Age of individual in years
gender	Categorical	Gender of the individual (Male or Female)
ethnicity	Categorical	Ethnicity of the individual (5 categories)

E Table of results (Chapter 2)

In this section, we present the tables used for the performance profiles in Chapter 2. Section E.1 provides the tables with estimation time and Section E.2 the tables with the number of epochs.

E.1 Estimation time

The estimation time are provided in seconds. Table E.1 compares the estimation of the model LPMC_DC with the three size of datasets on all the optimization methods presented in Table 2.3. Table E.2 provides the same results for the model LPMC_RR and Table E.3 for the models LPMC_Full and MTMC.

Table E.1: Time in seconds used for the estimation of the models LPMC_DC by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	LPMC_DC_S	LPMC_DC_M	LPMC_DC_L
GD	52.16 ± 0.24	95.07 ± 0.45	140.95 ± 0.49
BFGS	6.78 ± 0.03	13.11 ± 0.06	17.96 ± 0.06
BFGS^{-1}	7.20 ± 0.02	12.96 ± 0.10	19.61 ± 0.12
TR-BFGS	5.40 ± 0.04	11.05 ± 0.11	17.17 ± 0.07

Continues on next page...

Table E.1 – continued from previous page

Algorithms	LPMC_DC_S	LPMC_DC_M	LPMC_DC_L
NM	0.65 ± 0.01	1.23 ± 0.01	1.95 ± 0.02
TR	**0.40 ± 0.01**	**0.74 ± 0.01**	**1.03 ± 0.01**
GD-ABS	50.53 ± 0.32	92.80 ± 0.51	138.20 ± 0.52
BFGS-ABS	6.46 ± 0.29	10.80 ± 0.37	15.27 ± 0.63
BFGS^{-1}-ABS	6.85 ± 0.10	11.13 ± 0.14	15.62 ± 0.16
TR-BFGS-ABS	6.90 ± 0.37	12.49 ± 0.76	21.56 ± 1.04
NM-ABS	7.64 ± 18.97	36.79 ± 52.75	116.95 ± 74.20
TR-ABS	3.20 ± 0.06	6.43 ± 0.15	11.50 ± 0.13
H-NM-ABS	2.83 ± 0.15	4.77 ± 0.25	6.97 ± 0.39
H-TR-ABS	2.16 ± 0.14	3.82 ± 0.20	6.88 ± 0.38
HAMABS	1.86 ± 0.12	3.11 ± 0.20	4.59 ± 0.32

Table E.2: Time in seconds used for the estimation of the models LPMC_RR by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	LPMC_RR_S	LPMC_RR_M	LPMC_RR_L
GD	303.21 ± 0.25	567.61 ± 0.45	820.38 ± 1.46
BFGS	183.83 ± 0.50	337.25 ± 0.30	492.90 ± 0.39
BFGS^{-1}	177.66 ± 0.25	332.93 ± 0.40	473.32 ± 0.98
TR-BFGS	227.83 ± 0.51	405.82 ± 0.60	627.43 ± 0.67
NM	36.05 ± 0.98	70.28 ± 1.50	126.85 ± 2.18
TR	23.63 ± 0.49	46.16 ± 0.66	67.13 ± 1.02
GD-ABS	310.59 ± 1.21	568.12 ± 1.74	824.15 ± 2.54
BFGS-ABS	169.83 ± 1.58	314.56 ± 3.83	474.42 ± 5.53
BFGS^{-1}-ABS	169.36 ± 0.62	312.78 ± 2.79	460.49 ± 1.92
TR-BFGS-ABS	194.49 ± 8.57	387.56 ± 12.13	590.62 ± 18.80
NM-ABS	29.72 ± 2.41	53.28 ± 3.10	76.44 ± 5.78
TR-ABS	50.64 ± 2.10	97.66 ± 4.82	175.75 ± 5.21
H-NM-ABS	29.80 ± 1.38	48.46 ± 2.50	20.78 ± 1.96
H-TR-ABS	34.97 ± 6.11	60.87 ± 13.06	157.69 ± 10.85

Continues on next page...

Table E.2 – continued from previous page

Algorithms	LPMC_RR_S	LPMC_RR_M	LPMC_RR_L
HAMABS	**11.98 ± 1.23**	**18.46 ± 1.06**	**18.14 ± 1.06**

Table E.3: Time in seconds used for the estimation of the models LPMC_Full and MTMC by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	LPMC_Full_S	LPMC_Full_M	LPMC_Full_L	MTMC
GD	2785.66 ± 28.12	5344.42 ± 33.60	8031.02 ± 49.62	7842.92 ± 53.75
BFGS	3173.86 ± 22.37	6289.47 ± 34.27	9333.66 ± 63.08	10308.49 ± 61.36
BFGS^{-1}	3129.08 ± 10.83	5929.40 ± 27.85	8812.09 ± 66.60	10090.74 ± 56.43
TR-BFGS	2043.68 ± 16.10	4014.60 ± 24.31	5861.06 ± 39.09	5568.99 ± 38.86
NM	1398.81 ± 52.16	3230.83 ± 69.75	3984.04 ± 91.00	17199.76 ± 190.18
TR	540.07 ± 24.73	1021.88 ± 38.01	1501.90 ± 53.11	10613.62 ± 152.59
GD-ABS	2782.30 ± 23.46	5508.52 ± 27.21	7989.58 ± 64.22	7984.13 ± 65.41
BFGS-ABS	3163.19 ± 57.56	6119.99 ± 109.08	8944.95 ± 130.30	10225.68 ± 59.41
BFGS^{-1}-ABS	3021.65 ± 28.72	5814.22 ± 47.75	8721.32 ± 92.05	10083.01 ± 58.33
TR-BFGS-ABS	2035.42 ± 17.94	4001.22 ± 29.88	5796.55 ± 47.00	5586.36 ± 39.60
NM-ABS	4359.60 ± 14019.82	1982.06 ± 229.38	2721.13 ± 270.25	14535.94 ± 501.67
TR-ABS	1203.69 ± 84.31	2240.94 ± 99.74	4089.42 ± 134.02	13695.15 ± 757.32
H-NM-ABS	522.65 ± 47.01	939.05 ± 85.38	1376.32 ± 93.92	2749.46 ± 103.71
H-TR-ABS	591.33 ± 80.36	1085.78 ± 137.27	2209.45 ± 160.07	7633.65 ± 386.77
HAMABS	**257.02 ± 42.85**	**405.43 ± 43.77**	**486.31 ± 63.38**	**1243.95 ± 56.21**

E.2 Number of epochs

Table E.4 compares the number of epochs used to estimate the model LPMC_DC with the three size of datasets on all the optimization methods presented in Table 2.3. Table E.5 provides the same results for the model LPMC_RR and Table E.6 for the models LPMC_Full and MTMC.

Table E.4: Number of epochs used for the estimation of the models LPMC_DC by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	LPMC_DC_S	LPMC_DC_M	LPMC_DC_L
GD	1000	1000	1000
BFGS	108	109	99
BFGS^{-1}	111	112	111
TR-BFGS	132	148	151
NM	9	10	11
TR	**8**	**8**	**8**
GD-ABS	1000.71 ± 0.09	1000.53 ± 0.04	1000.28 ± 0.02
BFGS-ABS	85.05 ± 5.75	81.91 ± 3.91	78.88 ± 4.63
BFGS^{-1}-ABS	90.02 ± 0.97	87.79 ± 1.55	85.64 ± 1.21
TR-BFGS-ABS	101.91 ± 12.76	104.98 ± 11.82	109.02 ± 11.43
NM-ABS	106.97 ± 297.92	304.96 ± 455.35	702.37 ± 455.05
TR-ABS	15.62 ± 0.34	15.91 ± 0.33	20.41 ± 0.18
H-NM-ABS	30.68 ± 2.86	30.26 ± 2.39	28.07 ± 2.40
H-TR-ABS	37.53 ± 3.55	37.54 ± 2.78	47.26 ± 3.90
HAMABS	13.79 ± 1.70	13.50 ± 2.05	12.57 ± 1.93

Table E.5: Number of epochs used for the estimation of the models LPMC_RR by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	LPMC_RR_S	LPMC_RR_M	LPMC_RR_L
GD	1000	1000	1000
BFGS	480	461	478
BFGS^{-1}	468	464	462
TR-BFGS	989	935	1000
NM	12	12	15
TR	**8**	**8**	**8**
GD-ABS	1000.62 ± 0.26	1000.57 ± 0.16	1000.29 ± 0.11

Continues on next page...

Table E.5 – continued from previous page

Algorithms	`LPMC_RR_S`	`LPMC_RR_M`	`LPMC_RR_L`
BFGS-ABS	445.97 ± 4.61	442.97 ± 6.35	441.12 ± 5.45
BFGS^{-1}-ABS	445.21 ± 1.64	439.54 ± 3.81	437.46 ± 1.80
TR-BFGS-ABS	838.64 ± 38.49	880.85 ± 28.64	937.42 ± 31.23
NM-ABS	10.13 ± 0.78	9.28 ± 0.60	9.11 ± 0.75
TR-ABS	16.77 ± 0.35	16.71 ± 0.59	20.54 ± 0.42
H-NM-ABS	67.84 ± 4.32	60.45 ± 4.39	11.96 ± 1.15
H-TR-ABS	147.98 ± 30.61	144.95 ± 36.72	218.49 ± 21.42
HAMABS	15.31 ± 2.53	13.95 ± 1.67	9.55 ± 0.56

Table E.6: Number of epochs used for the estimation of the models `LPMC_Full` and `MTMC` by all the algorithms presented in Table 2.3. The values in light gray mean that the algorithms was not able to converge in the required number of epochs. The values in bold, in a gray cell, correspond to the the fastest optimization time.

Algorithms	`LPMC_Full_S`	`LPMC_Full_M`	`LPMC_Full_L`	`MTMC`
GD	1000	1000	1000	1000
BFGS	872	901	885	1000
BFGS^{-1}	878	859	868	1000
TR-BFGS	1000	1000	1000	1000
NM	20	24	20	23
TR	$\boxed{\mathbf{7}}$	$\boxed{\mathbf{7}}$	$\boxed{\mathbf{7}}$	$\boxed{\mathbf{14}}$
GD-ABS	1000.71 ± 0.15	1000.54 ± 0.05	1000.29 ± 0.03	1000.30 ± 0.03
BFGS-ABS	877.97 ± 16.90	870.00 ± 13.58	867.96 ± 11.58	1000.32 ± 0.05
BFGS^{-1}-ABS	856.91 ± 4.67	841.77 ± 6.28	843.83 ± 3.78	1000.37 ± 0.06
TR-BFGS-ABS	1000.67 ± 0.07	1000.28 ± 0.04	1000.79 ± 0.37	1000.74 ± 0.30
NM-ABS	65.91 ± 214.46	15.21 ± 1.62	14.11 ± 1.20	20.12 ± 0.60
TR-ABS	17.41 ± 1.06	17.10 ± 0.50	21.34 ± 0.39	18.40 ± 1.03
H-NM-ABS	98.20 ± 4.60	94.75 ± 5.36	99.43 ± 4.52	145.89 ± 8.99
H-TR-ABS	219.83 ± 37.59	216.80 ± 34.95	344.72 ± 27.79	564.62 ± 81.41
HAMABS	24.98 ± 2.16	20.42 ± 1.84	21.36 ± 2.05	18.63 ± 1.58

F Table of results (Chapter 3)

In this section, we present the tables used to rank the models in Chapter 3. Section F.1 provides the results to compare the DATGAN versions, Section F.2 the results to compare DATGAN with state-of-the-art generative models, and Section F.3 the results to study the effect of the DAG on DATGAN.

F.1 Comparison of DATGAN versions

The comparison of DATGAN versions is performed on three case studies: CMAP, LPMC, and LPMC_half. For each case study, we provide six tables:

1. Statistical assessment on the first aggregation level on all the columns
2. Statistical assessment on the first aggregation level on the continuous columns
3. Statistical assessment on the first aggregation level on the categorical columns
4. Statistical assessment on the second aggregation level
5. Statistical assessment on the third aggregation level
6. ML efficacy metrics

CMAP case study

Table F.1: Results of the statistics on the first aggregation level (all columns) for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	12	7.52e − 3	12	9.31e − 3	17	9.80e − 1	15	6.63e − 2	24	9.92e − 1	16.0
SGAN_NO_AS	**06**	6.50e − 3	**06**	8.19e − 3	15	9.82e − 1	**09**	6.19e − 2	23	9.92e − 1	11.8
SGAN_NO_SA	**07**	6.70e − 3	**07**	8.55e − 3	**09**	9.85e − 1	**07**	6.15e − 2	17	9.94e − 1	**9.4**
SGAN_NO_SS	**04**	6.13e − 3	**05**	7.80e − 3	11	9.85e − 1	**05**	5.76e − 2	13	9.94e − 1	**7.6**
SGAN_OS_AA	24	9.80e − 3	24	1.22e − 2	20	9.73e − 1	24	8.31e − 2	21	9.93e − 1	22.6
SGAN_OS_AS	34	3.36e − 2	33	4.06e − 2	27	8.75e − 1	34	2.70e − 1	22	9.93e − 1	30.0
SGAN_OS_SA	23	9.70e − 3	23	1.21e − 2	19	9.74e − 1	23	8.28e − 2	16	9.94e − 1	20.8
SGAN_OS_SS	32	3.32e − 2	31	4.04e − 2	25	8.78e − 1	33	2.68e − 1	**10**	9.94e − 1	26.2
SGAN_TS_AA	11	7.19e − 3	11	8.92e − 3	**06**	9.86e − 1	12	6.32e − 2	**08**	9.95e − 1	**9.6**
SGAN_TS_AS	**09**	6.94e − 3	**08**	8.59e − 3	**03**	9.87e − 1	**06**	6.08e − 2	**07**	9.95e − 1	**6.6**
SGAN_TS_SA	**08**	6.94e − 3	**09**	8.72e − 3	**04**	9.87e − 1	**10**	6.22e − 2	**06**	9.95e − 1	**7.4**

Continues on next page...

Table F.1 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_TS_SS	**10**	7.02e − 3	**10**	8.79e − 3	**05**	9.87e − 1	**08**	6.17e − 2	**05**	9.95e − 1	**7.6**
WGAN_NO_AA	22	9.56e − 3	22	1.18e − 2	18	9.80e − 1	18	7.90e − 2	20	9.93e − 1	20.0
WGAN_NO_AS	**01**	5.53e − 3	**01**	7.07e − 3	12	9.85e − 1	**04**	5.36e − 2	19	9.93e − 1	**7.4**
WGAN_NO_SA	21	9.55e − 3	21	1.17e − 2	16	9.81e − 1	17	7.87e − 2	15	9.94e − 1	18.0
WGAN_NO_SS	**02**	5.65e − 3	**02**	7.09e − 3	**07**	9.85e − 1	**02**	5.29e − 2	14	9.94e − 1	**5.4**
WGAN_OS_AA	13	7.58e − 3	14	9.48e − 3	14	9.83e − 1	14	6.58e − 2	12	9.94e − 1	13.4
WGAN_OS_AS	31	3.32e − 2	32	4.04e − 2	26	8.75e − 1	31	2.66e − 1	18	9.93e − 1	27.6
WGAN_OS_SA	15	7.64e − 3	15	9.51e − 3	13	9.83e − 1	16	6.67e − 2	**09**	9.94e − 1	13.6
WGAN_OS_SS	33	3.35e − 2	34	4.07e − 2	28	8.75e − 1	32	2.67e − 1	11	9.94e − 1	27.6
WGAN_TS_AA	16	7.75e − 3	16	9.58e − 3	**10**	9.85e − 1	13	6.41e − 2	**04**	9.96e − 1	11.8
WGAN_TS_AS	**03**	6.11e − 3	**03**	7.58e − 3	**02**	9.87e − 1	**01**	5.23e − 2	**02**	9.96e − 1	**2.2**
WGAN_TS_SA	14	7.63e − 3	13	9.34e − 3	**08**	9.85e − 1	11	6.23e − 2	**01**	9.96e − 1	**9.4**
WGAN_TS_SS	**05**	6.33e − 3	**04**	7.76e − 3	**01**	9.88e − 1	**03**	5.32e − 2	**03**	9.96e − 1	**3.2**
WGGP_NO_AA	20	8.76e − 3	20	1.10e − 2	23	9.65e − 1	22	8.26e − 2	28	9.86e − 1	22.6
WGGP_NO_AS	18	8.28e − 3	18	1.06e − 2	21	9.66e − 1	20	8.06e − 2	25	9.87e − 1	20.4
WGGP_NO_SA	19	8.65e − 3	19	1.08e − 2	22	9.66e − 1	21	8.14e − 2	27	9.86e − 1	21.6
WGGP_NO_SS	17	8.26e − 3	17	1.05e − 2	24	9.65e − 1	19	7.96e − 2	26	9.87e − 1	20.6
WGGP_OS_AA	30	2.65e − 2	30	3.45e − 2	34	6.03e − 1	30	2.52e − 1	36	9.61e − 1	32.0
WGGP_OS_AS	36	3.85e − 2	36	4.84e − 2	36	5.85e − 1	36	3.49e − 1	35	9.63e − 1	35.8
WGGP_OS_SA	29	2.49e − 2	29	3.17e − 2	29	7.79e − 1	29	2.25e − 1	34	9.68e − 1	30.0
WGGP_OS_SS	35	3.72e − 2	35	4.59e − 2	30	7.58e − 1	35	3.24e − 1	33	9.71e − 1	33.6
WGGP_TS_AA	28	1.71e − 2	28	2.05e − 2	35	5.97e − 1	28	1.35e − 1	31	9.79e − 1	30.0
WGGP_TS_AS	26	1.54e − 2	26	1.89e − 2	33	6.16e − 1	27	1.26e − 1	32	9.79e − 1	28.8
WGGP_TS_SA	27	1.62e − 2	27	1.95e − 2	32	6.20e − 1	26	1.25e − 1	29	9.81e − 1	28.2
WGGP_TS_SS	25	1.47e − 2	25	1.78e − 2	31	6.34e − 1	25	1.15e − 1	30	9.80e − 1	27.2

Table F.2: Results of the statistics on the first aggregation level (continuous columns) for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	24	1.23e − 2	24	1.61e − 2	24	9.31e − 1	24	1.61e − 1	24	9.67e − 1	24.0
SGAN_NO_AS	23	1.21e − 2	23	1.61e − 2	23	9.31e − 1	23	1.61e − 1	23	9.67e − 1	23.0
SGAN_NO_SA	17	1.02e − 2	17	1.43e − 2	14	9.47e − 1	17	1.43e − 1	18	9.75e − 1	16.6
SGAN_NO_SS	18	1.03e − 2	18	1.43e − 2	15	9.47e − 1	18	1.43e − 1	16	9.76e − 1	17.0
SGAN_OS_AA	21	1.16e − 2	21	1.51e − 2	21	9.37e − 1	21	1.51e − 1	15	9.76e − 1	19.8

Continues on next page...

Table F.2 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_OS_AS	22	1.18e−2	22	1.55e−2	22	9.36e−1	22	1.55e−1	19	9.75e−1	21.4
SGAN_OS_SA	20	1.09e−2	20	1.47e−2	18	9.45e−1	20	1.47e−1	**08**	9.83e−1	17.2
SGAN_OS_SS	19	1.07e−2	19	1.45e−2	16	9.47e−1	19	1.45e−1	**07**	9.83e−1	16.0
SGAN_TS_AA	12	9.59e−3	**07**	1.25e−2	**08**	9.60e−1	**07**	1.25e−1	**10**	9.82e−1	**8.8**
SGAN_TS_AS	**09**	9.44e−3	**05**	1.24e−2	**07**	9.61e−1	**05**	1.24e−1	**09**	9.82e−1	**7.0**
SGAN_TS_SA	**08**	9.42e−3	**08**	1.27e−2	**06**	9.61e−1	**08**	1.27e−1	**06**	9.83e−1	**7.2**
SGAN_TS_SS	**07**	9.24e−3	**06**	1.24e−2	**05**	9.62e−1	**06**	1.24e−1	**05**	9.84e−1	**5.8**
WGAN_NO_AA	13	9.86e−3	16	1.37e−2	20	9.39e−1	16	1.37e−1	22	9.70e−1	17.4
WGAN_NO_AS	14	9.91e−3	15	1.37e−2	19	9.40e−1	15	1.37e−1	21	9.70e−1	16.8
WGAN_NO_SA	15	9.93e−3	13	1.34e−2	13	9.47e−1	13	1.34e−1	17	9.75e−1	14.2
WGAN_NO_SS	16	1.00e−2	14	1.36e−2	17	9.46e−1	14	1.36e−1	20	9.75e−1	16.2
WGAN_OS_AA	**06**	8.91e−3	**09**	1.27e−2	11	9.56e−1	**09**	1.27e−1	13	9.79e−1	**9.6**
WGAN_OS_AS	**05**	8.89e−3	**10**	1.28e−2	12	9.55e−1	**10**	1.28e−1	14	9.79e−1	10.2
WGAN_OS_SA	11	9.46e−3	12	1.30e−2	**10**	9.58e−1	12	1.30e−1	12	9.82e−1	11.4
WGAN_OS_SS	**10**	9.44e−3	11	1.29e−2	**09**	9.59e−1	11	1.29e−1	11	9.82e−1	10.4
WGAN_TS_AA	**02**	7.58e−3	**03**	1.05e−2	**04**	9.69e−1	**03**	1.05e−1	**04**	9.85e−1	**3.2**
WGAN_TS_AS	**01**	7.48e−3	**01**	1.04e−2	**01**	9.71e−1	**01**	1.04e−1	**02**	9.86e−1	**1.2**
WGAN_TS_SA	**03**	7.85e−3	**02**	1.04e−2	**02**	9.71e−1	**02**	1.04e−1	**01**	9.86e−1	**2.0**
WGAN_TS_SS	**04**	7.98e−3	**04**	1.05e−2	**03**	9.70e−1	**04**	1.05e−1	**03**	9.86e−1	**3.6**
WGGP_NO_AA	28	1.64e−2	28	2.21e−2	28	8.58e−1	28	2.21e−1	28	9.39e−1	28.0
WGGP_NO_AS	27	1.63e−2	27	2.21e−2	25	8.61e−1	27	2.21e−1	26	9.40e−1	26.4
WGGP_NO_SA	26	1.61e−2	26	2.14e−2	26	8.59e−1	26	2.14e−1	25	9.40e−1	25.8
WGGP_NO_SS	25	1.61e−2	25	2.13e−2	27	8.58e−1	25	2.13e−1	27	9.40e−1	25.8
WGGP_OS_AA	36	3.67e−2	36	5.46e−2	36	−5.05e−1	36	5.46e−1	36	8.31e−1	36.0
WGGP_OS_AS	35	3.67e−2	35	5.45e−2	35	−5.01e−1	35	5.45e−1	35	8.32e−1	35.0
WGGP_OS_SA	34	3.01e−2	34	4.18e−2	34	3.68e−1	34	4.18e−1	34	8.70e−1	34.0
WGGP_OS_SS	33	2.99e−2	33	4.15e−2	33	3.74e−1	33	4.15e−1	33	8.72e−1	33.0
WGGP_TS_AA	32	2.40e−2	32	3.32e−2	31	7.38e−1	32	3.32e−1	31	9.12e−1	31.6
WGGP_TS_AS	31	2.39e−2	31	3.31e−2	32	7.37e−1	31	3.31e−1	32	9.12e−1	31.4
WGGP_TS_SA	30	2.02e−2	30	2.81e−2	30	8.12e−1	30	2.81e−1	29	9.22e−1	29.8
WGGP_TS_SS	29	2.02e−2	29	2.80e−2	29	8.12e−1	29	2.80e−1	30	9.22e−1	29.2

Table F.3: Results of the statistics on the first aggregation level (categorical columns) for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	12	6.33e − 3	**08**	7.60e − 3	11	9.92e − 1	**08**	4.25e − 2	11	9.98e − 1	**10.0**
SGAN_NO_AS	**03**	5.09e − 3	**04**	6.23e − 3	**04**	9.95e − 1	**04**	3.73e − 2	**05**	9.99e − 1	**4.0**
SGAN_NO_SA	**06**	5.83e − 3	**07**	7.12e − 3	**03**	9.95e − 1	**07**	4.12e − 2	**07**	9.99e − 1	**6.0**
SGAN_NO_SS	**04**	5.10e − 3	**03**	6.17e − 3	**05**	9.94e − 1	**03**	3.62e − 2	**04**	9.99e − 1	**3.8**
SGAN_OS_AA	21	9.36e − 3	23	1.15e − 2	23	9.82e − 1	23	6.63e − 2	26	9.97e − 1	23.2
SGAN_OS_AS	33	3.90e − 2	33	4.69e − 2	28	8.60e − 1	32	2.99e − 1	24	9.97e − 1	30.0
SGAN_OS_SA	22	9.40e − 3	24	1.15e − 2	24	9.82e − 1	24	6.68e − 2	28	9.97e − 1	24.4
SGAN_OS_SS	31	3.89e − 2	31	4.69e − 2	27	8.61e − 1	31	2.98e − 1	25	9.97e − 1	29.0
SGAN_TS_AA	14	6.59e − 3	14	8.03e − 3	**09**	9.93e − 1	14	4.78e − 2	20	9.98e − 1	14.2
SGAN_TS_AS	**10**	6.32e − 3	**09**	7.65e − 3	**06**	9.94e − 1	**09**	4.50e − 2	14	9.98e − 1	**9.6**
SGAN_TS_SA	11	6.32e − 3	**10**	7.73e − 3	**07**	9.93e − 1	11	4.60e − 2	18	9.98e − 1	11.4
SGAN_TS_SS	13	6.47e − 3	13	7.88e − 3	**08**	9.93e − 1	12	4.60e − 2	17	9.98e − 1	12.6
WGAN_NO_AA	24	9.48e − 3	21	1.13e − 2	18	9.90e − 1	21	6.46e − 2	**03**	9.99e − 1	17.4
WGAN_NO_AS	**01**	4.43e − 3	**01**	5.42e − 3	**01**	9.96e − 1	**02**	3.28e − 2	**02**	9.99e − 1	**1.4**
WGAN_NO_SA	23	9.45e − 3	22	1.13e − 2	20	9.89e − 1	22	6.49e − 2	**06**	9.99e − 1	18.6
WGAN_NO_SS	**02**	4.56e − 3	**02**	5.46e − 3	**02**	9.95e − 1	**01**	3.22e − 2	**01**	9.99e − 1	**1.6**
WGAN_OS_AA	18	7.25e − 3	18	8.67e − 3	17	9.90e − 1	17	5.04e − 2	21	9.98e − 1	18.2
WGAN_OS_AS	35	3.93e − 2	35	4.74e − 2	30	8.56e − 1	33	3.00e − 1	27	9.97e − 1	32.0
WGAN_OS_SA	17	7.19e − 3	17	8.64e − 3	19	9.90e − 1	18	5.08e − 2	22	9.98e − 1	18.6
WGAN_OS_SS	36	3.95e − 2	36	4.76e − 2	32	8.54e − 1	36	3.02e − 1	23	9.97e − 1	32.6
WGAN_TS_AA	20	7.79e − 3	20	9.34e − 3	21	9.89e − 1	20	5.37e − 2	12	9.98e − 1	18.6
WGAN_TS_AS	**05**	5.77e − 3	**05**	6.89e − 3	16	9.91e − 1	**05**	3.95e − 2	**09**	9.98e − 1	**8.0**
WGAN_TS_SA	19	7.57e − 3	19	9.08e − 3	22	9.89e − 1	19	5.19e − 2	**10**	9.98e − 1	17.8
WGAN_TS_SS	**07**	5.92e − 3	**06**	7.06e − 3	13	9.92e − 1	**06**	4.01e − 2	**08**	9.98e − 1	**8.0**
WGGP_NO_AA	16	6.84e − 3	15	8.19e − 3	14	9.92e − 1	15	4.80e − 2	16	9.98e − 1	15.2
WGGP_NO_AS	**08**	6.27e − 3	11	7.74e − 3	**10**	9.92e − 1	**10**	4.55e − 2	13	9.98e − 1	10.4
WGGP_NO_SA	15	6.78e − 3	16	8.19e − 3	12	9.92e − 1	16	4.84e − 2	19	9.98e − 1	15.6
WGGP_NO_SS	**09**	6.31e − 3	12	7.81e − 3	15	9.91e − 1	13	4.61e − 2	15	9.98e − 1	12.8
WGGP_OS_AA	30	2.39e − 2	30	2.95e − 2	26	8.81e − 1	30	1.79e − 1	36	9.93e − 1	30.4
WGGP_OS_AS	32	3.89e − 2	32	4.69e − 2	29	8.57e − 1	34	3.00e − 1	29	9.96e − 1	31.2
WGGP_OS_SA	29	2.37e − 2	29	2.91e − 2	25	8.81e − 1	29	1.77e − 1	35	9.93e − 1	29.4
WGGP_OS_SS	34	3.90e − 2	34	4.70e − 2	31	8.54e − 1	35	3.01e − 1	30	9.96e − 1	32.8
WGGP_TS_AA	28	1.53e − 2	28	1.74e − 2	36	5.62e − 1	27	8.60e − 2	31	9.95e − 1	30.0
WGGP_TS_AS	25	1.33e − 2	26	1.53e − 2	34	5.85e − 1	26	7.45e − 2	34	9.95e − 1	29.0

Continues on next page...

Table F.3 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGGP_TS_SA	27	1.52e − 2	27	1.73e − 2	35	5.72e − 1	28	8.61e − 2	32	9.95e − 1	29.8
WGGP_TS_SS	26	1.33e − 2	25	1.52e − 2	33	5.89e − 1	25	7.34e − 2	33	9.95e − 1	28.4

Table F.4: Results of the statistics on the second aggregation level for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	14	3.19e − 3	14	4.95e − 3	16	9.72e − 1	15	1.86e − 1	19	9.87e − 1	15.6
SGAN_NO_AS	**07**	3.09e − 3	**08**	4.77e − 3	15	9.73e − 1	13	1.82e − 1	17	9.87e − 1	12.0
SGAN_NO_SA	**05**	3.00e − 3	**06**	4.66e − 3	12	9.76e − 1	**09**	1.75e − 1	12	9.89e − 1	**8.8**
SGAN_NO_SS	**03**	2.96e − 3	**03**	4.54e − 3	11	9.77e − 1	**05**	1.71e − 1	11	9.89e − 1	**6.6**
SGAN_OS_AA	24	3.87e − 3	24	6.08e − 3	20	9.66e − 1	24	2.17e − 1	20	9.86e − 1	22.4
SGAN_OS_AS	34	1.18e − 2	34	1.80e − 2	32	8.16e − 1	33	6.25e − 1	30	9.63e − 1	32.6
SGAN_OS_SA	23	3.83e − 3	23	6.01e − 3	19	9.68e − 1	23	2.13e − 1	14	9.88e − 1	20.4
SGAN_OS_SS	33	1.18e − 2	33	1.79e − 2	31	8.19e − 1	32	6.21e − 1	28	9.65e − 1	31.4
SGAN_TS_AA	12	3.17e − 3	13	4.83e − 3	**10**	9.77e − 1	11	1.77e − 1	**10**	9.89e − 1	11.2
SGAN_TS_AS	**10**	3.15e − 3	11	4.81e − 3	**07**	9.78e − 1	12	1.78e − 1	**08**	9.90e − 1	**9.6**
SGAN_TS_SA	**08**	3.09e − 3	**07**	4.73e − 3	**05**	9.78e − 1	**06**	1.74e − 1	**05**	9.90e − 1	**6.2**
SGAN_TS_SS	11	3.16e − 3	12	4.82e − 3	**06**	9.78e − 1	**10**	1.77e − 1	**06**	9.90e − 1	**9.0**
WGAN_NO_AA	22	3.68e − 3	22	5.69e − 3	18	9.70e − 1	19	2.03e − 1	16	9.87e − 1	19.4
WGAN_NO_AS	15	3.28e − 3	16	5.02e − 3	14	9.73e − 1	16	1.86e − 1	18	9.87e − 1	15.8
WGAN_NO_SA	21	3.66e − 3	21	5.65e − 3	17	9.72e − 1	18	2.02e − 1	13	9.88e − 1	18.0
WGAN_NO_SS	16	3.31e − 3	15	5.02e − 3	13	9.74e − 1	14	1.85e − 1	15	9.88e − 1	14.6
WGAN_OS_AA	**09**	3.12e − 3	**09**	4.77e − 3	**09**	9.78e − 1	**07**	1.74e − 1	**09**	9.89e − 1	**8.6**
WGAN_OS_AS	31	1.16e − 2	31	1.77e − 2	29	8.23e − 1	30	6.15e − 1	25	9.69e − 1	29.2
WGAN_OS_SA	13	3.18e − 3	**10**	4.80e − 3	**08**	9.78e − 1	**08**	1.75e − 1	**07**	9.90e − 1	**9.2**
WGAN_OS_SS	32	1.16e − 2	32	1.78e − 2	30	8.22e − 1	31	6.17e − 1	26	9.69e − 1	30.2
WGAN_TS_AA	**06**	3.01e − 3	**05**	4.63e − 3	**04**	9.81e − 1	**04**	1.64e − 1	**03**	9.92e − 1	**4.4**
WGAN_TS_AS	**01**	2.78e − 3	**01**	4.21e − 3	**01**	9.83e − 1	**01**	1.53e − 1	**02**	9.92e − 1	**1.2**
WGAN_TS_SA	**04**	2.98e − 3	**04**	4.54e − 3	**03**	9.82e − 1	**03**	1.61e − 1	**01**	9.92e − 1	**3.0**
WGAN_TS_SS	**02**	2.83e − 3	**02**	4.27e − 3	**02**	9.83e − 1	**02**	1.56e − 1	**04**	9.92e − 1	**2.4**
WGGP_NO_AA	20	3.52e − 3	20	5.33e − 3	24	9.57e − 1	21	2.03e − 1	24	9.81e − 1	21.8
WGGP_NO_AS	17	3.42e − 3	18	5.30e − 3	21	9.58e − 1	20	2.03e − 1	21	9.81e − 1	19.4
WGGP_NO_SA	19	3.48e − 3	17	5.27e − 3	22	9.57e − 1	17	2.02e − 1	22	9.81e − 1	19.4
WGGP_NO_SS	18	3.42e − 3	19	5.31e − 3	23	9.57e − 1	22	2.04e − 1	23	9.81e − 1	21.0

Continues on next page...

Table F.4 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGGP_OS_AA	30	9.46e − 3	30	1.69e − 2	36	4.37e − 1	34	6.40e − 1	34	9.31e − 1	32.8
WGGP_OS_AS	36	1.35e − 2	36	2.13e − 2	35	5.23e − 1	36	7.78e − 1	36	9.18e − 1	35.8
WGGP_OS_SA	29	8.87e − 3	29	1.53e − 2	33	7.06e − 1	29	5.61e − 1	33	9.44e − 1	30.6
WGGP_OS_SS	35	1.29e − 2	35	2.00e − 2	34	7.05e − 1	35	7.20e − 1	35	9.31e − 1	34.8
WGGP_TS_AA	28	6.30e − 3	28	9.80e − 3	28	8.91e − 1	28	3.31e − 1	31	9.62e − 1	28.6
WGGP_TS_AS	27	6.08e − 3	27	9.39e − 3	27	8.97e − 1	27	3.19e − 1	32	9.61e − 1	28.0
WGGP_TS_SA	26	6.04e − 3	26	9.29e − 3	26	9.07e − 1	26	3.10e − 1	27	9.65e − 1	26.2
WGGP_TS_SS	25	5.80e − 3	25	8.88e − 3	25	9.12e − 1	25	2.99e − 1	29	9.64e − 1	25.8

Table F.5: Results of the statistics on the third aggregation level for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	13	1.19e − 3	14	2.19e − 3	14	9.51e − 1	14	3.75e − 1	16	9.77e − 1	14.2
SGAN_NO_AS	**10**	1.19e − 3	12	2.14e − 3	13	9.53e − 1	13	3.70e − 1	15	9.77e − 1	12.6
SGAN_NO_SA	**06**	1.15e − 3	**08**	2.09e − 3	12	9.57e − 1	**10**	3.58e − 1	12	9.80e − 1	**9.6**
SGAN_NO_SS	**05**	1.15e − 3	**05**	2.06e − 3	**10**	9.58e − 1	**08**	3.53e − 1	11	9.80e − 1	**7.8**
SGAN_OS_AA	24	1.37e − 3	24	2.58e − 3	20	9.42e − 1	24	4.16e − 1	18	9.76e − 1	22.0
SGAN_OS_AS	34	3.30e − 3	33	6.42e − 3	32	7.42e − 1	33	1.14	32	9.27e − 1	32.8
SGAN_OS_SA	23	1.36e − 3	23	2.53e − 3	19	9.46e − 1	23	4.06e − 1	14	9.78e − 1	20.4
SGAN_OS_SS	33	3.30e − 3	32	6.39e − 3	31	7.45e − 1	32	1.14	31	9.28e − 1	31.8
SGAN_TS_AA	14	1.20e − 3	13	2.15e − 3	11	9.57e − 1	**09**	3.58e − 1	**10**	9.80e − 1	11.4
SGAN_TS_AS	12	1.19e − 3	11	2.13e − 3	**09**	9.58e − 1	12	3.66e − 1	**09**	9.80e − 1	10.6
SGAN_TS_SA	**08**	1.18e − 3	**09**	2.10e − 3	**07**	9.59e − 1	**07**	3.50e − 1	**05**	9.81e − 1	**7.2**
SGAN_TS_SS	**09**	1.18e − 3	**10**	2.11e − 3	**08**	9.59e − 1	11	3.62e − 1	**07**	9.81e − 1	**9.0**
WGAN_NO_AA	21	1.32e − 3	22	2.39e − 3	17	9.50e − 1	18	3.90e − 1	17	9.77e − 1	19.0
WGAN_NO_AS	20	1.31e − 3	20	2.36e − 3	18	9.49e − 1	22	4.03e − 1	20	9.75e − 1	20.0
WGAN_NO_SA	19	1.31e − 3	21	2.38e − 3	15	9.51e − 1	17	3.89e − 1	13	9.78e − 1	17.0
WGAN_NO_SS	22	1.32e − 3	19	2.35e − 3	16	9.50e − 1	21	4.01e − 1	19	9.76e − 1	19.4
WGAN_OS_AA	**07**	1.17e − 3	**06**	2.07e − 3	**06**	9.60e − 1	**05**	3.48e − 1	**08**	9.80e − 1	**6.4**
WGAN_OS_AS	31	3.21e − 3	30	6.26e − 3	29	7.52e − 1	30	1.13	29	9.37e − 1	29.8
WGAN_OS_SA	11	1.19e − 3	**07**	2.08e − 3	**05**	9.60e − 1	**06**	3.48e − 1	**06**	9.81e − 1	**7.0**
WGAN_OS_SS	32	3.23e − 3	31	6.28e − 3	30	7.51e − 1	31	1.13	30	9.36e − 1	30.8
WGAN_TS_AA	**04**	1.11e − 3	**04**	1.97e − 3	**04**	9.65e − 1	**02**	3.26e − 1	**02**	9.84e − 1	**3.2**
WGAN_TS_AS	**01**	1.08e − 3	**01**	1.90e − 3	**01**	9.66e − 1	**03**	3.27e − 1	**03**	9.83e − 1	**1.8**

Continues on next page...

Table F.5 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGAN_TS_SA	**03**	1.10e − 3	**03**	1.95e − 3	**02**	9.66e − 1	**01**	3.22e − 1	**01**	9.84e − 1	**2.0**
WGAN_TS_SS	**02**	1.09e − 3	**02**	1.92e − 3	**03**	9.65e − 1	**04**	3.31e − 1	**04**	9.83e − 1	**3.0**
WGGP_NO_AA	18	1.30e − 3	18	2.29e − 3	24	9.37e − 1	16	3.84e − 1	24	9.70e − 1	20.0
WGGP_NO_AS	16	1.26e − 3	17	2.27e − 3	21	9.39e − 1	19	3.92e − 1	21	9.71e − 1	18.8
WGGP_NO_SA	17	1.28e − 3	15	2.26e − 3	23	9.38e − 1	15	3.83e − 1	22	9.71e − 1	18.4
WGGP_NO_SS	15	1.25e − 3	16	2.27e − 3	22	9.38e − 1	20	3.96e − 1	23	9.70e − 1	19.2
WGGP_OS_AA	30	2.92e − 3	34	7.01e − 3	36	9.66e − 2	34	1.19	34	8.97e − 1	33.6
WGGP_OS_AS	36	3.79e − 3	36	7.52e − 3	35	4.50e − 1	36	1.36	36	8.67e − 1	35.8
WGGP_OS_SA	29	2.71e − 3	29	6.18e − 3	34	5.38e − 1	29	1.03	33	9.15e − 1	30.8
WGGP_OS_SS	35	3.60e − 3	35	7.04e − 3	33	6.35e − 1	35	1.27	35	8.84e − 1	34.6
WGGP_TS_AA	28	2.06e − 3	28	3.93e − 3	28	8.57e − 1	28	5.92e − 1	27	9.44e − 1	27.8
WGGP_TS_AS	27	1.97e − 3	27	3.71e − 3	27	8.70e − 1	27	5.85e − 1	28	9.43e − 1	27.2
WGGP_TS_SA	26	1.97e − 3	26	3.70e − 3	26	8.77e − 1	26	5.66e − 1	25	9.48e − 1	25.8
WGGP_TS_SS	25	1.88e − 3	25	3.51e − 3	25	8.85e − 1	25	5.65e − 1	26	9.46e − 1	25.2

Table F.6: Results of the ML efficacy for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
SGAN_NO_AA	**02**	3.04	**02**	6.23e − 1	**2.0**
SGAN_NO_AS	**04**	3.05	**04**	6.51e − 1	**4.0**
SGAN_NO_SA	**06**	3.06	**01**	6.15e − 1	**3.5**
SGAN_NO_SS	**09**	3.06	**03**	6.38e − 1	**6.0**
SGAN_OS_AA	13	3.07	27	1.61e2	20.0
SGAN_OS_AS	27	3.38	23	2.95	25.0
SGAN_OS_SA	17	3.13	28	4.01e2	22.5
SGAN_OS_SS	28	3.43	24	2.97	26.0
SGAN_TS_AA	**10**	3.06	11	7.25e − 1	10.5
SGAN_TS_AS	**07**	3.06	12	7.34e − 1	**9.5**
SGAN_TS_SA	12	3.07	**09**	7.02e − 1	10.5
SGAN_TS_SS	14	3.08	**10**	7.22e − 1	12.0
WGAN_NO_AA	21	3.16	32	7.04e3	26.5
WGAN_NO_AS	19	3.14	17	1.18	18.0
WGAN_NO_SA	23	3.17	31	6.64e3	27.0
WGAN_NO_SS	22	3.17	18	1.19	20.0

Continues on next page...

Table F.6 – continued from previous page

Name	Continuous		Categorical		rank
WGAN_OS_AA	11	3.07	15	8.78e−1	13.0
WGAN_OS_AS	25	3.32	21	2.80	23.0
WGAN_OS_SA	15	3.08	16	8.92e−1	15.5
WGAN_OS_SS	26	3.34	22	2.82	24.0
WGAN_TS_AA	**01**	3.04	**07**	6.78e−1	**4.0**
WGAN_TS_AS	**03**	3.04	13	7.47e−1	**8.0**
WGAN_TS_SA	**08**	3.06	**08**	6.87e−1	**8.0**
WGAN_TS_SS	**05**	3.05	14	7.56e−1	**9.5**
WGGP_NO_AA	20	3.15	35	1.13e4	27.5
WGGP_NO_AS	16	3.12	**06**	6.72e−1	11.0
WGGP_NO_SA	24	3.18	36	1.23e4	30.0
WGGP_NO_SS	18	3.13	**05**	6.71e−1	11.5
WGGP_OS_AA	30	3.89	30	4.48e3	30.0
WGGP_OS_AS	36	4.50	25	3.10	30.5
WGGP_OS_SA	29	3.87	29	4.08e3	29.0
WGGP_OS_SS	35	4.25	26	3.30	30.5
WGGP_TS_AA	33	4.10	33	8.00e3	33.0
WGGP_TS_AS	34	4.15	20	1.28	27.0
WGGP_TS_SA	32	3.94	34	8.80e3	33.0
WGGP_TS_SS	31	3.91	19	1.20	25.0

LPMC case study

Table F.7: Results of the statistics on the first aggregation level (all columns) for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	22	1.26e−2	22	1.74e−2	24	8.72e−1	22	1.36e−1	24	9.69e−1	22.8
SGAN_NO_AS	16	9.78e−3	16	1.40e−2	16	9.06e−1	18	1.22e−1	22	9.73e−1	17.6
SGAN_NO_SA	20	1.22e−2	20	1.65e−2	21	8.86e−1	19	1.27e−1	20	9.74e−1	20.0
SGAN_NO_SS	13	9.40e−3	14	1.32e−2	14	9.17e−1	14	1.13e−1	21	9.73e−1	15.2
SGAN_OS_AA	24	1.41e−2	24	1.84e−2	30	3.91e−1	23	1.43e−1	36	9.36e−1	27.4
SGAN_OS_AS	34	2.34e−2	33	2.89e−2	22	8.85e−1	33	1.81e−1	32	9.62e−1	30.8
SGAN_OS_SA	23	1.31e−2	21	1.68e−2	29	4.07e−1	20	1.27e−1	35	9.40e−1	25.6
SGAN_OS_SS	31	2.24e−2	31	2.74e−2	17	8.95e−1	29	1.66e−1	29	9.65e−1	27.4
SGAN_TS_AA	15	9.71e−3	13	1.28e−2	15	9.13e−1	**09**	1.00e−1	18	9.74e−1	14.0

Continues on next page...

Table F.7 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_TS_AS	**05**	8.22e − 3	**05**	1.10e − 2	**09**	9.27e − 1	**04**	9.24e − 2	14	9.76e − 1	**7.4**
SGAN_TS_SA	14	9.41e − 3	11	1.25e − 2	13	9.17e − 1	**06**	9.75e − 2	13	9.76e − 1	11.4
SGAN_TS_SS	**02**	7.94e − 3	**02**	1.07e − 2	**08**	9.29e − 1	**03**	8.99e − 2	12	9.77e − 1	**5.4**
WGAN_NO_AA	36	2.42e − 2	36	3.25e − 2	26	8.01e − 1	36	2.30e − 1	17	9.75e − 1	30.2
WGAN_NO_AS	18	1.17e − 2	23	1.77e − 2	12	9.20e − 1	28	1.65e − 1	15	9.76e − 1	19.2
WGAN_NO_SA	32	2.25e − 2	34	2.93e − 2	25	8.22e − 1	35	1.98e − 1	**06**	9.80e − 1	26.4
WGAN_NO_SS	17	9.98e − 3	17	1.45e − 2	**05**	9.41e − 1	21	1.33e − 1	**05**	9.81e − 1	13.0
WGAN_OS_AA	26	1.61e − 2	26	2.12e − 2	32	2.09e − 1	27	1.63e − 1	26	9.67e − 1	27.4
WGAN_OS_AS	35	2.40e − 2	35	2.98e − 2	23	8.76e − 1	34	1.89e − 1	33	9.60e − 1	32.0
WGAN_OS_SA	25	1.52e − 2	25	1.97e − 2	31	2.21e − 1	24	1.48e − 1	31	9.63e − 1	27.2
WGAN_OS_SS	33	2.29e − 2	32	2.81e − 2	18	8.92e − 1	30	1.72e − 1	25	9.68e − 1	27.6
WGAN_TS_AA	28	1.71e − 2	28	2.26e − 2	36	−7.22e − 1	32	1.79e − 1	16	9.75e − 1	28.0
WGAN_TS_AS	11	9.03e − 3	15	1.32e − 2	**04**	9.49e − 1	16	1.18e − 1	**03**	9.85e − 1	**9.8**
WGAN_TS_SA	27	1.70e − 2	27	2.21e − 2	35	−7.16e − 1	31	1.74e − 1	**10**	9.79e − 1	26.0
WGAN_TS_SS	**10**	8.98e − 3	12	1.27e − 2	**02**	9.59e − 1	13	1.12e − 1	**01**	9.88e − 1	**7.6**
WGGP_NO_AA	**09**	8.80e − 3	**09**	1.18e − 2	**10**	9.22e − 1	**10**	1.00e − 1	**08**	9.79e − 1	**9.2**
WGGP_NO_AS	**04**	8.09e − 3	**06**	1.11e − 2	**06**	9.37e − 1	**05**	9.53e − 2	11	9.78e − 1	**6.4**
WGGP_NO_SA	12	9.11e − 3	**10**	1.23e − 2	11	9.22e − 1	12	1.05e − 1	**07**	9.80e − 1	10.4
WGGP_NO_SS	**07**	8.48e − 3	**08**	1.17e − 2	**07**	9.36e − 1	**08**	1.00e − 1	**09**	9.79e − 1	**7.8**
WGGP_OS_AA	21	1.25e − 2	19	1.59e − 2	28	5.81e − 1	17	1.20e − 1	34	9.56e − 1	23.8
WGGP_OS_AS	30	2.20e − 2	30	2.68e − 2	20	8.87e − 1	26	1.62e − 1	28	9.65e − 1	26.8
WGGP_OS_SA	19	1.21e − 2	18	1.55e − 2	27	5.89e − 1	15	1.15e − 1	27	9.65e − 1	21.2
WGGP_OS_SS	29	2.17e − 2	29	2.64e − 2	19	8.90e − 1	25	1.58e − 1	30	9.63e − 1	26.4
WGGP_TS_AA	**08**	8.59e − 3	**07**	1.14e − 2	33	2.03e − 1	11	1.04e − 1	23	9.72e − 1	16.4
WGGP_TS_AS	**06**	8.26e − 3	**04**	1.09e − 2	**03**	9.56e − 1	**02**	8.77e − 2	**04**	9.82e − 1	**3.8**
WGGP_TS_SA	**03**	8.04e − 3	**03**	1.07e − 2	34	2.01e − 1	**07**	9.77e − 2	19	9.74e − 1	13.2
WGGP_TS_SS	**01**	7.64e − 3	**01**	1.02e − 2	**01**	9.63e − 1	**01**	8.13e − 2	**02**	9.85e − 1	**1.2**

Table F.8: Results of the statistics on the first aggregation level (continuous columns) for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	30	1.39e − 2	30	2.16e − 2	33	8.87e − 1	30	2.16e − 1	33	9.67e − 1	31.2
SGAN_NO_AS	31	1.39e − 2	29	2.16e − 2	34	8.86e − 1	29	2.16e − 1	34	9.67e − 1	31.4

Continues on next page...

Table F.8 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_SA	23	1.28e − 2	23	1.96e − 2	31	9.18e − 1	23	1.96e − 1	27	9.75e − 1	25.4
SGAN_NO_SS	24	1.29e − 2	24	1.97e − 2	32	9.17e − 1	24	1.97e − 1	28	9.75e − 1	26.4
SGAN_OS_AA	22	1.28e − 2	22	1.91e − 2	21	9.36e − 1	22	1.91e − 1	23	9.76e − 1	22.0
SGAN_OS_AS	21	1.28e − 2	21	1.90e − 2	23	9.36e − 1	21	1.90e − 1	24	9.76e − 1	22.0
SGAN_OS_SA	14	1.07e − 2	14	1.58e − 2	11	9.56e − 1	14	1.58e − 1	**08**	9.84e − 1	12.2
SGAN_OS_SS	13	1.07e − 2	13	1.58e − 2	12	9.56e − 1	13	1.58e − 1	**07**	9.84e − 1	11.6
SGAN_TS_AA	12	1.05e − 2	11	1.51e − 2	18	9.47e − 1	11	1.51e − 1	16	9.80e − 1	13.6
SGAN_TS_AS	11	1.05e − 2	12	1.52e − 2	17	9.47e − 1	12	1.52e − 1	15	9.80e − 1	13.4
SGAN_TS_SA	**07**	9.97e − 3	**07**	1.46e − 2	13	9.54e − 1	**07**	1.46e − 1	**10**	9.83e − 1	**8.8**
SGAN_TS_SS	**08**	9.97e − 3	**08**	1.47e − 2	14	9.54e − 1	**08**	1.47e − 1	**09**	9.83e − 1	**9.4**
WGAN_NO_AA	36	1.99e − 2	36	3.18e − 2	35	8.75e − 1	36	3.18e − 1	35	9.64e − 1	35.6
WGAN_NO_AS	35	1.99e − 2	35	3.17e − 2	36	8.75e − 1	35	3.17e − 1	36	9.64e − 1	35.4
WGAN_NO_SA	34	1.65e − 2	34	2.51e − 2	30	9.25e − 1	34	2.51e − 1	25	9.76e − 1	31.4
WGAN_NO_SS	33	1.64e − 2	33	2.50e − 2	29	9.25e − 1	33	2.50e − 1	26	9.76e − 1	30.8
WGAN_OS_AA	26	1.35e − 2	26	2.04e − 2	24	9.36e − 1	26	2.04e − 1	20	9.79e − 1	24.4
WGAN_OS_AS	25	1.35e − 2	25	2.03e − 2	22	9.36e − 1	25	2.03e − 1	19	9.79e − 1	23.2
WGAN_OS_SA	18	1.15e − 2	18	1.72e − 2	**04**	9.63e − 1	18	1.72e − 1	**04**	9.87e − 1	12.4
WGAN_OS_SS	15	1.13e − 2	17	1.69e − 2	**03**	9.63e − 1	17	1.69e − 1	**03**	9.88e − 1	11.0
WGAN_TS_AA	29	1.38e − 2	31	2.17e − 2	19	9.39e − 1	31	2.17e − 1	11	9.82e − 1	24.2
WGAN_TS_AS	32	1.40e − 2	32	2.18e − 2	20	9.38e − 1	32	2.18e − 1	14	9.82e − 1	26.0
WGAN_TS_SA	28	1.37e − 2	28	2.06e − 2	**08**	9.59e − 1	28	2.06e − 1	**01**	9.89e − 1	18.6
WGAN_TS_SS	27	1.37e − 2	27	2.06e − 2	**07**	9.59e − 1	27	2.06e − 1	**02**	9.89e − 1	18.0
WGGP_NO_AA	17	1.14e − 2	15	1.67e − 2	27	9.29e − 1	15	1.67e − 1	31	9.72e − 1	21.0
WGGP_NO_AS	16	1.14e − 2	16	1.67e − 2	28	9.28e − 1	16	1.67e − 1	32	9.72e − 1	21.6
WGGP_NO_SA	19	1.19e − 2	19	1.76e − 2	25	9.30e − 1	19	1.76e − 1	29	9.73e − 1	22.2
WGGP_NO_SS	20	1.19e − 2	20	1.77e − 2	26	9.30e − 1	20	1.77e − 1	30	9.73e − 1	23.2
WGGP_OS_AA	**10**	1.03e − 2	**10**	1.50e − 2	16	9.52e − 1	**10**	1.50e − 1	22	9.77e − 1	13.6
WGGP_OS_AS	**09**	1.03e − 2	**09**	1.50e − 2	15	9.53e − 1	**09**	1.50e − 1	21	9.77e − 1	12.6
WGGP_OS_SA	**03**	9.49e − 3	**03**	1.40e − 2	**10**	9.56e − 1	**03**	1.40e − 1	17	9.79e − 1	**7.2**
WGGP_OS_SS	**04**	9.49e − 3	**04**	1.40e − 2	**09**	9.56e − 1	**04**	1.40e − 1	18	9.79e − 1	**7.8**
WGGP_TS_AA	**06**	9.84e − 3	**06**	1.43e − 2	**06**	9.61e − 1	**06**	1.43e − 1	12	9.82e − 1	**7.2**
WGGP_TS_AS	**05**	9.82e − 3	**05**	1.42e − 2	**05**	9.61e − 1	**05**	1.42e − 1	13	9.82e − 1	**6.6**
WGGP_TS_SA	**01**	8.91e − 3	**01**	1.31e − 2	**02**	9.69e − 1	**01**	1.31e − 1	**06**	9.86e − 1	**2.2**
WGGP_TS_SS	**02**	8.93e − 3	**02**	1.31e − 2	**01**	9.69e − 1	**02**	1.31e − 1	**05**	9.86e − 1	**2.4**

Table F.9: Results of the statistics on the first aggregation level (categorical columns) for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	19	1.13e − 2	19	1.34e − 2	17	8.58e − 1	17	6.17e − 2	19	9.70e − 1	18.2
SGAN_NO_AS	**07**	5.93e − 3	**07**	6.93e − 3	**09**	9.23e − 1	**07**	3.43e − 2	13	9.79e − 1	**8.6**
SGAN_NO_SA	20	1.15e − 2	20	1.37e − 2	18	8.56e − 1	18	6.24e − 2	16	9.72e − 1	18.4
SGAN_NO_SS	**10**	6.18e − 3	**10**	7.17e − 3	11	9.16e − 1	**09**	3.54e − 2	17	9.72e − 1	11.4
SGAN_OS_AA	24	1.53e − 2	24	1.78e − 2	30	−1.14e − 1	24	9.85e − 2	36	8.99e − 1	27.6
SGAN_OS_AS	34	3.33e − 2	34	3.82e − 2	20	8.38e − 1	33	1.73e − 1	29	9.49e − 1	30.0
SGAN_OS_SA	23	1.52e − 2	23	1.77e − 2	29	−1.03e − 1	23	9.79e − 2	35	9.00e − 1	26.6
SGAN_OS_SS	33	3.33e − 2	33	3.81e − 2	19	8.39e − 1	34	1.74e − 1	31	9.47e − 1	30.0
SGAN_TS_AA	18	8.98e − 3	18	1.06e − 2	16	8.81e − 1	16	5.26e − 2	22	9.69e − 1	18.0
SGAN_TS_AS	**09**	6.13e − 3	**09**	7.08e − 3	13	9.08e − 1	12	3.75e − 2	14	9.72e − 1	11.4
SGAN_TS_SA	17	8.90e − 3	17	1.05e − 2	15	8.83e − 1	15	5.20e − 2	18	9.70e − 1	16.4
SGAN_TS_SS	**08**	6.06e − 3	**08**	6.99e − 3	14	9.05e − 1	11	3.71e − 2	15	9.72e − 1	11.2
WGAN_NO_AA	29	2.81e − 2	29	3.32e − 2	25	7.32e − 1	29	1.49e − 1	**07**	9.85e − 1	23.8
WGAN_NO_AS	**02**	4.04e − 3	**02**	4.75e − 3	**01**	9.62e − 1	**02**	2.45e − 2	**03**	9.87e − 1	**2.0**
WGAN_NO_SA	30	2.81e − 2	30	3.32e − 2	26	7.27e − 1	30	1.50e − 1	**10**	9.84e − 1	25.2
WGAN_NO_SS	**01**	3.98e − 3	**01**	4.67e − 3	**05**	9.56e − 1	**01**	2.42e − 2	**04**	9.86e − 1	**2.4**
WGAN_OS_AA	25	1.86e − 2	25	2.20e − 2	31	−4.65e − 1	25	1.26e − 1	25	9.56e − 1	26.2
WGAN_OS_AS	36	3.39e − 2	36	3.87e − 2	24	8.20e − 1	36	1.75e − 1	32	9.42e − 1	32.8
WGAN_OS_SA	26	1.87e − 2	26	2.21e − 2	32	−4.68e − 1	26	1.26e − 1	33	9.40e − 1	28.6
WGAN_OS_SS	35	3.37e − 2	35	3.85e − 2	23	8.25e − 1	35	1.74e − 1	28	9.50e − 1	31.2
WGAN_TS_AA	28	2.00e − 2	28	2.35e − 2	35	−2.26	27	1.44e − 1	20	9.69e − 1	27.6
WGAN_TS_AS	**03**	4.45e − 3	**03**	5.21e − 3	**02**	9.59e − 1	**03**	2.48e − 2	**02**	9.87e − 1	**2.6**
WGAN_TS_SA	27	2.00e − 2	27	2.35e − 2	36	−2.27	28	1.44e − 1	21	9.69e − 1	27.8
WGAN_TS_SS	**04**	4.61e − 3	**04**	5.38e − 3	**03**	9.59e − 1	**04**	2.50e − 2	**01**	9.88e − 1	**3.2**
WGGP_NO_AA	11	6.36e − 3	11	7.25e − 3	**10**	9.16e − 1	13	3.77e − 2	**05**	9.86e − 1	**10.0**
WGGP_NO_AS	**05**	5.01e − 3	**05**	5.79e − 3	**07**	9.45e − 1	**05**	2.83e − 2	11	9.83e − 1	**6.6**
WGGP_NO_SA	13	6.48e − 3	12	7.39e − 3	12	9.14e − 1	14	3.77e − 2	**06**	9.85e − 1	11.4
WGGP_NO_SS	**06**	5.26e − 3	**06**	6.05e − 3	**08**	9.42e − 1	**06**	2.86e − 2	**08**	9.84e − 1	**6.8**
WGGP_OS_AA	21	1.45e − 2	21	1.68e − 2	28	2.37e − 1	21	9.11e − 2	34	9.36e − 1	25.0
WGGP_OS_AS	31	3.30e − 2	31	3.77e − 2	22	8.26e − 1	31	1.72e − 1	26	9.53e − 1	28.2
WGGP_OS_SA	22	1.45e − 2	22	1.68e − 2	27	2.49e − 1	22	9.13e − 2	27	9.52e − 1	24.0

Continues on next page...

Table F.9 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGGP_OS_SS	32	3.30e − 2	32	3.79e − 2	21	8.29e − 1	32	1.73e − 1	30	9.48e − 1	29.4
WGGP_TS_AA	16	7.43e − 3	16	8.74e − 3	33	−5.01e − 1	20	6.80e − 2	23	9.63e − 1	21.6
WGGP_TS_AS	14	6.82e − 3	14	7.88e − 3	**06**	9.51e − 1	**10**	3.70e − 2	12	9.82e − 1	11.2
WGGP_TS_SA	15	7.23e − 3	15	8.49e − 3	34	−5.12e − 1	19	6.70e − 2	24	9.62e − 1	21.4
WGGP_TS_SS	12	6.44e − 3	13	7.47e − 3	**04**	9.56e − 1	**08**	3.52e − 2	**09**	9.84e − 1	**9.2**

Table F.10: Results of the statistics on the second aggregation level for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	21	5.21e − 3	21	9.27e − 3	21	8.90e − 1	21	3.50e − 1	26	9.62e − 1	22.0
SGAN_NO_AS	17	4.89e − 3	16	8.70e − 3	18	9.02e − 1	18	3.37e − 1	22	9.64e − 1	18.2
SGAN_NO_SA	18	5.07e − 3	19	8.96e − 3	17	9.08e − 1	19	3.38e − 1	14	9.67e − 1	17.4
SGAN_NO_SS	15	4.79e − 3	15	8.47e − 3	11	9.18e − 1	15	3.27e − 1	13	9.69e − 1	13.8
SGAN_OS_AA	22	5.50e − 3	22	9.34e − 3	30	8.15e − 1	22	3.69e − 1	36	9.49e − 1	26.4
SGAN_OS_AS	34	8.75e − 3	32	1.38e − 2	22	8.90e − 1	32	4.51e − 1	30	9.60e − 1	30.0
SGAN_OS_SA	20	5.21e − 3	18	8.80e − 3	28	8.32e − 1	20	3.48e − 1	35	9.55e − 1	24.2
SGAN_OS_SS	32	8.50e − 3	31	1.34e − 2	19	9.01e − 1	28	4.35e − 1	20	9.65e − 1	26.0
SGAN_TS_AA	12	4.34e − 3	12	7.38e − 3	**10**	9.23e − 1	11	2.85e − 1	12	9.70e − 1	11.4
SGAN_TS_AS	**08**	4.19e − 3	**08**	7.15e − 3	**06**	9.30e − 1	**08**	2.79e − 1	11	9.71e − 1	**8.2**
SGAN_TS_SA	**09**	4.24e − 3	**09**	7.30e − 3	**08**	9.28e − 1	**09**	2.82e − 1	**09**	9.72e − 1	**8.8**
SGAN_TS_SS	**07**	4.13e − 3	**07**	7.10e − 3	**04**	9.34e − 1	**07**	2.78e − 1	**06**	9.73e − 1	**6.2**
WGAN_NO_AA	33	8.53e − 3	36	1.58e − 2	29	8.28e − 1	36	5.49e − 1	33	9.59e − 1	33.4
WGAN_NO_AS	26	6.19e − 3	27	1.14e − 2	24	8.88e − 1	29	4.41e − 1	29	9.60e − 1	27.0
WGAN_NO_SA	31	8.26e − 3	35	1.46e − 2	27	8.59e − 1	35	4.99e − 1	21	9.65e − 1	29.8
WGAN_NO_SS	23	5.92e − 3	23	1.04e − 2	13	9.16e − 1	25	3.98e − 1	17	9.66e − 1	20.2
WGAN_OS_AA	25	6.14e − 3	25	1.09e − 2	32	7.97e − 1	27	4.11e − 1	27	9.62e − 1	27.2
WGAN_OS_AS	36	8.99e − 3	34	1.42e − 2	23	8.88e − 1	34	4.61e − 1	32	9.60e − 1	31.8
WGAN_OS_SA	24	6.14e − 3	24	1.06e − 2	31	8.12e − 1	26	4.03e − 1	18	9.66e − 1	24.6
WGAN_OS_SS	35	8.94e − 3	33	1.41e − 2	20	9.01e − 1	33	4.59e − 1	23	9.64e − 1	28.8
WGAN_TS_AA	27	6.27e − 3	26	1.14e − 2	36	6.26e − 1	30	4.46e − 1	34	9.57e − 1	30.6
WGAN_TS_AS	16	4.89e − 3	17	8.80e − 3	**05**	9.34e − 1	16	3.28e − 1	**04**	9.75e − 1	11.6
WGAN_TS_SA	28	6.49e − 3	28	1.14e − 2	35	6.38e − 1	31	4.47e − 1	31	9.60e − 1	30.6

Continues on next page...

Table F.10 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGAN_TS_SS	19	5.14e−3	20	8.97e−3	**03**	9.44e−1	17	3.33e−1	**02**	9.77e−1	12.2
WGGP_NO_AA	11	4.29e−3	**10**	7.32e−3	14	9.15e−1	**10**	2.85e−1	**10**	9.71e−1	11.0
WGGP_NO_AS	**06**	4.04e−3	**05**	7.00e−3	**09**	9.28e−1	**04**	2.75e−1	**08**	9.72e−1	**6.4**
WGGP_NO_SA	**10**	4.27e−3	11	7.32e−3	12	9.17e−1	12	2.86e−1	**07**	9.72e−1	10.4
WGGP_NO_SS	**05**	4.02e−3	**06**	7.01e−3	**07**	9.29e−1	**05**	2.77e−1	**05**	9.73e−1	**5.6**
WGGP_OS_AA	14	4.70e−3	14	7.79e−3	26	8.70e−1	14	2.97e−1	24	9.64e−1	18.4
WGGP_OS_AS	30	7.81e−3	30	1.22e−2	16	9.08e−1	24	3.91e−1	16	9.66e−1	23.2
WGGP_OS_SA	13	4.64e−3	13	7.67e−3	25	8.77e−1	13	2.88e−1	19	9.65e−1	16.6
WGGP_OS_SS	29	7.78e−3	29	1.22e−2	15	9.10e−1	23	3.90e−1	15	9.67e−1	22.2
WGGP_TS_AA	**04**	3.85e−3	**04**	6.39e−3	34	7.89e−1	**06**	2.77e−1	28	9.62e−1	15.2
WGGP_TS_AS	**02**	3.70e−3	**02**	6.16e−3	**02**	9.47e−1	**02**	2.35e−1	**03**	9.75e−1	**2.2**
WGGP_TS_SA	**03**	3.71e−3	**03**	6.23e−3	33	7.91e−1	**03**	2.71e−1	25	9.64e−1	13.4
WGGP_TS_SS	**01**	3.52e−3	**01**	5.91e−3	**01**	9.51e−1	**01**	2.26e−1	**01**	9.78e−1	**1.0**

Table F.11: Results of the statistics on the third aggregation level for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	20	1.87e−3	21	4.21e−3	23	8.64e−1	20	6.48e−1	24	9.49e−1	21.6
SGAN_NO_AS	18	1.84e−3	18	4.09e−3	20	8.75e−1	18	6.37e−1	20	9.51e−1	18.8
SGAN_NO_SA	16	1.81e−3	17	4.02e−3	18	8.86e−1	17	6.33e−1	18	9.55e−1	17.2
SGAN_NO_SS	15	1.80e−3	16	3.98e−3	15	8.94e−1	16	6.32e−1	15	9.56e−1	15.4
SGAN_OS_AA	21	1.97e−3	20	4.20e−3	33	8.25e−1	22	6.96e−1	36	9.35e−1	26.4
SGAN_OS_AS	34	2.79e−3	32	5.75e−3	24	8.57e−1	32	8.38e−1	29	9.44e−1	30.2
SGAN_OS_SA	19	1.86e−3	15	3.94e−3	27	8.54e−1	21	6.64e−1	30	9.44e−1	22.4
SGAN_OS_SS	32	2.71e−3	31	5.60e−3	21	8.72e−1	31	8.34e−1	23	9.50e−1	27.6
SGAN_TS_AA	13	1.62e−3	13	3.49e−3	**10**	8.99e−1	14	5.62e−1	12	9.58e−1	12.4
SGAN_TS_AS	**10**	1.59e−3	11	3.44e−3	**09**	9.06e−1	12	5.60e−1	**10**	9.59e−1	10.4
SGAN_TS_SA	**09**	1.58e−3	**10**	3.40e−3	**06**	9.08e−1	11	5.55e−1	**07**	9.61e−1	**8.6**
SGAN_TS_SS	**07**	1.56e−3	**09**	3.39e−3	**04**	9.13e−1	13	5.60e−1	**05**	9.62e−1	**7.6**
WGAN_NO_AA	33	2.73e−3	36	6.71e−3	34	7.81e−1	36	9.66e−1	32	9.43e−1	34.2
WGAN_NO_AS	27	2.27e−3	30	5.22e−3	30	8.48e−1	29	8.08e−1	31	9.44e−1	29.4
WGAN_NO_SA	31	2.71e−3	35	6.20e−3	32	8.25e−1	34	8.89e−1	27	9.47e−1	31.8
WGAN_NO_SS	28	2.28e−3	29	5.00e−3	19	8.79e−1	27	7.73e−1	26	9.48e−1	25.8
WGAN_OS_AA	23	2.05e−3	24	4.75e−3	31	8.34e−1	26	7.47e−1	22	9.50e−1	25.2

Continues on next page...

Table F.11 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGAN_OS_AS	35	2.86e−3	33	5.86e−3	25	8.57e−1	33	8.59e−1	34	9.43e−1	32.0
WGAN_OS_SA	25	2.11e−3	23	4.65e−3	26	8.55e−1	25	7.42e−1	19	9.53e−1	23.6
WGAN_OS_SS	36	2.92e−3	34	6.00e−3	22	8.65e−1	35	9.06e−1	28	9.46e−1	31.0
WGAN_TS_AA	24	2.08e−3	28	4.98e−3	36	7.47e−1	30	8.09e−1	35	9.41e−1	30.6
WGAN_TS_AS	17	1.83e−3	19	4.14e−3	**08**	9.06e−1	15	6.22e−1	**04**	9.62e−1	12.6
WGAN_TS_SA	26	2.21e−3	26	4.95e−3	35	7.66e−1	28	8.02e−1	33	9.43e−1	29.6
WGAN_TS_SS	22	1.99e−3	22	4.30e−3	**03**	9.14e−1	19	6.47e−1	**03**	9.63e−1	13.8
WGGP_NO_AA	14	1.64e−3	14	3.51e−3	14	8.96e−1	**09**	5.49e−1	11	9.58e−1	12.4
WGGP_NO_AS	**06**	1.54e−3	**08**	3.35e−3	**07**	9.08e−1	**04**	5.36e−1	**08**	9.60e−1	**6.6**
WGGP_NO_SA	12	1.61e−3	12	3.48e−3	11	8.99e−1	**10**	5.50e−1	**09**	9.59e−1	10.8
WGGP_NO_SS	**05**	1.52e−3	**06**	3.33e−3	**05**	9.10e−1	**05**	5.37e−1	**06**	9.61e−1	**5.4**
WGGP_OS_AA	11	1.60e−3	**07**	3.33e−3	13	8.97e−1	**08**	5.44e−1	16	9.56e−1	11.0
WGGP_OS_AS	30	2.43e−3	27	4.95e−3	17	8.89e−1	24	7.31e−1	17	9.56e−1	23.0
WGGP_OS_SA	**08**	1.57e−3	**05**	3.29e−3	12	8.98e−1	**07**	5.42e−1	13	9.57e−1	**9.0**
WGGP_OS_SS	29	2.40e−3	25	4.91e−3	16	8.90e−1	23	7.31e−1	14	9.56e−1	21.4
WGGP_TS_AA	**04**	1.45e−3	**04**	3.03e−3	29	8.48e−1	**06**	5.39e−1	25	9.48e−1	13.6
WGGP_TS_AS	**02**	1.39e−3	**02**	2.90e−3	**02**	9.31e−1	**02**	4.61e−1	**02**	9.66e−1	**2.0**
WGGP_TS_SA	**03**	1.40e−3	**03**	2.96e−3	28	8.51e−1	**03**	5.33e−1	21	9.50e−1	11.6
WGGP_TS_SS	**01**	1.32e−3	**01**	2.79e−3	**01**	9.35e−1	**01**	4.50e−1	**01**	9.69e−1	**1.0**

Table F.12: Results of the ML efficacy for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
SGAN_NO_AA	15	2.81e1	32	5.98	23.5
SGAN_NO_AS	16	2.94e1	19	3.89	17.5
SGAN_NO_SA	29	4.59e1	23	4.55	26.0
SGAN_NO_SS	30	4.74e1	18	3.77	24.0
SGAN_OS_AA	17	3.08e1	25	4.61	21.0
SGAN_OS_AS	20	3.37e1	27	4.73	23.5
SGAN_OS_SA	27	4.16e1	21	4.18	24.0
SGAN_OS_SS	28	4.44e1	24	4.55	26.0
SGAN_TS_AA	13	2.78e1	26	4.67	19.5
SGAN_TS_AS	14	2.80e1	16	3.71	15.0
SGAN_TS_SA	18	3.21e1	17	3.73	17.5

Continues on next page...

Table F.12 – continued from previous page

Name	Continuous		Categorical		rank
SGAN_TS_SS	19	3.31e1	14	3.48	16.5
WGAN_NO_AA	24	3.93e1	35	1.49e4	29.5
WGAN_NO_AS	26	4.14e1	29	5.00	27.5
WGAN_NO_SA	35	6.60e1	36	1.62e4	35.5
WGAN_NO_SS	36	6.87e1	28	4.94	32.0
WGAN_OS_AA	21	3.55e1	13	3.45	17.0
WGAN_OS_AS	22	3.90e1	30	5.30	26.0
WGAN_OS_SA	31	5.87e1	15	3.49	23.0
WGAN_OS_SS	33	6.15e1	31	5.40	32.0
WGAN_TS_AA	23	3.91e1	33	7.28e3	28.0
WGAN_TS_AS	25	4.05e1	20	4.14	22.5
WGAN_TS_SA	32	6.15e1	34	7.68e3	33.0
WGAN_TS_SS	34	6.34e1	22	4.31	28.0
WGGP_NO_AA	**09**	2.31e1	**10**	3.09	**9.5**
WGGP_NO_AS	**10**	2.32e1	**06**	2.65	**8.0**
WGGP_NO_SA	11	2.57e1	**09**	3.04	**10.0**
WGGP_NO_SS	12	2.60e1	**05**	2.61	**8.5**
WGGP_OS_AA	**01**	1.96e1	**04**	2.56	**2.5**
WGGP_OS_AS	**02**	2.03e1	**07**	2.94	**4.5**
WGGP_OS_SA	**05**	2.09e1	**03**	2.50	**4.0**
WGGP_OS_SS	**06**	2.16e1	**08**	2.94	**7.0**
WGGP_TS_AA	**03**	2.03e1	12	3.19	**7.5**
WGGP_TS_AS	**04**	2.03e1	**02**	2.21	**3.0**
WGGP_TS_SA	**08**	2.21e1	11	3.14	**9.5**
WGGP_TS_SS	**07**	2.20e1	**01**	2.19	**4.0**

LPMC_half case study

Table F.13: Results of the statistics on the first aggregation level (all columns) for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	22	1.59e − 2	23	2.18e − 2	19	8.15e − 1	21	1.62e − 1	**10**	9.63e − 1	19.0
SGAN_NO_AS	**10**	1.12e − 2	11	1.60e − 2	**05**	9.04e − 1	12	1.38e − 1	**04**	9.71e − 1	**8.4**
SGAN_NO_SA	19	1.55e − 2	21	2.09e − 2	17	8.28e − 1	17	1.55e − 1	**05**	9.67e − 1	15.8

Continues on next page...

Table F.13 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_SS	**07**	1.09e−2	**08**	1.53e−2	**02**	9.17e−1	**06**	1.31e−1	**02**	9.74e−1	**5.0**
SGAN_OS_AA	21	1.57e−2	22	2.12e−2	26	1.56e−1	22	1.72e−1	36	9.20e−1	25.4
SGAN_OS_AS	32	2.56e−2	31	3.23e−2	20	8.08e−1	28	2.10e−1	32	9.37e−1	28.6
SGAN_OS_SA	18	1.48e−2	18	1.97e−2	25	1.94e−1	19	1.57e−1	34	9.27e−1	22.8
SGAN_OS_SS	30	2.45e−2	29	3.06e−2	14	8.43e−1	26	1.94e−1	26	9.50e−1	25.0
SGAN_TS_AA	16	1.34e−2	16	1.85e−2	18	8.18e−1	16	1.50e−1	19	9.55e−1	17.0
SGAN_TS_AS	11	1.12e−2	**10**	1.58e−2	15	8.39e−1	11	1.38e−1	18	9.55e−1	13.0
SGAN_TS_SA	14	1.26e−2	14	1.70e−2	16	8.37e−1	**09**	1.34e−1	16	9.57e−1	13.8
SGAN_TS_SS	**03**	1.01e−2	**03**	1.41e−2	11	8.67e−1	**03**	1.21e−1	11	9.62e−1	**6.2**
WGAN_NO_AA	36	4.92e−2	36	6.44e−2	32	−3.48e−1	36	4.37e−1	35	9.21e−1	35.0
WGAN_NO_AS	24	2.01e−2	30	3.09e−2	22	6.11e−1	34	2.94e−1	33	9.30e−1	28.6
WGAN_NO_SA	35	4.28e−2	35	5.33e−2	29	−8.45e−2	35	3.26e−1	21	9.52e−1	31.0
WGAN_NO_SS	17	1.37e−2	19	1.99e−2	**08**	8.78e−1	25	1.84e−1	12	9.62e−1	16.2
WGAN_OS_AA	31	2.51e−2	33	3.37e−2	28	2.59e−2	32	2.53e−1	30	9.46e−1	30.8
WGAN_OS_AS	34	2.79e−2	34	3.63e−2	21	7.77e−1	33	2.55e−1	31	9.39e−1	30.6
WGAN_OS_SA	27	2.27e−2	27	2.94e−2	27	1.13e−1	29	2.11e−1	13	9.59e−1	24.6
WGAN_OS_SS	33	2.57e−2	32	3.24e−2	13	8.60e−1	30	2.14e−1	22	9.52e−1	26.0
WGAN_TS_AA	26	2.23e−2	26	2.93e−2	36	−1.09	31	2.23e−1	15	9.57e−1	26.8
WGAN_TS_AS	12	1.13e−2	12	1.65e−2	**06**	8.92e−1	14	1.47e−1	**06**	9.67e−1	**10.0**
WGAN_TS_SA	25	2.14e−2	24	2.73e−2	35	−1.04	27	2.05e−1	**07**	9.66e−1	23.6
WGAN_TS_SS	**04**	1.07e−2	**06**	1.48e−2	**01**	9.26e−1	**05**	1.30e−1	**01**	9.75e−1	**3.4**
WGGP_NO_AA	15	1.29e−2	15	1.74e−2	34	−5.08e−1	20	1.60e−1	28	9.47e−1	22.4
WGGP_NO_AS	**09**	1.12e−2	**09**	1.54e−2	**10**	8.68e−1	**08**	1.33e−1	14	9.57e−1	**10.0**
WGGP_NO_SA	13	1.25e−2	13	1.69e−2	33	−4.74e−1	18	1.55e−1	20	9.54e−1	19.4
WGGP_NO_SS	**05**	1.08e−2	**07**	1.49e−2	**07**	8.86e−1	**04**	1.27e−1	**09**	9.63e−1	**6.4**
WGGP_OS_AA	23	1.63e−2	20	2.04e−2	24	4.53e−1	15	1.49e−1	24	9.50e−1	21.2
WGGP_OS_AS	29	2.45e−2	28	2.97e−2	12	8.65e−1	24	1.82e−1	27	9.48e−1	24.0
WGGP_OS_SA	20	1.57e−2	17	1.96e−2	23	4.59e−1	13	1.42e−1	29	9.47e−1	20.4
WGGP_OS_SS	28	2.39e−2	25	2.89e−2	**09**	8.72e−1	23	1.75e−1	23	9.50e−1	21.6
WGGP_TS_AA	**08**	1.11e−2	**05**	1.44e−2	30	−2.19e−1	**10**	1.35e−1	25	9.50e−1	15.6
WGGP_TS_AS	**02**	9.61e−3	**02**	1.28e−2	**04**	9.09e−1	**02**	1.05e−1	**08**	9.64e−1	**3.6**

Continues on next page...

Table F.13 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGGP_TS_SA	**06**	1.08e−2	**04**	1.41e−2	31	−2.20e−1	**07**	1.33e−1	17	9.57e−1	13.0
WGGP_TS_SS	**01**	9.32e−3	**01**	1.25e−2	**03**	9.11e−1	**01**	1.03e−1	**03**	9.73e−1	**1.8**

Table F.14: Results of the statistics on the first aggregation level (continuous columns) for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	20	1.57e−2	25	2.43e−2	15	8.97e−1	25	2.43e−1	15	9.69e−1	20.0
SGAN_NO_AS	21	1.57e−2	26	2.45e−2	16	8.96e−1	26	2.45e−1	16	9.69e−1	21.0
SGAN_NO_SA	15	1.51e−2	15	2.29e−2	**09**	9.31e−1	15	2.29e−1	**01**	9.78e−1	11.0
SGAN_NO_SS	16	1.51e−2	16	2.30e−2	**10**	9.31e−1	16	2.30e−1	**02**	9.78e−1	12.0
SGAN_OS_AA	22	1.58e−2	24	2.43e−2	30	8.30e−1	24	2.43e−1	20	9.66e−1	24.0
SGAN_OS_AS	19	1.56e−2	23	2.41e−2	29	8.33e−1	23	2.41e−1	19	9.67e−1	22.6
SGAN_OS_SA	12	1.38e−2	12	2.11e−2	17	8.95e−1	12	2.11e−1	**08**	9.73e−1	12.2
SGAN_OS_SS	11	1.37e−2	11	2.09e−2	18	8.94e−1	11	2.09e−1	**09**	9.73e−1	12.0
SGAN_TS_AA	17	1.55e−2	21	2.40e−2	31	8.27e−1	21	2.40e−1	21	9.65e−1	22.2
SGAN_TS_AS	18	1.56e−2	22	2.40e−2	32	8.26e−1	22	2.40e−1	22	9.65e−1	23.2
SGAN_TS_SA	**10**	1.37e−2	**10**	2.06e−2	22	8.69e−1	**10**	2.06e−1	14	9.71e−1	13.2
SGAN_TS_SS	**09**	1.36e−2	**09**	2.05e−2	21	8.70e−1	**09**	2.05e−1	12	9.72e−1	12.0
WGAN_NO_AA	36	3.61e−2	36	5.76e−2	35	2.86e−1	36	5.76e−1	36	8.80e−1	35.8
WGAN_NO_AS	35	3.60e−2	35	5.75e−2	36	2.85e−1	35	5.75e−1	35	8.80e−1	35.2
WGAN_NO_SA	34	2.27e−2	34	3.47e−2	28	8.34e−1	34	3.47e−1	30	9.47e−1	32.0
WGAN_NO_SS	33	2.27e−2	33	3.47e−2	27	8.35e−1	33	3.47e−1	29	9.47e−1	31.0
WGAN_OS_AA	32	2.17e−2	31	3.41e−2	33	7.35e−1	31	3.41e−1	32	9.46e−1	31.8
WGAN_OS_AS	31	2.17e−2	32	3.41e−2	34	7.33e−1	32	3.41e−1	31	9.46e−1	32.0
WGAN_OS_SA	27	1.68e−2	27	2.53e−2	13	9.16e−1	27	2.53e−1	11	9.72e−1	21.0
WGAN_OS_SS	28	1.69e−2	28	2.54e−2	14	9.15e−1	28	2.54e−1	13	9.71e−1	22.2
WGAN_TS_AA	30	1.74e−2	30	2.71e−2	26	8.47e−1	30	2.71e−1	26	9.54e−1	28.4
WGAN_TS_AS	29	1.73e−2	29	2.71e−2	25	8.49e−1	29	2.71e−1	25	9.54e−1	27.4
WGAN_TS_SA	23	1.58e−2	17	2.33e−2	11	9.30e−1	17	2.33e−1	**05**	9.74e−1	14.6
WGAN_TS_SS	26	1.59e−2	18	2.35e−2	12	9.30e−1	18	2.35e−1	**07**	9.73e−1	16.2
WGGP_NO_AA	25	1.59e−2	19	2.36e−2	24	8.53e−1	19	2.36e−1	34	9.40e−1	24.2
WGGP_NO_AS	24	1.59e−2	20	2.36e−2	23	8.53e−1	20	2.36e−1	33	9.40e−1	24.0
WGGP_NO_SA	14	1.48e−2	14	2.23e−2	20	8.83e−1	14	2.23e−1	28	9.51e−1	18.0

Continues on next page...

Table F.14 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGGP_NO_SS	13	1.47e−2	13	2.21e−2	19	8.84e−1	13	2.21e−1	27	9.51e−1	17.0
WGGP_OS_AA	**08**	1.30e−2	**08**	1.85e−2	**08**	9.31e−1	**08**	1.85e−1	24	9.64e−1	11.2
WGGP_OS_AS	**07**	1.29e−2	**07**	1.83e−2	**07**	9.32e−1	**07**	1.83e−1	23	9.64e−1	10.2
WGGP_OS_SA	**06**	1.20e−2	**06**	1.71e−2	**05**	9.37e−1	**06**	1.71e−1	17	9.67e−1	**8.0**
WGGP_OS_SS	**05**	1.20e−2	**03**	1.69e−2	**06**	9.36e−1	**03**	1.69e−1	18	9.67e−1	**7.0**
WGGP_TS_AA	**04**	1.18e−2	**05**	1.71e−2	**04**	9.43e−1	**05**	1.71e−1	**10**	9.73e−1	**5.6**
WGGP_TS_AS	**03**	1.17e−2	**04**	1.70e−2	**03**	9.45e−1	**04**	1.70e−1	**06**	9.74e−1	**4.0**
WGGP_TS_SA	**02**	1.13e−2	**02**	1.66e−2	**01**	9.48e−1	**02**	1.66e−1	**03**	9.75e−1	**2.0**
WGGP_TS_SS	**01**	1.12e−2	**01**	1.65e−2	**02**	9.47e−1	**01**	1.65e−1	**04**	9.75e−1	**1.8**

Table F.15: Results of the statistics on the first aggregation level (categorical columns) for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	22	1.62e−2	22	1.94e−2	21	7.40e−1	16	8.70e−2	14	9.57e−1	19.0
SGAN_NO_AS	**06**	6.97e−3	**07**	8.10e−3	**05**	9.11e−1	**07**	3.93e−2	**07**	9.73e−1	**6.4**
SGAN_NO_SA	21	1.59e−2	21	1.91e−2	22	7.32e−1	15	8.63e−2	15	9.56e−1	18.8
SGAN_NO_SS	**08**	7.02e−3	**08**	8.14e−3	**06**	9.04e−1	**08**	3.97e−2	**09**	9.71e−1	**7.8**
SGAN_OS_AA	20	1.57e−2	20	1.84e−2	26	−4.69e−1	22	1.07e−1	36	8.77e−1	24.8
SGAN_OS_AS	32	3.48e−2	32	3.99e−2	20	7.86e−1	32	1.80e−1	34	9.10e−1	30.0
SGAN_OS_SA	19	1.57e−2	19	1.83e−2	25	−4.56e−1	21	1.07e−1	35	8.84e−1	23.8
SGAN_OS_SS	31	3.46e−2	31	3.96e−2	19	7.96e−1	30	1.79e−1	32	9.28e−1	28.6
SGAN_TS_AA	17	1.13e−2	17	1.33e−2	15	8.10e−1	13	6.68e−2	23	9.45e−1	17.0
SGAN_TS_AS	**10**	7.15e−3	**10**	8.20e−3	12	8.51e−1	**10**	4.29e−2	21	9.47e−1	12.6
SGAN_TS_SA	18	1.17e−2	18	1.37e−2	17	8.08e−1	14	6.77e−2	24	9.44e−1	18.2
SGAN_TS_SS	**07**	6.97e−3	**06**	8.05e−3	11	8.63e−1	**09**	4.23e−2	18	9.54e−1	10.2
WGAN_NO_AA	36	6.15e−2	36	7.07e−2	29	−9.35e−1	36	3.07e−1	12	9.59e−1	29.8
WGAN_NO_AS	**01**	5.26e−3	**02**	6.17e−3	**04**	9.14e−1	**03**	3.23e−2	**03**	9.76e−1	**2.6**
WGAN_NO_SA	35	6.14e−2	35	7.05e−2	30	−9.37e−1	35	3.07e−1	13	9.58e−1	29.6
WGAN_NO_SS	**02**	5.29e−3	**01**	6.15e−3	**03**	9.19e−1	**01**	3.20e−2	**04**	9.76e−1	**2.2**
WGAN_OS_AA	28	2.82e−2	28	3.33e−2	27	−6.32e−1	26	1.72e−1	22	9.46e−1	26.2
WGAN_OS_AS	29	3.36e−2	29	3.84e−2	13	8.18e−1	27	1.74e−1	29	9.32e−1	25.4

Continues on next page...

Table F.15 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
WGAN_OS_SA	27	2.81e−2	27	3.33e−2	28	−6.34e−1	25	1.72e−1	20	9.47e−1	25.4
WGAN_OS_SS	30	3.39e−2	30	3.88e−2	16	8.09e−1	28	1.76e−1	27	9.35e−1	26.2
WGAN_TS_AA	26	2.69e−2	26	3.13e−2	36	−2.89	31	1.79e−1	**10**	9.61e−1	25.8
WGAN_TS_AS	**03**	5.71e−3	**03**	6.60e−3	**01**	9.31e−1	**02**	3.20e−2	**01**	9.79e−1	**2.0**
WGAN_TS_SA	25	2.67e−2	25	3.10e−2	35	−2.87	29	1.78e−1	11	9.60e−1	25.0
WGAN_TS_SS	**04**	5.79e−3	**04**	6.72e−3	**02**	9.23e−1	**04**	3.23e−2	**02**	9.77e−1	**3.2**
WGGP_NO_AA	13	1.02e−2	13	1.16e−2	34	−1.77	17	9.02e−2	19	9.53e−1	19.2
WGGP_NO_AS	**05**	6.78e−3	**05**	7.74e−3	**08**	8.82e−1	**05**	3.80e−2	**06**	9.73e−1	**5.8**
WGGP_NO_SA	15	1.04e−2	14	1.19e−2	33	−1.73	18	9.15e−2	16	9.56e−1	19.2
WGGP_NO_SS	**09**	7.12e−3	**09**	8.14e−3	**07**	8.88e−1	**06**	3.90e−2	**05**	9.75e−1	**7.2**
WGGP_OS_AA	24	1.95e−2	24	2.23e−2	24	9.64e−3	24	1.16e−1	26	9.37e−1	24.4
WGGP_OS_AS	34	3.53e−2	34	4.03e−2	18	8.03e−1	34	1.81e−1	30	9.32e−1	30.0
WGGP_OS_SA	23	1.92e−2	23	2.19e−2	23	1.54e−2	23	1.15e−1	33	9.28e−1	25.0
WGGP_OS_SS	33	3.49e−2	33	4.00e−2	14	8.12e−1	33	1.81e−1	28	9.34e−1	28.2
WGGP_TS_AA	16	1.04e−2	16	1.19e−2	31	−1.30	19	1.02e−1	31	9.29e−1	22.6
WGGP_TS_AS	12	7.64e−3	12	8.90e−3	**10**	8.75e−1	12	4.59e−2	17	9.55e−1	12.6
WGGP_TS_SA	14	1.03e−2	15	1.19e−2	32	−1.31	20	1.02e−1	25	9.40e−1	21.2
WGGP_TS_SS	11	7.56e−3	11	8.80e−3	**09**	8.78e−1	11	4.57e−2	**08**	9.71e−1	**10.0**

Table F.16: Results of the statistics on the second aggregation level for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	21	6.14e−3	21	1.09e−2	13	8.69e−1	17	4.03e−1	**09**	9.58e−1	16.2
SGAN_NO_AS	**09**	5.40e−3	11	9.56e−3	**06**	8.94e−1	11	3.72e−1	**06**	9.62e−1	**8.6**
SGAN_NO_SA	19	5.99e−3	18	1.05e−2	**07**	8.92e−1	14	3.85e−1	**05**	9.64e−1	12.6
SGAN_NO_SS	**08**	5.33e−3	**09**	9.32e−3	**04**	9.15e−1	**05**	3.59e−1	**03**	9.68e−1	**5.8**
SGAN_OS_AA	22	6.37e−3	22	1.13e−2	26	7.08e−1	24	4.58e−1	30	9.35e−1	24.8
SGAN_OS_AS	31	9.39e−3	30	1.51e−2	20	8.30e−1	26	5.14e−1	25	9.45e−1	26.4
SGAN_OS_SA	20	6.03e−3	20	1.05e−2	24	7.50e−1	20	4.21e−1	26	9.42e−1	22.0
SGAN_OS_SS	30	9.09e−3	29	1.45e−2	15	8.62e−1	25	4.83e−1	15	9.52e−1	22.8
SGAN_TS_AA	17	5.82e−3	19	1.05e−2	22	8.27e−1	19	4.18e−1	20	9.50e−1	19.4
SGAN_TS_AS	11	5.45e−3	15	9.75e−3	18	8.44e−1	16	4.00e−1	16	9.52e−1	15.2
SGAN_TS_SA	12	5.51e−3	13	9.67e−3	16	8.54e−1	13	3.81e−1	11	9.55e−1	13.0

Continues on next page...

Table F.16 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_TS_SS	**06**	5.11e – 3	**07**	8.94e – 3	12	8.72e – 1	**07**	3.63e – 1	**07**	9.58e – 1	**7.8**
WGAN_NO_AA	36	1.63e – 2	36	3.13e – 2	36	1.27e – 1	36	1.09	36	8.76e – 1	36.0
WGAN_NO_AS	32	9.52e – 3	34	1.86e – 2	34	5.18e – 1	34	7.77e – 1	35	8.95e – 1	33.8
WGAN_NO_SA	35	1.43e – 2	35	2.52e – 2	33	5.22e – 1	35	8.02e – 1	34	9.20e – 1	34.4
WGAN_NO_SS	23	7.42e – 3	24	1.33e – 2	17	8.51e – 1	29	5.19e – 1	27	9.42e – 1	24.0
WGAN_OS_AA	29	8.92e – 3	32	1.65e – 2	29	6.29e – 1	33	6.13e – 1	32	9.33e – 1	31.0
WGAN_OS_AS	34	1.03e – 2	33	1.70e – 2	23	7.77e – 1	32	5.94e – 1	33	9.31e – 1	31.0
WGAN_OS_SA	26	8.29e – 3	28	1.44e – 2	25	7.41e – 1	28	5.18e – 1	19	9.50e – 1	25.2
WGAN_OS_SS	33	9.82e – 3	31	1.56e – 2	14	8.67e – 1	30	5.26e – 1	21	9.49e – 1	25.8
WGAN_TS_AA	25	7.99e – 3	27	1.42e – 2	35	4.90e – 1	31	5.49e – 1	29	9.37e – 1	29.4
WGAN_TS_AS	15	5.70e – 3	17	1.02e – 2	11	8.80e – 1	15	3.94e – 1	12	9.55e – 1	14.0
WGAN_TS_SA	24	7.92e – 3	26	1.35e – 2	32	5.43e – 1	27	5.17e – 1	23	9.47e – 1	26.4
WGAN_TS_SS	16	5.70e – 3	16	9.88e – 3	**03**	9.20e – 1	12	3.75e – 1	**04**	9.66e – 1	10.2
WGGP_NO_AA	13	5.65e – 3	14	9.71e – 3	31	5.90e – 1	22	4.29e – 1	31	9.35e – 1	22.2
WGGP_NO_AS	**07**	5.12e – 3	**06**	8.93e – 3	**10**	8.82e – 1	**09**	3.63e – 1	17	9.51e – 1	**9.8**
WGGP_NO_SA	**10**	5.42e – 3	**10**	9.39e – 3	30	6.07e – 1	18	4.15e – 1	28	9.41e – 1	19.2
WGGP_NO_SS	**04**	4.87e – 3	**05**	8.54e – 3	**05**	8.98e – 1	**03**	3.45e – 1	**08**	9.58e – 1	**5.0**
WGGP_OS_AA	18	5.86e – 3	12	9.58e – 3	21	8.30e – 1	**06**	3.61e – 1	18	9.51e – 1	15.0
WGGP_OS_AS	28	8.58e – 3	25	1.34e – 2	**09**	8.88e – 1	23	4.37e – 1	13	9.55e – 1	19.6
WGGP_OS_SA	14	5.69e – 3	**08**	9.30e – 3	19	8.34e – 1	**04**	3.52e – 1	14	9.53e – 1	11.8
WGGP_OS_SS	27	8.41e – 3	23	1.31e – 2	**08**	8.91e – 1	21	4.28e – 1	**10**	9.57e – 1	17.8
WGGP_TS_AA	**05**	4.91e – 3	**04**	8.18e – 3	28	6.55e – 1	**10**	3.67e – 1	24	9.46e – 1	14.2
WGGP_TS_AS	**02**	4.31e – 3	**02**	7.24e – 3	**02**	9.29e – 1	**02**	2.82e – 1	**02**	9.69e – 1	**2.0**
WGGP_TS_SA	**03**	4.76e – 3	**03**	8.02e – 3	27	6.55e – 1	**08**	3.63e – 1	22	9.48e – 1	12.6
WGGP_TS_SS	**01**	4.19e – 3	**01**	7.12e – 3	**01**	9.30e – 1	**01**	2.79e – 1	**01**	9.70e – 1	**1.0**

Table F.17: Results of the statistics on the third aggregation level for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_NO_AA	18	2.16e – 3	20	4.84e – 3	15	8.30e – 1	16	7.36e – 1	11	9.41e – 1	16.0
SGAN_NO_AS	13	2.06e – 3	12	4.46e – 3	11	8.55e – 1	**08**	6.92e – 1	**08**	9.45e – 1	10.4
SGAN_NO_SA	16	2.11e – 3	15	4.57e – 3	**07**	8.66e – 1	**10**	6.94e – 1	**04**	9.50e – 1	10.4
SGAN_NO_SS	11	2.04e – 3	11	4.36e – 3	**04**	8.84e – 1	**06**	6.72e – 1	**03**	9.53e – 1	**7.0**
SGAN_OS_AA	22	2.31e – 3	22	5.22e – 3	29	7.10e – 1	24	8.67e – 1	31	9.14e – 1	25.6

Continues on next page...

Table F.17 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
SGAN_OS_AS	31	3.03e − 3	30	6.29e − 3	21	7.93e − 1	28	9.40e − 1	26	9.24e − 1	27.2
SGAN_OS_SA	19	2.18e − 3	19	4.77e − 3	23	7.71e − 1	19	7.91e − 1	24	9.26e − 1	20.8
SGAN_OS_SS	30	2.93e − 3	26	6.00e − 3	16	8.29e − 1	25	9.00e − 1	17	9.33e − 1	22.8
SGAN_TS_AA	20	2.19e − 3	21	4.99e − 3	24	7.70e − 1	22	8.04e − 1	23	9.27e − 1	22.0
SGAN_TS_AS	14	2.09e − 3	16	4.67e − 3	20	7.95e − 1	17	7.75e − 1	19	9.30e − 1	17.2
SGAN_TS_SA	12	2.05e − 3	14	4.53e − 3	18	8.13e − 1	15	7.33e − 1	16	9.36e − 1	15.0
SGAN_TS_SS	**09**	1.96e − 3	**09**	4.28e − 3	14	8.35e − 1	12	7.10e − 1	13	9.40e − 1	11.4
WGAN_NO_AA	36	5.00e − 3	36	1.38e − 2	36	−1.01e − 1	36	1.98	36	8.33e − 1	36.0
WGAN_NO_AS	34	3.41e − 3	34	8.55e − 3	35	3.75e − 1	34	1.41	35	8.58e − 1	34.4
WGAN_NO_SA	35	4.42e − 3	35	1.06e − 2	34	5.02e − 1	35	1.42	34	8.85e − 1	34.6
WGAN_NO_SS	28	2.82e − 3	29	6.29e − 3	19	7.97e − 1	30	9.81e − 1	30	9.14e − 1	27.2
WGAN_OS_AA	29	2.89e − 3	33	7.10e − 3	32	6.27e − 1	33	1.08	32	9.10e − 1	31.8
WGAN_OS_AS	33	3.31e − 3	32	6.97e − 3	27	7.36e − 1	32	1.05	33	9.04e − 1	31.4
WGAN_OS_SA	27	2.76e − 3	27	6.10e − 3	22	7.75e − 1	26	9.10e − 1	18	9.32e − 1	24.0
WGAN_OS_SS	32	3.22e − 3	31	6.58e − 3	17	8.25e − 1	31	9.88e − 1	22	9.27e − 1	26.6
WGAN_TS_AA	24	2.62e − 3	28	6.14e − 3	33	6.18e − 1	29	9.80e − 1	28	9.16e − 1	28.4
WGAN_TS_AS	17	2.13e − 3	18	4.73e − 3	13	8.45e − 1	14	7.29e − 1	14	9.40e − 1	15.2
WGAN_TS_SA	26	2.66e − 3	25	5.78e − 3	31	6.91e − 1	27	9.14e − 1	25	9.25e − 1	26.8
WGAN_TS_SS	21	2.21e − 3	17	4.72e − 3	**03**	8.86e − 1	13	7.19e − 1	**05**	9.49e − 1	11.8
WGGP_NO_AA	15	2.09e − 3	13	4.51e − 3	30	7.03e − 1	23	8.05e − 1	29	9.16e − 1	22.0
WGGP_NO_AS	**07**	1.91e − 3	**08**	4.15e − 3	12	8.55e − 1	**07**	6.89e − 1	15	9.38e − 1	**9.8**
WGGP_NO_SA	**10**	1.99e − 3	**10**	4.33e − 3	28	7.21e − 1	18	7.80e − 1	27	9.23e − 1	18.6
WGGP_NO_SS	**04**	1.81e − 3	**06**	3.94e − 3	**05**	8.72e − 1	**05**	6.56e − 1	**07**	9.46e − 1	**5.4**
WGGP_OS_AA	**08**	1.95e − 3	**07**	4.02e − 3	**10**	8.62e − 1	**04**	6.55e − 1	12	9.40e − 1	**8.2**
WGGP_OS_AS	25	2.65e − 3	24	5.38e − 3	**08**	8.66e − 1	21	8.03e − 1	**09**	9.44e − 1	17.4
WGGP_OS_SA	**06**	1.90e − 3	**05**	3.93e − 3	**09**	8.65e − 1	**03**	6.47e − 1	**10**	9.42e − 1	**6.6**
WGGP_OS_SS	23	2.60e − 3	23	5.27e − 3	**06**	8.69e − 1	20	7.92e − 1	**06**	9.46e − 1	15.6
WGGP_TS_AA	**05**	1.83e − 3	**04**	3.84e − 3	26	7.44e − 1	11	7.00e − 1	21	9.28e − 1	13.4
WGGP_TS_AS	**02**	1.62e − 3	**02**	3.37e − 3	**02**	9.09e − 1	**02**	5.37e − 1	**02**	9.59e − 1	**2.0**
WGGP_TS_SA	**03**	1.77e − 3	**03**	3.76e − 3	25	7.46e − 1	**09**	6.94e − 1	20	9.30e − 1	12.0
WGGP_TS_SS	**01**	1.57e − 3	**01**	3.32e − 3	**01**	9.11e − 1	**01**	5.34e − 1	**01**	9.60e − 1	**1.0**

Table F.18: Results of the ML efficacy for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
SGAN_NO_AA	18	4.04e1	24	5.66	21.0
SGAN_NO_AS	20	4.27e1	12	4.34	16.0
SGAN_NO_SA	29	6.41e1	17	4.94	23.0
SGAN_NO_SS	30	6.55e1	**10**	4.07	20.0
SGAN_OS_AA	11	3.41e1	19	5.10	15.0
SGAN_OS_AS	16	3.78e1	20	5.11	18.0
SGAN_OS_SA	24	4.66e1	21	5.12	22.5
SGAN_OS_SS	26	4.85e1	18	4.96	22.0
SGAN_TS_AA	12	3.60e1	16	4.70	14.0
SGAN_TS_AS	15	3.73e1	11	4.16	13.0
SGAN_TS_SA	22	4.50e1	15	4.64	18.5
SGAN_TS_SS	23	4.57e1	**09**	4.01	16.0
WGAN_NO_AA	27	5.88e1	36	4.79e4	31.5
WGAN_NO_AS	28	6.08e1	22	5.59	25.0
WGAN_NO_SA	35	1.01e2	35	4.74e4	35.0
WGAN_NO_SS	36	1.03e2	23	5.62	29.5
WGAN_OS_AA	21	4.42e1	32	6.48e3	26.5
WGAN_OS_AS	25	4.80e1	26	5.79	25.5
WGAN_OS_SA	33	8.27e1	31	6.40e3	32.0
WGAN_OS_SS	34	8.64e1	25	5.77	29.5
WGAN_TS_AA	17	3.84e1	33	1.93e4	25.0
WGAN_TS_AS	19	4.13e1	13	4.59	16.0
WGAN_TS_SA	31	6.74e1	34	1.95e4	32.5
WGAN_TS_SS	32	6.76e1	14	4.59	23.0
WGGP_NO_AA	**09**	3.39e1	30	5.60e3	19.5
WGGP_NO_AS	**10**	3.39e1	**04**	2.60	**7.0**
WGGP_NO_SA	13	3.61e1	29	3.20e3	21.0
WGGP_NO_SS	14	3.63e1	**03**	2.57	**8.5**
WGGP_OS_AA	**01**	2.36e1	**06**	2.70	**3.5**
WGGP_OS_AS	**02**	2.43e1	**08**	3.10	**5.0**
WGGP_OS_SA	**03**	2.52e1	**05**	2.68	**4.0**
WGGP_OS_SS	**04**	2.59e1	**07**	3.10	**5.5**
WGGP_TS_AA	**05**	2.59e1	27	1.36e3	16.0
WGGP_TS_AS	**06**	2.62e1	**02**	2.54	**4.0**

Continues on next page...

Table F.18 – continued from previous page

Name	Continuous		Categorical		rank
WGGP_TS_SA	**07**	2.88e1	28	3.04e3	17.5
WGGP_TS_SS	**08**	2.90e1	**01**	2.54	**4.5**

F.2 Comparison with state-of-the-art models

The comparison between DATGAN and the state-of-the-art generative models is performed on the four case studies: CMAP, LPMC, LPMC_half, and ADULT. For each case study, we provide two tables:

1. Statistical assessments
2. ML efficacy metrics

CMAP case study

Table F.19: Results of the statistical assessments between the best DATGAN version and the state-of-the-art models for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
CTABGAN	**03**	2.55e − 2	**03**	3.18e − 2	**03**	8.17e − 1	**03**	1.97e − 1	**04**	9.30e − 1	**3.2**
CTGAN	**05**	3.59e − 2	**05**	4.31e − 2	**04**	6.37e − 1	**04**	2.59e − 1	**05**	8.54e − 1	**4.6**
DATGAN (WGAN)	**01**	6.33e − 3	**01**	7.76e − 3	**01**	9.88e − 1	**01**	5.32e − 2	**01**	9.96e − 1	**1.0**
TGAN	**02**	1.16e − 2	**02**	1.50e − 2	**02**	9.49e − 1	**02**	1.14e − 1	**02**	9.81e − 1	**2.0**
TVAE	**04**	3.33e − 2	**04**	4.21e − 2	**05**	5.59e − 1	**05**	2.72e − 1	**03**	9.37e − 1	**4.2**
First aggregation level - continuous columns											
CTABGAN	**02**	1.23e − 2	**02**	1.71e − 2	**02**	9.37e − 1	**02**	1.71e − 1	**03**	9.73e − 1	**2.2**
CTGAN	**04**	2.10e − 2	**04**	2.87e − 2	**04**	8.27e − 1	**04**	2.87e − 1	**04**	9.24e − 1	**4.0**
DATGAN (WGAN)	**01**	7.98e − 3	**01**	1.05e − 2	**01**	9.70e − 1	**01**	1.05e − 1	**01**	9.86e − 1	**1.0**
TGAN	**05**	2.21e − 2	**05**	3.07e − 2	**05**	7.98e − 1	**05**	3.07e − 1	**05**	9.18e − 1	**5.0**
TVAE	**03**	1.32e − 2	**03**	1.72e − 2	**03**	9.32e − 1	**03**	1.72e − 1	**02**	9.80e − 1	**2.8**
First aggregation level - categorical columns											
CTABGAN	**03**	2.88e − 2	**03**	3.54e − 2	**03**	7.87e − 1	**03**	2.03e − 1	**04**	9.19e − 1	**3.2**
CTGAN	**05**	3.97e − 2	**04**	4.67e − 2	**04**	5.90e − 1	**04**	2.52e − 1	**05**	8.36e − 1	**4.4**
DATGAN (WGAN)	**01**	5.92e − 3	**01**	7.06e − 3	**01**	9.92e − 1	**01**	4.01e − 2	**01**	9.98e − 1	**1.0**

Continues on next page...

Table F.19 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
TGAN	**02**	9.01e−3	**02**	1.10e−2	**02**	9.87e−1	**02**	6.63e−2	**02**	9.96e−1	**2.0**
TVAE	**04**	3.83e−2	**05**	4.84e−2	**05**	4.66e−1	**05**	2.97e−1	**03**	9.27e−1	**4.4**
Second aggregation level											
CTABGAN	**03**	8.17e−3	**03**	1.32e−2	**03**	8.61e−1	**03**	4.39e−1	**03**	9.43e−1	**3.0**
CTGAN	**04**	1.04e−2	**04**	1.68e−2	**04**	7.79e−1	**04**	5.53e−1	**05**	9.03e−1	**4.2**
DATGAN (WGAN)	**01**	2.83e−3	**01**	4.27e−3	**01**	9.83e−1	**01**	1.56e−1	**01**	9.92e−1	**1.0**
TGAN	**02**	4.73e−3	**02**	7.66e−3	**02**	9.29e−1	**02**	2.92e−1	**02**	9.71e−1	**2.0**
TVAE	**05**	1.15e−2	**05**	1.96e−2	**05**	5.92e−1	**05**	6.64e−1	**04**	9.13e−1	**4.8**
Third aggregation level											
CTABGAN	**03**	2.36e−3	**03**	4.74e−3	**03**	8.30e−1	**03**	7.47e−1	**03**	9.26e−1	**3.0**
CTGAN	**04**	2.86e−3	**04**	5.81e−3	**04**	7.51e−1	**04**	9.15e−1	**04**	8.85e−1	**4.0**
DATGAN (WGAN)	**01**	1.09e−3	**01**	1.92e−3	**01**	9.65e−1	**01**	3.31e−1	**01**	9.83e−1	**1.0**
TGAN	**02**	1.63e−3	**02**	3.24e−3	**02**	8.93e−1	**02**	5.43e−1	**02**	9.57e−1	**2.0**
TVAE	**05**	3.47e−3	**05**	7.79e−3	**05**	4.60e−1	**05**	1.18	**05**	8.80e−1	**5.0**

Table F.20: Results of the ML efficacy between the best DATGAN version and the state-of-the-art models for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
CTABGAN	**03**	3.33	**02**	1.43	**2.5**
CTGAN	**05**	3.45	**03**	1.49	**4.0**
DATGAN (WGAN)	**01**	3.05	**01**	7.56e−1	**1.0**
TGAN	**02**	3.24	**04**	8.10e1	**3.0**
TVAE	**04**	3.36	**05**	8.04e2	**4.5**

LPMC case study

Table F.21: Results of the statistical assessments between the best DATGAN version and the state-of-the-art models for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
CTABGAN	**04**	2.72e−2	**04**	3.47e−2	**05**	−3.97	**04**	2.60e−1	**05**	8.22e−1	**4.4**

Continues on next page...

Table F.21 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
CTGAN	**03**	2.62e−2	**03**	3.28e−2	**04**	−2.92	**03**	2.27e−1	**04**	8.71e−1	**3.4**
DATGAN (WGGP)	**01**	7.64e−3	**01**	1.02e−2	**01**	9.63e−1	**01**	8.13e−2	**01**	9.85e−1	**1.0**
TGAN	**02**	1.19e−2	**02**	1.62e−2	**02**	5.88e−1	**02**	1.48e−1	**02**	9.45e−1	**2.0**
TVAE	**05**	2.79e−2	**05**	3.71e−2	**03**	−2.24	**05**	2.76e−1	**03**	8.96e−1	**4.2**
First aggregation level - continuous columns											
CTABGAN	**04**	1.68e−2	**04**	2.60e−2	**02**	9.44e−1	**04**	2.60e−1	**02**	9.80e−1	**3.2**
CTGAN	**02**	1.57e−2	**02**	2.35e−2	**03**	9.25e−1	**02**	2.35e−1	**03**	9.70e−1	**2.4**
DATGAN (WGGP)	**01**	8.93e−3	**01**	1.31e−2	**01**	9.69e−1	**01**	1.31e−1	**01**	9.86e−1	**1.0**
TGAN	**03**	1.68e−2	**03**	2.46e−2	**05**	8.72e−1	**03**	2.46e−1	**05**	9.50e−1	**3.8**
TVAE	**05**	1.72e−2	**05**	2.79e−2	**04**	8.98e−1	**05**	2.79e−1	**04**	9.58e−1	**4.6**
First aggregation level - categorical columns											
CTABGAN	**04**	3.68e−2	**04**	4.28e−2	**05**	−8.54	**04**	2.59e−1	**05**	6.76e−1	**4.4**
CTGAN	**03**	3.60e−2	**03**	4.14e−2	**04**	−6.48	**03**	2.18e−1	**04**	7.79e−1	**3.4**
DATGAN (WGGP)	**01**	6.44e−3	**01**	7.47e−3	**01**	9.56e−1	**01**	3.52e−2	**01**	9.84e−1	**1.0**
TGAN	**02**	7.39e−3	**02**	8.45e−3	**02**	3.24e−1	**02**	5.69e−2	**02**	9.40e−1	**2.0**
TVAE	**05**	3.79e−2	**05**	4.56e−2	**03**	−5.16	**05**	2.74e−1	**03**	8.39e−1	**4.2**
Second aggregation level											
CTABGAN	**04**	8.63e−3	**04**	1.50e−2	**05**	3.11e−2	**04**	6.13e−1	**05**	8.93e−1	**4.4**
CTGAN	**03**	8.30e−3	**03**	1.42e−2	**03**	4.28e−1	**03**	5.39e−1	**03**	9.07e−1	**3.0**
DATGAN (WGGP)	**01**	3.52e−3	**01**	5.91e−3	**01**	9.51e−1	**01**	2.26e−1	**01**	9.78e−1	**1.0**
TGAN	**02**	4.83e−3	**02**	8.48e−3	**02**	8.19e−1	**02**	3.74e−1	**02**	9.47e−1	**2.0**
TVAE	**05**	9.77e−3	**05**	1.77e−2	**04**	1.34e−1	**05**	6.82e−1	**04**	8.95e−1	**4.6**
Third aggregation level											
CTABGAN	**04**	2.52e−3	**04**	5.82e−3	**04**	5.09e−1	**04**	1.10	**04**	8.75e−1	**4.0**
CTGAN	**03**	2.44e−3	**03**	5.54e−3	**03**	7.07e−1	**03**	9.67e−1	**03**	8.99e−1	**3.0**
DATGAN (WGGP)	**01**	1.32e−3	**01**	2.79e−3	**01**	9.35e−1	**01**	4.50e−1	**01**	9.69e−1	**1.0**
TGAN	**02**	1.71e−3	**02**	3.76e−3	**02**	8.30e−1	**02**	6.89e−1	**02**	9.34e−1	**2.0**
TVAE	**05**	3.15e−3	**05**	7.54e−3	**05**	4.07e−1	**05**	1.21	**05**	8.70e−1	**5.0**

Table F.22: Results of the ML efficacy between the best DATGAN version and the state-of-the-art models for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
CTABGAN	**03**	3.49e1	**04**	5.48	**3.5**
CTGAN	**04**	4.33e1	**02**	2.89	**3.0**

Continues on next page...

Table F.22 – continued from previous page

Name	Continuous		Categorical		rank
DATGAN (WGGP)	**01**	2.20e1	**01**	2.19	**1.0**
TGAN	**02**	2.82e1	**03**	2.91	**2.5**
TVAE	**05**	4.91e1	**05**	4.88e2	**5.0**

LPMC_half case study

Table F.23: Results of the statistical assessments between the best DATGAN version and the state-of-the-art models for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
CTABGAN	**03**	2.21e−2	**03**	2.79e−2	**04**	−2.46	**03**	2.06e−1	**04**	8.54e−1	**3.4**
CTGAN	**04**	2.44e−2	**04**	3.00e−2	**03**	−1.58	**04**	2.06e−1	**05**	8.24e−1	**4.0**
DATGAN (WGGP)	**01**	9.32e−3	**01**	1.25e−2	**01**	9.11e−1	**01**	1.03e−1	**01**	9.73e−1	**1.0**
TGAN	**02**	1.40e−2	**02**	1.88e−2	**02**	4.04e−1	**02**	1.65e−1	**02**	9.39e−1	**2.0**
TVAE	**05**	2.97e−2	**05**	3.83e−2	**05**	−2.69	**05**	2.68e−1	**03**	8.71e−1	**4.6**
First aggregation level - continuous columns											
CTABGAN	**02**	1.43e−2	**03**	2.21e−2	**01**	9.61e−1	**03**	2.21e−1	**01**	9.86e−1	**2.0**
CTGAN	**03**	1.51e−2	**02**	2.19e−2	**03**	9.15e−1	**02**	2.19e−1	**03**	9.66e−1	**2.6**
DATGAN (WGGP)	**01**	1.12e−2	**01**	1.65e−2	**02**	9.47e−1	**01**	1.65e−1	**02**	9.75e−1	**1.4**
TGAN	**05**	1.81e−2	**05**	2.64e−2	**05**	8.46e−1	**05**	2.64e−1	**05**	9.36e−1	**5.0**
TVAE	**04**	1.62e−2	**04**	2.57e−2	**04**	9.11e−1	**04**	2.57e−1	**04**	9.60e−1	**4.0**
First aggregation level - categorical columns											
CTABGAN	**03**	2.93e−2	**03**	3.33e−2	**04**	−5.63	**03**	1.92e−1	**04**	7.31e−1	**3.4**
CTGAN	**04**	3.30e−2	**04**	3.76e−2	**03**	−3.89	**04**	1.94e−1	**05**	6.92e−1	**4.0**
DATGAN (WGGP)	**01**	7.56e−3	**01**	8.80e−3	**01**	8.78e−1	**01**	4.57e−2	**01**	9.71e−1	**1.0**
TGAN	**02**	1.03e−2	**02**	1.19e−2	**02**	−5.40e−	**02**	7.36e−2	**02**	9.42e−1	**2.0**
TVAE	**05**	4.22e−2	**05**	5.01e−2	**05**	−6.04	**05**	2.78e−1	**03**	7.89e−1	**4.6**
Second aggregation level											
CTABGAN	**03**	7.16e−3	**03**	1.24e−2	**04**	4.22e−1	**03**	5.06e−1	**03**	9.18e−1	**3.2**
CTGAN	**04**	8.05e−3	**04**	1.36e−2	**03**	5.59e−1	**04**	5.08e−1	**04**	9.09e−1	**3.8**
DATGAN (WGGP)	**01**	4.19e−3	**01**	7.12e−3	**01**	9.30e−1	**01**	2.79e−1	**01**	9.70e−1	**1.0**
TGAN	**02**	5.48e−3	**02**	9.52e−3	**02**	7.64e−1	**02**	4.13e−1	**02**	9.36e−1	**2.0**
TVAE	**05**	1.01e−2	**05**	1.81e−2	**05**	1.54e−1	**05**	6.54e−1	**05**	8.94e−1	**5.0**
Third aggregation level											

Continues on next page...

Table F.23 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
CTABGAN	**03**	2.21e−3	**03**	4.98e−3	**04**	6.85e−1	**03**	9.24e−1	**03**	9.04e−1	**3.2**
CTGAN	**04**	2.49e−3	**04**	5.52e−3	**03**	7.34e−1	**04**	9.24e−1	**04**	8.99e−1	**3.8**
DATGAN (WGGP)	**01**	1.57e−3	**01**	3.32e−3	**01**	9.11e−1	**01**	5.34e−1	**01**	9.60e−1	**1.0**
TGAN	**02**	1.94e−3	**02**	4.20e−3	**02**	7.85e−1	**02**	7.53e−1	**02**	9.20e−1	**2.0**
TVAE	**05**	3.24e−3	**05**	7.66e−3	**05**	4.22e−1	**05**	1.16	**05**	8.69e−1	**5.0**

Table F.24: Results of the ML efficacy between the best DATGAN version and the state-of-the-art models for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
CTABGAN	**04**	4.25e1	**04**	4.14	**4.0**
CTGAN	**05**	5.13e1	**02**	3.09	**3.5**
DATGAN (WGGP)	**01**	2.90e1	**01**	2.54	**1.0**
TGAN	**02**	3.05e1	**03**	3.34	**2.5**
TVAE	**03**	4.24e1	**05**	3.93e3	**4.0**

ADULT case study

Table F.25: Results of the statistical assessments between the best DATGAN version and the state-of-the-art models for the ADULT dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
CTABGAN	**06**	2.31e−2	**06**	3.04e−2	**06**	8.64e−1	**06**	2.79e−1	**06**	9.60e−1	**6.0**
CTGAN	**05**	1.98e−2	**05**	2.56e−2	**05**	9.40e−1	**05**	2.45e−1	**05**	9.79e−1	**5.0**
DATGAN (WGAN)	**01**	3.83e−3	**01**	4.99e−3	**01**	9.98e−1	**01**	4.12e−2	**01**	9.99e−1	**1.0**
DATGAN (WGGP)	**03**	1.01e−2	**03**	1.34e−2	**03**	9.81e−1	**03**	1.11e−1	**03**	9.96e−1	**3.0**
TGAN	**02**	5.01e−3	**02**	6.89e−3	**02**	9.94e−1	**02**	7.48e−2	**02**	9.98e−1	**2.0**
TVAE	**04**	1.06e−2	**04**	1.36e−2	**04**	9.64e−1	**04**	1.18e−1	**04**	9.84e−1	**4.0**
First aggregation level - continuous columns											
CTABGAN	**06**	1.56e−2	**06**	2.45e−2	**06**	9.26e−1	**06**	2.31e−1	**06**	9.77e−1	**6.0**
CTGAN	**04**	1.14e−2	**04**	1.70e−2	**05**	9.63e−1	**04**	1.66e−1	**05**	9.84e−1	**4.4**
DATGAN (WGAN)	**01**	5.88e−3	**01**	8.10e−3	**01**	9.96e−1	**01**	7.13e−2	**01**	9.98e−1	**1.0**

Continues on next page...

Table F.25 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
`DATGAN (WGGP)`	**05**	1.29e−2	**05**	2.03e−2	**04**	9.66e−1	**05**	1.85e−1	**04**	9.92e−1	**4.6**
`TGAN`	**02**	6.73e−3	**02**	1.00e−2	**02**	9.88e−1	**02**	9.62e−2	**02**	9.95e−1	**2.0**
`TVAE`	**03**	7.61e−3	**03**	1.09e−2	**03**	9.87e−1	**03**	1.04e−1	**03**	9.94e−1	**3.0**
First aggregation level - categorical columns											
`CTABGAN`	**06**	2.61e−2	**06**	3.28e−2	**06**	8.40e−1	**06**	2.99e−1	**06**	9.54e−1	**6.0**
`CTGAN`	**05**	2.32e−2	**05**	2.91e−2	**05**	9.31e−1	**05**	2.77e−1	**05**	9.78e−1	**5.0**
`DATGAN (WGAN)`	**01**	3.01e−3	**01**	3.74e−3	**01**	9.99e−1	**01**	2.91e−2	**01**	1.00	**1.0**
`DATGAN (WGGP)`	**03**	8.93e−3	**03**	1.07e−2	**03**	9.87e−1	**03**	8.13e−2	**03**	9.98e−1	**3.0**
`TGAN`	**02**	4.32e−3	**02**	5.65e−3	**02**	9.97e−1	**02**	6.62e−2	**02**	9.99e−1	**2.0**
`TVAE`	**04**	1.18e−2	**04**	1.46e−2	**04**	9.55e−1	**04**	1.24e−1	**04**	9.80e−1	**4.0**
Second aggregation level											
`CTABGAN`	**06**	7.18e−3	**06**	1.47e−2	**06**	8.42e−1	**06**	9.00e−1	**06**	9.39e−1	**6.0**
`CTGAN`	**05**	5.94e−3	**05**	1.21e−2	**05**	9.25e−1	**05**	7.44e−1	**05**	9.70e−1	**5.0**
`DATGAN (WGAN)`	**02**	1.82e−3	**01**	3.23e−3	**01**	9.95e−1	**01**	1.69e−1	**01**	9.98e−1	**1.2**
`DATGAN (WGGP)`	**03**	3.22e−3	**03**	6.45e−3	**03**	9.78e−1	**03**	3.45e−1	**03**	9.93e−1	**3.0**
`TGAN`	**01**	1.76e−3	**02**	3.54e−3	**02**	9.91e−1	**02**	2.31e−1	**02**	9.96e−1	**1.8**
`TVAE`	**04**	3.46e−3	**04**	6.78e−3	**04**	9.60e−1	**04**	3.78e−1	**04**	9.81e−1	**4.0**
Third aggregation level											
`CTABGAN`	**06**	2.06e−3	**06**	6.43e−3	**06**	7.82e−1	**06**	1.82	**06**	9.12e−1	**6.0**
`CTGAN`	**05**	1.70e−3	**05**	5.26e−3	**05**	8.96e−1	**05**	1.46	**05**	9.56e−1	**5.0**
`DATGAN (WGAN)`	**02**	6.16e−4	**01**	1.57e−3	**01**	9.90e−1	**01**	4.18e−1	**01**	9.96e−1	**1.2**
`DATGAN (WGGP)`	**03**	9.61e−4	**03**	2.79e−3	**03**	9.68e−1	**03**	7.11e−1	**03**	9.89e−1	**3.0**
`TGAN`	**01**	5.90e−4	**02**	1.66e−3	**02**	9.87e−1	**02**	4.55e−1	**02**	9.94e−1	**1.8**
`TVAE`	**04**	1.09e−3	**04**	3.13e−3	**04**	9.45e−1	**04**	7.70e−1	**04**	9.73e−1	**4.0**

Table F.26: Results of the ML efficacy between the best DATGAN version and the state-of-the-art models for the ADULT dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
`CTABGAN`	**05**	4.73	**04**	8.56e3	**4.5**
`CTGAN`	**03**	4.32	**03**	1.04e3	**3.0**
`DATGAN (WGAN)`	**04**	4.41	**02**	1.91	**3.0**
`DATGAN (WGGP)`	**06**	4.81	**01**	1.09	**3.5**
`TGAN`	**01**	4.13	**06**	1.00e4	**3.5**
`TVAE`	**02**	4.31	**05**	8.64e3	**3.5**

F.3 Sensitivity anaylsis of the DAG

The sensitivity analysis of the DAG is performed on three case studies: CMAP, LPMC, and LPMC_half. For each case study, we provide two tables:

1. Statistical assessments
2. ML efficacy metrics

CMAP case study

Table F.27: Results of the statistical assessments with the different DAGs for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
`full`	**05**	7.31e − 3	**05**	9.45e − 3	**03**	9.85e − 1	**05**	6.49e − 2	**03**	9.95e − 1	**4.2**
`trans. red.`	**03**	6.70e − 3	**04**	8.68e − 3	**01**	9.88e − 1	**03**	6.19e − 2	**01**	9.95e − 1	**2.4**
`linear`	**04**	6.71e − 3	**03**	8.48e − 3	**02**	9.86e − 1	**02**	6.00e − 2	**02**	9.95e − 1	**2.6**
`prediction`	**02**	6.49e − 3	**02**	8.01e − 3	**05**	9.72e − 1	**04**	6.25e − 2	**05**	9.88e − 1	**3.6**
`no links`	**01**	6.11e − 3	**01**	7.57e − 3	**04**	9.78e − 1	**01**	5.90e − 2	**04**	9.90e − 1	**2.2**
First aggregation level - continuous columns											
`full`	**01**	8.75e − 3	**01**	1.26e − 2	**02**	9.61e − 1	**01**	1.26e − 1	**03**	9.81e − 1	**1.6**
`trans. red.`	**02**	9.34e − 3	**03**	1.35e − 2	**01**	9.63e − 1	**03**	1.35e − 1	**01**	9.83e − 1	**2.0**
`linear`	**03**	9.61e − 3	**02**	1.32e − 2	**03**	9.61e − 1	**02**	1.32e − 1	**02**	9.83e − 1	**2.4**
`prediction`	**05**	1.56e − 2	**05**	1.96e − 2	**05**	8.82e − 1	**05**	1.96e − 1	**05**	9.42e − 1	**5.0**
`no links`	**04**	1.39e − 2	**04**	1.77e − 2	**04**	9.06e − 1	**04**	1.77e − 1	**04**	9.54e − 1	**4.0**
First aggregation level - categorical columns											
`full`	**05**	6.94e − 3	**05**	8.65e − 3	**05**	9.91e − 1	**05**	4.95e − 2	**04**	9.98e − 1	**4.8**
`trans. red.`	**04**	6.04e − 3	**04**	7.47e − 3	**02**	9.94e − 1	**04**	4.36e − 2	**03**	9.98e − 1	**3.4**
`linear`	**03**	5.99e − 3	**03**	7.30e − 3	**04**	9.93e − 1	**03**	4.20e − 2	**05**	9.98e − 1	**3.6**
`prediction`	**02**	4.20e − 3	**02**	5.10e − 3	**03**	9.94e − 1	**01**	2.91e − 2	**01**	9.99e − 1	**1.8**
`no links`	**01**	4.16e − 3	**01**	5.03e − 3	**01**	9.96e − 1	**02**	2.94e − 2	**02**	9.99e − 1	**1.4**
Second aggregation level											
`full`	**03**	3.22e − 3	**03**	5.08e − 3	**02**	9.77e − 1	**03**	1.85e − 1	**02**	9.89e − 1	**2.6**
`trans. red.`	**01**	3.13e − 3	**01**	4.93e − 3	**01**	9.78e − 1	**01**	1.81e − 1	**01**	9.90e − 1	**1.0**
`linear`	**02**	3.21e − 3	**02**	4.95e − 3	**03**	9.76e − 1	**02**	1.84e − 1	**03**	9.89e − 1	**2.4**

Continues on next page...

Table F.27 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
`prediction`	**04**	5.87e − 3	**04**	9.27e − 3	**05**	9.14e − 1	**05**	3.48e − 1	**05**	9.56e − 1	**4.6**
`no links`	**05**	6.03e − 3	**05**	9.49e − 3	**04**	9.16e − 1	**04**	3.48e − 1	**04**	9.57e − 1	**4.4**
Third aggregation level											
`full`	**02**	1.19e − 3	**02**	2.20e − 3	**02**	9.57e − 1	**02**	3.80e − 1	**02**	9.79e − 1	**2.0**
`trans. red.`	**01**	1.19e − 3	**01**	2.19e − 3	**01**	9.58e − 1	**01**	3.76e − 1	**01**	9.80e − 1	**1.0**
`linear`	**03**	1.23e − 3	**03**	2.24e − 3	**03**	9.54e − 1	**03**	3.90e − 1	**03**	9.78e − 1	**3.0**
`prediction`	**04**	2.36e − 3	**04**	4.55e − 3	**04**	8.31e − 1	**04**	7.85e − 1	**04**	9.11e − 1	**4.0**
`no links`	**05**	2.45e − 3	**05**	4.70e − 3	**05**	8.31e − 1	**05**	7.91e − 1	**05**	9.11e − 1	**5.0**

Table F.28: Results of the ML efficacy with the different DAGs for the CMAP dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
`full`	**01**	3.13	**01**	8.69e − 1	**1.0**
`trans. red.`	**02**	3.13	**02**	9.17e − 1	**2.0**
`linear`	**03**	3.20	**03**	1.11	**3.0**
`prediction`	**04**	6.08	**04**	3.62	**4.0**
`no links`	**05**	6.22	**05**	3.95	**5.0**

LPMC case study

Table F.29: Results of the statistical assessments with the different DAGs for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
`full`	**02**	7.74e − 3	**03**	1.03e − 2	**02**	9.54e − 1	**03**	8.29e − 2	**02**	9.83e − 1	**2.4**
`trans. red.`	**01**	7.04e − 3	**01**	9.50e − 3	**01**	9.55e − 1	**02**	7.73e − 2	**01**	9.83e − 1	**1.2**
`linear`	**03**	7.99e − 3	**02**	1.03e − 2	**03**	9.20e − 1	**01**	7.60e − 2	**03**	9.72e − 1	**2.4**
`prediction`	**04**	1.59e − 2	**04**	2.21e − 2	**05**	8.71e − 1	**04**	2.06e − 1	**05**	9.37e − 1	**4.4**
`no links`	**05**	1.60e − 2	**05**	2.27e − 2	**04**	8.79e − 1	**05**	2.12e − 1	**04**	9.49e − 1	**4.6**
First aggregation level - continuous columns											
`full`	**03**	9.34e − 3	**03**	1.35e − 2	**02**	9.55e − 1	**03**	1.35e − 1	**03**	9.80e − 1	**2.8**
`trans. red.`	**02**	8.46e − 3	**02**	1.26e − 2	**03**	9.53e − 1	**02**	1.26e − 1	**02**	9.81e − 1	**2.2**
`linear`	**01**	7.92e − 3	**01**	1.15e − 2	**01**	9.75e − 1	**01**	1.15e − 1	**01**	9.88e − 1	**1.0**

Continues on next page...

Table F.29 – continued from previous page

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
`prediction`	**04**	2.79e − 2	**04**	4.01e − 2	**05**	7.84e − 1	**04**	4.01e − 1	**05**	8.93e − 1	**4.4**
`no links`	**05**	2.80e − 2	**05**	4.14e − 2	**04**	7.92e − 1	**05**	4.14e − 1	**04**	9.06e − 1	**4.6**
First aggregation level - categorical columns											
`full`	**04**	6.27e − 3	**04**	7.34e − 3	**03**	9.53e − 1	**04**	3.41e − 2	**03**	9.86e − 1	**3.6**
`trans. red.`	**03**	5.72e − 3	**03**	6.61e − 3	**02**	9.57e − 1	**03**	3.20e − 2	**02**	9.86e − 1	**2.6**
`linear`	**05**	8.05e − 3	**05**	9.10e − 3	**05**	8.69e − 1	**05**	3.94e − 2	**05**	9.57e − 1	**5.0**
`prediction`	**01**	4.70e − 3	**02**	5.39e − 3	**04**	9.52e − 1	**02**	2.57e − 2	**04**	9.77e − 1	**2.6**
`no links`	**02**	4.74e − 3	**01**	5.37e − 3	**01**	9.59e − 1	**01**	2.47e − 2	**01**	9.88e − 1	**1.2**
Second aggregation level											
`full`	**03**	3.53e − 3	**03**	5.94e − 3	**02**	9.52e − 1	**03**	2.28e − 1	**02**	9.78e − 1	**2.6**
`trans. red.`	**02**	3.36e − 3	**02**	5.75e − 3	**03**	9.51e − 1	**02**	2.23e − 1	**03**	9.78e − 1	**2.4**
`linear`	**01**	2.95e − 3	**01**	4.90e − 3	**01**	9.60e − 1	**01**	1.87e − 1	**01**	9.83e − 1	**1.0**
`prediction`	**04**	9.08e − 3	**04**	1.58e − 2	**04**	8.16e − 1	**04**	6.64e − 1	**05**	9.06e − 1	**4.2**
`no links`	**05**	9.18e − 3	**05**	1.63e − 2	**05**	8.14e − 1	**05**	6.74e − 1	**04**	9.11e − 1	**4.8**
Third aggregation level											
`full`	**03**	1.33e − 3	**03**	2.78e − 3	**02**	9.37e − 1	**03**	4.47e − 1	**02**	9.70e − 1	**2.6**
`trans. red.`	**02**	1.29e − 3	**02**	2.76e − 3	**03**	9.36e − 1	**02**	4.43e − 1	**03**	9.70e − 1	**2.4**
`linear`	**01**	1.06e − 3	**01**	2.14e − 3	**01**	9.55e − 1	**01**	3.53e − 1	**01**	9.79e − 1	**1.0**
`prediction`	**04**	3.39e − 3	**04**	7.64e − 3	**04**	7.39e − 1	**05**	1.38	**05**	8.64e − 1	**4.4**
`no links`	**05**	3.45e − 3	**05**	7.92e − 3	**05**	7.29e − 1	**04**	1.37	**04**	8.66e − 1	**4.6**

Table F.30: Results of the ML efficacy with the different DAGs for the LPMC dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
`full`	**01**	2.14e1	**02**	2.06	**1.5**
`trans. red.`	**02**	2.16e1	**03**	2.09	**2.5**
`linear`	**03**	2.18e1	**01**	1.96	**2.0**
`prediction`	**04**	4.94e2	**04**	6.14	**4.0**
`no links`	**05**	5.06e2	**05**	6.45	**5.0**

LPMC_half case study

Table F.31: Results of the statistical assessments with the different DAGs for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	MAE		RMSE		R^2		SRMSE		ρ_{Pearson}		rank
First aggregation level - all columns											
`full`	**01**	9.29e−3	**01**	1.22e−2	**01**	9.24e−1	**02**	9.85e−2	**02**	9.67e−1	**1.4**
`trans. red.`	**03**	1.38e−2	**05**	1.98e−2	**05**	7.75e−1	**03**	1.71e−1	**04**	9.45e−1	**4.0**
`linear`	**02**	9.74e−3	**02**	1.26e−2	**02**	8.85e−1	**01**	9.79e−2	**01**	9.72e−1	**1.6**
`prediction`	**05**	1.47e−2	**04**	1.97e−2	**04**	8.60e−1	**05**	1.78e−1	**05**	9.41e−1	**4.6**
`no links`	**04**	1.42e−2	**03**	1.90e−2	**03**	8.62e−1	**04**	1.72e−1	**03**	9.45e−1	**3.4**
First aggregation level - continuous columns											
`full`	**02**	1.10e−2	**02**	1.60e−2	**02**	9.45e−1	**02**	1.60e−1	**02**	9.73e−1	**2.0**
`trans. red.`	**03**	1.90e−2	**03**	2.97e−2	**05**	7.12e−1	**03**	2.97e−1	**03**	9.29e−1	**3.4**
`linear`	**01**	1.06e−2	**01**	1.53e−2	**01**	9.47e−1	**01**	1.53e−1	**01**	9.79e−1	**1.0**
`prediction`	**05**	2.39e−2	**05**	3.34e−2	**04**	7.67e−1	**05**	3.34e−1	**05**	8.96e−1	**4.8**
`no links`	**04**	2.34e−2	**04**	3.25e−2	**03**	7.69e−1	**04**	3.25e−1	**04**	9.02e−1	**3.8**
First aggregation level - categorical columns											
`full`	**03**	7.66e−3	**03**	8.69e−3	**03**	9.06e−1	**03**	4.11e−2	**04**	9.61e−1	**3.2**
`trans. red.`	**04**	8.91e−3	**05**	1.06e−2	**04**	8.34e−1	**05**	5.43e−2	**05**	9.61e−1	**4.6**
`linear`	**05**	8.92e−3	**04**	1.01e−2	**05**	8.26e−1	**04**	4.70e−2	**03**	9.66e−1	**4.2**
`prediction`	**02**	6.07e−3	**02**	7.00e−3	**02**	9.45e−1	**02**	3.20e−2	**02**	9.83e−1	**2.0**
`no links`	**01**	5.65e−3	**01**	6.46e−3	**01**	9.48e−1	**01**	2.93e−2	**01**	9.86e−1	**1.0**
Second aggregation level											
`full`	**02**	4.16e−3	**02**	6.98e−3	**01**	9.37e−1	**02**	2.69e−1	**02**	9.71e−1	**1.8**
`trans. red.`	**03**	5.90e−3	**03**	1.11e−2	**05**	7.68e−1	**03**	4.58e−1	**03**	9.36e−1	**3.4**
`linear`	**01**	3.88e−3	**01**	6.51e−3	**02**	9.29e−1	**01**	2.54e−1	**01**	9.73e−1	**1.2**
`prediction`	**05**	8.40e−3	**05**	1.46e−2	**03**	8.17e−1	**05**	6.10e−1	**05**	9.10e−1	**4.6**
`no links`	**04**	8.38e−3	**04**	1.45e−2	**04**	8.17e−1	**04**	6.03e−1	**04**	9.12e−1	**4.0**
Third aggregation level											
`full`	**02**	1.52e−3	**02**	3.21e−3	**01**	9.22e−1	**02**	5.19e−1	**02**	9.63e−1	**1.8**
`trans. red.`	**03**	2.05e−3	**03**	5.03e−3	**05**	6.95e−1	**03**	8.69e−1	**03**	9.19e−1	**3.4**
`linear`	**01**	1.38e−3	**01**	2.90e−3	**02**	9.18e−1	**01**	4.87e−1	**01**	9.65e−1	**1.2**
`prediction`	**04**	3.26e−3	**05**	7.36e−3	**03**	7.44e−1	**05**	1.28	**05**	8.69e−1	**4.4**
`no links`	**05**	3.27e−3	**04**	7.35e−3	**04**	7.43e−1	**04**	1.28	**04**	8.70e−1	**4.2**

Table F.32: Results of the ML efficacy with the different DAGs for the LPMC_half dataset. Lighter grey tone corresponds to better results compared to darker ones.

Name	Continuous		Categorical		rank
`full`	**02**	2.86e1	**02**	2.55	**2.0**
`trans. red.`	**03**	3.67e1	**03**	2.68	**3.0**
`linear`	**01**	2.81e1	**01**	2.44	**1.0**
`prediction`	**04**	4.90e2	**04**	6.07	**4.0**
`no links`	**05**	4.99e2	**05**	6.45	**5.0**

Bibliography

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mane, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viegas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467 [cs]*, March 2016. URL http://arxiv.org/abs/1603.04467. arXiv: 1603.04467.

Aemmer, Zack and MacKenzie, Don. Generative population synthesis for joint household and individual characteristics. *Computers, Environment and Urban Systems*, 96:101852, September 2022. ISSN 0198-9715. doi: 10.1016/j.compenvurbsys.2022.101852. URL https://www.sciencedirect.com/science/article/pii/S0198971522000965.

Agarwal, Naman, Bullins, Brian, and Hazan, Elad. Second-Order Stochastic Optimization for Machine Learning in Linear Time. *arXiv:1602.03943 [cs, stat]*, February 2016. URL http://arxiv.org/abs/1602.03943. arXiv: 1602.03943.

Alqahtani, Hamed, Kavakli-Thorne, Manolya, and Kumar, Gulshan. Applications of Generative Adversarial Networks (GANs): An Updated Review. *Archives of Computational Methods in Engineering*, 28(2):525–552, March 2021. ISSN 1886-1784. doi: 10.1007/s11831-019-09388-y. URL https://doi.org/10.1007/s11831-019-09388-y.

Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*, December 2017. URL http://arxiv.org/abs/1701.07875. arXiv: 1701.07875.

Auld, Joshua A., Mohammadian, Abolfazl (Kouros), and Wies, Kermit. Population Synthesis with Subregion-Level Control Variable Aggregation. *Journal of Transportation Engineering*, 135(9):632–639, September 2009. ISSN 0733-947X. doi: 10.1061/(ASCE)TE.1943-5436.0000040. URL https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29TE.1943-5436.0000040. Publisher: American Society of Civil Engineers.

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer Normalization.

arXiv:1607.06450 [cs, stat], July 2016. URL http://arxiv.org/abs/1607.06450. arXiv: 1607.06450.

Badu-Marfo, Godwin, Farooq, Bilal, and Paterson, Zachary. Composite Travel Generative Adversarial Networks for Tabular and Sequential Population Synthesis. *arXiv:2004.06838 [cs, stat]*, April 2020. URL http://arxiv.org/abs/2004.06838. arXiv: 2004.06838.

Balles, Lukas, Romero, Javier, and Hennig, Philipp. Coupling Adaptive Batch Sizes with Learning Rates. *arXiv:1612.05086 [cs, stat]*, December 2016. URL http://arxiv.org/abs/1612.05086. arXiv: 1612.05086.

Barbedo, Jayme Garcia Arnal. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and Electronics in Agriculture*, 153:46–53, October 2018. ISSN 0168-1699. doi: 10.1016/j.compag.2018.08.013. URL https://www.sciencedirect.com/science/article/pii/S0168169918304617.

Barthelemy, Johan and Toint, Philippe L. Synthetic Population Generation Without a Sample. *Transportation Science*, 47(2):266–279, 2013. ISSN 0041-1655. URL https://www.jstor.org/stable/43666649. Publisher: INFORMS.

Beckman, Richard J., Baggerly, Keith A., and McKay, Michael D. Creating synthetic baseline populations. *Transportation Research Part A: Policy and Practice*, 30(6):415–429, November 1996. ISSN 0965-8564. doi: 10.1016/0965-8564(96)00004-3. URL http://www.sciencedirect.com/science/article/pii/0965856496000043.

Beiranvand, Vahid, Hare, Warren, and Lucet, Yves. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848, December 2017. ISSN 1573-2924. doi: 10.1007/s11081-017-9366-1. URL https://doi.org/10.1007/s11081-017-9366-1.

Ben-Akiva, Moshe E. and Lerman, Steven R. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, 1985. ISBN 978-0-262-02217-0. URL https://mitpress.mit.edu/books/discrete-choice-analysis. Google-Books-ID: oLC6ZYPs9UoC.

Bierlaire, Michel. BIOGEME: a free package for the estimation of discrete choice models. *Swiss Transport Research Conference 2003*, March 2003. URL https://infoscience.epfl.ch/record/117133.

Bierlaire, Michel. PandasBiogeme: a short introduction. Technical Report TRANSP-OR 181219, Transport and Mobility Laboratory, EPFL, 2018. URL https://transp-or.epfl.ch/documents/technicalReports/Bier18.pdf.

Bishop, Christopher M. *Pattern Recognition and Machine Learning*. Springer: New York, 2006. URL https://link.springer.com/book/9780387310732.

Bollapragada, Raghu, Mudigere, Dheevatsa, Nocedal, Jorge, Shi, Hao-Jun Michael, and Tang, Ping Tak Peter. A Progressive Batching L-BFGS Method for Machine Learning. *arXiv:1802.05374 [cs, math, stat]*, February 2018. URL http://arxiv.org/abs/1802.05374. arXiv: 1802.05374.

Bonabeau, Eric. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, May 2002. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.082080899. URL https://www.pnas.org/content/99/suppl_3/7280. Publisher: National Academy of Sciences Section: Colloquium Paper.

Bordes, Antoine, Bottou, Léon, Gallinari, Patrick, Chang, Jonathan, and Smith, S. Alex. Erratum: SGDQN is Less Careful than Expected. *Journal of Machine Learning Research*, 11(Aug):2229–2240, 2010. ISSN ISSN 1533-7928. URL http://www.jmlr.org/papers/v11/bordes10a.html.

Borysov, Stanislav S., Rich, Jeppe, and Pereira, Francisco C. How to generate micro-agents? A deep generative modeling approach to population synthesis. *Transportation Research Part C: Emerging Technologies*, 106:73–97, September 2019. ISSN 0968-090X. doi: 10.1016/j.trc.2019.07.006. URL http://www.sciencedirect.com/science/article/pii/S0968090X1831180X.

Brathwaite, Timothy, Vij, Akshay, and Walker, Joan L. Machine Learning Meets Microeconomics: The Case of Decision Trees and Discrete Choice. *arXiv:1711.04826 [stat]*, November 2017. URL http://arxiv.org/abs/1711.04826. arXiv: 1711.04826.

Brock, Andrew, Donahue, Jeff, and Simonyan, Karen. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *arXiv:1809.11096 [cs, stat]*, February 2019. URL http://arxiv.org/abs/1809.11096. arXiv: 1809.11096.

Brown, Tom B., Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel M., Wu, Jeffrey, Winter, Clemens, Hesse, Christopher, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, and Amodei, Dario. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. URL http://arxiv.org/abs/2005.14165. arXiv: 2005.14165.

Casati, Daniele, Müller, Kirill, Fourie, Pieter J., Erath, Alexander, and Axhausen, Kay W. Synthetic Population Generation by Combining a Hierarchical, Simulation-Based Approach with Reweighting by Generalized Raking. *Transportation Research Record: Journal of the Transportation Research Board*, (2493), 2015. ISSN 0361-1981. URL https://trid.trb.org/view.aspx?id=1339142. ISBN: 9780309369633 Number: 15-5284.

Chen, Mark, Radford, Alec, Child, Rewon, Wu, Jeffrey, Jun, Heewoo, Luan, David, and Sutskever, Ilya. Generative Pretraining From Pixels. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1691–1703. PMLR, November 2020. URL https://proceedings.mlr.press/v119/chen20s.html. ISSN: 2640-3498.

Conn, A., Gould, N., and Toint, P. *Trust Region Methods*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, January 2000. ISBN 978-0-89871-460-9.

doi: 10.1137/1.9780898719857. URL https://epubs.siam.org/doi/book/10.1137/1.9780898719857.

Danalet, Antonin and Mathys, Nicole. Mobility Resources in Switzerland in 2015. In *Proceedings of the 18th Swiss Transport Research Conference*, Ascona, Switzerland, 2018. URL http://www.strc.ch/2018/Danalet_Mathys.pdf.

de Vries, Harm, Strub, Florian, Mary, Jérémie, Larochelle, Hugo, Pietquin, Olivier, and Courville, Aaron. Modulating early visual processing by language. *arXiv:1707.00683 [cs]*, December 2017. URL http://arxiv.org/abs/1707.00683. arXiv: 1707.00683.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/ede7e2b6d13a41ddf9f4bdef84fdc737-Paper.pdf.

Deming, W. Edwards and Stephan, Frederick F. On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known. *Annals of Mathematical Statistics*, 11(4):427–444, December 1940. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177731829. URL https://projecteuclid.org/euclid.aoms/1177731829. Publisher: Institute of Mathematical Statistics.

Dennis, J. and Schnabel, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 1996. ISBN 978-0-89871-364-0. doi: 10.1137/1.9781611971200. URL https://epubs.siam.org/doi/book/10.1137/1.9781611971200.

Devarakonda, Aditya, Naumov, Maxim, and Garland, Michael. AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks. *arXiv:1712.02029 [cs, stat]*, December 2017. URL http://arxiv.org/abs/1712.02029. arXiv: 1712.02029.

Dolan, Elizabeth D. and Moré, Jorge J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, January 2002. ISSN 1436-4646. doi: 10.1007/s101070100263. URL https://doi.org/10.1007/s101070100263.

Dozat, Timothy. Incorporating Nesterov Momentum into Adam. February 2016. URL https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011. ISSN ISSN 1533-7928. URL http://jmlr.org/papers/v12/duchi11a.html.

Dumoulin, Vincent, Shlens, Jonathon, and Kudlur, Manjunath. A Learned Representation For Artistic Style. *arXiv:1610.07629 [cs]*, February 2017. URL http://arxiv.org/abs/1610.07629. arXiv: 1610.07629.

Farooq, Bilal, Bierlaire, Michel, Hurtubia, Ricardo, and Flötteröd, Gunnar. Simulation based population synthesis. *Transportation Research Part B: Methodological*, 58:243–263, December 2013. ISSN 0191-2615. doi: 10.1016/j.trb.2013.09.012. URL http://www.sciencedirect.com/science/article/pii/S0191261513001720.

Fletcher, R. *Practical Methods of Optimization; (2nd Ed.)*. Wiley-Interscience, New York, NY, USA, 1987. ISBN 978-0-471-91547-8. URL https://www.wiley.com/en-us/Practical+Methods+of+Optimization%2C+2nd+Edition-p-9780471494638.

Frégier, Yaël and Gouray, Jean-Baptiste. Mind2Mind : transfer learning for GANs. *arXiv:1906.11613 [cs, stat]*, October 2020. URL http://arxiv.org/abs/1906.11613. arXiv: 1906.11613.

Garrido, Sergio, Borysov, Stanislav S., Pereira, Francisco C., and Rich, Jeppe. Prediction of rare feature combinations in population synthesis: Application of deep generative modelling. *arXiv:1909.07689 [cs, stat]*, September 2019. URL http://arxiv.org/abs/1909.07689. arXiv: 1909.07689.

Geman, Stuart and Geman, Donald. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, November 1984. ISSN 1939-3539. doi: 10.1109/TPAMI.1984.4767596. URL https://ieeexplore.ieee.org/document/4767596. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Gers, Felix A., Schmidhuber, Jürgen, and Cummins, Fred. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471, October 2000. ISSN 0899-7667. doi: 10.1162/089976600300015015. URL https://ieeexplore.ieee.org/document/818041. Conference Name: Neural Computation.

Goodfellow, Ian. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160 [cs]*, April 2017. URL http://arxiv.org/abs/1701.00160. arXiv: 1701.00160.

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. URL http://arxiv.org/abs/1406.2661. arXiv: 1406.2661.

Gower, Robert M., Hanzely, Filip, Richtárik, Peter, and Stich, Sebastian. Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization. *arXiv:1802.04079 [cs, math]*, February 2018. URL http://arxiv.org/abs/1802.04079. arXiv: 1802.04079.

Goyal, Priya, Dollár, Piotr, Girshick, Ross, Noordhuis, Pieter, Wesolowski, Lukasz, Kyrola, Aapo, Tulloch, Andrew, Jia, Yangqing, and He, Kaiming. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arXiv:1706.02677 [cs]*, April 2018. URL http://arxiv.org/abs/1706.02677. arXiv: 1706.02677.

Gu, Tian and Duan, Rui. SynTL: A synthetic-data-based transfer learning approach for multi-center risk prediction, March 2022. URL https://www.medrxiv.org/content/10.1101/2022.03.23.22272834v1. Pages: 2022.03.23.22272834.

Gulrajani, Ishaan, Ahmed, Faruk, Arjovsky, Martin, Dumoulin, Vincent, and Courville, Aaron. Improved Training of Wasserstein GANs. *arXiv:1704.00028 [cs, stat]*, December 2017. URL http://arxiv.org/abs/1704.00028. arXiv: 1704.00028.

Hagberg, Aric, Swart, Pieter, and Schult, Daniel. Exploring Network Structure, Dynamics, and Function Using NetworkX. In *Proceedings of the 7th Python in Science Conference*, United States, January 2008. URL https://www.researchgate.net/publication/236407765_Exploring_Network_Structure_Dynamics_and_Function_Using_NetworkX.

Hillel, Tim. *Understanding travel mode choice: A new approach for city scale simulation.* PhD thesis, University of Cambridge, Cambridge, January 2019. URL https://www.repository.cam.ac.uk/handle/1810/293576.

Hillel, Tim, Elshafie, Mohammed Z E B, and Jin, Ying. Recreating passenger mode choice-sets for transport simulation: A case study of London, UK. *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction*, 171(1):29–42, March 2018. doi: 10.1680/jsmic.17.00018. URL https://www.icevirtuallibrary.com/doi/full/10.1680/jsmic.17.00018.

Hindupur, Avinash. The GAN Zoo, May 2022. URL https://github.com/hindupuravinash/the-gan-zoo. original-date: 2017-04-14T16:45:24Z.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://ieeexplore.ieee.org/abstract/document/6795963. Conference Name: Neural Computation.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-To-Image Translation With Conditional Adversarial Networks. pages 1125–1134, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Isola_Image-To-Image_Translation_With_CVPR_2017_paper.html.

Jeon, Hyeonseong, Bang, Youngoh, Kim, Junyaup, and Woo, Simon S. T-GD: Transferable GAN-generated Images Detection Framework. *arXiv:2008.04115 [cs]*, August 2020. URL http://arxiv.org/abs/2008.04115. arXiv: 2008.04115.

Jha, Anurag, Chandrasekaran, Anand, Kim, Chiho, and Ramprasad, Rampi. Impact of dataset uncertainties on machine learning model predictions: the example of polymer glass transition temperatures. *Modelling and Simulation in Materials Science and Engineering*, 27(2):024002, January 2019. ISSN 0965-0393. doi: 10.1088/1361-651X/aaf8ca. URL https://doi.org/10.1088/1361-651x/aaf8ca. Publisher: IOP Publishing.

Jones, Eric, Oliphant, Travis, and Peterson, Pearu. SciPy: open source scientific tools for Python. 2014. URL https://scipy.org/.

Jordon, James, Yoon, Jinsung, and Schaar, Mihaela van der. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. September 2018. URL https://openreview.net/forum?id=S1zk9iRqF7.

Kagho, Grace O., Balac, Milos, and Axhausen, Kay W. Agent-Based Models in Transport Planning: Current State, Issues, and Expectations. *Procedia Computer Science*, 170:726–732, January 2020. ISSN 1877-0509. doi: 10.1016/j.procs.2020.03.164. URL https://www.sciencedirect.com/science/article/pii/S187705092030627X.

Kahng, Minsuk, Thorat, Nikhil, Chau, Duen Horng, Viégas, Fernanda B., and Wattenberg, Martin. GAN Lab: Understanding Complex Deep Generative Models using Interactive Visual Experimentation. *IEEE Transactions on Visualization and Computer Graphics*, 25 (1):310–320, January 2019. ISSN 1941-0506. doi: 10.1109/TVCG.2018.2864500. URL https://ieeexplore.ieee.org/document/8440049. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

Karimpanal, Thommen George and Bouffanais, Roland. Self-Organizing Maps for Storage and Transfer of Knowledge in Reinforcement Learning. *Adaptive Behavior*, 27(2):111–126, April 2019. ISSN 1059-7123, 1741-2633. doi: 10.1177/1059712318818568. URL http://arxiv.org/abs/1811.08318. arXiv:1811.08318 [cs].

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, and Liu, Tie-Yan. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Keskar, Nitish Shirish and Berahas, Albert S. adaQN: An Adaptive Quasi-Newton Algorithm for Training RNNs. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 1–16. Springer, Cham, September 2016. ISBN 978-3-319-46127-4 978-3-319-46128-1. doi: 10.1007/978-3-319-46128-1_1. URL https://link.springer.com/chapter/10.1007/978-3-319-46128-1_1.

Kim, Jooyoung and Lee, Seungjae. A simulated annealing algorithm for the creation of synthetic population in activity-based travel demand model. *KSCE Journal of Civil Engineering*, 20(6):2513–2523, September 2016. ISSN 1976-3808. doi: 10.1007/s12205-015-0691-7. URL https://doi.org/10.1007/s12205-015-0691-7.

Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. URL http://arxiv.org/abs/1412.6980. arXiv: 1412.6980.

Kingma, Diederik P. and Welling, Max. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114. arXiv: 1312.6114.

Kiros, Ryan. Training Neural Networks with Stochastic Hessian-Free Optimization. *arXiv:1301.3641 [cs, stat]*, January 2013. URL http://arxiv.org/abs/1301.3641. arXiv: 1301.3641.

Kohavi, Ron. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 202–207, Portland, Oregon, August 1996. AAAI Press. URL https://www.aaai.org/Papers/KDD/1996/KDD96-033.pdf.

Kukić, Marija and Bierlaire, Michel. One-step simulator for synthetic households generation. May 2022. URL http://www.strc.ch/2022/Kukic_Bierlaire.pdf.

Lederrey, Gael, Lurkin, Virginie, and Bierlaire, Michel. Optimization of Discrete Choice Models using first-order methods. In *Proceedings of the 7th Symposium of the European Association for Research in Transportation*, Athens, Greece, 2018a. URL https://transp-or.epfl.ch/heart/2018/abstracts/5489.pdf.

Lederrey, Gael, Lurkin, Virginie, and Bierlaire, Michel. SNM: Stochastic Newton Methodfor Optimization of Discrete Choice Models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3199–3204, November 2018b. doi: 10.1109/ITSC.2018.8569539. URL https://ieeexplore.ieee.org/document/8569539. ISSN: 2153-0017.

Lederrey, Gael, Hillel, Tim, and Bierlaire, Michel. DATGAN: Integrating expert knowledge into deep learning for synthetic tabular data. *arXiv:2203.03489 [cs]*, March 2022. URL http://arxiv.org/abs/2203.03489. arXiv: 2203.03489.

Lin, J. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991. ISSN 1557-9654. doi: 10.1109/18.61115. URL https://ieeexplore.ieee.org/abstract/document/61115. Conference Name: IEEE Transactions on Information Theory.

Linjordet, Trond and Balog, Krisztian. Impact of Training Dataset Size on Neural Answer Selection Models. In Azzopardi, Leif, Stein, Benno, Fuhr, Norbert, Mayr, Philipp, Hauff, Claudia, and Hiemstra, Djoerd, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 828–835, Cham, 2019. Springer International Publishing. ISBN 978-3-030-15712-8. doi: 10.1007/978-3-030-15712-8_59. URL https://link.springer.com/chapter/10.1007/978-3-030-15712-8_59.

Liu, Ming-Yu and Tuzel, Oncel. Coupled Generative Adversarial Networks. *arXiv:1606.07536 [cs]*, September 2016. URL http://arxiv.org/abs/1606.07536. arXiv: 1606.07536.

Liu, Yi, Peng, Jialiang, Yu, James J. Q., and Wu, Yi. PPGAN: Privacy-preserving Generative Adversarial Network. *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 985–989, December 2019. doi: 10.1109/ICPADS47876.2019.00150. URL http://arxiv.org/abs/1910.02007. arXiv: 1910.02007.

Lucas, Peter J., Gaag, Linda C., and Abu-Hanna, Ameen. Bayesian Networks in biomedicine and health-care. *Artificial Intelligence in Medicine*, 30:201–214, April 2004. doi: 10.1016/j.artmed.2003.11.001. URL https://www.researchgate.net/publication/224818117_Bayesian_Networks_in_biomedicine_and_health-care.

Martens, James. Deep learning via Hessian-free optimization. *ICML*, 27:735–742, 2010. URL https://www.cs.toronto.edu/~jmartens/docs/Deep_HessianFree.pdf.

Mirza, Mehdi and Osindero, Simon. Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*, November 2014. URL http://arxiv.org/abs/1411.1784. arXiv: 1411.1784.

Miyato, Takeru and Koyama, Masanori. cGANs with Projection Discriminator. *arXiv:1802.05637 [cs, stat]*, August 2018. URL http://arxiv.org/abs/1802.05637. arXiv: 1802.05637.

Müller, Kirill and Axhausen, Kay W. Population synthesis for microsimulation: State of the art. *Arbeitsberichte Verkehrs- und Raumplanung*, 638, August 2010. doi: 10.3929/ethz-a-006127782. URL https://www.research-collection.ethz.ch/handle/20.500.11850/30298. Accepted: 2017-08-22T09:43:21Z Publisher: IVT, ETH Zurich.

Müller, Kirill and Axhausen, Kay W. Hierarchical IPF: Generating a synthetic population for Switzerland. Technical Report ersa11p305, European Regional Science Association, September 2011. URL https://ideas.repec.org/p/wiw/wiwrsa/ersa11p305.html. Publication Title: ERSA conference papers.

Mokhtari, A. and Ribeiro, A. RES: Regularized Stochastic BFGS Algorithm. *IEEE Transactions on Signal Processing*, 62(23):6089–6104, December 2014. ISSN 1053-587X. doi: 10.1109/TSP.2014.2357775. URL https://arxiv.org/abs/1401.7625.

Mutny, Mojmir. Stochastic Second-Order Optimization via von Neumann Series. *arXiv:1612.04694 [cs, math]*, December 2016. URL http://arxiv.org/abs/1612.04694. arXiv: 1612.04694.

Nazeri, Kamyar, Ng, Eric, Joseph, Tony, Qureshi, Faisal Z., and Ebrahimi, Mehran. EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning. *arXiv:1901.00212 [cs]*, January 2019. URL http://arxiv.org/abs/1901.00212. arXiv: 1901.00212.

Nesterov, Yurii E. A method for solving the convex programming problem with convergence rate O (1/k^ 2). In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983. URL https://vsokolov.org/courses/750/2018/files/nesterov.pdf.

Newman, Jeffrey P., Lurkin, Virginie, and Garrow, Laurie A. Computational methods for estimating multinomial, nested, and cross-nested logit models that account for semi-aggregate data. *Journal of Choice Modelling*, 26:28–40, March 2018. ISSN 1755-5345. doi: 10.1016/j.jocm.2017.11.001. URL http://www.sciencedirect.com/science/article/pii/S1755534517300179.

Noguchi, Atsuhiro and Harada, Tatsuya. Image Generation From Small Datasets via Batch Statistics Adaptation. *arXiv:1904.01774 [cs]*, October 2019. URL http://arxiv.org/abs/1904.01774. arXiv: 1904.01774.

Odena, Augustus, Olah, Christopher, and Shlens, Jonathon. Conditional Image Synthesis With Auxiliary Classifier GANs. *arXiv:1610.09585 [cs, stat]*, July 2017. URL http://arxiv.org/abs/1610.09585. arXiv: 1610.09585.

Ortelli, Nicola, Hillel, Tim, Pereira, Francisco C., de Lapparent, Matthieu, and Bierlaire, Michel. Assisted specification of discrete choice models. *Journal of Choice Modelling*, 39:100285, June 2021. ISSN 1755-5345. doi: 10.1016/j.jocm.2021.100285. URL https://www.sciencedirect.com/science/article/pii/S175553452100018X.

Park, Noseong, Mohammadi, Mahmoud, Gorde, Kshitij, Jajodia, Sushil, Park, Hongkyu, and Kim, Youngmin. Data Synthesis based on Generative Adversarial Networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, June 2018. ISSN 21508097. doi: 10.14778/3231751.3231757. URL http://arxiv.org/abs/1806.03384. arXiv: 1806.03384.

Pathak, Deepak, Krahenbuhl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei A. Context Encoders: Feature Learning by Inpainting. *arXiv:1604.07379 [cs]*, November 2016. URL http://arxiv.org/abs/1604.07379. arXiv: 1604.07379.

Philips, Ian, Clarke, Graham, and Watling, David. A Fine Grained Hybrid Spatial Microsimulation Technique for Generating Detailed Synthetic Individuals from Multiple Data Sources: An Application To Walking And Cycling. *International Journal of Microsimulation*, 10(1): 167–200, 2017. URL https://ideas.repec.org/a/ijm/journl/v10y2017i1p167-200.html. Publisher: International Microsimulation Association.

Polyak, B. and Juditsky, A. Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, July 1992. ISSN 0363-0129. doi: 10.1137/0330046. URL https://epubs.siam.org/doi/abs/10.1137/0330046.

Polyak, Boris T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. URL https://www.sciencedirect.com/science/article/pii/0041555364901375.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*, January 2016. URL http://arxiv.org/abs/1511.06434. arXiv: 1511.06434.

Rafati, Jacob, DeGuchy, Omar, and Marcia, Roummel F. Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2015–2019, September 2018. doi: 10.23919/EUSIPCO.2018.8553243. URL https://ieeexplore.ieee.org/document/8553243. ISSN: 2076-1465, 2219-5491.

Reddi, Sashank J., Kale, Satyen, and Kumar, Sanjiv. On the Convergence of Adam and Beyond. February 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

Reed, Scott, Akata, Zeynep, Yan, Xinchen, Logeswaran, Lajanugen, Schiele, Bernt, and Lee, Honglak. Generative Adversarial Text to Image Synthesis. *arXiv:1605.05396 [cs]*, June 2016. URL http://arxiv.org/abs/1605.05396. arXiv: 1605.05396.

Rich, Jeppe. Large-scale spatial population synthesis for Denmark. *European Transport Research Review*, 10(2):63, December 2018. ISSN 1866-8887. doi: 10.1186/s12544-018-0336-2. URL https://doi.org/10.1186/s12544-018-0336-2.

Robbins, Herbert and Monro, Sutton. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 0003-4851. URL https://www.jstor.org/stable/2236626.

Rubin, Donald B. The Use of Matched Sampling and Regression Adjustment to Remove Bias in Observational Studies. *Biometrics*, 29(1):185–203, 1973. ISSN 0006-341X. doi: 10.2307/2529685. URL https://www.jstor.org/stable/2529685. Publisher: [Wiley, International Biometric Society].

Ruder, Sebastian. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, September 2016. URL http://arxiv.org/abs/1609.04747. arXiv: 1609.04747.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*, June 2016. URL http://arxiv.org/abs/1606.03498. arXiv: 1606.03498.

Satell, Greg. *Mapping Innovation: A Playbook for Navigating a Disruptive Age*. McGraw Hill, New York, 1st edition edition, May 2017. ISBN 978-1-259-86225-0. URL https://gregsatell.com/read-the-book/.

Schmidt, Mark, Roux, Nicolas Le, and Bach, Francis. Minimizing Finite Sums with the Stochastic Average Gradient. *arXiv:1309.2388 [cs, math, stat]*, September 2013. URL http://arxiv.org/abs/1309.2388. arXiv: 1309.2388.

Shin, Minchul Craig. gans-awesome-applications, May 2022. URL https://github.com/nashory/gans-awesome-applications. original-date: 2017-10-12T03:19:02Z.

Shorten, Connor and Khoshgoftaar, Taghi M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, July 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0. URL https://doi.org/10.1186/s40537-019-0197-0.

Sun, Lijun and Erath, Alexander. A Bayesian network approach for population synthesis. *Transportation Research Part C: Emerging Technologies*, 61:49–62, December 2015. ISSN 0968-090X. doi: 10.1016/j.trc.2015.10.010. URL http://www.sciencedirect.com/science/article/pii/S0968090X15003599.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. URL https://www.youtube.com/watch?v=SJ48OZ_qlrc.

Wang, Xiao, Ma, Shiqian, and Liu, Wei. Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. *arXiv:1412.1196 [math]*, December 2014. URL http://arxiv.org/abs/1412.1196. arXiv: 1412.1196.

Wang, Yaxing, Gonzalez-Garcia, Abel, Berga, David, Herranz, Luis, Khan, Fahad Shahbaz, and van de Weijer, Joost. MineGAN: effective knowledge transfer from GANs to target domains with few images. *arXiv:1912.05270 [cs]*, April 2020. URL http://arxiv.org/abs/1912.05270. arXiv: 1912.05270.

Wen, Bingyang, Colon, Luis Oliveros, Subbalakshmi, K. P., and Chandramouli, R. Causal-TGAN: Generating Tabular Data Using Causal Generative Adversarial Networks. *arXiv:2104.10680 [cs]*, April 2021. URL http://arxiv.org/abs/2104.10680. arXiv: 2104.10680.

Wolfe, P. Convergence Conditions for Ascent Methods. *SIAM Review*, 11(2):226–235, April 1969. ISSN 0036-1445. doi: 10.1137/1011036. URL https://epubs.siam.org/doi/abs/10.1137/1011036.

Wolfe, Philip. Convergence Conditions for Ascent Methods. II: Some Corrections. *SIAM Review*, 13(2):185–188, 1971. ISSN 0036-1445. URL https://www.jstor.org/stable/2028821.

Wu, Chai Wah. ProdSumNet: reducing model parameters in deep neural networks via product-of-sums matrix decompositions. *arXiv:1809.02209 [cs, stat]*, May 2019. URL http://arxiv.org/abs/1809.02209. arXiv: 1809.02209.

Xu, Lei and Veeramachaneni, Kalyan. Synthesizing Tabular Data using Generative Adversarial Networks. *arXiv:1811.11264 [cs, stat]*, November 2018. URL http://arxiv.org/abs/1811.11264. arXiv: 1811.11264.

Xu, Lei, Skoularidou, Maria, Cuesta-Infante, Alfredo, and Veeramachaneni, Kalyan. Modeling Tabular data using Conditional GAN. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d\textquotesingle, Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 7335–7345. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8953-modeling-tabular-data-using-conditional-gan.pdf.

Ye, Haishan and Zhang, Zhihua. Nestrov's Acceleration For Second Order Method. *arXiv:1705.07171 [cs]*, May 2017. URL http://arxiv.org/abs/1705.07171. arXiv: 1705.07171.

Yeh, Raymond A., Chen, Chen, Lim, Teck Yian, Schwing, Alexander G., Hasegawa-Johnson, Mark, and Do, Minh N. Semantic Image Inpainting with Deep Generative Models. *arXiv:1607.07539 [cs]*, July 2017. URL http://arxiv.org/abs/1607.07539. arXiv: 1607.07539.

Yin, Dan and Yang, Qing. GANs Based Density Distribution Privacy-Preservation on Mobility Data. *Security and Communication Networks*, 2018:e9203076, December 2018. ISSN 1939-0114. doi: 10.1155/2018/9203076. URL https://www.hindawi.com/journals/scn/2018/9203076/. Publisher: Hindawi.

You, Z. and Xu, B. Investigation of stochastic Hessian-Free optimization in Deep neural networks for speech recognition. In *The 9th International Symposium on Chinese Spoken Language Processing*, pages 450–453, September 2014. doi: 10.1109/ISCSLP.2014.6936597. URL https://ieeexplore.ieee.org/document/6936597/.

Zeiler, Matthew D. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, December 2012. URL http://arxiv.org/abs/1212.5701. arXiv: 1212.5701.

Zhang, Danqing, Cao, Junyu, Feygin, Sid, Tang, Dounan, Shen, Zuo-Jun(Max), and Pozdnoukhov, Alexei. Connected population synthesis for transportation simulation. *Transportation Research Part C: Emerging Technologies*, 103:1–16, June 2019a. ISSN 0968-090X. doi: 10.1016/j.trc.2018.12.014. URL http://www.sciencedirect.com/science/article/pii/S0968090X18318515.

Zhang, Han, Goodfellow, Ian, Metaxas, Dimitris, and Odena, Augustus. Self-Attention Generative Adversarial Networks. *arXiv:1805.08318 [cs, stat]*, June 2019b. URL http://arxiv.org/abs/1805.08318. arXiv: 1805.08318.

Zhao, Zilong, Kunar, Aditya, Van der Scheer, Hiek, Birke, Robert, and Chen, Lydia Y. CTAB-GAN: Effective Table Data Synthesizing. *arXiv:2102.08369 [cs]*, May 2021. URL http://arxiv.org/abs/2102.08369. arXiv: 2102.08369.

Zheng, Chuanxia, Cham, Tat-Jen, and Cai, Jianfei. Pluralistic Image Completion. *arXiv:1903.04227 [cs]*, April 2019. URL http://arxiv.org/abs/1903.04227. arXiv: 1903.04227.

Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks. pages 2223–2232, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Zhu_Unpaired_Image-To-Image_Translation_ICCV_2017_paper.html.

Gael **Lederrey**

PH.D. STUDENT IN MACHINE LEARNING AND DATA SCIENCE

gael.lederrey@epfl.ch | glederrey | gael-lederrey | Gael Lederrey

Strengths

During my Ph.D., I specialized in **Machine Learning and Data Science methodologies** combined with modelization techniques. During my Bachelor's and Master's studies, I **explored multiple engineering fields** such as Biology, Mechanics, and Physics. My mathematics and engineering background compliments my Ph.D. since it allows me to **apply out-of-the-box solutions to practical problems**.

Education

Ph.D. in Machine Learning and Data Science *2017 - 2022*
(TRANSP-OR LABORATORY) · EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*
Title: "*Bridging the gap between model-driven and data-driven methods in the era of Big Data*"

MSc in Computational Science and Engineering (CSE) *2014 - 2017*
EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*

Propaedeutic year in Mathematics and BSc in Physics *2010 - 2014*
EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*

Experiences

Ph.D. in Machine Learning and Data Science *Sep. 2017 - Sep. 2022*
(TRANSP-OR LABORATORY) · EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*

- This thesis aims to use the strengths of both data-driven and model-driven methods to improve its counterparts. The first part of the thesis used knowledge from Machine Learning to optimize large models and applied it to Discrete Choice Models. The second part combined the strengths of modelization and Deep Learning to develop a new methodology for generating synthetic data. This research is comprised of three distinct projects:
  1. Development of stochastic algorithms to improve the optimization performance of discrete choice models.
  2. Development of a new architecture for Generative Adversarial Networks (GANs) based on the modelization of causal links in the data.
  3. Generation of a large, detailed, and representative synthetic population of Greater London while safe-guarding the privacy of individuals.
- Other activities include 7 talks in conferences, 5 conference proceedings, and the supervision of 11 semester projects and 2 MSc thesis.

Internship - Data scientist/Researcher *Jul. 2017 - Aug. 2017*
(DATA SCIENCE LAB) · EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*

- Developed a scraper in Python to download data about beer reviews data from websites, including cleaning and analysis.
- Preparation of a conference proceeding following the results of the Master thesis.

Master thesis in Data Science (CSE degree) *Feb. 2017- Jul. 2017*
(DATA SCIENCE LAB) · EPFL (ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE) *Lausanne, Switzerland*

- Title: "*Who likes this beer? Me or the community? A matched observational study of beer review from two aligned communities*"
- The workload included cleaning the data, matching the elements between the communities, and studying the data using statistics and Machine Learning to compare both communities and the impact of the rating scales on the final reviews.

Junior Backend Engineer *Feb. 2016 - Aug. 2016*
(nViso SA) *Lausanne, Switzerland*
- Continuation of the internship at 50% during the semester (Feb. to Jun.) and 100% during the summer.

Internship - Backend Engineer *Jul. 2015 - Jan. 2016*
(nViso SA) *Lausanne, Switzerland*
- Development of the backend of a test product based on emotion-recognition technologies; Prototype was developed using Swagger; Final product was built in collaboration with a foreign team of developers; Worked on several other internal projects.

Teaching assistant *Sep. 2012 - Sep. 2022*
(École Polytechnique Fédérale de Lausanne) *Lausanne, Switzerland*
- Sep. 2017 - Sep. 2022: TA in Optimization and Machine Learning.
- Jan. 2017 - Aug. 2017: Main assistant on a MOOC on Matlab and Octave.
- Sep. 2016 - Dec. 2016: Scrum master for a Neuroscience C++ project for a team of students; TA in C++ for beginners.
- Feb. 2015 - Jun. 2015: TA in Numerical Analysis.
- Sep. 2012 - Dec. 2014: TA in Physics.

Extracurricular Activity

Ph.D. Student representative *Sep. 2018 - Aug. 2020*
(École Polytechnique Fédérale de Lausanne) *Lausanne, Switzerland*
- Ensure the link between Ph.D. students and the various EPFL bodies.
- Organisation of social events and working with the EDCE Ph.D. school.

Communication and Treasury *Dec. 2017 - Dec. 2021*
(PolyDoc - Association of Ph.D. students) *Lausanne, Switzerland*
- Creation of the association; Communication for two years and Treasurer for two years.
- Built a new system for the treasury; Creation of an exchange platform for Ph.D. representatives; Organisation of social events.

Voluntary driver *Dec. 2015 - present*
(Association Nez Rouge) *Lausanne, Switzerland*
- Drive people during the cold nights of the winter holidays.

Member and President *Sep. 2015 - Aug. 2017*
(CQFD - Association of Math students) *Lausanne, Switzerland*
- Link between the association and the administration of the Mathematic section; Organisation of social events.

Skills

General
- Strong mathematical background: statistics, linear algebra, calculus, and numerical methods.
- Knowledge in various engineering fields: Physics, Mechanics, Neurobiology, Chemistry, etc.
- Main language: Python; Knowledge in Javascript, Bash, C++ (OpenMP and MPI), Java, and Matlab.
- Software Engineering: extensive use of GIT and knowledge of Scrum.
- Main OS: Linux; Knowledge in Windows.

Data Science
- Scraping the web (with or without APIs) and cleaning/wrangling data.
- Interactive visualization in JavaScript.
- Analysis of the data using statistical and Machine Learning methodologies.
- Main focus on tabular data; knowledge in NLP, time series, and images.

Machine Learning
- Building Machine Learning models or using them with sklearn.
- Evaluation and optimization of Machine Learning models.
- Knowledge in Tensorflow and Pytorch.
- Cloud computing using slurm.

Languages French (Native), English (C1-C2) & German (A2-B1)