EPFL

Thèse n° 8328

Methods for strong gravitational lens detection and analysis using machine learning and high performance computing

Présentée le 4 septembre 2020

à la Faculté des sciences de base
Laboratoire d'astrophysique
Programme doctoral en physique

pour l'obtention du grade de Docteur ès Sciences

par

Christoph Ernst René SCHÄFER

Acceptée sur proposition du jury

Prof. R. Houdré, président du jury
Prof. J.-P. R. Kneib, directeur de thèse
Prof. S. Suyu, rapporteuse
Prof. J. Richard, rapporteur
Dr C. Gheller, rapporteur

École polytechnique fédérale de Lausanne

2020

You have power over your mind - not outside events.
Realize this, and you will find strength.
— Marcus Aurelius

To my friends and family . . .

Acknowledgements

Completing this thesis has been one of the most interesting and challenging experiences of my life. I must thank first my PhD advisor, Jean-Paul Kneib, for the amazing opportunities he has given me to work, travel and experience the scientific world. He has driven me out of my comfort zone so much that I forgot what it looked like after two years. I also wish to thank from the bottom of my heart Markus Rexroth, my predecessor and one could say mentor in the ways of the PhD and Gilles Fourestey, the HPC expert working with me on Lenstool-HPC, always there with a helpful depreciating comment on why our code was not working how it was supposed to. Without them this thesis would not have reached its present level. I cannot forget to thank Frédéric Courbin, who helped me a lot in my thesis when clearly he did not have to. A thank you also to Bruno Altieri at ESAC, who invited me regularly to visit to understand the working of ESA. Both of them helped finance my thesis.

I could not have finished these four years of intense research without the support of my amazing colleagues from Lastro like Vivien, Loic, Damien, Sophie, Elodie, Claudio, Thibault and all the others. This would have been a lot less fun without you although maybe easier to concentrate. The concerned parties know what I am talking about. Another big hug goes to my Zinal PhD buddies, Cagil, Pierre and Killian. Our monthly sessions quickly became therapeutic for all four of us when we were bemoaning our still unpublished papers. Rest in peace, Killian... Another shoutout goes to my friends from EPFL and around the world: Joackim, Naomi, Gautier, Thais, Ben, Raphael, Marius, David and all the others. You all make life worth living. A big thank you also goes to my parents and sister. Their support meant the world to me and made me who I am. A big kiss to my newborn son Zachary, whose presence, while extremely motivating, might have made my thesis defense slightly harder then it had to be. Last, but most importantly, a huge thank you to my life's partner Julia Campanico who supported me in every aspect of this thesis. She cheered me up, left me alone to work when I needed it and helped me proofread my thesis to an amazing degree. Thank you all!

Lausanne, 8 July 2020 C. S.

Preface

When Jean-Paul Kneib offered me the opportunity to work within the Euclid framework with the European Space Agency (ESA) on strong lensing detection and analysis, for myself coming from an astrophysics and informatics background, this challenge intrigued me strongly because of the unexploited potential there. Strong lens astronomers had already many extremely powerful semi-automated optimisation and machine learning tools. Bayesian based machine learning especially had taken center stage for the last 30 years and astronomers were only then really starting to feel the limits of these techniques in speed and efficiency. There was still little mention of more modern deep learning image recognition techniques like CNN for object detection. Speed focused HPC implementation of lensing modelling software were underway but still in the first stages and only developed by astronomers. Contacts with HPC experts and machine learning experts were rare. Therefore, initially using Jean-Pauls connections, I worked heavily together with experts from the HPC department (a big shout out to Gilles Fourestey here without whom Lenstool-HPC would simply not have existed) at EPFL and various deep learning experts from EPFL to fuel my thesis. This expert help has certainly participated to transforming my thesis into something a lot more technical than one would expect from an astrophysical thesis.

Lausanne, 20 May 2020 C. S.

Abstract

The study of strong gravitational lenses is a relatively new scientific field in astronomy with many applications in cosmology. Its unique observables allow astronomers to trace dark matter, determine the expansion of the universe and study galaxy evolution. While strong lenses are relatively rare phenomena, recent and future improvements in observational instrumentation are expected to lead to a sharp increase in the number and quality of their observation. With that increase comes however an observational challenge from the sheer quantity of data the lenses are hidden in. Current classification and modelling techniques are simply not capable of handling the new data in a reasonable timeframe. As a consequence this thesis concerns itself with the development and improvement of numerical methods necessary to find and analyse these newly found lenses, borrowing from deep learning and high performance computing techniques.

Deep learning, a field of machine learning, shows significant success in finding lenses in simulations. Performing significantly better than any other classification method, CNN-based lens finders reach almost the required classification efficiency and accuracy for a fully automated Euclid strong lensing pipeline. Its dependency on a varied and complete training set and the difficulty creating it make its immediate application to new surveys however more difficult.

Incorporating high performance computing techniques and more concurrent algorithm design into lens analysis results in crucial speed-ups for strong lens analysis. Applied to Lenstool, a popular mass modelling tool for gravitational lenses, the resulting software can be run on modern supercomputers, using Graphics Processing Units (GPU) and CPUs simultaneously. Running it on state-of-the-art hardware makes it possible to reduce computation time to an acceptable level when mass modelling complex cluster lenses.

Keywords: Strong Gravitational Lenses – High Performance Computing – Lens-modelling – Deep Learning – Euclid

Résumé

L'étude des lentilles gravitationnelles est un domaine de recherche récent en astronomie qui possède une multitude d'application en cosmologie. Les observables uniques que présentent les lentilles permettent aux astronomes d'observer indirectement la matière noire, de calculer le taux d'expansion de l'univers et d'étudier l'évolution des galaxies. Bien que les lentilles gravitationnelles soient un phénomène rare, on prédit une forte augmentation du nombre et de la qualité des lentilles gravitationnelles trouvées due aux avancées technologiques en matière d'instrumentation observationnelle. Cette forte augmentation met les astronomes face à des nouveaux problèmes liés à la quantité de données ayant besoin de traitement. Les techniques de classification et modélisation actuelles ne sont simplement pas capables d'analyser autant de données dans un délai raisonnable. C'est donc à cause de ceci que le sujet de cette thèse est le développement et l'amélioration de méthodes numériques utilisées pour la recherche et l'analyse de lentilles gravitationnelles.

Le deep learning, un sous-domaine du machine learning, produit des résultats impressionnants pour la recherche de lentilles. Basé sur des concours de classification de lentilles, les algorithmes de recherche basés sur des CNN se sont révélés plus efficaces que n'importe quelle autre méthode sur le marché. Leur efficacité actuelle est presque suffisante pour atteindre les critères de qualité demandés par la mission Euclid dans le but de construire une pipeline automatisée de recherche de lentilles. En revanche la dépendance de ces modèles à un set d'entraînement varié et réel rend leur application problématique sur des missions d'observation n'ayant pas encore produit de données.

L'utilisation de techniques de calcul de haute performance et le design plus parallèle d'algorithmes permet l'augmentation de la vitesse de calcul nécessaire dans l'analyse de lentilles gravitationnelles. Basé sur Lenstool, un outil réputé de modélisation de masse de lentilles gravitationnelles, l'incorporation de ces techniques dans Lenstool-HPC permet son utilisation sur les superordinateurs les plus modernes, utilisant simultanément GPU (Graphics Processing Unit) et CPU. L'utilisation de matériel informatique de pointe permet donc de réduire le temps de calcul à un niveau acceptable lors de la modélisation de lentilles gravitationnelles.

Contents

List of Figures

List of Tables

Introduction

Astronomy is historically a science constantly starved for more data. The physical phenomena studied by astronomers are of an order of magnitude in terms of mass, energy and duration which simply cannot be recreated in a controlled lab environment. While numerical simulations can help with this, astronomers are essentially limited to what they can observe in the night sky. Luckily the universe is big enough to provide us with an ample amount of almost every type of situation possible as long as we can find and observe them. This means that the data available to us is mostly limited by our observational capacities, based on the running of extremely expensive and specialised ground-based and space-based telescopes that each time produce seemingly trickle amounts of new data for data-starved astronomers. Observation proposals for observation time on the best performing telescopes were and still are a matter of huge competition between scientists.

Improvements in observational techniques and the more and more international projects coming together are now starting to reverse this almost fundamental fact of astronomy. New telescopes generate so much data that in certain cases, astronomers have access to more data than is humanely possible to sort through using the human eye. Astronomers have to contend more and more with the so called "Big Data revolution" that is slowly becoming a staple in many other sectors. The exciting new possibilities that this new data presents come with their own set of challenges that astronomers are starting to grapple with. How does one handle such a huge quantity of data quickly and safely? How do we need to change the methods we already use to handle the amount of data and still reach our science goals? Strong lens astronomers have already many extremely powerful semi automated optimisation and machine learning tools but the limits of these techniques are starting to be felt strongly.

In my case, due to my involvement in the EUCLID Strong Lensing Working group and my professor Jean-Paul Kneib's involvement in cluster modelling, the problems I tackled were specifically the search for strong gravitational lenses and the modelling of cluster lenses. To solve these challenges, I worked heavily together with the HPC experts from the SCITAS department at EPFL and various machine-learning experts around the world. This thesis is therefore a technical one, firmly set in the world of astroinformatics.

The thesis is organised into three main parts: strong lensing theory, strong lensing detection and strong lensing analysis. The strong lensing theory gives an overview of the science objectives and the mathematical framework behind strong gravitational lenses. The strong lensing detection part comes directly from my work on the Euclid pipeline with the Euclid Strong Lensing Working Group and handles the detection of galaxy scale lenses using deep learning models like CNN. The strong lensing analysis part describes the work we did to rewrite Lenstool, a popular mass modelling tool for cluster scale lenses, to handle the increasingly precise available data for strongly lensed clusters.

Scientific Background Part I

1 Strong Lensing

"Matter bends light". This seemingly simple statement lies at the source of one of the most striking and visually impressive astrophysical phenomenon visible in the night sky: the strong gravitational lens.

Figure 1.1 – Galaxy cluster Abell 370. Each streak of light is in fact a magnified image of an even more distant galaxy behind the cluster. These images appear distorted and displaced because of the gravitational pull of Abell 370 which acts as a lens. Around 100 galaxies are multiply imaged by this effect. Credit: NASA, ESA, and J. Lotz and the HFF Team (STScI)

The photons emitted by distant light-sources are affected by the gravitational pull of the objects in their paths and therefore slightly deviated resulting in what astronomers call the weak-lensing effect. Because matter is not homogeneously distributed in the universe, most light-sources appear to us

slightly displaced, distorted and magnified compared to how they should appear.

In rare cases, when a light source is almost perfectly aligned with a high density mass like clusters or single galaxies, the deflection is sufficient to create multiple images of the source. The capacity of creating multiple images sets the distinction between strong and weak lensing effect.

Three interesting properties set strong gravitational lensing apart as an interesting research subject. First, its observables like relative positions and time-delays depend on the gravitational potential of the foreground object or "lens". This allows the study of the spatial distribution of Dark Matter (DM) inside galaxies which is the main provider of gravitational potential at these scales. Secondly, these observables also depend on the overall geometry of the universe through the angular diameter distances between observer, deflector and source. This makes them effective additional constraints for cosmology. Thirdly, the magnification created by the deflection of light which can increase flux by order of magnitudes allows for the discovery and study of objects too faint for actual instruments to detect.

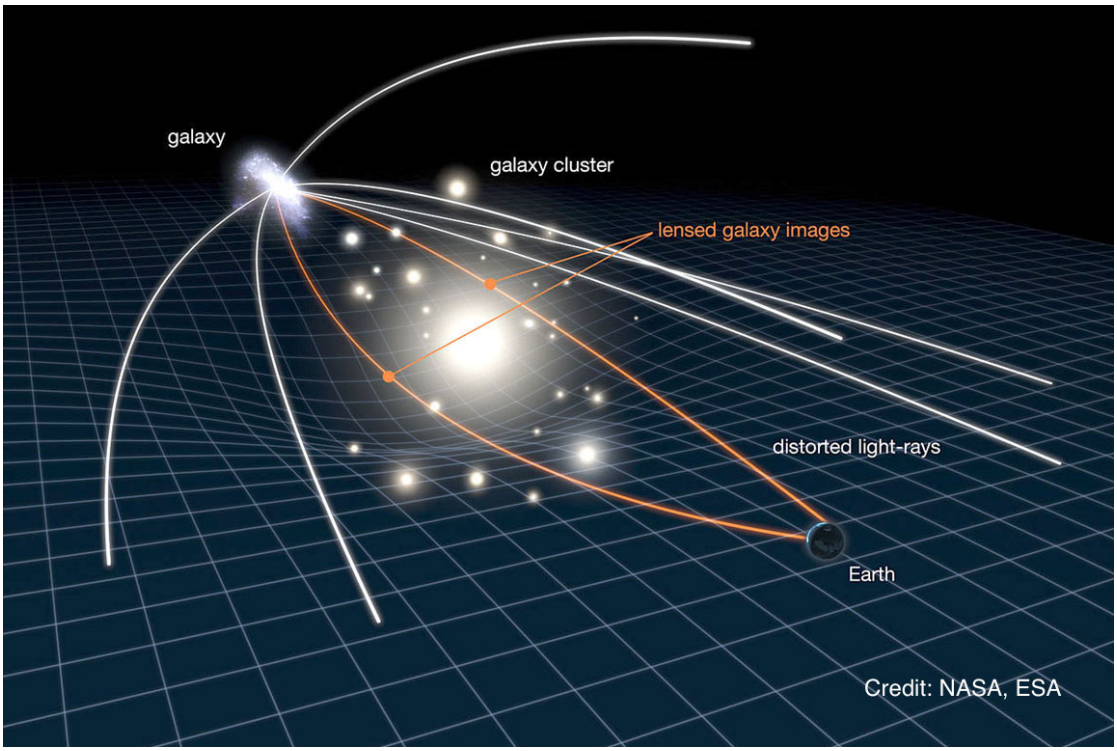
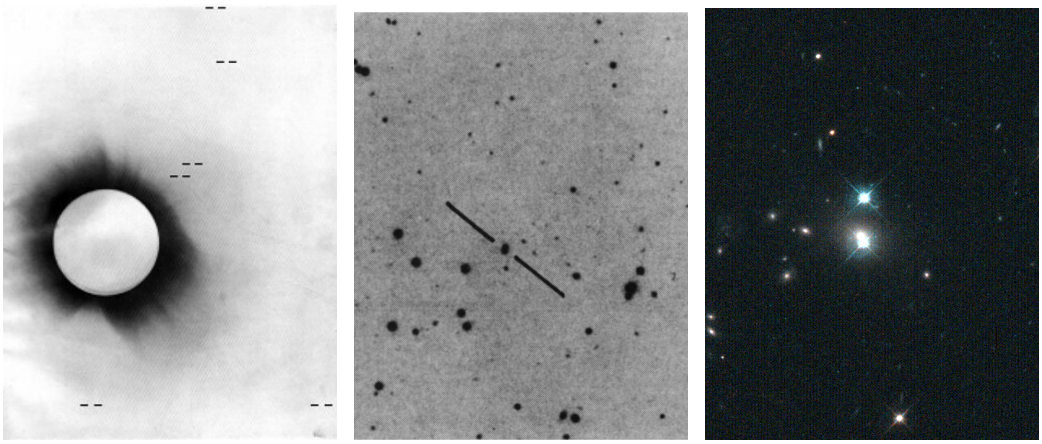


Figure 1.2 – A strong gravitational lens system: Photons emitted by a distant light sources are deflected by the gravitational potential of objects in their path, in this case a cluster of galaxies. The distorted light-rays that arrive at earth make the initial galaxy appear to us multiple times, magnified and distorted. Credit NASA, ESA

In this chapter we will go over the basic theory of strong gravitational lensing, the main science objectives using strong lenses and the difficulties involved in finding and using these lenses for science purposes. Since this thesis is not intended as a literature review, for more details the reader can refer to the book by Schneider et al. (1992) and the excellent annual reviews by Treu (2010) and Blandford and Narayan (1992).

1.1 History

The concept that matter could bend light is not a new one and was already formulated by Isaac Newton as an open question in 1704 as a logical consequence of his gravity theory. If light was made up of small particles, it would stand to reason that they too would be affected by the gravitational pull of other objects. Following this line of thought, assuming the corpuscular theory of light and Newtons laws of gravitation, Pierre Simon Laplace noted in 1799 that "the attractive force of a heavenly body could be so large, that light could not flow out of it" (Laplace, 1799). This represents one of the first theories on the existence of black holes. Based on these ideas in 1801, J. Soldner (Soldner, 1801) and Einstein (Einstein, 1911) 100 years later calculated respectively the "newtonian" and "relativistic" deflection of light by a gravitational potential. From these calculations came the idea to observe if and how much the position of stars near the sun was deflected by the mass of the sun. This was done during a solar eclipse in 1919, constituting the first observational proof of relativity theory and corroborating Einstein's calculations,



(a) Solar Eclipse snapshot from 1919 with the first deflection measurement of stars near the sun corroborating Einstein relativity theory.

(b) 0957+561, The first gravitational lens found, a quasar doubly lensed by a galaxy observed in 1979 (Walsh et al., 1979)

(c) 0957+561, The same doubly lensed quasar as to the left this time zoomed in and observed using the Wide Field and Planetary Camera 2 (WFPC2).

Figure 1.3 – Early gravitational lens observation

The idea that gravity could not only deflect but also act as a lens and create multiple images seems to have been first stated by A.S Eddington (Eddington, 1920) and followed up by R.W Mandl (Einstein, 1936). However the idea of stars bending light in such a fashion was considered very unlikely to be observed with the then generation of telescopes. Only in 1937, did Zwicky (Zwicky, 1937) go on to consider nebulae as potentials massive enough to act as lenses, establishing then, that with nebulae acting as lenses, it should be almost certain to observe a lens in a certain area of the sky. Theory continued to develop on how to exploit gravitational lensing if ever found following the discovery of quasars as possible lensed objects but it was still considered an esoteric field until D. Walsh, R.F Carswell and R.J Weymann announced the detection of the first gravitational lens candidate 0957+561, a doubly lensed quasar lensed by a galaxy in 1979 (Walsh et al., 1979). The subsequent search and findings of more gravitational lensing candidates convinced more and more of the research community of the validity of the lensing hypothesis with the first international conference on lensing held in Toronto in 1986. Since then gravitational lenses have become an established science sector with many applications in cosmology and galaxy morphology.

1.2 The Lens Equation

What is initially surprising when first starting to work with lenses is the relative simplicity of the physical theory behind them compared to other fields.

At the core of lensing theory lies a simple geometrical relation, called the lens equation. Consider a homogeneous Friedmann-Robertson-Walker (FRW) universe with cosmological density parameter Ω_0 in which we have a lens system comprised of a luminous source S, a high density mass object acting as a lens L and an observer O. The angular diameter distances between them can be described as by convention as $D_s = D(0, z_s)$, $D_l = D(0, z_L)$, and $D_{ls} = D(z_l, z_s)$.

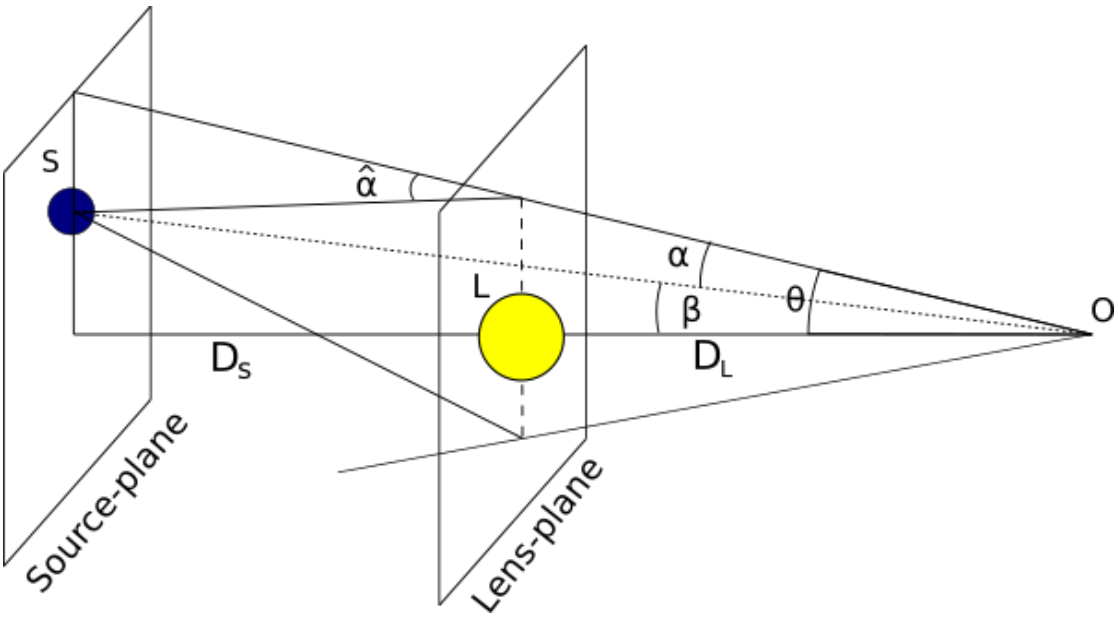


Figure 1.4 – A Lens-system is comprised of a luminous source S, a high density mass L and the observer O. The light emitted in all directions by the source S is bent around the mass L by its gravitational pull and arrives on multiple different paths at the observer O.

The lens equation is derived using simple geometry from a schematized lens-system (Fig. 1.4) and connects the observed image position θ to the source position β

$$\beta = \theta - \alpha(\theta) \tag{1.1}$$

through the reduced deflection angle $\alpha(\theta)$ (also called scaled deflection angle). The reduced deflection can be calculated from the deflection angle $\hat{\alpha}(D_l\theta)$ as follows.

$$\alpha(\theta) = \frac{D_{ls}}{D_s}\hat{\alpha}(D_l\theta) \tag{1.2}$$

$\hat{\alpha}(D_l\theta)$ is the deflection angle that a light ray would experience when passing through a specific point

of the lens-plane. The lens-plane is defined as the plane perpendicular to the line between lens and observer centered around the lens.

Because the distances between the objects of interest are of order of magnitudes bigger than their sizes, the deflection angles can always be considered small. This allows us to use the weak field approximation of general relativity, meaning that for a thin lens the whole problem can be described as the following series of Poisson like equations:

$$\nabla^2\psi(D_l\theta) = 8\pi G\Sigma(D_l\theta) \tag{1.3}$$

where $\psi(D_l\theta)$ is the 2D Newtonian gravitational potential obtained and $\Sigma(D_l\theta)$ the 2D-surface mass density projected unto the lens-plane . From that potential we can derive the deflection angle:

$$\hat{\alpha}(D_l\theta) = \nabla\psi(D_l\theta)/c^2 \tag{1.4}$$

These equations establish a link between the 2D-surface mass density of the lens ($\Sigma(D_l\theta)$), the distance between observer, source and lens and the observed position of the multiply imaged sources which is crucial for the science applications of gravitational lensing. For more details on the hypothesis involved and the formal proof of the equations above, please refer to Blandford and Narayan (1992); Schneider et al. (1992).

The lens equation represents a non-linear problem. For each image position θ there exists an unique source position β but the inverse is not necessarily true. When source, lens and observer are sufficiently aligned unto one axis, the lens equation can have multiple different θ solutions. This is characteristic of the strong lensing regime with each solution being the position of one multiple image.

From the lens equation we can derive also other lens properties and observables. The Jacobian of the lens equation defines the inverse magnification tensor written as

$$\frac{\partial\vec{\beta}}{\partial\vec{\theta}} = \delta_{ij} - \frac{\partial^2\psi}{\partial\vec{\theta}_i\vec{\theta}_j} = \begin{pmatrix} 1-\kappa-\gamma_1 & -\gamma_2 \\ -\gamma_2 & 1-\kappa+\gamma_1 \end{pmatrix} , \tag{1.5}$$

with the shear components γ_1 and γ_2 representing the local distortion of a source image. The convergence parameter κ measures the isotropic part of the magnification. From its relation to $\Sigma(D_l\theta)$ the 2D-surface mass density projected unto the lens-plane, we can define the critical density Σ_{cr}

$$\kappa(D_l\theta) = \frac{\Sigma(D_l\theta)}{\Sigma_{cr}}, \quad \Sigma_{cr} = \frac{c^2}{4\pi G}\frac{D_s}{D_l D_{ls}} = 0.35\ \mathrm{g\ cm^{-2}} \left(\frac{\mathrm{D_s}}{1\mathrm{Gpc\ D_l D_{ls}}}\right)^{-1}. \tag{1.6}$$

Σ_{cr} defines the limit between the weak and strong lensing regime with a lens with multiple images having $\Sigma > \Sigma_{cr}$. That limit also defines the lines (on the image plane) called critical lines when the

magnification of the lens becomes formally infinite. As an example in the case of a point source, the local magnification is given by the determinant of the magnification tensor.

$$\mu = \frac{1}{(1-\kappa)^2 - \gamma_1^2 - \gamma_2^2} \tag{1.7}$$

For non-point sources the magnification also depends on the surface brightness distribution of the source. When $\kappa \approx 1$, the magnification becomes formally infinite allowing compact images close to the critical line to be magnified by up to two orders of magnitude. Critical lines can be unlensed back unto the sourceplane, forming caustics. Every source crossing a caustic will generate two more or two less images, with each pair being stretched along the critical lines. A schematised example for a spherical and elliptical lens can be found in Fig 1.5.

1.3 Galaxy and Cluster Scale Lenses

Gravitational lenses come in many shapes and forms but a general classification can be made based upon the object acting as a lens. Typically strong gravitational lenses can be separated into galaxy and cluster lenses. Examples of galaxy lenses are Fig. 1.6a and 1.6b and for cluster lenses, Fig. 1.1, 1.8 and 1.12.

For **galaxy lenses**, any type of galaxy can act as a lens provided it is massive enough. The most commonly found galaxy type acting as a lens are massive elliptical galaxies of velocity dispersion between 200-300 km.s^{-1}. The resulting lens images are relatively simple compared to cluster lenses because the lensing potential of one galaxy is fairly homogeneous. These lenses have an einstein radii of approximately $E_r = 1.''0$ and are typically lens quasars (QSO) or radio sources. These objects are found more often because they are typically bright and are easier to pick out. Spiral galaxy lenses are less likely to be found despite being more numerous than elliptical galaxies because their lesser mass leads to a lower einstein radius and a lower angular separation between the images. They exist but are simply much more difficult to detect since the multiple images cannot be distinguished from each other or the lens. A good example is the Sloan Lens ACS survey (SLACS) using the Hubble Space telescope (HST) which has observed and confirmed in detail 130 lenses (Auger et al., 2009). The lensing galaxies have an average velocity dispersion of 248 km.s^{-1}, with an rms scatter of 46 km.s^{-1}. They consist of 80% elliptical, 10% lenticular and 10% massive spiral lenses. Lenses with lower velocity dispersion exist but for them the lensed image separation drops to below $0.''3 - 0.''4$ which is the current detection limit for the HST and the Very Large Array (VLA). As the resolution of telescopes improves further, less massive lenses should be discovered.

Most confirmed galaxy lenses are observed up to redshift $z < 0.4$, mainly because for these redshift ranges spectroscopic information is easier to get than for higher redshift objects. Spectroscopy in turn is necessary for confirming if an object is a lens or not and for accurate redshift measurement. The source objects redshift can go back as far as redshift $z < 4$ (see Fig.1.7a,Treu (2010)) with some even higher redshift sources having been confirmed.

Cluster lenses are objects so massive that they always act in some way as a lens. They are composed of hundred of galaxies, making them the most massive virially bound objects in the universe, and the deflection maps they form are a lot more complex than for galaxy scale lenses. A massive DM

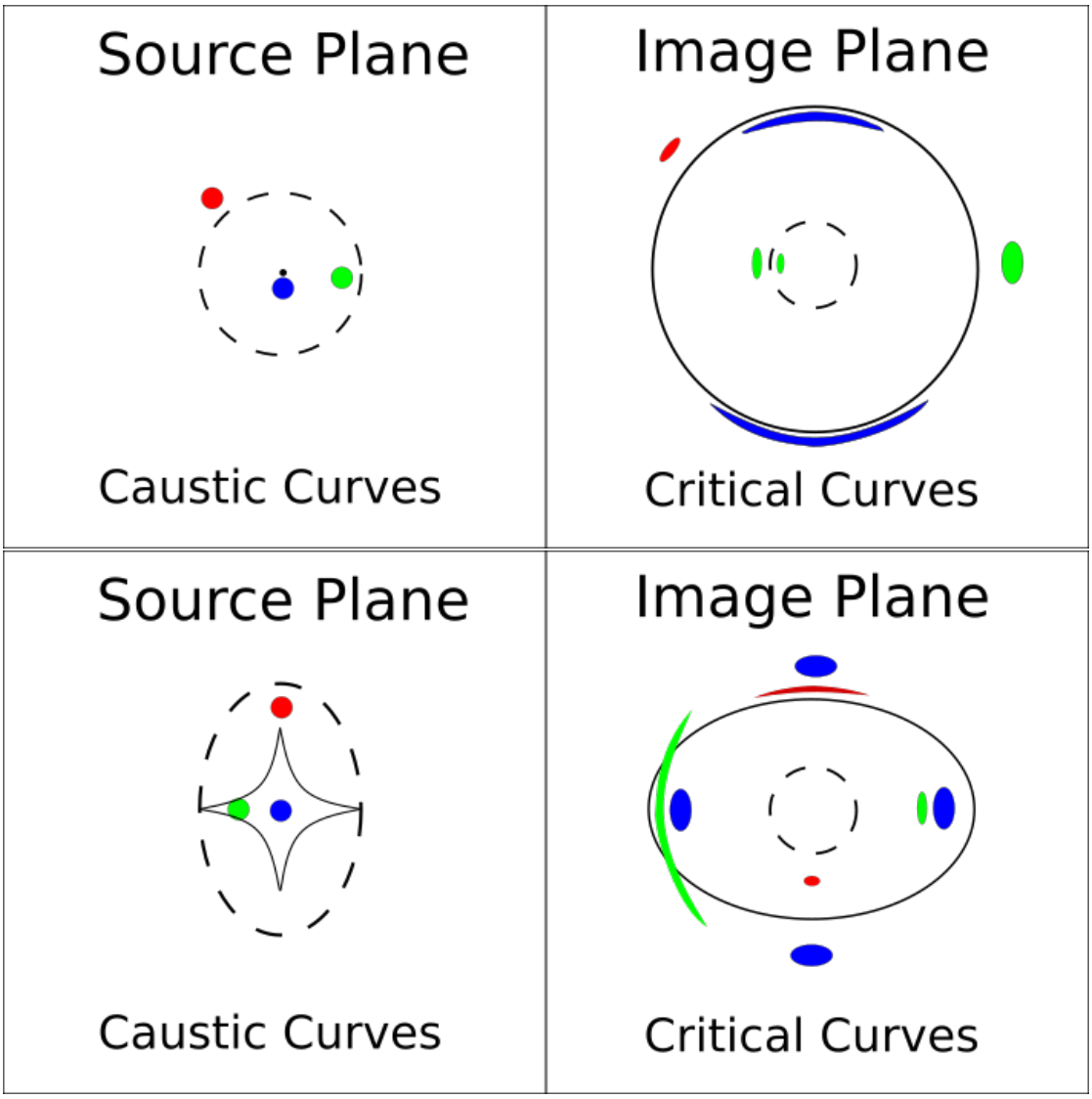
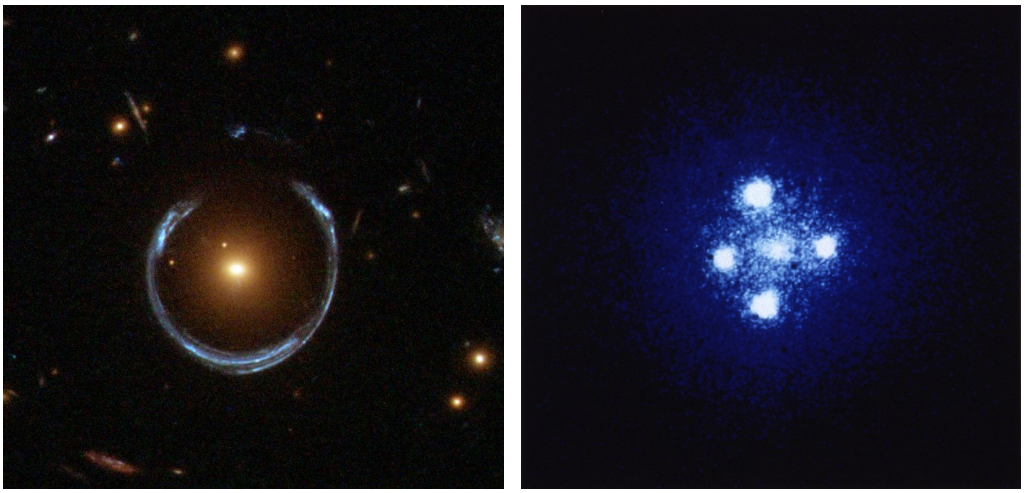


Figure 1.5 – Caustic and critical lines generated respectively by a spherical and elliptical lens. As each source crosses a caustic line in the source plane, it generates two more or two less multiple images. The multiple images appear on the image plane in the regions set by the critical lines. Near the caustic lines, the amplification and deformation effect is highest, so that sources start forming distorted arcs and Einstein rings. Caustic lines closest to the center correspond to the furthest critical lines. The ellipticity of the lens influences if an Einstein ring is formed: the more spherical a lens, the more perfect the resulting Einstein ring.



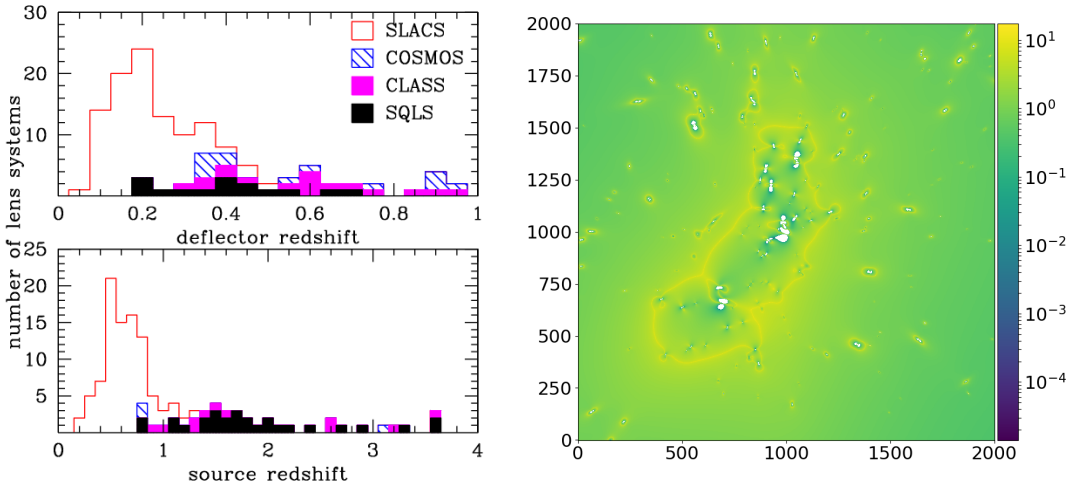
(a) Horseshoe Einstein Ring (b) Einstein cross

Figure 1.6 – Gravitational Lens observation using the Hubble Space telescope (HST). The left image shows a background galaxy distorted into an almost perfect Einstein ring (LRG 3-757). The right is a quadruply lensed quasar (Q2237+030) with a fifth image at the center too faint to observe. Credit: NASA, STScI, and ESA

halo contributes the bulk of the deflection potential with each separate galaxy member of the cluster contributing an additional significant amount of gravitational potential (see Fig 1.7b & 1.8). The higher mass and larger size means that they affect a much bigger area than galaxy lenses. The MAssive Cluster Survey (MACS) clusters (Ebeling et al., 2001; Kneib and Natarajan, 2011) have for example a median Einstein radii of around 28.″. The corresponding critical and caustic lines are extremely dependent on the complex underlying mass distribution of the cluster, making clusters a prime case studies for DM observation. The higher mass also results in higher amplification factors. Amplification factors of 4 are commonly observed and can reach up to 40 close to the critical lines. The higher amplification allows fainter sources at even higher redshifts to be observed with Lyman alpha emitters having been observed up to $z < 8$ (Stark et al., 2007; Clément et al., 2011). Most sources are however observed between redshift $3 < z < 7$. Cluster redshifts tend to be between $0.2 < z < 0.8$.

1.4 Science Objectives and Applications

Strong lenses are interesting objects to study because of their unique observables which can be used to answer many open questions. Astronomers can use them to measure the mass of the lensing galaxies without needing the mass to be luminous. This is crucial in dark matter studies. The magnification of distant objects by strong lenses allows us to probe the Universe at the highest redshifts ever and they have a unique application in cosmography through time delay calculation using multiply lensed quasars. The following section goes briefly over these science applications, to give the reader an overview of the potential of lenses. For more details on the corresponding science applications, please refer to the corresponding citations.



(a) Redshift distribution of galaxy scale lenses and of corresponding lensed sources in SLACS, COSMOS, CLASS and SQLS survey. Credit: Treu (2010)

(b) Amplification map example for Abell 2744 based on the lens model by Jauzac et al. (2016). The clear lines where the amplification is highest trace perfectly the critical lines of the lens system. Cluster lenses have typically more complex lens models because of the perturbative potential of each member galaxy.

Figure 1.7

1.4.1 Study of Dark Matter

The standard cosmological model which stands at the center of modern astronomy predicts that the mass energy content of the universe consists of 5% baryonic matter, 27% dark matter, and 68% dark energy. The two latter parts scientists know almost nothing about. However Dark matter (DM) is considered to interact with baryonic matter mainly through gravitational interaction. With strong lens deflection depending only on the gravitational interaction, the mass measurements of lensing galaxies and clusters through strong lenses become an excellent way for us to constrain DM.

Around galaxies and clusters the **existence of massive DM haloes** is now a widely accepted fact. DM halos have been detected by kinematic studies of galaxies, through weak lensing studies for statistical sample up to $z < 5$ (Wechsler and Tinker, 2018) but also by strong lensing. Using strong gravitational lenses one can calculate the mass of the lensing galaxy inside its Einstein radius. For massive elliptical galaxies however, up to $z \sim 1$ acting as lens galaxies, the masses found inside of the Einstein ring clearly exceed the stellar mass M_* inferred from the flux of the galaxies. Another type of non-luminous matter therefore has to exist to explain this surplus mass which is typically accepted to be DM.

The precise mass measurements strong lensing provides can be used to accurately compute the **fraction of DM** inside galaxies which is necessary to the understanding of the interplay of baryons and DM (Jiang and Kochanek, 2007). These studies also serve to constrain and explain the **fundamental plane relation** for early type galaxies, which links effective radius, effective surface brightness and stellar velocity dispersion (Bolton et al., 2007, 2008; Thomas et al., 2011).

Strong gravitational lens mass-measurements can be combined with dynamical studies to explore

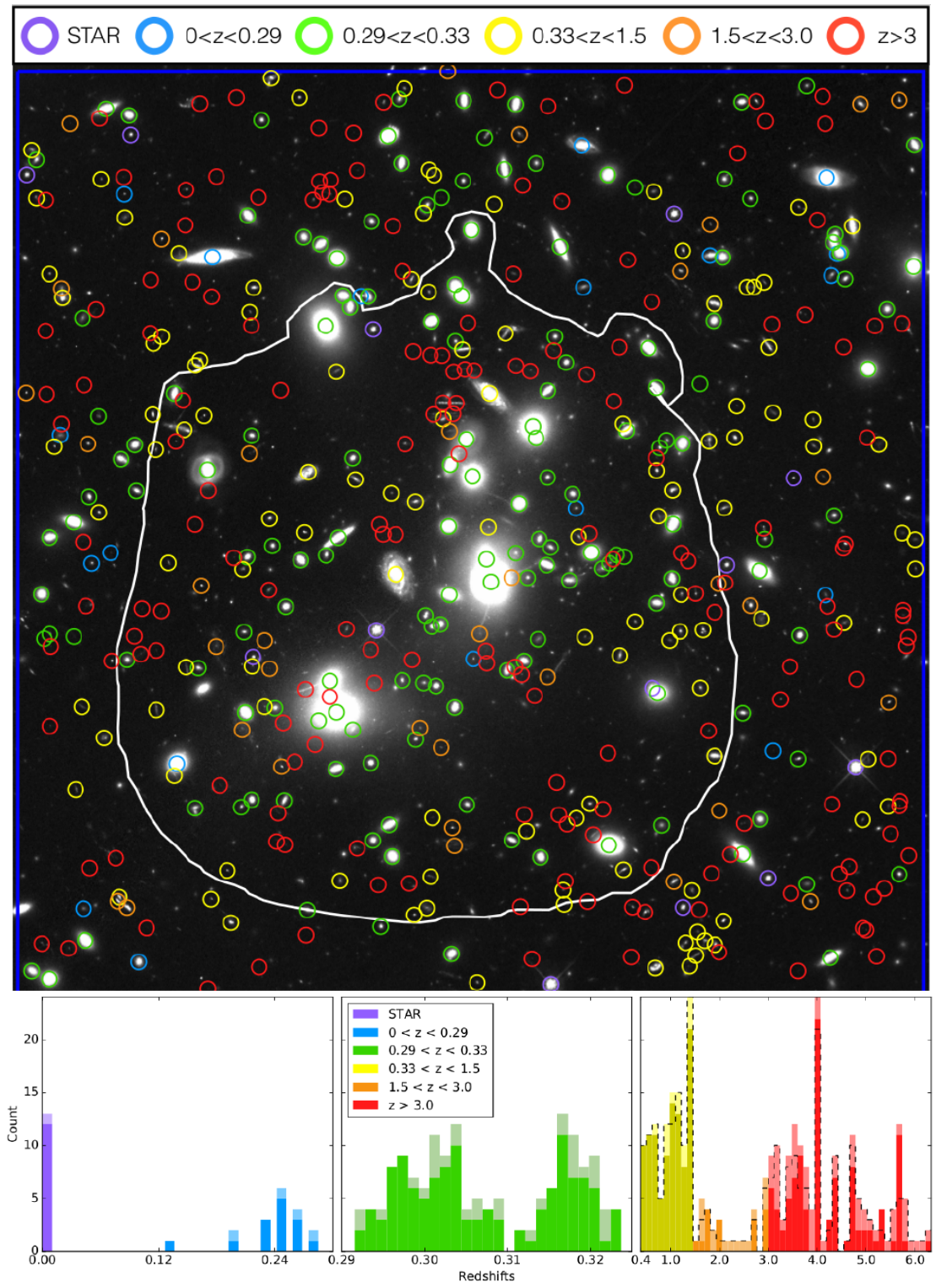


Figure 1.8 – Redshift distribution of object in cluster Abell 2744: The different redshifts ranges are color-coded and spatially represented on the image. The corresponding redshift distribution is shown in the histogram in the lower panel. The blue and purple redshifts belong to objects in the foreground between us and the cluster. Green corresponds to the cluster members and red and yellow to the sources behind the cluster being lensed. Credit: Mahler et al. (2018)

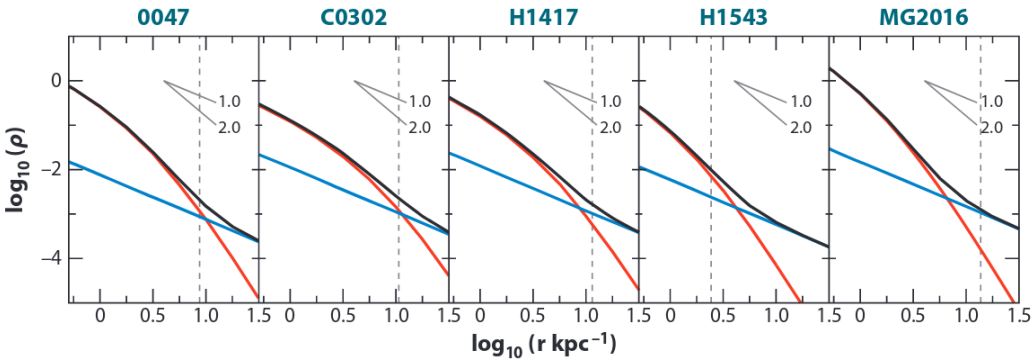


Figure 1.9 – Mass-density profiles of lens galaxies inferred from a strong lensing and dynamical analysis. The luminous matter (red) combines with the dark matter halo (blue) to form an almost perfect isothermal mass profile (black) with $\gamma = 2$. The vertical dashed line shows the location of the Einstein radius. (Figure from Treu & Koopmans 2004)

the mass distribution of galaxies because they resolve a mass anisotropy degeneracy that dynamical studies face (Treu and Koopmans, 2004; Barnabè et al., 2009). The resulting mass-profiles show that there seems to be a remarkable regularity in the mass structures of early-type galaxies coined the **bulge-halo conspiracy**. Early-type galaxies all seem to have approximately isothermal mass-profiles best described by a powerlaw $\rho(r) \approx r^{\gamma}$ with $\gamma \approx -2$ out to very large radii with almost flat rotation curves (Rusin and Kochanek, 2005; Koopmans et al., 2009). Recent studies (Ruff et al., 2011; Sonnenfeld et al., 2015; Shankar et al., 2017, 2018) have shown that variations do occur in function of projected stellar mass density and redshift but no definite explanation has been found yet.

Strong lensing can also be used to probe the **core-cusp problem**. Early dark matter Nb-body simulations have predicted that dark matter halos all possess similar "cuspy" density profiles that can be fitted by simple analytical functions like the Navarro-Frenk-White (NFW) profile $\rho_{DM} \approx r^{-\gamma}$ with $\gamma \approx 1$ (Navarro et al., 1997). In other words they predicted a "cuspy" mass density in their center. Observations however show that dwarf, spiral galaxies and galaxy clusters tend to have shallower inner mass density profiles, forming more of a "core" at the center of the DM halos (Sand et al., 2008; Zitrin et al., 2015).

The core-cusp problem is an exceptional observational constraint which can be solved by making some revision to Cold Dark Matter (CDM) theory (Yoshida et al., 2000; Ahn and Shapiro, 2005) or by adapting the nature of DM. It could also mean that certain baryonic processes like stellar feedback have a significant effect on DM halo evolution and cannot simply be neglected (Navarro et al., 1996; Teyssier et al., 2013; Peirani et al., 2017).

Another open question for which strong lensing is interesting is the **excess subhalo problem**. Following standard cosmology, DM halos should host a certain number of subhalos also called substructures. However when comparing N-body simulations to observational data, significantly more substructures are generated in the simulations than actually observed. While clusters have hundreds of verified galaxies within their own halos, galaxies have very few luminous satellites that could correspond to the predicted substructures. This could be because these substructures consist only of DM and therefore are not visible or that cosmological theory is wrong on the number of subhalos generated. Strong lenses are critical for answering this question because they are the best at detecting DM substructure and

comparing it to simulation. This can be mostly done through the study of flux anomalities of lenses. These flux anomalies are due to the perturbation of the magnification pattern of the lens which could be generated by the presence of DM substructures. More details on the excess subhalo problem can be found in the review by Kravtsov (2010).

1.4.2 Cosmic telescope

A straightforward use case for strong gravitational lenses is as a nature-made telescope. With it, it is possible to resolve galaxies of the distant universe (at redshift above $z >> 0.1$) with more details than using only man-made telescopes (Marshall et al. (2007); Riechers et al. (2008); Sharon et al. (2019)). This can be used for example to study the stellar initial mass function (IMF) of high redshift galaxies as with "The Cosmic Horseshoe" lens (Fig. 1.6a, Quider et al. (2009)). More details on IMF can be found in the review on IMF by Bastian et al. (2010).

Cluster lenses are especially useful in this regard because of the huge magnification they can reach, up to factors of 40x (Kneib et al., 2004; Richard et al., 2011). With the magnification being wavelength independent, cluster observations have been done not only in the visible but also in the submiliter (Smail et al., 1998), mid infrared (Altieri et al., 1999) and far infrared domain (Altieri et al., 2010).

Submillimeter observations are used for high redshift submillimeter galaxy detection ($1 < z < 5.5$) (Smail et al., 1998), to measure their exact contribution to the background radiation. From this we know that submillimeter galaxies generate a significant fraction of the energy output of all galaxies in the early Universe (Blain et al., 2002). Submillimeter observations also allow us to study the starforming regions of these galaxies in great detail (Swinbank et al., 2010). Mid-infrared observations probe similarly to submillimiter observations the faint and distant mid-infrared galaxy population and their contribution to the cosmic mid-infrared background radiation. Far-infrared observations were used for the study of Extremely Red Objects (ERO), some particularly peculiar galaxies (peculiar galaxy morphology study) which revealed themselves mostly to be young dusty star bursts at $2 < z < 3$ (Schaerer et al., 2007; Vieira et al., 2013).

The extreme amplification region around critical lines allows for the detection of Lyman alpha emitters present at very high redshift ($4 < z < 7$). Lyman alpha emitters are extremely young galaxies which act as probes crucial for the study of the reionisation period of the Universe (Stark et al., 2007; Clément et al., 2011).

1.4.3 H_0 measuring through time-delay monitoring

One other exciting science application for strong gravitational lenses concerns the Hubble H_0 constant which measures the expansion rate of the universe. The true value of H_0 is another open question in astronomy which is leading to contention inside the astronomical community. Not considering strong gravitational lenses, two famous H_0 measurement methods find significantly different values for H_0. With both methods, one based on Cepheid and supernova measurement and the other on cosmic microwave background (CMB) predictions using the Planck satellite, being independent from each other, this H_0 tension hints strongly to a problem in our cosmological models.

H_0 measurements using time-delays was pioneered by the H0LiCOW collaboration (Suyu et al., 2017), a program specialised in monitoring strongly lensed quasars for time delay calculation. Quasars (QSO) are extremely bright sources with variable luminosities of time scales ranging around days and months.

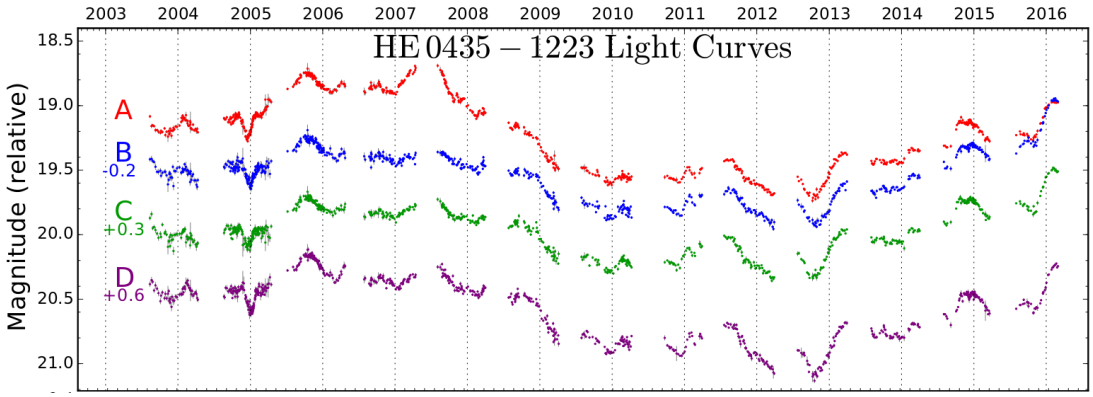


Figure 1.10 – Light curves for the four lensed images of the quasar HE 04351223. The relative shifts in magnitude are chosen to ease visualization, and do not influence the time-delay measurements. Credit: (Bonvin et al., 2017)

When observing multiply imaged quasars, it is possible to observe a time-delay between the multiply imaged signals (Fig. 1.10).

This time delay depends on the matter distribution in the lens galaxy, on the overall matter distribution along the line of sight (which can perturb measurements) and on the cosmological parameters, primarily H_0. This approach to measure H_0 was first suggested by Refsdal (1964) but only after an extensive monitoring campaign by the COSMOGRAIL project and complex modelling efforts were first results achieved (Bonvin et al., 2017). Instead of bridging the results of both preceding methods and pointing to some unaccounted for observation bias, it exacerbated the H_0 tension by corroborating the Cepheid based result (Fig. 1.11).

1.5 Finding Strong Lenses

One of the main problems astronomers encounter when using gravitational lenses is the scarcity of data. Lenses are rare phenomena with only roughly a 1 to 1000 chance of appearing for every other astronomical object observed (Collett, 2015). To find lenses, we have to rely on major ground based sky surveys, and then use expensive spectroscopic follow-up to confirm the lens. In these surveys the first lenses were found serendipitously by simple human eye observation of the data, but the ever increasing amount of data quickly made this an unreasonable proposition. Semi-automatic machine learning tools like arc finders (More et al., 2012), PCA based finders (Paraficz, D. et al., 2016) and, when available, spectroscopic analysis were quickly developed and used to help in the task. This made possible the creation of the first catalogs of lenses, allowing strong gravitational lenses to be finally used for science purposes.

We can mention here the Cosmic Lens All Sky Survey (CLASS) (Browne et al., 2003), searching for radio-loud gravitationally lensed systems with approximately 30 confirmed lenses to its name, or the Sloan Lens ACS (SLACS) Survey with some 150 confirmed lenses with subsequent Hubble follow-up observation (Shu et al., 2017).

However present-day and upcoming surveys like KIDS (Kilo Degree Survey), Euclid, DES (Dark Energy Survey) (Diehl et al., 2017), HSC (Hyper Suprime Cam) (Chan et al., 2019) and others (More et al., 2012;

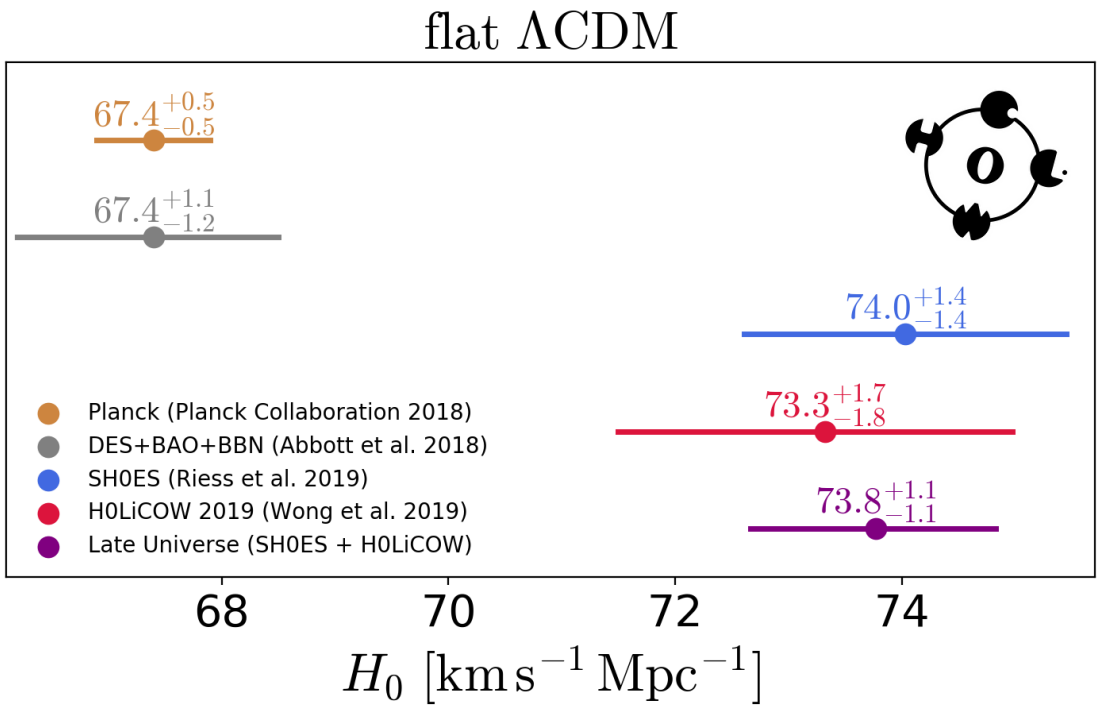


Figure 1.11 – Comparison of H0 constraints for early-universe and late-universe probes in a flat CDM cosmology (H0LICOW XIII.). One can clearly see the non agreement between the cosmic microwave background (CMB) based results and the Cepheid and time-delay results. Figure credits : Vivien Bonvin/Martin Millon.

Talbot et al., 2018) generate so much data that detecting promising lens-candidates by human eye interaction is becoming impossible. To give an idea of the magnitude of the problem, only around 600 confirmed lenses have been found to date and Euclid alone proposes to find 100 000 galaxy lenses and 5 000 cluster lenses hidden in a corresponding amount of information (Collett, 2015). More efficient and automated methods have therefore become even more crucial. This is the subject of the strong lensing detection part of this thesis.

1.6 Gravitational Lens Mass Modeling

Another problematic for strong lens astronomers is the scarcity of precise mass-models. To model the mass distribution inside galaxy clusters, the standard procedure is to use the multiply imaged lensed sources as constraints to calculate the fit of a specific mass distribution. As additional metric, one can use the magnitude of the images and the shape of the images, since lensing theory affects and predicts these values, but the image position remains the main constraint used for lens-modelling. The best-fit mass distribution is then found by using a domain sampler like a Markov Chain Monte Carlo sampler (MCMC).

The problem astronomers face in producing mass models is the sheer complexity of the models. We will go more into details in the chapter dedicated to this but the models we produce can depend on hundreds to thousands of free parameters to accurately depict the complexity of the mass distribution

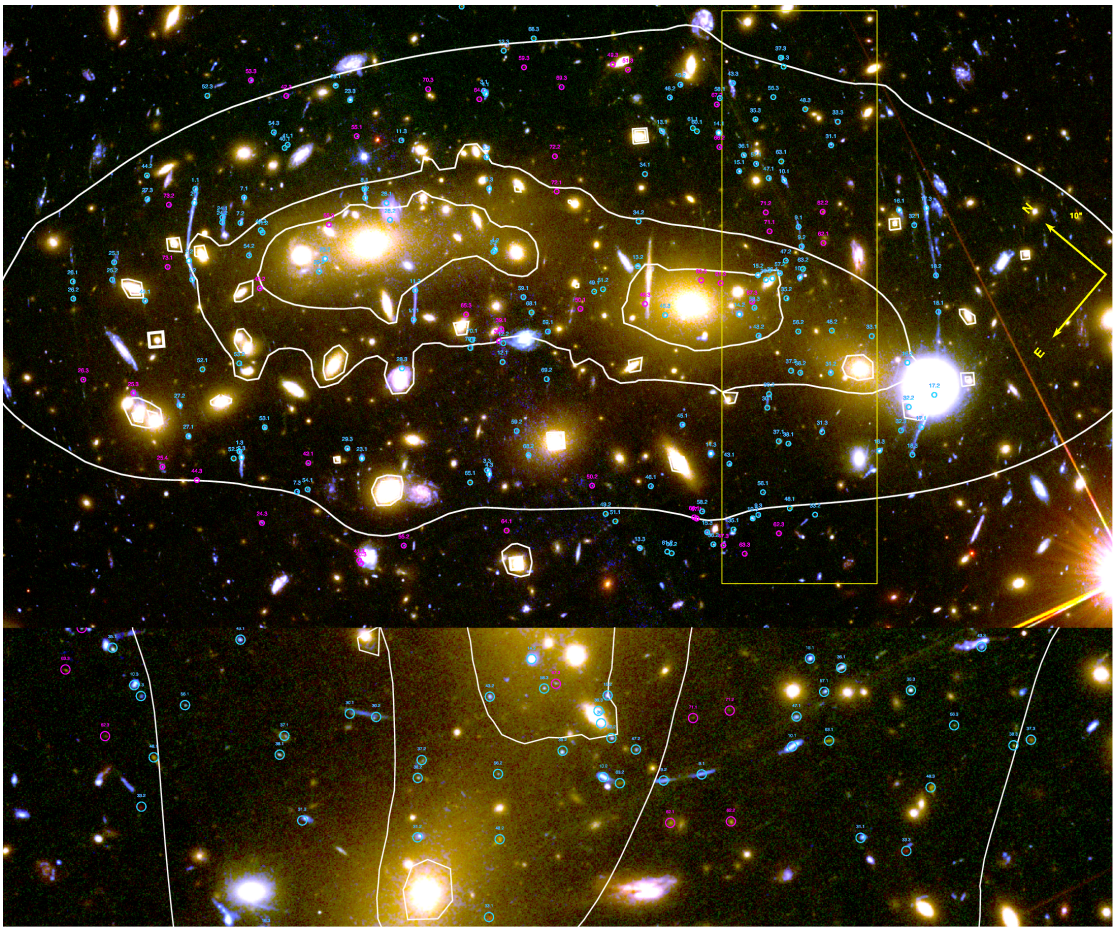


Figure 1.12 – MACSJ0416-2403: a strongly lensed cluster with approximately 200 images. The white lines give an overview of the complex underlying mass distribution. The lower part of the image is a zoomed cutout of the cluster indicated by the yellow rectangle. The multiple images are indicated by small colored circles. Credit: Jauzac et al. (2014)

of a cluster. Strong lens modellers are therefore affected by the well-known curse of dimensionality. Simply put, the parameter space we have to explore to find the close to optimum solution expands exponentially in function of the free parameters making a full exploration extremely lengthy in the case of numerous free parameters.

In addition to this for each step, the fit computation of the tested mass distribution is computationally costly. The non-invertible nature of the lens equation that allows it to have multiple solution and therefore to generate multiple images for one source precludes an analytical solution. It is easy to unlens a point unto the source plane but impossible to relens it. Instead the algorithm is forced to rasterise the lens-plane, unlens every pixel of the grid unto the source plane and check for a source.

As a consequence one optimisation run of a complex cluster like the MACSJ0416-2403 (Fig. 1.12) can last for weeks, potentially even months. These computation times are simply not acceptable anymore with observational data of massive clusters likely to improve in the future. To handle this problem, we worked on developing a High Performance Computing mass modeling tool for cluster lenses. This is the subject of the strong lensing analysis part of this thesis.

Part II Strong Gravitational Lens detection

2 Deep Learning

The amount of data that astronomers, especially observers, are required to handle has long reached a critical amount. We are reliant on machine-learning based semi-automatic and automatic methods to classify the objects we detect and to predict the variables of our cosmological models. As a consequence, machine learning has been for some time an area of high interest for astronomers. Among the many fields of machine learning however, deep learning is still something of a new field for astronomy. While by the time of writing this thesis the process is already well underway, astronomers are still discovering many different possible applications of deep-learning in their research.

Since in this thesis, we explored the viability of deep learning for strong gravitational lens classification, in this chapter we will go over the core concepts of machine learning and deep learning in particular. Most of these core concepts come from the book "Deep Learning" by Goodfellow et al. (2018) which I recommend for further theoretical details. If the reader is interested in more practical examples, I strongly recommend the classic MNIST and Keras tutorials.

2.1 History

For classification problems in general, the last decade has seen the resurgence of a particular field of machine learning called Deep Learning. Deep learning is a machine learning field that groups together methods using artificial neural networks. The initial idea, which dates back to the early fifties, was to create models that imitated the human brain. Perceptrons (Rosenblatt, 1957) organised into a layer, each representing a simplified version of a biological neuron, were able to learn linear functions allowing for some simple classication tasks. The initial attempts of single-layer perceptrons had some success but were unable to learn any non linear functions until 1986. It was then that Rumelhart et al. (1986) showed that by backpropagating training errors through layers it was possible to stack multiple layers of perceptrons. Following the universal approximation theorem (Cybenko, 1989; Hornik, 1991), the more layers were stacked the more complex the functions learned could be. The prohibitive computation cost of the training and high success rate of other methods like Support Vector Machines (SVM) however slowed down research in the domain until 2012. Interest picked up, and continues up to today, when a convolutional neural network (CNN) called "ALexNet" (Krizhevsky et al., 2012) beat the best preceding score in the Large Scale Visual Recognition Challenge(LSVRC) by halving the errors rates from 26% to 16% using graphics processing units (GPU). With more advanced versions of neural networks with less prohibitive training costs available and GPUs to compensate for those, deep learning has now become one of the fastest expanding current research fields, specialised in image

recognition and parameter estimation.

2.2 Learning for Machines

Deep learning belongs to the machine learning algorithm category. One thing that differentiates machine learning algorithms from others is that they are capable of **learning**. What that exactly means is the critical component to understand when using them.

According to Mitchell (1997), "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The concept of learning here refers to the capability of the software to "learn" how to best perform a certain task on data based on some metric that we give it. Any task for which a performance metric can be written could theoretically be learned: classification, regression, data denoising, anomaly detection, etc. The performance metric which is often specific to the task is what allows us to measure how well a program is doing. From it we can define a loss or cost function which has to be minimized to increase performance. The learning or minimization process of the loss function is done by letting the program experience a data set. At each step, depending on its performance, it updates its model parameters following an optimisation procedure to reduce the loss.

Generalisation to unseen data

The key concept here is that the goal of learning is not to perform well on already seen data but on data it has never seen before. This means the software has to **generalise** its learned experience. This is the main difference between machine learning problems and optimisation problems. To evaluate the generalisation, or more commonly known, the test error, data sets are often divided into two sets, a training set and a test set. The training set is used to train the predictive model while the test set is used only to measure its generalisation performance and not used at all for training.

How can we expect an algorithm learning on training data to perform well on test data it has never seen before? This is possible because we make a certain set of assumptions on the data-generating process behind the training and test. These are the i.i.d assumptions which suppose that the examples in the data sets are **independent** from each other and **identically distributed**, meaning generated using the same underlying probability distribution. The model is trained using the training set to learn the underlying probability distribution and using this knowledge to perform well on any data using that distribution. As a consequence of this, the test error will almost always be higher than the training error.

Performance

Determining how well a machine learning algorithm is performing therefore depends on two factors: Its ability to have a small training error, and to generalise well, meaning to keep the difference between test and training error small. Avoiding **underfitting** (high training errors) and **overfitting** (big difference between training and test scores) are the two major challenges of the learning process. A model's tendency to overfit or underfit depends on the models capacity, its capability of modelling a wide variety of complex functions. An excellent example for this is the linear regression model (see Fig.2.1).

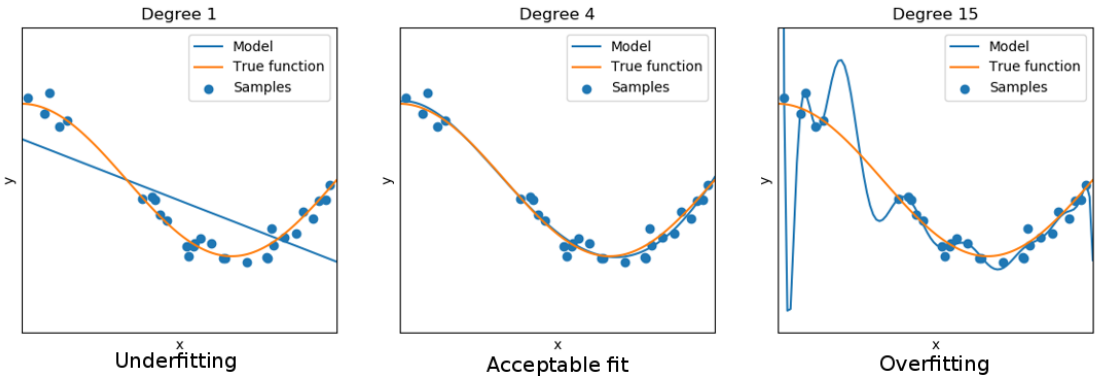


Figure 2.1 – Overfitting and underfitting with linear regression: The blue points are randomly generated from a distribution function (orange) with some added noise. The blue line is a linear regression model with polynomial features of degree 1, 4 and 15 fitting the data (blue points) trying to estimate the underlying distribution function. At degree 1 it is underfitting: the model is not complex enough to fit the data (high training error) or the underlying function. At degree 15 the model is too complex, it captures perfectly every point but does not generalise well the underlying distribution function: it is overfitting. To correctly estimate the distribution function, a balance between both has to be achieved. The figure was generated using the scikit python module.

A models tendency to overfit can also come from a training set which is not big enough to truly sample the underlying probability distribution. In these cases increasing the size of the training set can often help the model reduce overfitting.

Essentially statistical learning theory (Vapnik, 1995) tells us that "the discrepancy between training and test error is upper bound by the models complexity which shrinks with increasing training size". This gives us the theoretical proof that machine learning models can learn to work with never seen before data. In practice, achieving this can be a bit more complex.

Summary

To summarise the preceding, a basic machine learning problem and solution consists of four things: An extensive data set divided into training and test set following the i.i.d assumptions, a costfunction adapted to the task needed, an optimisation procedure to minimise the costfunction and a machine learning model to perform the task.

The model must be sufficiently complex to not underfit the problem while not so complex as to overfit. Overfitting can be worked against using larger dataset and better optimisation functions. Every algorithm belonging to the machine learning class follows these concepts.

2.3 Deep Learning Models

Deep learning is a term that is used to describe a network using artificial neurons. Given an input x, neural networks try to learn a function $y = f(x;\theta)$ with y being for example some classification of x.

They are called networks because to model complex functions, they are often composed of multiple minor functions $f(x) = f3(f2(f1(x)))$ called **layers** of the network. The overall length is associated with the **depth** of the network thus the name "deep". The deeper it is the more complex functions it can represent. The last layer of the network is called the output layer which will give the result of the function learned by the network. All layers in between input and output layer are commonly coined as **hidden layers**.

The term "neural" originates historically from the decision to organise these layers into units that act in parallel each representing a vector to scalar function. The units resemble neurons because they output a single value from multiple output and in most cases imitate how neurons fire when receiving sufficient input. However while the initial research was partly inspired by neuroscience, modern neural network research has now branched out into different directions.

2.3.1 Artificial Neural Network

The previous concepts being a bit abstract, I will go over them using the Artificial Neural Network (ANN) as a more detailed example. The ANN is among the simplest neural networks models still used to date and is an excellent example for deep learning concepts and tutorials.

The model

Like most neural networks, it is composed of multiple layers of artificial neurons and can be used to model a response variable like class labels or target variables from a series of input features.

Each neuron in a layer is a weighted linear function which takes as input all the output of the preceding layers (see fig.**??**). The output is then fed through a non linear **activation function** before being passed to the neurons of the next line. The output is therefore

$$a(\sum w_i x_i + b) \tag{2.1}$$

with w_i the weights for each input, b a bias and $a()$ the activation function. The **weights** and **biases** of each neuron are trainable parameters meaning they will be updated to improve the capabilities of the network. The non-linear activation functions are considered **hyperparameters**, they do not change during training and are decided at the conception of the model. These non-linear functions are important because they make it possible for the ANN to learn non-linear models. Activation functions essentially determine if a neuron is activated ("fires") or not and therefore if its input is to be considered by the subsequent layers. They also typically normalise the output which avoids overstimulation of the following neurons. They are many different types of activation functions that are commonly used with different advantages and disadvantages.

Gradient based learning

Neural networks and therefore ANN use the same gradient descent based learning as most machine learning algorithms. Using a predefined costfunction one computes an error gradient that defines how to change the weights in a layer so as to improve the final output. To update the parameters of the

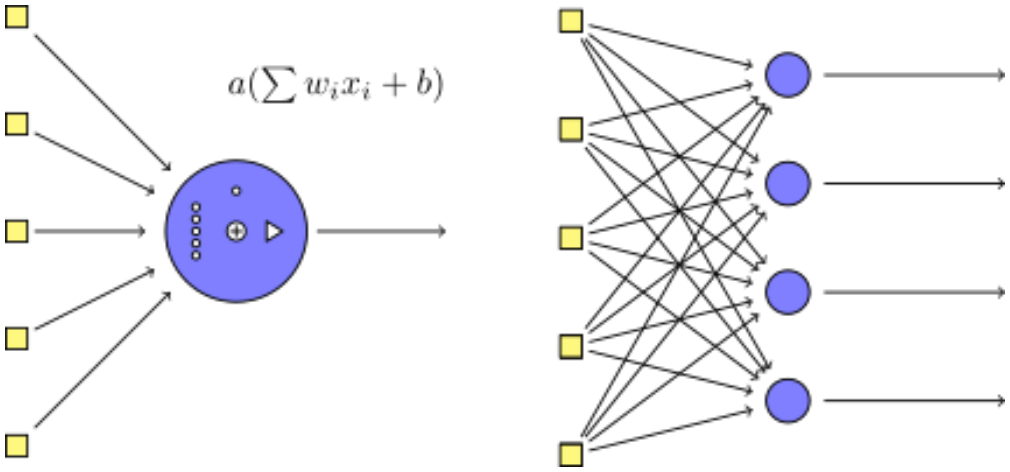


Figure 2.2 – Typical layer of an Artificial Neural Network (ANN): Composed of multiple neurons. Each neuron outputs a weighted sums of its inputs passed through a non-linear activation function. Each layer uses the outputs from the preceding layers as inputs.

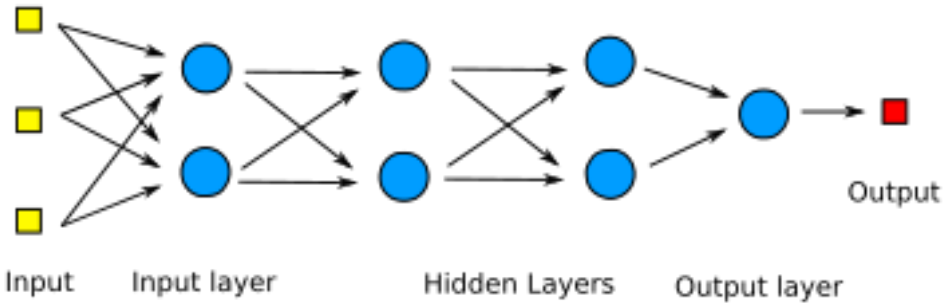


Figure 2.3 – A four layer deep artificial neural network composed of an input layer, two hidden layers and one output layer. Input and hidden layers can be composed of any number of neurons and are fully connected to the preceding one. The output layer scales its size to the desired output size, which in the case of a simple regression is one.

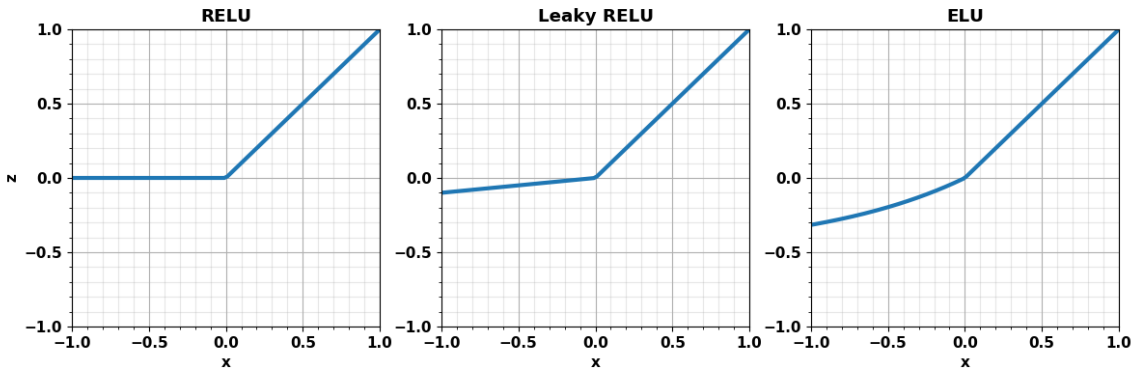


Figure 2.4 – Three commonly used activation functions for hidden layers: ReLU, Leaky ReLU and ELU. ReLUs by removing the possibility of low firing neurons to learn, they introduce greater sparsity into the neural networks which allows for quicker learning. Leaky ReLUs and ELUs are evolved versions of ReLU that avoid its drawback of "killing" too many neurons. All three are efficient against the vanishing gradient problem.

hidden layers, we use the back propagation algorithm (Rumelhart et al., 1986). The backpropagation algorithm calculates the partial derivatives of the cost-function with respect to any weight or bias in the network. Using these partial derivatives, a gradient descent method can be used to change the parameters so as to minimise the costfunction and shift closer to the desired output. The main difference with other machine learning techniques is that the nonlinearity of deep learning models means that most interesting cost functions we might want to use are not convex anymore. As a consequence using gradient descent based learning we will rarely find the global minima (the optimum solution) but only something close to it. This means that training a deep learning network is a more heuristic affair than for classic machine learning. The initial parameters at the start of the training will have a significant impact on the networks capabilities and multiple different training runs will result in different models.

Additionally gradient descent based learning possesses one notable weakness which leads to the vanishing and exploding gradient problems. As the error gradient is back -propagated to deeper layers it can become unstable and can either vanish or increase exponentially (Pascanu et al., 2012; Pascanu et al., 2012). If it vanishes, training will not improve the network further, while if it explodes, the network will probably simply not work. Activation functions are extremely useful to mitigate the vanishing gradient problem. The Rectified Linear Unit activation function (ReLU) (see Fig.2.4) for example forces the gradient to be either 1 or 0.

$$\mathrm{ReLU(x)} = \max(0,\mathrm{x}), \quad \mathrm{ReLU}'(\mathrm{x}) = \begin{Bmatrix} 1 & x > 0 \\ 0 & x < 0 \end{Bmatrix} \tag{2.2}$$

In this way it avoids the vanishing gradient problem entirely and introduces greater sparsity into the network by "killing" low firing neurons(Glorot et al., 2011). This has drawbacks since it can kill too many neurons which is why the leaky ReLU and ELU activation have been developed. By allowing dead neurons to learn just a bit, they avoid the dead ReLU problem and still are efficient against the vanishing gradient problem.

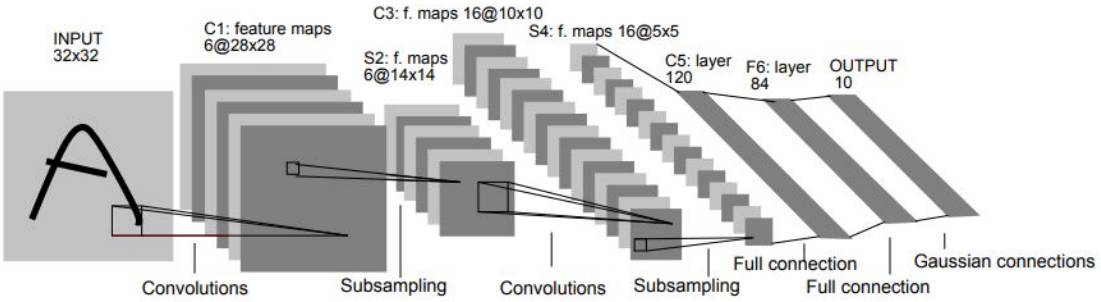


Figure 2.5 – Example of a typical CNN "LeNet5" by LeCun et al. (1998) for handwritten number recognition. It's composed of two convolution layers with each layer followed by a pooling layer reducing the dimension of the output. The network is finished by two fully connected layers that flatten the output from the convolution layers to the desired output of 10. Image Credit LeCun et al. (1998)

Other types of activation function exist but the field is so diverse that I will not go into details. Activation function research is extremely active and changes rapidly. While ELU and ReLU were considered state of the art at the writing of this thesis, this can change in the future.

2.3.2 Convolutional Neural Network

Convolutional neural networks (CNN), first introduced by LeCun et al. (1998), are widely credited for the actual deep learning renaissance. Most commonly used for 2D image processing, they were the first deep learning network truly being used for commercial application, for bank checks and handwriting recognition in the 1990s. The current interest in them however began when they won the annual ImageNet object recognition challenge of 2012 setting them as the new gold standard of image recognition. Since then, similar networks have won continuously other object recognition challenges and almost every research and industry branch has expressed their interest in them.

Convolutional neural networks are a variant of neural networks where the neurons of every layer are not fully connected to the preceding layers but take as input only a few neighbouring neurons. Every neuron of the same layer also shares the same weights and bias. The resulting operation corresponds to a convolution operation, with the neurons acting as a convolution matrix more commonly known as kernel. The output of the layer becomes a feature map which is then given to the next layer. Since every layer corresponds to only one kernel, which is not enough for complex models, CNNs use one more dimension than classic ANNs. Each CNN layer will be composed of multiple neuron layers each with a different kernel looking for different features. A CNN layer for a 2D image will output a feature map for each kernel it has which taken together create a 3D feature map. The concept of depth of a CNN layer should not be confused with the depth of a neural network which is why they are commonly known as features.

There are three reasons why CNNs are so powerful compared to ANNs and other machine learning models: Sparse connectivity, parameter sharing and translational invariance

Compared to ANNs, CNNs neurons are connected only to a few neurons of the preceding layer because the needed convolution kernel can be much smaller than the input. Typically we use 3 by 3 kernels

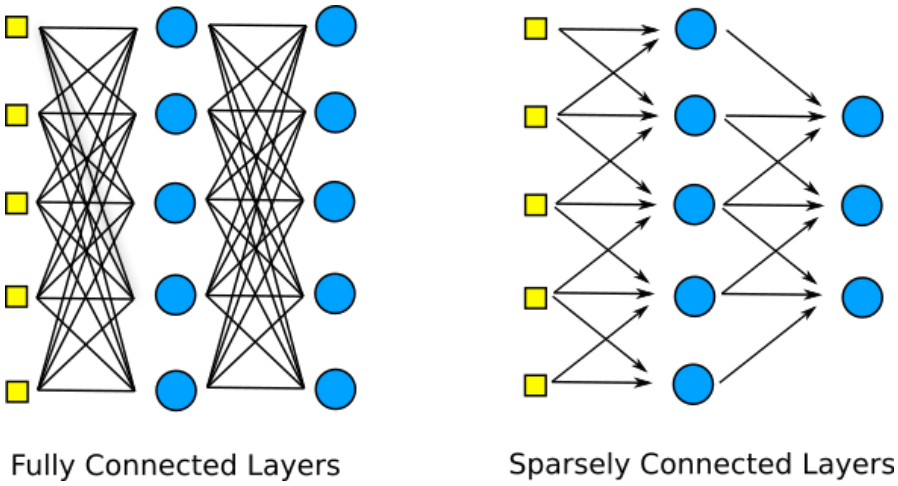


Figure 2.6 – CNNs neurons are connected to only a few preceding neuron through a convolution kernel. The resulting fewer connections make it easier and less computationally costly to train CNNs compared to fully connected networks. Through consecutive layers, deeper neurons see more of the total picture. If the network is deep enough or the final layer is fully connected, the network sees the whole picture without losing parts of it.

meaning every neuron is connected only to 9 other neurons. Supposing we are processing a 6000 by 6000 pixel image, an ANN neuron would have $3,6 \times 10^7$ connections compared to a CNN's 9 connections. These extremely **sparse interactions** make CNN a lot less computation intensive than ANN, allowing deeper and more complex models. While the neurons see only the closest neurons of the preceding layers, because of the depth of the networks and because the last layer is often fully connected, with each layer the deeper neurons see more of the whole picture.

In addition to the sparse interaction as stated before the neurons of each layer share their weights and biases. This **parameter sharing** means that during training the parameters of each feature layer are identical and updated in exactly the same way. This again reduces training computation costs considerably and makes deeper networks possible. It also justifies the terminology of "feature map" because the kernel of each layer becomes specialised in a specific feature detection, like edges for example. Parameter sharing also means that the layer will be **translation equivariant**, which should not to be confused with invariance. A function f is equivariant with respect to a transformation T if $f(T(x)) = T(f(x))$. In other words, applying the transformation to x is equivalent to applying it to the result $f(x)$. In the case of invariance we would have $f(T(x)) = f(x)$,which CNNs achieve an approximation of using pooling layers.

To further reduce the dimension of the CNN, a typical CNN set of layers will use a pooling layer after a convolution layer. The pooling layer replaces the output of a convolution with a summary statistic of the nearby outputs, effectively reducing the dimensions of the output layer. The most commonly used pooling layer, the max pooling layer, keeps only the maximum value of a rectangular 2 by 2 grid, reducing the size of the output layer by four. Others keep only the norm or a weighted average. The pooling layers make the CNN model approximately **invariant to small translation** in the input. This can be important in image detection where one is often more interested in knowing if a feature is present than where it is exactly.

These three features are the main reason why CNNs are still today considered state-of-the-art for visual

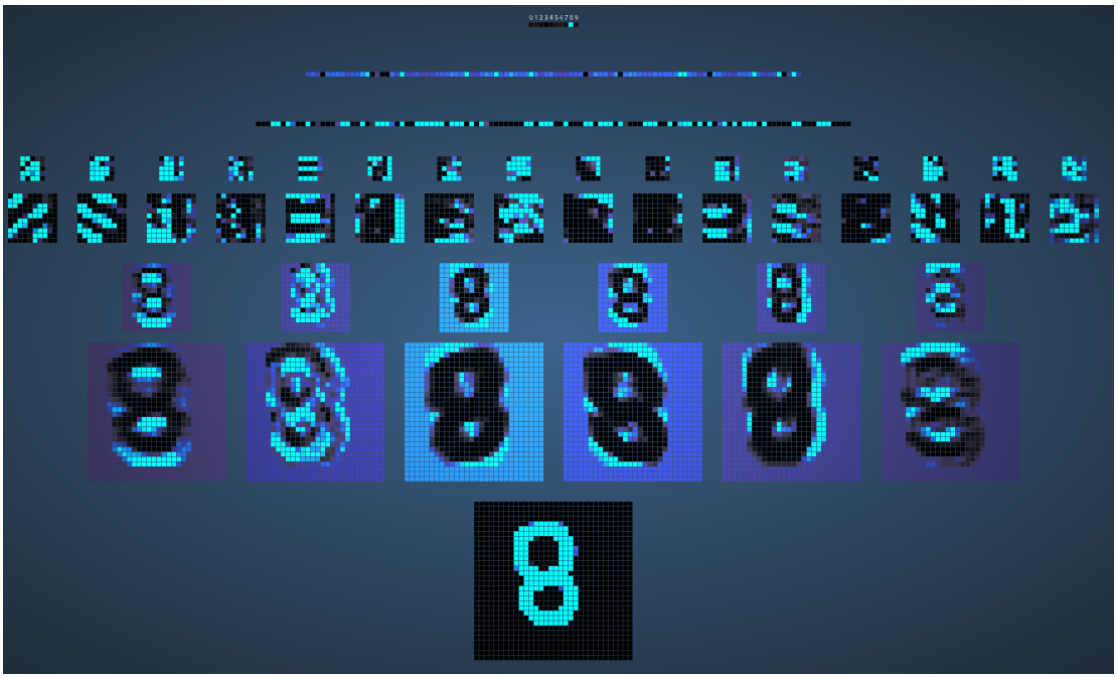


Figure 2.7 – Example of a LeNet 5 style CNN with two convolutional layers, two pooling layers and two fully connected layers classifying a handwritten 8. The closer to light blue a point is, the more the corresponding neuron has been activated. The first convolution layer highlights easy to understand points of interest like the edges of the form 8 and the presence of two circles inside the handwriting. Already with the second convolutional layer (third layer from the bottom) however, the relationship between neurons become too complex for us to keep track of them. This image was created using the interactive tool developed by Harley (2015).

classification tasks. But while they have been applied to many generic classification problems, at the beginning of this thesis they hadn't been applied to lens finding. The next chapter shows therefore how these CNNs can be adapted to the strong gravitational lens detection problem.

3 Euclid Lens Finder

3.1 Preface

Euclid is one of the major space surveys planned for the next decade. Covering 15000 deg^2 of the observable night sky at half the precision of the Hubble Space telescope with one visual and 3 near-infrared bands, it has immense scientific potential especially for strong lensing application. Based on early prediction (Collett, 2015), we expect to find around 200000 lenses and 5000 cluster lenses in the wide survey. The predicted abundance of objects is staggering and necessitates the development of fully automated detection methods for the mission. As part of the Euclid Strong Lensing Science Working Group (SLSWG), we organised and participated the Galaxy Galaxy Strong Lensing challenge (GGSLC) to test ideas for the strong lensing pipeline. The paper below details the CNN version we developed for it.

3.2 Paper

This chapter is presented in the form of a published paper as Christoph Schaefer, Mario Geiger, Thibault Kuntzer, and Jean-Paul Kneib, Deep Convolutional Neural Networks as strong gravitational lens detectors, Astronomy & Astrophysics, 611, A2 (2018).

A&A 611, A2 (2018)
DOI: 10.1051/0004-6361/201731201

Astronomy & Astrophysics

Deep convolutional neural networks as strong gravitational lens detectors

C. Schaefer[1], M. Geiger[1], T. Kuntzer[1], and J.-P. Kneib[1,2]

[1] Institute of Physics, Laboratory of Astrophysics, École Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, 1250 Versoix, Switzerland
e-mail: christophernstrerne.schaefer@epfl.ch
[2] Aix-Marseille Université, CNRS, LAM (Laboratoire d'Astrophysique de Marseille) UMR 7326, 13388 Marseille, France

Received 19 May 2017 / Accepted 22 October 2017

ABSTRACT

Context. Future large-scale surveys with high-resolution imaging will provide us with approximately 10^5 new strong galaxy-scale lenses. These strong-lensing systems will be contained in large data amounts, however, which are beyond the capacity of human experts to visually classify in an unbiased way.
Aims. We present a new strong gravitational lens finder based on convolutional neural networks (CNNs). The method was applied to the strong-lensing challenge organized by the Bologna Lens Factory. It achieved first and third place, respectively, on the space-based data set and the ground-based data set. The goal was to find a fully automated lens finder for ground-based and space-based surveys that minimizes human inspection.
Methods. We compared the results of our CNN architecture and three new variations ("invariant" "views" and "residual") on the simulated data of the challenge. Each method was trained separately five times on 17 000 simulated images, cross-validated using 3000 images, and then applied to a test set with 100 000 images. We used two different metrics for evaluation, the area under the receiver operating characteristic curve (AUC) score, and the recall with no false positive ($\text{Recall}_{0\text{FP}}$).
Results. For ground-based data, our best method achieved an AUC score of 0.977 and a $\text{Recall}_{0\text{FP}}$ of 0.50. For space-based data, our best method achieved an AUC score of 0.940 and a $\text{Recall}_{0\text{FP}}$ of 0.32. Adding dihedral invariance to the CNN architecture diminished the overall score on space-based data, but achieved a higher no-contamination recall. We found that using committees of five CNNs produced the best recall at zero contamination and consistently scored better AUC than a single CNN.
Conclusions. We found that for every variation of our CNN lensfinder, we achieved AUC scores close to 1 within 6%. A deeper network did not outperform simpler CNN models either. This indicates that more complex networks are not needed to model the simulated lenses. To verify this, more realistic lens simulations with more lens-like structures (spiral galaxies or ring galaxies) are needed to compare the performance of deeper and shallower networks.

Key words. gravitational lensing: strong – methods: numerical – methods: data analysis – techniques: image processing – cosmology: observations – dark matter

1. Introduction

Future strong gravitational lense (SL) studies will help further constrain cosmology and galaxy evolution. As of today, galaxy-scale lenses have been used successfully to constrain the Hubble constant by measuring the time-delay of lensed images of quasars independently of other measurement techniques (Bonvin et al. 2016; Suyu et al. 2017). The magnification of lensed source-objects allows observations and studies of background objects at much higher redshifts than are usually visible to telescopes (Kneib et al. 2004; Richard et al. 2011; Atek et al. 2015). Measurement of galaxy-scale SLs can accurately constrain the total mass of the galaxy by probing the dark matter structure. This can be used to estimate the fraction of dark matter in galaxy halos when used in combination with weak-lensing analysis (Gavazzi et al. 2007) or by itself (Jiang & Kochanek 2007; More et al. 2011; Sonnenfeld et al. 2015). It can also be used to constrain the slope of the inner mass density profile (Treu & Koopmans 2002a,b; More et al. 2008; Koopmans et al. 2009; Cao et al. 2016) and the initial stellar mass function (Treu et al. 2010; Ferreras et al. 2010; Leier et al. 2016). One of the largest lens catalogs was produced by the Sloan Lens ACS Survey (SLACS) with about 100 observed lenses (Bolton et al. 2008). These SLs were discovered by selecting lens candidates from the spectroscopic database of the Sloan Digital Sky Survey (SDSS). Lens candidates were chosen by identifying the spectroscopic signature of two galaxies in the spectra, one galaxy at a greater distance than the other. These candidates were then verified by follow-up observation using the *Hubble* Space Telescope.

Historically, SLs were found serendipitously by human inspection of data. However, a systematic search by experts is too time-consuming to be a practical proposition for future large-scale surveys unless it were to involve citizen scientists. For example, the number of new lens systems from the *Euclid* mission (Laureijs et al. 2011) and from the Large Synoptic Survey Telescope (LSST Science Collaboration et al. 2009) survey is expected to reach at least 10^5 SLs among 10^9 objects (LSST: Oguri & Marshall 2010; HST: Pawase et al. 2014; *Euclid*: Collett 2015). Similarly, the amount of SLs found by the

SKA survey is expected to be on the same order of magnitude (McKean et al. 2015). Efficient automated gravitational lens-finding techniques are urgently needed.

The Spacewarps project[1] was an attempt to use and train non-experts at lens classification. Through an interactive website, amateur scientists were trained to sort through data from CFHTLS (Marshall et al. 2015). They found 29 promising new lens candidates in the survey (More et al. 2016), but this method will likely be too slow and too much subject to human error for future data sets. Semi-automated methods like arc detectors using clustering techniques have been used with some success (Lenzen et al. 2004; Cabanac et al. 2007) and have been further improved. Joseph et al. (2014) and Paraficz et al. (2016) added machine-learning to these techniques, using a Principal Component Analysis (PCA) based approach to remove the foreground galaxy from the image and facilitate the detection of arcs. Recently, Petrillo et al. (2017) and Jacobs et al. (2017) started using convolutional neural networks (CNNs) for lens detection. CNNs belong to a class of efficient image-classifier techniques that have revolutionized image processing (Lecun et al. 1998). In astrophysics, they have been applied successfully to galaxy morphology (Huertas-Company 2015), redshift estimation (Hoyle 2016), and spectra analysis (Hála 2014).

The *Euclid* Strong-Lensing working group, in collaboration with the Bologna lens factory[2], has started the Galaxy-Galaxy Strong-Lensing challenge[3] (GGSLC: Metcalf et al., in prep.) in light of future large-scale imaging surveys such as the *Euclid* mission. The goal was to determine the best technique for finding gravitational lenses for both ground-based and space-based imaging.

Our goal was to explore and optimize CNN architectures for lens classification. We successfully applied it to the GGSLC and were awared first and third place in the two categories of the GGSLC. In this paper, we present the CNN lens finder in detail that we created for the GGSLC challenge and discuss the advantages and disadvantages of CNN lens classifiers when applied on simulated and real data. The paper is organized as follows. Section 2 gives a brief overview of artificial neural networks (ANN) and CNNs and their usage in image processing. Section 3 outlines the details of our algorithm implementation and the two winning CNN architectures of the challenge, while in Sect. 4 we present some interesting alternative architectures. Section 5 summarizes the results of the different architectures we applied to GGSLC data, and we discuss them.

2. Theory

2.1. Artificial neural network

Artificial neural networks are machine-learning techniques inspired by the study of the human brain (Hebbian learning: Hebb 1950). ANNs are capable of learning classification or regression tasks in N dimensions by training using a set of labeled examples. This makes them easily applicable to complex problems for which explicitly programmed solutions or mathematical models are difficult to write. The main drawback of ANNs is the computation cost of the training procedure. More modern training techniques coupled with advances in GPU processing power made ANNs versatile and capable of being applied to almost any data set. They are created by stacking layers of neurons together.

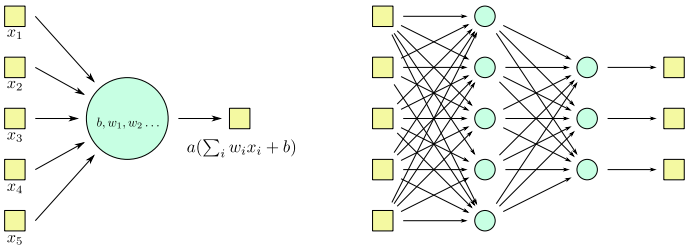


Fig. 1. *Left*: structure of a neuronal unit. Each neuron implements a linear combination (using weights w_i and a bias b) of its input $\boldsymbol{x}$ followed by a nonlinear activation function $a(\boldsymbol{x})$. *Right*: ANN structure. Neurons in the same layer all receive the same input. The stacking of layers allows the ANN to define a model parametrized by the weight variables of the network.

Each neuron implements a linear combination (using weights w_i and a bias b) of its input $\boldsymbol{x}$ followed by a nonlinear activation function $a(\boldsymbol{x})$,

$$y(\boldsymbol{x}) = a\left(\sum_{i=1}^{N} w_i x_i + b\right), \tag{1}$$

where N is the dimension of the inputs.

A layer consists of multiple neurons applied to the same input. Each output is passed as an input to the next layer. This cascade of nonlinear combinations of inputs ends at the output layer (see Fig. 1). In a classical ANN, all possible connections are established and exploited, in short, it is fully connected. A neuron in a given layer will transmit its outputs to all neurons in the next layer. Every layer between the input and the output layer is called a hidden layer. The initial input layer is sometimes also called a front layer. An ANN model is parametrized by the weights w and biases b of the neurons.

These weights and biases are trained iteratively. ANNs make predictions when presented with a training input. As the model parameters are randomly initialized, the first predictions are very different from the ground truth of the input.

The ANN then evaluates the error according to some predefined cost function and computes appropriate corrections to the parameters. These prediction errors are propagated backward through the layers, from the output to the front layer, and induce parameter updates. The technique is known as back-propagation (Rumelhart et al. 1986) and is commonly built on gradient descent for the computation of the parameter updates.

2.2. Convolutional neural network

Deep ANNs, models that have more than one or two hidden layers, perform better than shallow networks. The mathematical evidence for this statement is still scarce, but it is empirically observed. The continued growth in computation power made ANNs interesting for scientific application. However, computational cost of training increases with depth, and limitations in gradient-based procedures are challenging performance obstacles. Training with gradient methods generates a so-called vanishing-gradient problem, first identified by Hochreiter (1991). The magnitude of the gradient diminishes as it is back-propagated through the layers. The typical result is that layers close to the front layer effectively stop learning. While still affected by the vanishing-gradient problems, CNNs limit its effect by reducing the number of connections and sharing weights. This mitigation motivated the development of CNNs and their

[1] https://spacewarps.org/
[2] https://bolognalensfactory.wordpress.com/
[3] metcalf1.bo.astro.it/blf-portal/gg_challenge.html

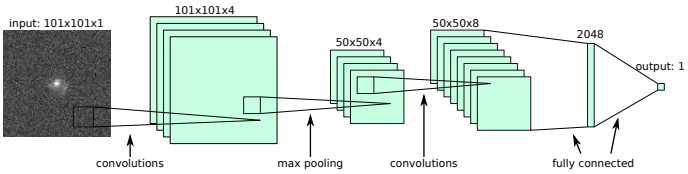


Fig. 2. Example of a CNN architecture: The input image undergoes a series of convolution layers into a series of feature maps. The first convolution transforms the 101×101 pixel image into four 101×101 pixel feature maps. To lower computation cost, max-pooling layers are used in between convolutions. They reduce the dimensionality of the image, dividing the size of the image by two. A fully connected layer then combines all feature maps for the classification.

subsequent application to image recognition (Lenet-5 model, Lecun et al. 1998).

The breakthrough for CNN came when Krizhevsky et al. (2012) created an architecture that won the 2012 ImageNet Large-Scale Visual Recognition Challenge[4]. His submission achieved a classification error of only 15.3% compared to the second-best submission with 26.2% obtained by a method not based on a neural network. CNNs have been used extensively in image processing ever since.

Our CNNs (Fig. 2) are created by stacking the following layers: convolution layers convolve the input image by a number of small kernels (or features maps, typically of dimension 3×3 to 7×7). The parameters to be optimized during training are the individual kernels. These weights are shared by all neurons in the layers (the kernels are the same for the whole layer). Pooling layers reduce the dimensionality of the input to decrease the number of parameters and avoid overfitting. The most common pooling technique is the max-pooling method. It partitions the input image into non-overlapping quadrants and yields the maximum value in the quadrant. Fully connected (fc) layers are the classic ANN neuron layer. Every input is connected to every neuron of the layers. They are used as the final CNN layers to merge the information contained in the feature maps into the desired output form. Dropout layers are only active during training. They randomly sever half the connections between the two layers they separate (Hinton et al. 2012). This is done to reduce coadaption of the neurons (learning the same features) and reduce overfitting. Batch normalization layers normalize and shift the output along a small input sample $B = \{x_{1...m}\}$ following the equation

$$y_i = \gamma \frac{x_i - \mu_B}{\sigma_B^2} + \beta, \tag{2}$$

where μ_B and σ_B are the mean and the variance over B. γ and β are two model parameters of the layer. Batch normalization is used to increase the training speed of the CNN (Ioffe & Szegedy 2015).

Convolution layers take advantage of the local spatial correlation in the data. Stacking multiple convolution layers implies a global treatment of the signal, making the network shift-invariant (i.e., features will be detected independently of their position). This make CNNs especially effective when treating images (Mallat 2016).

2.3. Data set of the Galaxy-Galaxy Strong-Lensing Challenge

The data for the GGSLC was provided by the Bologna lens factory challenge. The Bologna lens factory project is a complex lens-simulation project. It is based on the Millennium simulation (Lemson & Virgo Consortium 2006), with modeling of the gravitational lensing effect using the `Glamer` ray-tracing tool (Metcalf & Petkova 2014) and with `MOKA` to create the multi-plane dark halos and their substructures (Giocoli et al. 2012). The models and the parameters used to generate the simulations were blinded for the duration of the challenge.

Each image of the SL challenge was a 101×101 pixel stamp centered around an object. Participants had to submit a confidence value $p \in [0, 1]$ for each image. An object with a high confidence value was interpreted as a lens. Two categories of data were proposed with separate data sets, each with 20 000 training and 100 000 test images: (i) a space-based data set that consisted of images in a single visible band (simulating exposures of the *Euclid* instrument VIS); and (ii) a ground-based data set with images taken in four bands (U, G, R, and I) with a lower singal-to-noise ratio (S/N) and random masking of pixels, mimicking noisy data.

The ratio of lenses to non-lenses in the simulated data was much higher than in reality, around one-to-one, as an imbalance of examples (called skewed classes) can lead to biases. The results are have been made public, and a detailed discussion of the simulations and results will be provided in Metcalf et al. (in prep.). Our baseline architecture submission to GGSLC ranked first in the space-based data category and third in the ground-based category (Fig. 9). CNNs in general dominated the challenge. CNN-based methods filled the seven best submission in both categories.

3. CNN lensfinder: architectures

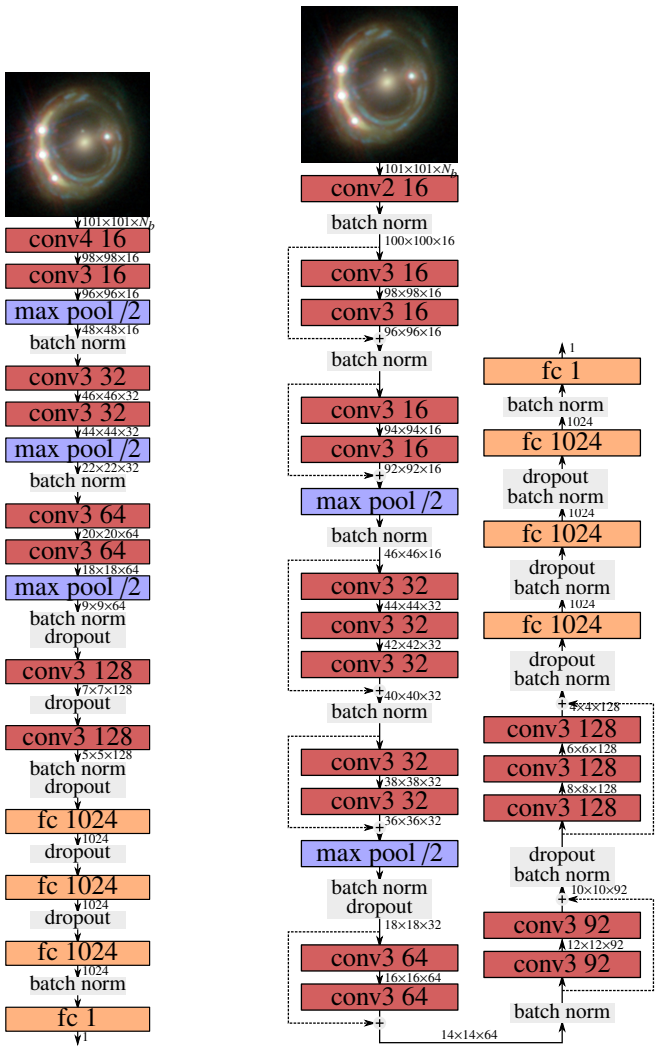
For this paper, four different types of CNN architectures were applied to the training data of the GGSLC: a simple CNN architecture that forms the baseline comparison for the paper, a so-called residual architecture based on the paper by He et al. (2015), and two further architectures with additional invariant properties. The final version of each architecture was selected after a heuristic study of the parameter space.

3.1. Baseline architecture

The baseline architecture as shown in Fig. 3a was inspired from typical CNN architectures that performed well in the ImageNet competition (Simonyan & Zisserman 2014). It is organized by stacking convolution blocks. This simple baseline architecture achieved first place in the space component of GGSLC. A convolution block is the superposition of 2 convolutional layers followed by a pooling layer to reduce the dimensionality of the image and a batch-normalization layer. The baseline architecture is comprised of 8 convolutional layers, organized into 3 convolution blocks and 2 stand-alone layers, and 3 fully connected layers at the top. There is thus a total of 11 layers. With the exception of the initial layer, every convolution layer uses 3×3 convolution kernels for efficiency reasons (Simonyan & Zisserman 2014). The first convolution layer uses a 4×4 kernel to yield an even number of pixels for easier manipulation. At each convolution block, the number of features was doubled, resulting in 256 features in the last block. The fully connected layers used either 1024 or 2048 features.

For each layer, we chose a modified version of the rectifier linear unit (ReLU) activation function because of its sparse representation capability (Glorot et al. 2011; Arpit et al. 2016). The

[4] http://image-net.org/



(a) Baseline **(b)** Residual

Fig. 3. Visualization of the baseline and residual architecture for the CNN lensfinder: the convolution blocks (red) indicate the size of the kernel and the number of features. The fully connected blocks (yellow) indicate the number of features. The arrows indicate the flow of the data, and between the blocks, we show the dimensionality of the input ($N_{\rm pixel} \times N_{\rm pixel} \times N_{\rm features}$). The last fully connected layer yields a confidence value of the object being a lens. The initial layer has $N_{\rm b}$ features, either one or four, depending on the category of the data (space and ground, respectively). Batch normalization and dropout layers are indicated as gray blocs.

activation is given by

$$f(x) = \frac{1}{\sqrt{\pi - 1}} \left(\sqrt{2\pi} \max(0, x) - 1 \right). \quad (3)$$

Inputs of the networks have dimension of $101 \times 101 \times N_{\rm b}$, where $N_{\rm b}$ is the number of bands ($N_{\rm b} = 1$ for space and $N_{\rm b} = 4$ for ground). The wavelength-dependent information (in the third dimension) is handled naturally by extending the kernel dimension from two to three (spatial to spatial plus wavelengths).

3.2. Residual architecture

A common way for improving CNN is to increase the depth, that is, the number of convolutional layers. With creating increasingly deeper CNNs comes the vanishing-gradient problem detailed before. At some point in the training process, the accuracy starts to saturate and degrade, generating an upper limit to the possible depth of CNNs. To compensate for this, He et al. (2015) introduced residual learning. In the GGSLC challenge, Francois Lanusse's deep lens classifier (Lanusse et al. 2018) used residual learning to create a 46-layer deep CNN that won the ground part of the challenge. We adapted our residual architecture to analyze the advantages and disadvantages compared to the baseline CNN.

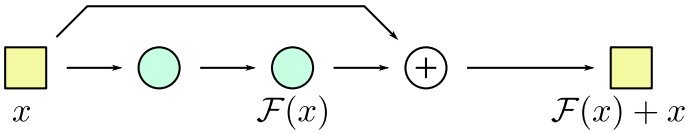


Fig. 4. Structure of a residual block: The feature maps $F(\boldsymbol{x})$ from two stacked convolutional layers are added to input $\boldsymbol{x}$. Each green circle represents a convolutional layer.

In a classical convolution layer, the feature map is created from scratch, that is, it learns an unreferenced mapping. The end-goal of the training process is to find parameters that minimize the cost function. We denote by $H(\boldsymbol{x})$ the optimum feature map and by $F(\boldsymbol{x})$ the map currently held in the parameters. In other words, the training updates $F(\boldsymbol{x})$ until

$$H(\boldsymbol{x}) = F(\boldsymbol{x}). \quad (4)$$

In contrast, residual networks train by optimizing a residual mapping $\boldsymbol{x}$, or the difference between the ideal and the real feature map. He et al. (2015) stated that it is easier to optimize the residual feature map than the unreferenced map,

$$H(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x}, \quad (5)$$

where $\boldsymbol{x}$ is the identity mapping obtained by using shortcut connections skipping the convolution layers (Fig. 4). Our residual architecture as shown in Fig. 3b is 20-layer deep with 3 small residual blocks, 4 large residual blocks, and 3 fully connected layers with 1024 features. The small residual block is composed of 2 convolutions and 1 shortcut, keeping the same number of features. The large residual block is composed of 3 convolutions and 1 shortcut followed by a convolution layer, doubling the number of features.

3.3. Implementation details

Other than the differences in the approach to the problem, the networks shared a number of implementation details that we outline here.

- Cost function: we chose the binary cross-entropy cost function as the cost function C driving the training,

$$C = -\frac{1}{N} \sum \left\{ y \ln(y_{\rm p}) + (1 - y) \left[(1 - \ln(y_{\rm p}) \right] \right\}, \quad (6)$$

  where N is the number of training examples, y is the ground truth, and $y_{\rm p}$ is the classification prediction.
- Data augmentation: to increase the number of training samples, we used data-augmentation techniques. The goal is to generate more examples out of the original training set by exploiting physically invariant transformations, for example, rotating the image by 90 degrees. The benefit of increasing the training set size is to reduce overfitting.Taking advantage of the dihedral group symmetry (Fig. 6) of the lens problem,

the training sets were augmented using 90-degree rotations and flipping operations. We did not use rotation angles different than 90 degrees to avoid having to interpolate in pixel space.

- Training: the challenge training set was subdivided into a training set (of 17 000 images) that was used by the networks to learn and a validation set (3000 images strong) to check the performances on an independent set. The performance was monitored every 1000 steps by evaluating predictions made on the validation set. At each training step, we randomly selected batches of 30 images (15 lenses and 15 non-lenses) and ran the learning procedure for ∼250–300 epochs using the `ADAM` minimization algorithm (Kingma & Ba 2015). We trained five networks with the same architecture and selected the best-performing individual.
- Library: the models were implemented using the `Tensorflow` library (Abadi et al. 2015) on a GeForce GTX 1060 graphic card. The training time took approximately 1 h/100 epochs for the baseline model and 2 h/100 epochs for the residual model. The final prediction of the classification for the challenge on the 100 000 test images took approximately 20 min.

3.4. Image invariance

The idea behind these two next architectures was to deal with the inability of most lens finders to recognize and handle the invariant features of gravitational lenses. CNNs are, by design, already invariant to translation, but not to rotation, scaling, and flipping. The pretraining data augmentation phase renders them more robust to these symmetry operations, but not invariant. By modifying the CNN architecture so as to be invariant or more robust to different types of symmetries, we expect to reduce identification errors. The following sections describe how we increased the invariance of our models.

3.4.1. Views architecture

Several models trained to accomplish the same task form a committee. Predictions of a committee typically result in some sort of weighted combination of its members' predictions. They have been used to improve classification results for example on the MNIST[5] problem (Ciresan et al. 2011) or to detect anomalies in the predictions (Nguyen et al. 2014).

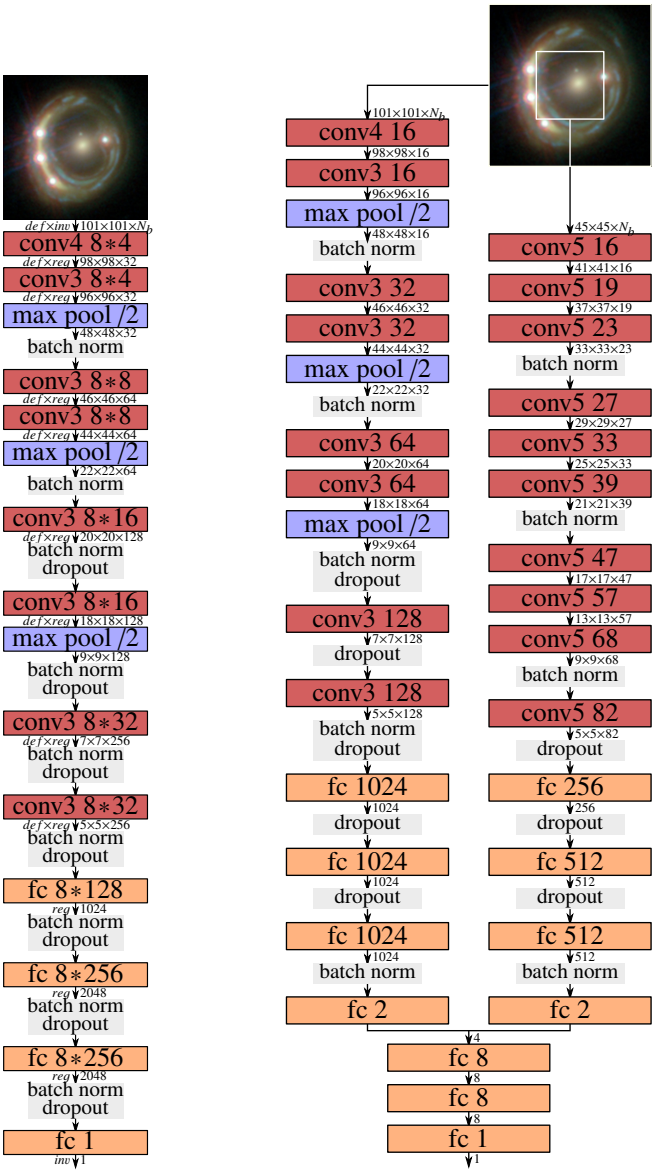
The views architecture (Fig. 5b) trains two neural networks separately to look for lenses of different sizes. The first network looks at the whole image, detecting large lenses spanning the whole image. The second uses only the central part of the image. By combining the prediction of the two networks, smaller lenses should be detected while not neglecting the detection of the larger lenses. In other words, the first network takes as input the whole image, like the baseline model, while the second only accepts a smaller stamp of 45×45 pixels. To simplify the smaller network, we used only 5 × 5 convolution layers and fewer features at each layer.

3.4.2. Invariant architecture

The invariant architecture adds additional invariant properties to the model. While relatively untested, this has been used with success for a galaxy morphology classifier on Galaxy Zoo data (Dieleman et al. 2015, 2016). The invariant architecture takes advantage of the dihedral symmetry of the lens-finding problem (Fig. 6) by using dihedral equivariant convolutional layers that we refer to as Dec layers.

At the level of the input layer, eight operations of the same convolution kernel, transformed by a different transformation of the dihedral group, are applied to the input image. The output is divided into eight different output channels (see Fig. 7),

$$y_i = \mathrm{Conv}(x, F_i) \quad i \in \{0, \ldots, 7\}, \tag{7}$$



(a) Invariant **(b)** Views

Fig. 5. Visualization of the invariant and views architecture for the CNN lensfinder: the convolution blocks (red) indicate the size of the kernel and the number of features. The fully connected blocks (yellow) indicate the number of features. The arrows indicate the flow of the data, and between the blocks, we show the dimensionality of the input ($N_{\mathrm{pixel}} \times N_{\mathrm{pixel}} \times N_{\mathrm{features}}$). The last fully connected layer yields a confidence value of the object being a lens. The initial layer has N_{b} features, either one or four, depending on the category of the data (space and ground, respectively). Batch normalization and dropout layers are indicated as gray blocs.

[5] MNIST database of handwritten digits, `http://yann.lecun.com/exdb/mnist/`

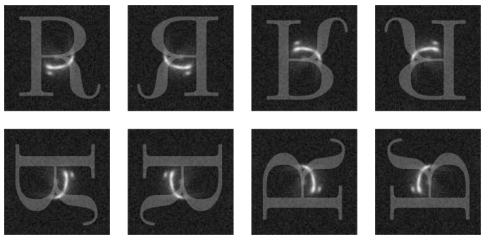


Fig. 6. Representation of the dihedral symmetry group: An optimal lens finder should be invariant to the operations of this group (i.e., flipping and rotation).

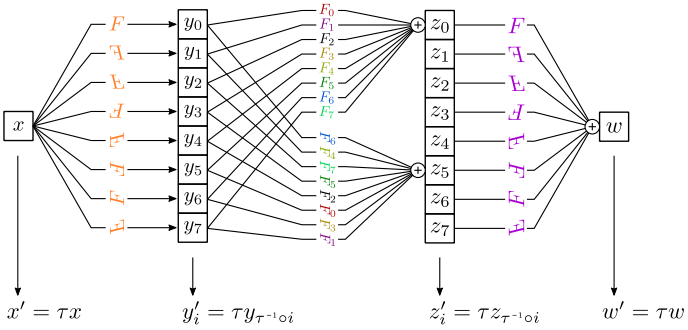


Fig. 7. Dihedral equivariant architecture: Kernels with identical colors but different orientation are identical kernels to which a different dihedral operations has been applied. Phase 1: seperation into eight channels, one for every input channel and member of the dihedral group. Phase 2: convolution of the eight channels with eight separate kernels. Each output channel from a Dec layer is the sum of all the input channels convolved by all feature kernels of the layer transformed by one of the dihedral operations. Phase 3: the eight channels are summed, giving a dihedral invariant result.

where i is one of the eight specific dihedral transformation and M_i is the filter to which a dihedral transformation has been applied.

Compared to the baseline version, for the Dec layer there are eight different convolution kernel instead of one: one kernel for each transformation of the dihedral group (F_i, $i \in \{0, \ldots, 7\}$). Each kernel is initialized and trained separately from each other. Each output channel in a Dec layer is the sum of all the input channels convolved by all the different feature kernels of the layer transformed by one of the dihedral operations (Fig. 7). The result of the eight channels, y_j, is a dihedral invariant quantity,

$$y_j = \sum_{i=0}^{7} \mathrm{Conv}(x_i, jF_{j^{-1}\circ i}) \quad j \in \{0, \ldots, 7\}. \tag{8}$$

The two layers illustrated in Fig. 7 have the property of being invariant with respect to the dihedral group. Our invariant architecture is shown in Fig. 5a and follows the same fundamental scheme as the baseline architecture. Since using eight channels increases computation time and makes the model more prone to overfitting, the number of features of the convolutional layers is divided by four. The invariance was tested by checking that rotated and flipped versions of the same image are attributed the same score by the classifier.

4. Results

In this section we describe the results of the different architectures applied to the GGSLC data.

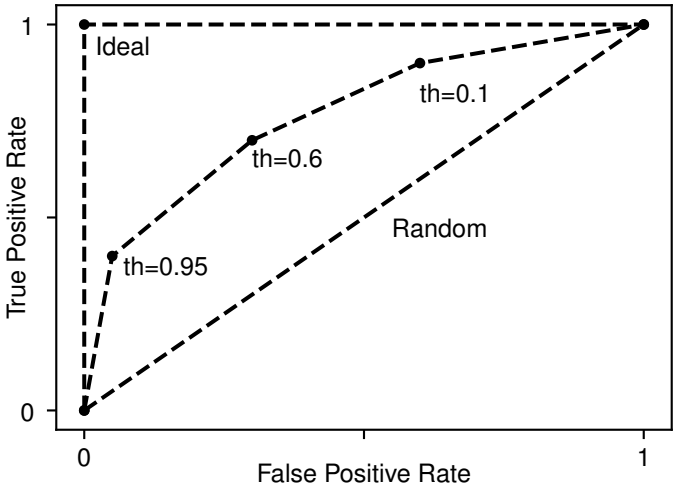


Fig. 8. Receiver operating characteristic (ROC) curve: The GGSLC ranked the classifiers as a function of the area under the ROC curve (AUC). For a perfect classifier, the score is 1, and for a random classifier, it is 0.5.

4.1. Performance metric

We first start by a brief overview of the performance metrics we used to quantify the performance of the lens classification.

- The true-positive rate (TPR) measures how well the classifier detects lenses from the whole population of objects,

$$\mathrm{TPR} = \frac{N_{\text{True positives}}}{N_{\text{True psitives}} + N_{\text{False negatives}}}. \tag{9}$$

  This metric is also known as recall. The best algorithms have a TPR close to 1.
- The false-positive rate (FPR) measures the contamination of the positive detections by false positives,

$$\mathrm{FPR} = \frac{N_{\text{False positives}}}{N_{\text{True negatives}} + N_{\text{False positives}}}. \tag{10}$$

  The best algorithms have an FPR close to 0.
- The receiver operating characteristic (ROC) is a visual representation of the TPR and FPR. Since they depend on the threshold $t \in (0, 1)$ defined to distinguish objects as lenses or non-lenses, the ROC curve (Fig. 8) is created by plotting TPR(t) as a function of FPR(t) for $t \in (0, 1)$. The challenge ranked the classifiers as a function of the area under the ROC curve (AUC), which is the integral of the ROC curve between an FPR of 0 and 1. A perfect classifier would score 1, while a randomly predicting classifier would score 0.5.

4.2. Training, submission, and results

After the challenge deadline, we tested our four architectures on the GGSLC data. As for the baseline architecture, we used our 17 000-image training set and the 3000-image validation set. Each architecture was trained five separate times. In Table 1 we show the result of this committee training. The performance is also evaluated in Table 3 on the challenge test data as the ground-truth values were released to participants after the submission deadline. The standard deviation of the five runs is also given. The two metrics used to evaluate the performance of the

Table 1. Training results: rach architecture was run five separate times.

Space	Training	Validation
baseline	0.9920 ± 0.0020	0.9764 ± 0.0017
views	0.9898 ± 0.0030	0.9753 ± 0.0021
residual	0.9958 ± 0.0028	0.9765 ± 0.0024
invariant	0.9997 ± 0.0003	0.9719 ± 0.0016
Ground	Training	Validation
baseline	0.9953 ± 0.0090	0.9905 ± 0.0082
views	0.9980 ± 0.0010	0.9924 ± 0.0006
residual	0.9990 ± 0.0023	0.9932 ± 0.0027
invariant	$0.9999 \pm 3 \times 10^{-6}$	0.9880 ± 0.0030

Notes. The training and validation AUC scores are the mean of these runs. The error is the standard deviation.

Table 2. Confusion matrix (baseline architecture, GGSLC challenge) for TPR0.

Space-based	Classified as non-lens	Classified as lens
Non-lens	59742	40
lens	20957	19264
Ground-based	Classified as non-lens	Classified as lens
Non-lens	50042	17
lens	21754	28194

Notes. The TPR0 threshold was chosen by the GGSLC organizers for no false-positive in the first 10 000 images of the test set.

Table 3. Test, $\text{Recall}_{0\text{FP}}$, and $\text{Recall}_{1\text{FP}}$ results.

Space	Test AUC	$\text{Recall}_{0\text{FP}}$	$\text{Recall}_{1\text{FP}}$
baseline	0.9322 ± 0.0016	0.01 ± 0.02	0.04 ± 0.04
committee b.	0.9326	0.01	0.01
views	0.9324 ± 0.0013	0.26 ± 0.06	0.28 ± 0.07
committee v.	0.9343	0.30	0.32
residual	0.9322 ± 0.0006	0.23 ± 0.04	0.29 ± 0.03
committee r.	0.9346	0.29	0.30
invariant	0.9332 ± 0.0006	0.27 ± 0.04	0.28 ± 0.05
committee i.	0.9399	0.32	0.33
Ground	Test AUC	$\text{Recall}_{0\text{FP}}$	$\text{Recall}_{1\text{FP}}$
baseline	0.9761 ± 0.0011	0.44 ± 0.13	0.49 ± 0.08
committee b.	0.9773	0.50	0.55
views	0.9746 ± 0.0011	0.35 ± 0.19	0.43 ± 0.17
committee v.	0.9759	0.35	0.39
residual	0.9775 ± 0.0006	0.44 ± 0.06	0.46 ± 0.07
committee r.	0.9795	0.50	0.55
invariant	0.9774 ± 0.002	0.39 ± 0.11	0.45 ± 0.05
committee i.	0.9813	0.49	0.49

Notes. Each architecture was run five times. The test scores are the mean of these runs.

methods are the (i) the AUC and (ii) the zero false-positives, $\text{Recall}_{0\text{FP}}$ (that is, the fraction of lenses recovered with zero false-positives).

The AUC results are much better for ground-based data than for space-based data (Fig. 9) although the images have a lower S/N than the space-based images. This increased performance could be due to the increased amount of information in the form of the four bands, instead of the single VIS-like band for space. The lower S/N in the ground-based data does not seem to hinder prediction.

The baseline, views, invariant and residual architectures achieved equally good AUC results on the validation set and the test set within the standard deviation of the runs. This is surprising because deeper networks, like the residual one, are expected to perform better than shallower models. The scores are too close to the optimum to confidently distinguish between the architectures. The most likely explanation is that the simulated data were too simple for the CNN lensfinder. The simulations did not include spiral galaxies or some other ring-like objects capable of confusing gravitational lens classifiers. A more complex method was therefore not needed to classify the data correctly. The difference that can be seen between the validation scores and the test scores in Tables 1 and 3 can be attributed to a slight overfitting. This is probably due to the small size of the validation set we used in comparison to the test set.

The invariant architecture has a lower validation AUC score than the others, but performs equally well on the test set. This may indicate that the invariant architecture generalizes the lens model better than other architectures. This could be due to the imposed invariant properties, as we have given the model some additional knowledge. This has no effect on the final test score but could become important when applying the CNN lensfinder to real data. Since the amount of known galaxy-scale lenses is small, a sufficiently large training set for a CNN lensfinder can only be obtained by simulated lenses (see Petrillo et al. 2017, for a CNN lensfinder applied to CHFTLS data). The caveat here is that CNNs trained on simulation might miss lenses because the simulated training set was unrealistic. The better the CNNs generalize the lens model, the lower the chance that they will missidentify objects. Ideas exist to force CNNs to focus on the lens model. One is to use multiple different simulations to create lenses (Jacobs et al. 2017). Adding dihedral invariance to CNNs could be another way of doing this.

The ground-based results are extremely encouraging, especially because of the purity of the score. In a classification problem with a 1-to-1000 ratio between lenses and non-lenses, algorithms with even a very small contamination can be dominated by false positives. The baseline model performs well on the metric $\text{Recall}_{0\text{FP}} = 0.44$ in the ground-based test set (Table 3). In a more realistic setting with a ratio of lenses to non-lens objects, we would have found 22 out of the 50 lenses in a test set containing 100 000 images without any false positives.

Table 3 shows that the standard deviation of $\text{Recall}_{0\text{FP}}$ is large. Using the CNN that performed best on the validation set does not guarantee the best $\text{Recall}_{0\text{FP}}$ or even best AUC score (Figs. 10 and 11). The metrics vary depending on the individual result of the training run. To mitigate this, we grouped the five training runs of our model in a committee of CNNs. The committee output is taken as the average of their prediction. By compensating for each other's shortcomings, committees stabilize the results and achieve a better-than-average result for the AUC metric as well as for the $\text{Recall}_{0\text{FP}}$ metric (Table 3, Figs. 12 and 13). The invariant model especially is improved by this and obtains the best scores for the space-based data.

5. Conclusions

We presented a strong gravitational lens finder based on convolutional neural networks (CNN). The method showed strong

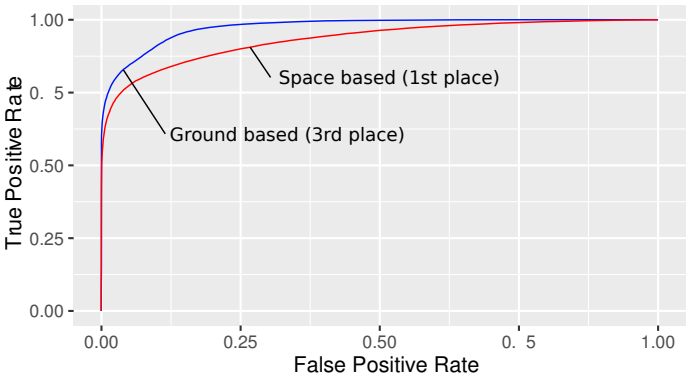


Fig. 9. ROC curve of our baseline architecture submission to the GGSLC challenge. The solid line is the curve from our submission. Blue is the ground-based data category, red is the space-based data category.

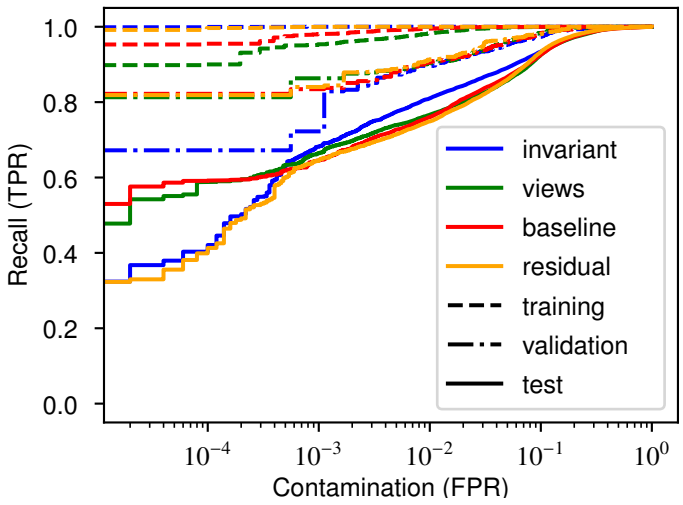


Fig. 10. Logaritmic ROC curves on ground-based data. Training (dotted line), validation (half-dotted line) and test (solid line) score of all four architectures. Data come from the best of five runs in terms of validation set score.

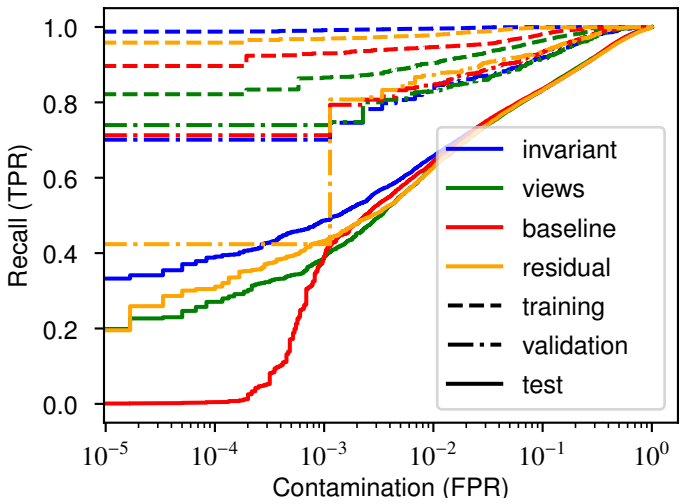


Fig. 11. Logaritmic ROC curves on space-based data. Training (dotted line), validation (half-dotted line) and test (solid line) score of all four architectures. Data come from the best of five runs in terms of validation set score.

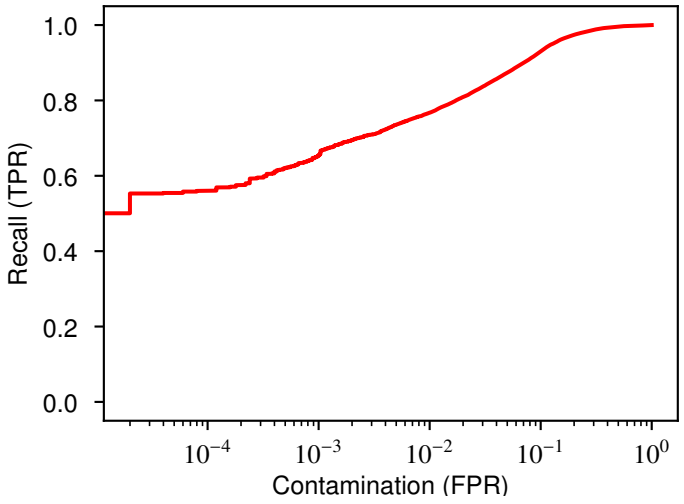


Fig. 12. Logaritmic ROC curve of the baseline committee on ground-based data. The curve is the result of the baseline committee (five baseline CNNs taken together). The shaded areas represent the minimum and maximum values from the five stand-alone baseline CNNs.

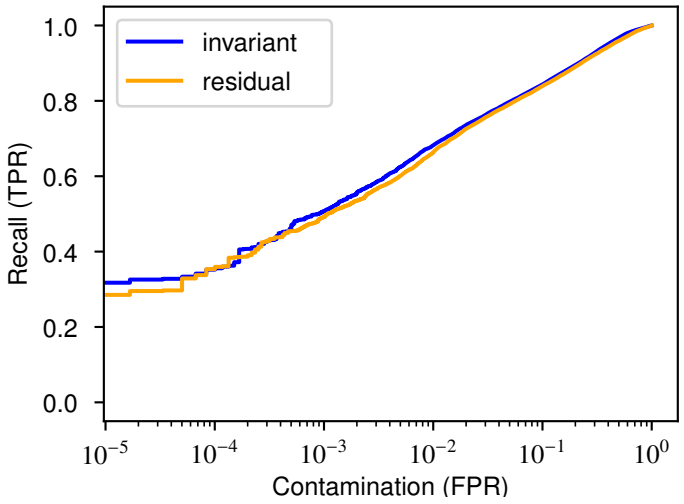


Fig. 13. Logaritmic ROC curve of the invariant and residual committee on space-based data. The curve is the result of the committee (five invariant or residual CNNs taken together). The shaded areas represent the minimum and maximum values from the five stand-alone invariant or residual CNNs.

performances on simulated data. It won the first place and third place in the Strong Gravitational Lens Challenge (GGSLC), respectively, in the space-based and ground-based data category. We have also presented three other variations of that lensfinder, among which, a residual CNN based on the recent architecture developed by He et al. (2015).

We found that CNNs perform better on ground-based data than on space-based data despite the lower S/N. This is probably due to the additional bands, which add information, but this still has to be confirmed. This can be done, for instance, by limiting the ground-based data to one band and comparing to the other results. All four CNNs achieved almost perfect ROC curve scores on the simulated data, with the highest area under the ROC curve (AUC) score up to 0.9775 for ground-based and 0.933 for space-based data. They also achieved a recall with a zero false-positive ($\text{Recall}_{0\text{FP}}$) of 50% for ground-based and of 32% for space-based data. We showed that the best $\text{Recall}_{0\text{FP}}$ results were achieved

by committees of CNNs instead of single CNNs. Committees of CNNs consistently scored the best AUC scores. We also observed that adding rotation invariance to CNNs grouped together in committees produces the best space-based $\text{Recall}_{0\text{FP}}$ score.

Because all results are almost equally good, more conclusions about the best CNN model cannot be drawn. Most likely the simulations did not include enough lens-like objects capable of inducing false positives in the lensfinder, that is, the simulations were likely not realistic enough. This might explain why, contrary to expectations, the residual CNN has not performed better than the others. We will further explore CNN algorithms in the future GGSLC.

Acknowledgements. The authors would like to acknowledge Frederic Courbin, Colin Jacobs, Ben Metcalf, and Markus Rexroth for their help and advice on the subject. C.S. and J.P.K. acknowledge support from the ERC advanced grant LIDA. T.K. acknowledges support from the Swiss National Science Foundation.

References

Abadi, M., Agarwal, A., Barham, P., et al. 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org
Arpit, D., Zhou, Y., Kota, B. U., & Govindaraju, V. 2016, ArXiv e-prints [arXiv:1603.01431]
Atek, H., Richard, J., Kneib, J.-P., et al. 2015, ApJ, 800, 18
Bolton, A. S., Burles, S., Koopmans, L. V. E., et al. 2008, ApJ, 682, 964
Bonvin, V., Tewes, M., Courbin, F., et al. 2016, A&A, 585, A88
Cabanac, R. A., Alard, C., Dantel-Fort, M., et al. 2007, A&A, 461, 813
Cao, S., Biesiada, M., Yao, M., & Zhu, Z.-H. 2016, MNRAS, 461, 2192
Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. 2011, in 2011 International Conference on Document Analysis and Recognition, 1135
Collett, T. E. 2015, ApJ, 811, 20
Dieleman, S., De Fauw, J., & Kavukcuoglu, K. 2016, ArXiv e-prints [arXiv:1602.02660]
Dieleman, S., Willett, K. W., & Dambre, J. 2015, MNRAS, 450, 1441
Ferreras, I., Saha, P., Leier, D., Courbin, F., & Falco, E. E. 2010, MNRAS, 409, L30
Gavazzi, R., Treu, T., Rhodes, J. D., et al. 2007, ApJ, 667, 176
Giocoli, C., Meneghetti, M., Bartelmann, M., Moscardini, L., & Boldrin, M. 2012, MNRAS, 421, 3343
Glorot, X., Bordes, A., & Bengio, Y. 2011
Hála, P. 2014, Ph.D. Thesis [arXiv:1412.8341]
He, K., Zhang, X., Ren, S., & Sun, J. 2015, ArXiv e-prints [arXiv:1502.01852]
Hebb, D. O. 1950, Science Education, 34, 336
Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. 2012, ArXiv e-prints [arXiv:1207.0580]
Hochreiter, S. 1991, Diploma, Technical University Munich, Institute of Computer Science
Hoyle, B. 2016, Astron. Comput., 16, 34
Huertas-Company, M. 2015, IAU General Assembly, 22, 2252228
Ioffe, S., & Szegedy, C. 2015, ArXiv e-prints [arXiv:1502.03167]
Jacobs, C., Glazebrook, K., Collett, T., More, A., & McCarthy, C. 2017, MNRAS, 471, 167
Jiang, G., & Kochanek, C. S. 2007, ApJ, 671, 1568
Joseph, R., Courbin, F., Metcalf, R. B., et al. 2014, A&A, 566, A63
Kingma, D. P., & Ba, J. 2015, in International Conference on Learning Representations [arXiv:1412.6980]
Kneib, J.-P., Ellis, R. S., Santos, M. R., & Richard, J. 2004, ApJ, 607, 697
Koopmans, L. V. E., Bolton, A., Treu, T., et al. 2009, ApJ, 703, L51
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, in Advances in Neural Information Processing Systems Conf.
Lanusse, F., Ma, Q., Li, N., et al. 2018, MNRAS, 473, 3895
Laureijs, R., Amiaux, J., Arduini, S., et al. 2011, ArXiv e-prints [arXiv:1110.3193]
Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proc. IEEE, 86, 2278
Leier, D., Ferreras, I., Saha, P., et al. 2016, MNRAS, 459, 3677
Lemson, G., & Virgo Consortium, T. 2006, ArXiv e-prints [arXiv:astro-ph/0608019]
Lenzen, F., Schindler, S., & Scherzer, O. 2004, A&A, 416, 391
LSST Science Collaboration, Abell, P. A., Allison, J., et al. 2009, ArXiv e-prints [arXiv:0912.0201]
Mallat, S. 2016, Philos. Trans. R. Soc. London Ser. A, 374, 20150203
Marshall, P. J., Lintott, C. J., & Fletcher, L. N. 2015, ARA&A, 53, 247
McKean, J., Jackson, N., Vegetti, S., et al. 2015, Advancing Astrophysics with the Square Kilometre Array (AASKA14), 84
Metcalf, R. B., & Petkova, M. 2014, MNRAS, 445, 1942
More, A., McKean, J. P., Muxlow, T. W. B., et al. 2008, MNRAS, 384, 1701
More, A., Verma, A., Marshall, P. J., et al. 2016, MNRAS, 455, 1191
More, S., van den Bosch, F. C., Cacciato, M., et al. 2011, MNRAS, 410, 210
Nguyen, A., Yosinski, J., & Clune, J. 2014, ArXiv e-prints [arXiv:1412.1897]
Oguri, M., & Marshall, P. J. 2010, MNRAS, 405, 2579
Paraficz, D., Courbin, F., Tramacere, A., et al. 2016, A&A, 592, A75
Pawase, R. S., Courbin, F., Faure, C., Kokotanekova, R., & Meylan, G. 2014, MNRAS, 439, 3392
Petrillo, C. E., Tortora, C., Chatterjee, S., et al. 2017, MNRAS, 472, 1129
Richard, J., Jones, T., Ellis, R., et al. 2011, MNRAS, 413, 643
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986, Nature, 323, 533
Simonyan, K., & Zisserman, A. 2014, ArXiv e-prints [arXiv:1406.2199]
Sonnenfeld, A., Treu, T., Marshall, P. J., et al. 2015, ApJ, 800, 94
Suyu, S. H., Bonvin, V., Courbin, F., et al. 2017, MNRAS, 468, 2590
Treu, T., & Koopmans, L. V. E. 2002a, ApJ, 575, 87
Treu, T., & Koopmans, L. V. E. 2002b, MNRAS, 337, L6
Treu, T., Auger, M. W., Koopmans, L. V. E., et al. 2010, ApJ, 709, 1195

3.3 Outlook

This section concentrates on the evolution of strong lens finding set after the writing of the above paper.

3.3.1 The problem of simulated data and applying on real data

The simulation problem

One of the main conclusions from the preceding paper is that in the first GGSLC challenge (Metcalf et al., 2019), the simulations were not complete enough to truly test and distinguish between the best performing algorithms. The problem here does not lie with the simulation of the lenses, which is not a problem of incredible complexity, but with the simulation of the non lens objects. Simulating all the different objects existing in the universe is simply an impossibility for us at the moment.

Yet one of the weaknesses of CNNs is that they can react badly to input they have never seen before. Confronted with such an input, CNNs will essentially generate a random classification which could rate higher than real lenses. We have no real fail-safe to guard against this eventuality since we do not have a metric for the "newness" of the input. The best way to avoid this is by letting the CNN "see" every permutation possible of the data, which means presenting it with a training set as complete as possible.

To address this problem, the most popular simulation concept proposed among the contestants during the evaluation of the challenge was the copy-paste concept. The idea is to combine the simulated lenses with real data from the survey to act as false positives. Beyond allowing possible false positives to be comprehensively studied by the network, the data has the added side-benefit to already have the correct point spread function (PSF) and other effects that the simulated data might not have perfectly recreated. They are a few unexplored drawbacks to it since we could conceivably tag real strong gravitational lenses as a non lenses and therefore contaminate the training set, depending on how carefully one constructs the training set. The amount of falsely tagged objects should affect the model only minimally but rigorous testing would be needed for certainty.

Petrillo and Jacobs, two of our competitors on the GGSLG challenge, followed up partially on the idea, applying their CNN from the GGSLG challenge to the KIDS, CFTHLS, and DES survey (Petrillo et al., 2017; Jacobs et al., 2017, 2019) in the objective to generate a lens candidate catalogue. They constructed the training set using simulated lenses and carefully selecting the objects acting as false positives, i.e. a set of elliptical, spiral and irregular galaxies taken from the survey population. Despite excellent training scores, the application on real data proves a lot more difficult than expected.

Application to non simulated data

To cite Jacobs et al. (2017) first try as an example: Having applied the CNN to the 171 square degrees of the CFHTLS survey, they produced a list of 18,861 of candidates which included a "total of 63 of 565 previously found lens candidates (11%) and 14 of 103 confirmed lenses (14%). Of the remaining candidates, using follow-up visual inspection 18,400 were found to be false positives by the authors and only 149 as potential lens candidate." The 199 true positives represent a purity of only 1% for the estimated completeness of 11-14 %.

These results, disappointing considering the promises CNNs showed on simulated data in the previ-

Name	type	AUROC	TPR_0	TPR_{10}	short description
CMU-DL-Resnet-ground3	Ground-Based	0.98	0.09	0.45	CNN
CMU-DL-Resnet-Voting	Ground-Based	0.98	0.02	0.10	CNN
LASTRO EPFL	Ground-Based	0.97	0.07	0.11	CNN
CAS Swinburne Melb	Ground-Based	0.96	0.02	0.08	CNN
AstrOmatic	Ground-Based	0.96	0.00	0.01	CNN
Manchester SVM	Ground-Based	0.93	0.22	0.35	SVM
GaborManchester2	Ground-Based	0.89	0.00	0.01	Human Inspection
ALL-star	Ground-Based	0.84	0.01	0.02	edges/grad./Log. Reg.
CAST	Ground-Based	0.83	0.00	0.00	CNN
SVMYattaLensLite	Ground-Based	0.82	0.00	0.00	SExtractor
LASTRO EPFL	Space-Based	0.93	0.00	0.08	CNN
CMU-DL-Resnet	Space-Based	0.92	0.22	0.29	CNN
GAMOCLASS	Space-Based	0.92	0.07	0.36	CNN
CMU-DL-Resnet-Voting	Space-Based	0.91	0.00	0.01	CNN
AstrOmatic	Space-Based	0.91	0.00	0.01	CNN
CMU-DL-Resnet-aug	Space-Based	0.91	0.00	0.00	CNN
Kapteyn Resnet	Space-Based	0.82	0.00	0.00	CNN
CAST	Space-Based	0.81	0.07	0.12	CNN
Manchester1	Space-Based	0.81	0.01	0.17	Human Inspection
Manchester SVM	Space-Based	0.81	0.03	0.08	SVM / Gabor
NeuralNet2	Space-Based	0.76	0.00	0.00	CNN / wavelets
YattaLensLite	Space-Based	0.76	0.00	0.00	Arcs / SExtractor
All-now	Space-Based	0.73	0.05	0.07	edges/grad./Log. Reg.
GAHEC IRAP	Space-Based	0.66	0.00	0.01	arc finder

Table 3.1 – GSLC challenge results sorted by AUROC (Area Under the Receiver Operating Characteristics curve) score. Results taken from Metcalf et al. (2019). In both space-based and ground-based categories, which imitate the respective observational conditions, CNNs overwhelmingly beat the other methods including human eye observation. Different types of CNN however did not seem to improve results beyond basic CNNs. TPR_0 is the ratio of true lenses before finding the first false positive, TPR_{10} the ratio before the first 10. These metrics are a non conventional way of measuring the recall capability of a classifier, i.e. how many true lenses it finds compared to the amount of false ones. The low performance there was addressed using K-fold variation/committee techniques and layered training phases (see my preceding paper or Jacobs et al. (2019)).

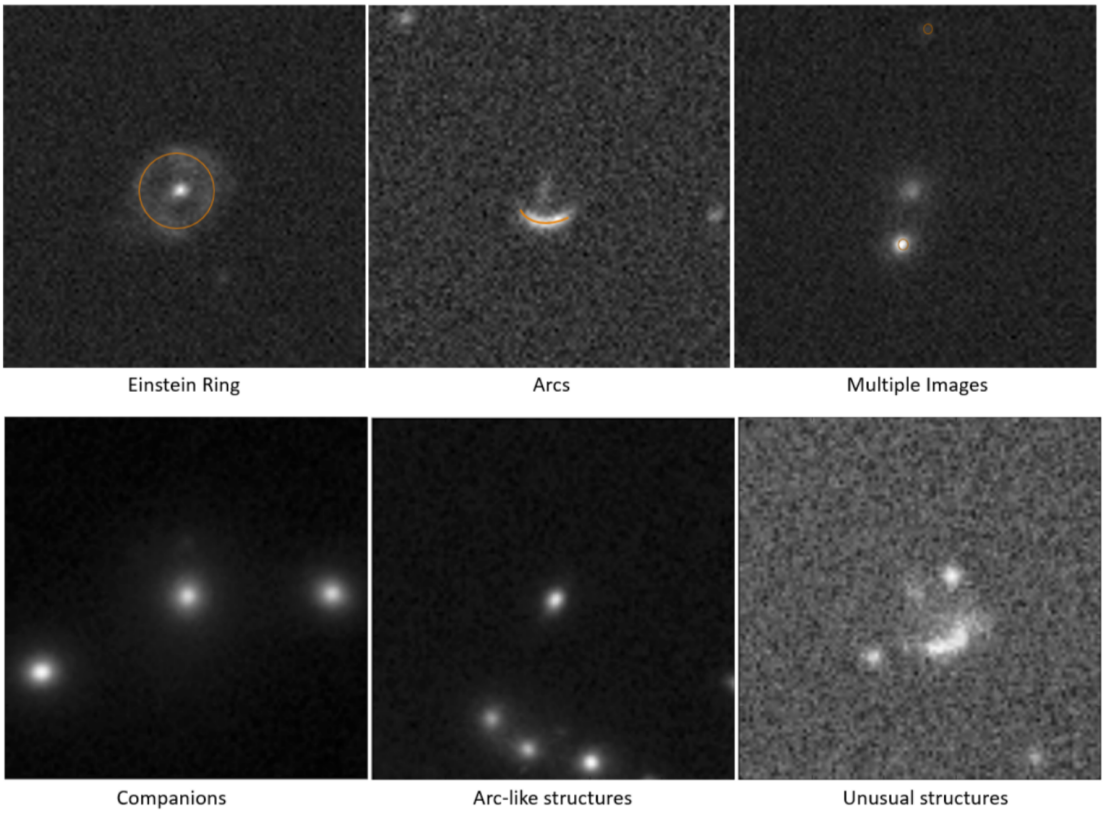


Figure 3.1 – Example of lenses vs false positives. The first row shows examples of correctly identified lensed with the orange line highlighting the lens features like the Einstein ring, the arcs and the position of multiple images. The second shows possible false positives examples that a network or human could easily mistake for lenses.

ous challenges, point towards an overfitting problem. The model trained on simulated data simply generalises badly to the test data mainly because the simulated training data is not realistic enough. My own paper and the one of Jacobs et al. (2019) explore a few commonly used options to improve the performance significantly, like the K-fold variation (similar to commitee) techniques and layered training phases. These improvements have managed to make CNNs extremely usable on large surveys, as a method for presorting obvious non lenses from possible lens candidates. Some projects have started using them to simplify and amplify the use of citizen science in strong lens finding (Sonnenfeld et al., 2020). CNNs have definitely found their place in strong lens finding there. Further improvements should be achievable through improvements to the training set with better simulations and better representation of likely false positives. Research efforts continue in this direction, with for example the lens finding group around Frédéric Courbin, who are currently testing a new batch of simulations.

Even with these improvements however, we are able to pick mainly the "low-hanging fruits" among strong lenses and these "easy" lenses might not be the most interesting lenses depending on our research objective. Our CNNs have not yet proven to be good at lens outlier detection. All of Jacob's applications also continue to require a certain amount of human interaction and verification of the lens candidates, simply because such a CNN model cannot be fully trusted due to the high false positive rate. This makes its application to large scale survey complicated since a fully automated system is

almost certainly needed for surveys of the size of Euclid.

3.3.2 Options for Euclid

Ideally the goal for Euclid was to create a fully automated strong lens finder capable of reliably finding strong lenses of any type. As stated above, the current generation of CNNs is not quite capable of reaching this ambitious goal, still having to work on the outlier detection aspect, the completeness and the high false detection rate of lenses. These are issues that further improvements to the training data might overcome but since Euclid has not generated any data yet that could be used to create such a training set, this could be problematic.

Stage-wise training and application

With Euclid not launched yet, we have no access to real data to create realistic training sets. To test their pipelines, the big survey missions use simulations, which accurately represent all observational defects but, as for the GGSLC challenge, do not show a complete representation of the astronomical objects. This essentially precludes us from presenting a CNN based strong lens finder before the mission is launched, since we cannot apply the copy-paste concept. What we can do however, is use a simulation-trained CNN lens finder and iteratively retrain it with the new incoming data as we get it. That this could work can be inferred from the studies made on the transferability of feature learned by neural networks and stage-wise training (Yosinski et al., 2014; Barshan and Fieguth, 2015). The network would essentially improve its performance the more we augment its training set with real observed data. Thanks to the classification efficiency of CNNs, we can classify data much faster than it is acquired by Euclid, making multiple classification runs eminently feasible. If we combine this with active learning where the learning algorithm can query an expert observer to label some highly uncertain data points for him, acting as a teacher, we could even further improve its capabilities.

Gravitational lens database

A critical component, that has to be done, is the creation of a comprehensive lensing database. With an exception for the non publicly accessible master lens database by Leonidas Moustakas and Joel Brownstein, no such database exist to the best of my knowledge. With the many different papers about lens finders, there exists a multitude of catalogues of lens candidates of various certainty in the literature but nobody has yet collected them all. For this reason I have worked on the development of a web accessible database that should assemble all lenses of the various available surveys. With the main infrastructure finished, there remains a literature review to be done in order to regroup all already found lensed candidates into it before the database can be published. Such a database would centralise all found lenses at one place, allow us to easily add confirmed lenses to training simulation needed for iterative training and to better plan subsequent confirmation of known lens candidates,

Method combination

Another possibility which has not gotten much attention yet for the pipeline preparation is to run multiple different types of methods concurrently and combine the results. Since combining multiple CNNs together has the tendency to create a more robust model with better test results, why not combine radically different methods? As long as the computational resources available are sufficient, it

is possible to run methods like Hartley's SVM (Hartley et al., 2017) alongside CNN and combine their results. With every method specialised in a certain kind of lens, we should be less biased on finding lenses. Knowing how much this is true would merit a more thorough investigation.

Another possibility would be to layer different methods on top of CNN. With CNN doing a presorting of the data, using semi automatic conventional lens finder or citizen science could become feasible again.

Visual check-up and bayesian statistics

While all the above mentioned problem can and will be solved, there is one aspect where current CNN cannot deliver and that is the possibility of avoiding some kind of visual check-up. Without a way to measure our confidence in the prediction, visual check-ups cannot be removed from the pipeline. Luckily, the uncertainty problem is well known and its solution is high on the wish-list of deep learners. An impressive research effort is taking place at the moment into combining bayesian probability theory to deep learning but to the best of my knowledge, no real breakthrough has been achieved yet (Gal and Ghahramani, 2016; Osawa et al., 2019).

Part III

Strong Gravitational Lens analysis

4 High Performance Computing

Software taking too much time is a problem common to all sectors of industry and science. Something any of us knows is the pain of waiting for weeks for your code to finish doing its tasks, creating the results you need for an urgent deadline. In this age of big data, astronomers surprisingly often work at the forefront, creating software that have to handle ludicrous amount of data. As I have shown in the previous chapter, for Euclid we use deep learning to create efficient and fast lens-finders to surmount the big data problem. But Euclid is still considered a small project at big data levels. The Square Kilometer Array (SKA), one of the most ambitious radio telescope projects available, will, if the actual technical parameters are to be believed, create every day an amount of data equal to the actual Internet. The Hubble Space telescope generates images of lens clusters of such a high quality that modelling the lenses inside them takes months.

At this level, creating software capable of handling these quantities of data can still be achieved but only through a rigorous software engineering effort using High Performance Computing (HPC). HPC is the discipline specialised in optimising software performance by taking into account computational hardware design. This thesis goes deeper into HPC that one would expect from an astronomer, which is why the following chapter is dedicated to an introduction of the basic HPC concepts needed to understand the following papers.

4.1 Brief History

4.1.1 The numerical revolution

The numerical revolution is best summarised by Moore's Law. In 1965 Gordon Moores stated that the number of transistors that can be fitted unto a microchip will double every two years with costs being halved. That statement became a truism known as Moore's Law, which was taken as the stated long-term planning goal of the semiconductor industry. With the clockspeed of processors following roughly the same evolution, this exponential growth was so important that software optimisation became slightly secondary. High performance software development was not essential since by waiting a few years, you achieved the desired speed-up anyway.

In recent years however, engineers have had difficulties increasing the clockspeed on processors. Transitor miniaturisation is starting to reach its limit at atomic levels with the smallest transistors to-date being 5 nm big. Every development step is accompanied by increasing cost and problems like

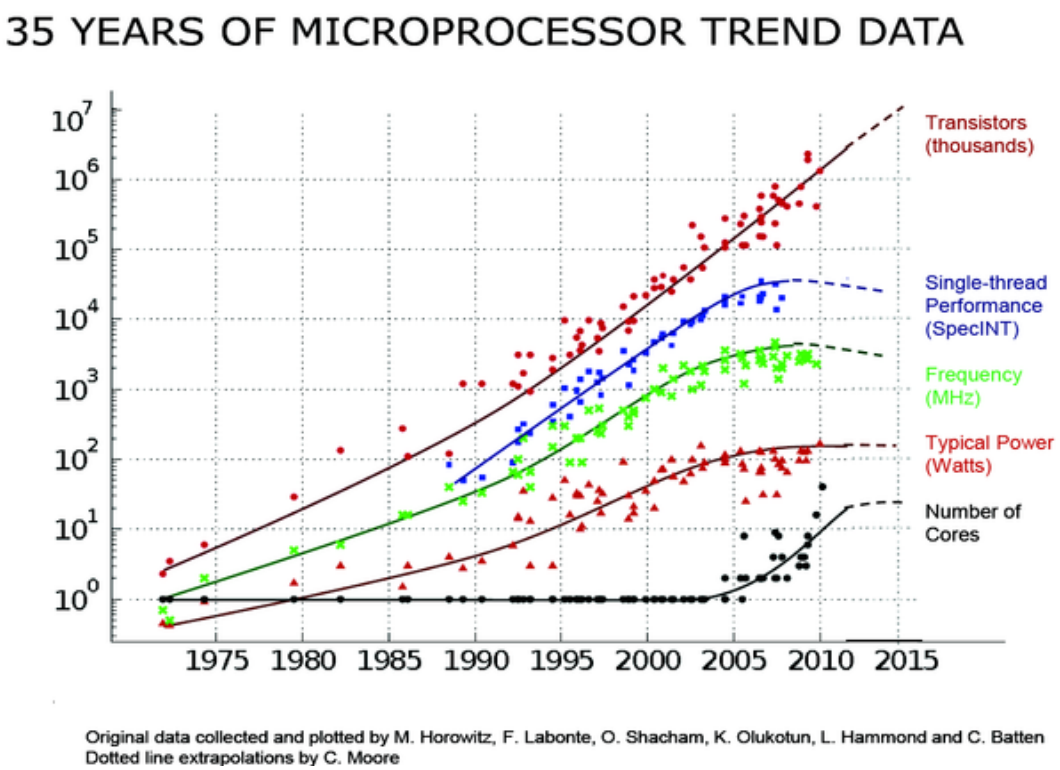


Figure 4.1 – Moore's Law: The number of transistors (red) that can be fitted unto a microchip is doubled every two years. With frequency (green) not keeping up however, the single thread performance (blue) of a CPU is stagnating, shifting hardware from serial to parallel with an increase in number of cores (black).

temperature diffusion show that Moore's Law is finally reaching its end.

4.1.2 Little's Law and the Ninja Gap

To keep up with increasing demands of more computation power and with hardware engineers not being able to increase clockspeed anymore, development ideology changed to take a parallel approach. Taken from queuing theory, John Little proved in 1961 (Little, 1961) that the "long-term average number L of customers in a stationary system is equal to the long-term average effective arrival rate multiplied by the average time that a customer spends in the system".

Applying Little's Law to the software optimisation problem, it translates to

$$T = \frac{P}{L} \tag{4.1}$$

with T the computation throughput, P the amount of parallel operations and L the latency of the operation. The latency is the time it takes the processor to do a specific operation, which is linked in part to its clockspeed.

Since latency improvement was reaching a dead-end, hardware developers used parallelism to further increase the throughput of their hardware. Single core CPUs evolved into multicore processors with every core capable of independent computation. CPU cores acquired the capability of computing the same scalar operation simultaneously as a vector on multiple different data. An extreme example of this design philosophy are Graphics Processing Unit (GPU). Developed for the computationally intensive task of displaying images, GPUs are specifically designed to execute thousand of operations concurrently. This allows them, despite having a higher latency than CPUs, to reach higher throughput values. The parallel design of the resulting hardware allowed engineers to continue to deliver the computation power promised by Moore's Law.

The consequence of this however, is that software and algorithm design has to change to take advantage of the existing parallelism. Especially in astronomy where most software is developed on a case to case basis and professionally maintained software is rare, we are experiencing an increasing performance gap called the "Ninja Gap". Coined by Satish et al. (2012) as "the performance gap between naively written C/C++ code that is parallelism unaware (often serial) and best-optimized code on modern multi-/many-core processors", this performance gap promises to become worse as more multicore processors are developed further. HPC specialises in reducing this Ninja gap to a minimum by introducing parallelism into serially designed software to take into account the parallel design of the hardware.

4.2 Strong Scaling vs Weak Scaling

Ideally, if we double the amount of computation units, the computation time of a task should be divided by two. However, depending on the amount of communication needed, tasks possess different amounts of parallelism potential. Any serial sub-task depending on the results of a previous sub-task will create a non parallelisable overhead and this overhead limits the amount of speed-up achievable through parallelisation. To study how much a task is limited by its serial part, it is necessary here to make a distinction between the strong and weak scaling of tasks.

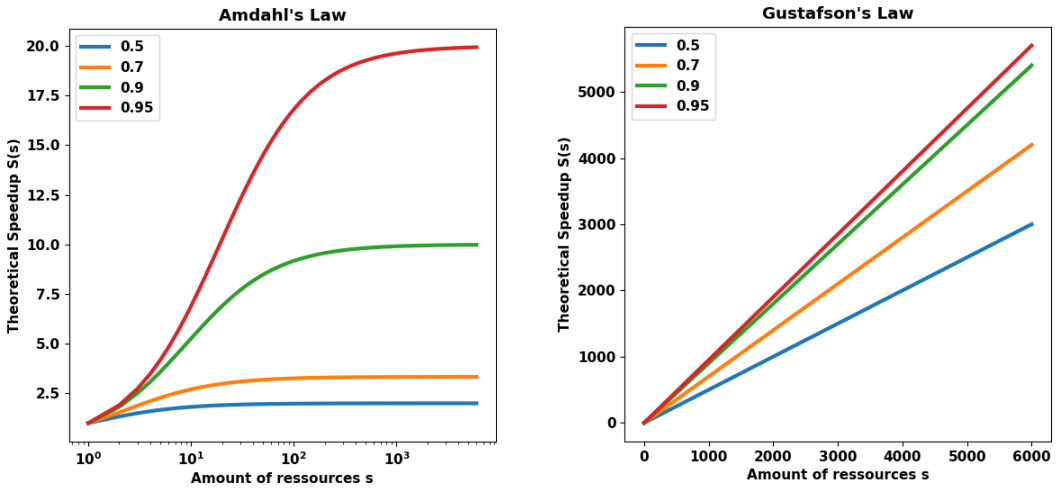


Figure 4.2 – Amdahl's and Guststafson's Law show the theoretical speed-up reached when distributing computational work over multiple resources. The evolution depends on the amount of non parallelisable portions of the algorithm (colored lines) and if the computational load scales with the resources (strong vs weak scaling). The legend (colored lines) shows the fraction of parallel parts in the algorithm.

Strong scaling studies the speed-up of a task in function of computation units for a fixed workload, while weak scaling assumes the workload increases in function of computation units. Using Amdahl's Law and Gustafson's Law for strong scaling and weak scaling respectively we can calculate the theoretical speed-up of a task based on its parallelisable portion. Amdahl's Law states:

$$S(s) = \frac{1}{(1-p) + \frac{p}{s}} \tag{4.2}$$

with S the theoretical speed-up, p the execution time of the parallelisable portion before parallelisation and s the speed-up of the parallelisable portion when distributed over more ressources.

Gustafson's Law states:

$$S(s) = (1-p) + p \cdot s \tag{4.3}$$

As can be seen in Fig.4.2, using Amdahl's Law, even for tasks with a low proportion of serial parts, the speed-up gained by increasing the amount of resources will quickly hit a maximum threshold defined by the serial part of the task. This showcases an apparent bottleneck that parallel computing is facing. In reality however, the size of the problems we try to resolve often scales with the amount of computation power at our disposal. While we may not be able to do a certain task faster, we could do the same task on a larger scale in the same time as before. Gustafson' Law (Fig.4.2) showcases exactly this, with no threshold and the amount of serial parts of task only impacting the efficiency of the linear

scaling.

Strong and weak scaling are studied during benchmarks to evaluate the efficiency of the parallelisation of a program and before deployment on HPC clusters. They allow users to optimally balance the cost of using multiple computation units with the desired speed-up when launching the software on HPC clusters.

Understanding the difference between weak and strong scaling is therefore crucial to understand how to reformulate a problem to best take advantage of parallelism and to develop an algorithm that can perform well on HPC clusters.

4.3 Parallelism

4.3.1 Parallelism levels

During algorithm design, depending on the task and the used hardware, parallelism can be extracted at multiple different levels. The following list is non exhaustive but the main levels to know about are Task Level Parallelism (TLP), Data-level Parallelism (DLP), and Instruction Level Parallelism (ILP). When designing a parallel algorithm that scales well, it is important to decide how to best distribute the computation load over these levels.

Task Level Parallelism

Task Level Parallelism measures the amount of different tasks performing concurrently over multiple processors. Supposing an algorithm can be subdivided into multiple independent tasks, Task Level Parallelism will concentrate on distributing these tasks to work in parallel over multiple computation units. Hardware-wise, this can be done on a multi-core CPU distributing tasks over the multiple cores of a CPU as threads, or over multiple different CPUs.

Data Level Parallelism

Data level Parallelism, in contrast to TLP, measures parallelism gained by running the same task simultaneously on different data. Matrix operation for example entails making the same operation on multiple rows of different data. Distributing this operation over multiple CPUs, cores or using the single core vectorisation capability, all belong to the DLP level.

Instruction Level parallelism

Instruction Level Parallelism measures the amount of instruction executed simultaneously by the processor. An example for an ILP technique is instruction pipelining. To complete an instruction, a CPU processor has to execute different tasks like fetching instructions, accessing memory and writing the results to the register. All of these tasks, modern CPUs are capable of doing simultaneously but not for the same instruction. To take advantage of this, in instruction pipelinining, the compiler will overlap independent instructions so as to fully occupy the CPU's task capabilities. These techniques also include out-of-order executions and branch prediction.

ILP techniques are mainly implemented automatically by the compiler level, without needing specific developer input, except for a few compiler commands.

4.4 Understanding the hardware

Designing fast software does not only require a well designed parallel algorithm but also has to take into account the capabilities of the underlying hardware. Historically Flynn's Taxonomy (Flynn, 1967) was used to describe the distinct classes of parallel architecture: Single Instruction Single Data stream (SISD), Single Instruction Multiple Data stream (SIMD), Multiple Instruction Multiple data stream (MIMD) and Multiple Instruction Single data stream (MISD). Clusters of CPUs & GPUs, multicore CPUs, CPU cores and GPUs, all are classified into these classes or their variations, although only SIMD and MIMD are really still used today.

4.4.1 CPUs & SIMD (Vectorisation)

SIMD is a class of parallel computers with multiple processing elements that perform the same operation on multiple data points simultaneously. These types of computers exploit Data Level Parallelism to increase their throughput. Most modern CPU processors are small-scale SIMD class computers because they include SIMD instructions which can be used to upgrade scalar operations to vector operations. This vectorisation can be done manually or using the compiler autovectorisation function.

Using the compiler autovectorisation is arguably the easiest option to exploit these capabilities but to be able to use them one has to pay attention to the memory layout. Software development usually uses the Abstract Data Structures (ADS) paradigm where data is organized into structures or classes with their attributes stored contiguously in memory. This "Array of Structure" (AoS) layout has several advantages, it being intuitive for humans to understand and that in a single memory access, the processor likely has access to the whole structure.

To use the SIMD units however the data has to be organised into a "Structure of Array" (SOA) layout where the attributes of multiple structures are sorted by type and not by specific object. Using the SOA layout, one memory access fills the register with data of the same attribute which is exactly what is needed for SIMD operations. More details on this subject can be found in both papers of Chapter 6.

4.4.2 Clusters & MIMD

Multicore CPUs and clusters of multiple CPUs and/or GPUs belong to the Multiple Instruction, Multiple Data (MIMD) class. MIMD machines have multiple processing elements that perform asynchronously and independently. They can apply different instructions on different types of data making them the most versatile parallel computer class. Variations exist depending on the memory model. Multicore CPUs "share" their memory, meaning they have access to a globally available memory. This makes memory access easier but is difficult to scale to more than a hundred of processing units. CPU clusters distribute memory over the distinct processing elements, each with their distinct memory. While scalable, it means the software has to handle all the memory transfer necessary for the computation, possibly creating memory bottlenecks.

4.4.3 GPUs & SIMT

Single Instruction Multiple Threads (SIMT) is a variant of SIMD which highlights the difference between CPU and GPU programming. SIMT is an execution model used mainly by GPUs which tries to hide the inevitable latency of memory access through multi-threading. In contrast to CPUs with few cores and low latency, GPUs possess an abundance of computation units organised into warps with a high latency. Only a few of these warps can be run simultaneously but by switching between tasks during memory access, the GPU can compute continuously without having to wait for memory access. This effectively "hides" the latency but it does require problems which can provide the necessary amount of computation.

Another downside is that threads inside warps cannot run different instructions simultaneously. Any thread divergence like those caused by if-else statements are handled by letting the whole warp first handle the "if" and then the "else" statement. The more divergent the threads the more performance you lose to a point of extreme inefficiency, which makes a coherent control-flow vital. If-else statements can still be used but need to take into account warp-size and behaviour.

4.4.4 CPU vs GPU?

The differences between CPUs and GPUs can be summarised thusly. CPUs are universal workhorses with few but very efficient cores. They are specialised in reducing latency to a minimum and any algorithm that possesses an important amount of serial tasks like a recursive algorithm will benefit from using them. GPUs are parallelism specialists with lots of slow cores. GPUs are more tricky to use because they will only be efficient if the problem has enough parallel computation to hide the high latencies. The computation units of GPUs are simpler than CPUs, with less arithmetic support, divergence problems and other peculiarities. However, the more computation, the better GPUs will be, even to the point of significantly beating CPUs. The choice of using one or the other or even both is therefore entirely dependent on the problem.

5 Cluster Lens Modelisation

Once strong lenses are found and confirmed comes another crucial step before being able to use them for any science application: the mass modelling. All scientific applications using strong gravitational lenses depend strongly on precise models. The more precise the model, the lower the error bars of the resulting science. Typically, parametric mass models are fitted to the lenses using the multiply imaged sources as constraints. Galaxy lenses are much more numerous and have, compared to cluster lenses, a relatively easy to produce mass model. Using simulation and analytical profiles, astronomers have already a good idea of the mass profile of a galaxy. This allows us to fit analytical models controlled by only a few free parameters to the observational data and the corresponding parameter space is quite small. Using one or two parametric models is often sufficient to model a well-resolved galaxy lens, giving us an idea of the Einstein radii and the mass behind.

Mass modeling however reaches a whole new level of complexity when studying cluster scale lenses. Clusters can be comprised of up to 200 galaxies, each having to be modelled by a parametric mass model. Each galaxy possesses a non negligible gravitational potential and can significantly change the lensing effect. Multiple images used as constraints can jump up to 200 and more, each which has to be checked at every step of the mass model fitting. All this makes cluster mass-modelling a non trivial task for strong lens astronomers. The following chapter will give an overview of the complexities involved with the process and an in-depth introduction to Lenstool, a highly successful lens-modelisation software first created by Kneib et al. (1996) and maintained and developed by Jullo et al. (2007); Jullo and Kneib (2009).

5.1 Clusters Lenses

Galaxy clusters are the largest virially bound structures in the universe. Consisting of hundreds of galaxies and some massive dark matter halos, the massive gravitational potential they possess distorts, amplifies and multiplies the galaxies lying behind them to create a sky image of considerable confusion and beauty (Fig. 1.12). Reaching up to an amplification factor up to ≈ 40 (Kneib and Natarajan, 2011) allowing astronomers to observe early universe galaxies with redshift of up to 9-12 and map complex DM halos merging, clusters are rife with scientific potential.

To take full advantage of them, galaxy cluster surveys have done amazing work to provide quality photometric and spectroscopic data of cluster lenses. Amongst the most significant are the three surveys done using the Hubble Space Telescope (HST): the CLASH (Cluster Lensing And Supernova

survey with Hubble) which observed in-depth 25 massive galaxy cluster (Postman et al., 2012), the HST Frontier Fields project (Lotz et al., 2017) for the observation of six deep fields centered on strong lensing galaxy clusters from Abell et al. (1989); Ebeling et al. (2001) and its current follow-up project Buffalo (Beyond Ultra-deep Frontier Fields And Legacy Observations) (Steinhardt et al., 2018).

5.2 Modelling Clusters

Modelling clusters is done using the multiply lensed images of distant light sources as constraints to fit multiple parametric mass models to a cluster. In theory the resulting composite mass model of this fitting process accurately describes the mass-distribution of the cluster. To make this process easier, astronomers use specialised mass modelling software like Lenstool (Kneib et al., 1996; Jullo et al., 2007), on which the following part of the thesis is based.

5.2.1 Parametric models

Lenstool for example proposes over 50 different mass models, with new ones added almost every year, allowing mass modellers to easily fit their preferred mass distribution. Among the more commonly used mass distributions feature the Pseudo Isothermal Elliptical Mass Distribution (PIEMD), first introduced by Kassiola and Kovner (1993), and its "dual PIEMD" variant by Elíasdóttir et al. (2007), which is essentially a truncated PIEMD with a core and a scaling radius. Lenstool supports also the popular Navarro-Frenk-White profile (NFW) (Navarro et al., 1996) created by studying dark matter N-body simulations predictions. To avoid overfitting, the complexity of mass-models that can be fitted to a cluster depends on the amount of constraints available. Luckily due to improvements in observations, the amount of constraints available for massive clusters are increasing making more complex mass-models possible. Using Multi Unit Spectroscopic Explorer (MUSE) spectroscopy for example, Lagattuta et al. (2017) & Lagattuta et al. (2019) increased the amount of multiply imaged sources with secure redshift from 4 to 45 for the cluster Abell 370.

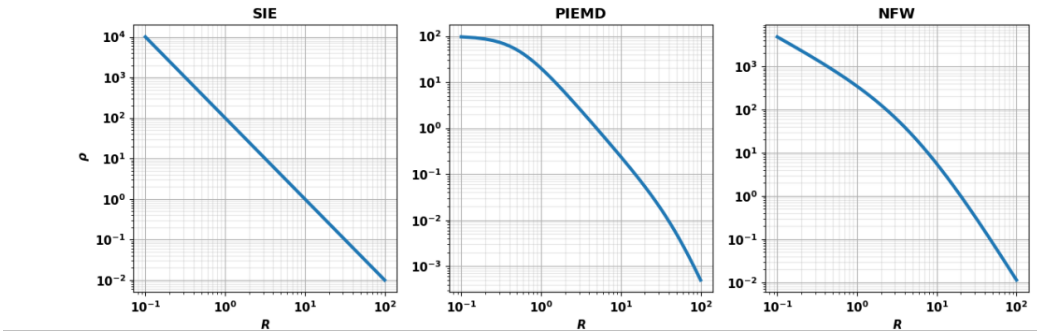
5.2.2 Constraining the model

Constraining the mass in cluster cores using multiple images lens systems is done through maximising a gaussian likelihood for the observed data D and parameters p of the model:

$$\mathscr{L} = \Pr(\mathrm{D}|\mathbf{p}) = \prod_{\mathrm{i}=1}^{\mathrm{N}} \frac{1}{\prod_{\mathrm{j}=1}^{\mathrm{n_i}} \sigma_{\mathrm{ij}} \sqrt{2\pi}} \exp^{-\frac{\chi_{\mathrm{i}}^2}{2}} \tag{5.1}$$

with N the number of systems, n_i the number of multiple images for the system i and χ^2 the error between model prediction and constraints given by:

$$\chi_i^2 = \sum_{j=1}^{n_i} \frac{[\theta^j_{obs} - \theta^j(\mathbf{p})]^2}{\sigma^2_{ij}} \tag{5.2}$$



$$\rho(R) = \frac{\rho_0}{R^2} \qquad \rho(R) = \frac{\rho_0}{(1+\frac{R^2}{r_{cut}^2})*(1+\frac{R^2}{r_{core}^2})} \qquad \rho(R) = \frac{\delta_c \rho_c}{\frac{R}{r_s}*(1+\frac{R}{r_s})^2}$$

Figure 5.1 – Three of Lenstool's most used parametric mass distributions: Singular Isothermal Elliptical mass distribution (SIE), Pseudo Isothermal Elliptical Mass Distribution (PIEMD) and the Navarro-Frenk-White profile (NFW). These distributions show well the core-cusp problem astronomers face when modelling galaxies with NFW belonging to the cuspy mass profiles and PIEMD introducing a core radius (here for r = 5) where mass stops increasing when reaching the center of the galaxy, modelling a smooth core. Here $\rho_0, \rho_c, r_c, r_{cut}, r_s$ are all parameters that will be constrained, with the multiple images acting as constraints.

with θ^j_{obs} the position of image j, $\theta^j(\vec{p})$ the position of image j predicted by the current model and σ_{ij} the error on the position of image j.

There also exists a variant using the source position as constraints instead of image positions:

$$\chi^2_{S_i} = \sum_{j=1}^{n_i} \frac{[\theta^j_S(\mathbf{p}) - <\theta^j_S(\mathbf{p})>]^2}{\mu_j^{-2}\sigma^2_{ij}} \tag{5.3}$$

with $\theta^j_S(\mathbf{p})$ the source position of the observed image j, $<\theta^j_S(\mathbf{p})>$ the barycenter position of the n_i source positions and μ_j the magnification for image j. Being less constraining but much faster to compute, it is usually used for an initial fitting phase to pre-constrain the model. Initially when the model is not well constrained, the solutions proposed are so far from the truth that the number of multiple images generated often do not match the number of constraints. This leads to the model simply being completely rejected, which does not give us much useful information about the parameter distribution. Using source positions as initial constraints to constrain the model makes it possible to constrain the model sufficiently that switching to multiple image constraints becomes viable.

In general the fitting process goes through multiple iterations, first using only more securely spectroscopy identified sources which in turn allows us to confirm or find less secure multiply imaged sources to gradually increase the accuracy of the mass models.

5.2.3 Cluster mass components

Clusters are not well represented by a single parametric model. Usually the cluster gravitational potential is described by the sum of multiple parametric models which can be grouped into two different types of potentials

$$\phi_{tot} = \sum_i \phi_{c_i} + \sum_j \phi_{g_j}$$

where ϕ_{c_i} are the smooth, large-scale potentials representing the cluster dark matter and the intra-cluster gas and ϕ_{g_j} are the smaller perturbing potentials due to the individual cluster galaxies (Natarajan and Kneib, 1997).

This separation is necessary because the lensing due to the mass of each individual galaxy member cannot be neglected. Following the studies of Kneib et al. (1996); Natarajan and Kneib (1997); Natarajan et al. (1998), the consensus is that the sum of the individual dark matter halos of each galaxy represents approximately 10% of the total cluster mass and therefore can significantly affect the lensing behaviour. While most galaxy members simply increase the total mass of the cluster, in some cases the lensing effect of a galaxy can affect the position and the multiplicity of lensed images. For this reason, when modelling clusters, the final model also includes a parametric model for each galaxy that is within two times the Einstein radius of the cluster.

5.2.4 Bayesian MCMC

To explore the parameter space of the parametric models used, we typically use a sampler, in our case a bayesian MCMC sampler. In general the bayesian approach is prefered in situations where we do not have enough data to fully constrain the model. The bayesian frame work (Jullo et al. (2007)) allows us to add our existing knowledge about the parameters as a prior function. This is especially useful in avoiding degeneracies (multiple solutions to the same problem). The bayesian approach is based on Bayes' theorem

$$\Pr(\mathbf{p}|\mathrm{D},\mathrm{M}) = \frac{\Pr(\mathrm{D}|\mathbf{p},\mathrm{M})\Pr(\mathbf{p}|\mathrm{M})}{\Pr(\mathrm{D}|\mathrm{M})} \tag{5.4}$$

with $\Pr(\mathbf{p}|\mathrm{D},\mathrm{M})$ the posterior Probability Density Function (PDF), $\Pr(\mathrm{D}|\mathbf{p},\mathrm{M})$ the likelihood of getting the observed data D given the parameters $\mathbf{p}$ of the model M, $\Pr(\mathbf{p}|\mathrm{M})$ the prior PDF for the parameters and $\Pr(\mathrm{D}|\mathrm{M})$ the evidence.

For each step of the sampling process, we calculate the Likelihood for a certain set of parameters $\mathbf{p}$ of the proposed model M. The sampler will calculate the posterior PDF using the above mentioned Bayes' theorem and then use it as a new prior. We are essentially incorporating with each step the new knowledge we gained when testing a set of parameters into the prior. The bayesian MCMC sampler converges to the posterior PDF which is highest for the set of parameters $\mathbf{p}$, which gives the best fit to the constraints and is consistent with the prior PDF, essentially representing the prior knowledge we have of the system.

The evidence Pr(D|M) acts as a sort of Occam's razor[1] because it measures the complexity of the model M and penalises more complex models. Lenstool uses the Bayesys3 package by John Skilling[2] which generates samples following the bayesian principle quickly. It is resistant to local optima and can propose new models even for a multidimensional parameter space quickly.

5.2.5 The curse of dimensionality

With the increase in quality of observational data and our need to model that data as accurately as possible, astronomers now face a common problem in optimisation and sampling which is called the "curse of dimensionality". This refers to the exponential growth of the volume associated with adding new dimensions to a mathematical space. In our case this is the parameter space, the mathematical space created by the total amount of different parameters possible for the current parametric mass model. As an example, to uniformly sample with 10^{-2} unit interval a 1 unit cube (3 dimensions), one would need 10^2 points. In our case, considering a model with 100 parametric mass models (1 for smooth cluster halo, and 99 for cluster member), and each model with 6 parameters, one would have to sample $\frac{(10^2)^{600}}{10^2} = 10^{1198}$ points for the same 10^{-2} unit interval.

Bayesian based MCMC sampling, which is much more efficient than uniform sampling, can actually constrain such a parameter space with fewer samples, but even using it, the process takes time. And the more complex the models astronomers wish to use, the bigger the parameter space and the more time it takes. As a rough estimate, one modelling run for a cluster can take up to a month. Truly finishing a model for a typical cluster necessitates multiple of these runs, making the publication of a precise cluster model a lengthy proposition.

5.3 Lenstool

Mass modelling software like Lenstool incorporates all the above concepts in a handy package to allow for easier mass modelling. Development on Lenstool was started by Jean-Paul Kneib in his Phd thesis (Kneib et al., 1996) with bayesian statistics later implemented by Eric Jullo (Jullo et al., 2007). It has been heavily and successfully used (Richard et al., 2007; Limousin et al., 2008; Richard et al., 2009; Jauzac et al., 2014, 2018) since then. It is already well optimised but the time problems of modelling stated above are starting to affect the software. This is what made us decide to rewrite the software to be more compliant with the current computer hardware in the hope of gaining efficiency. Before doing that however an in depth understanding of the bottlenecks is essential. One is of course the curse of dimensionality affecting the MCMC exploration mentioned previously, but another one is the fit computation.

5.3.1 Fit computation

For the image fit computation, Lenstool proposes two types of methods, the brute force and the image transport method. Both of these have advantages and disadvantages.

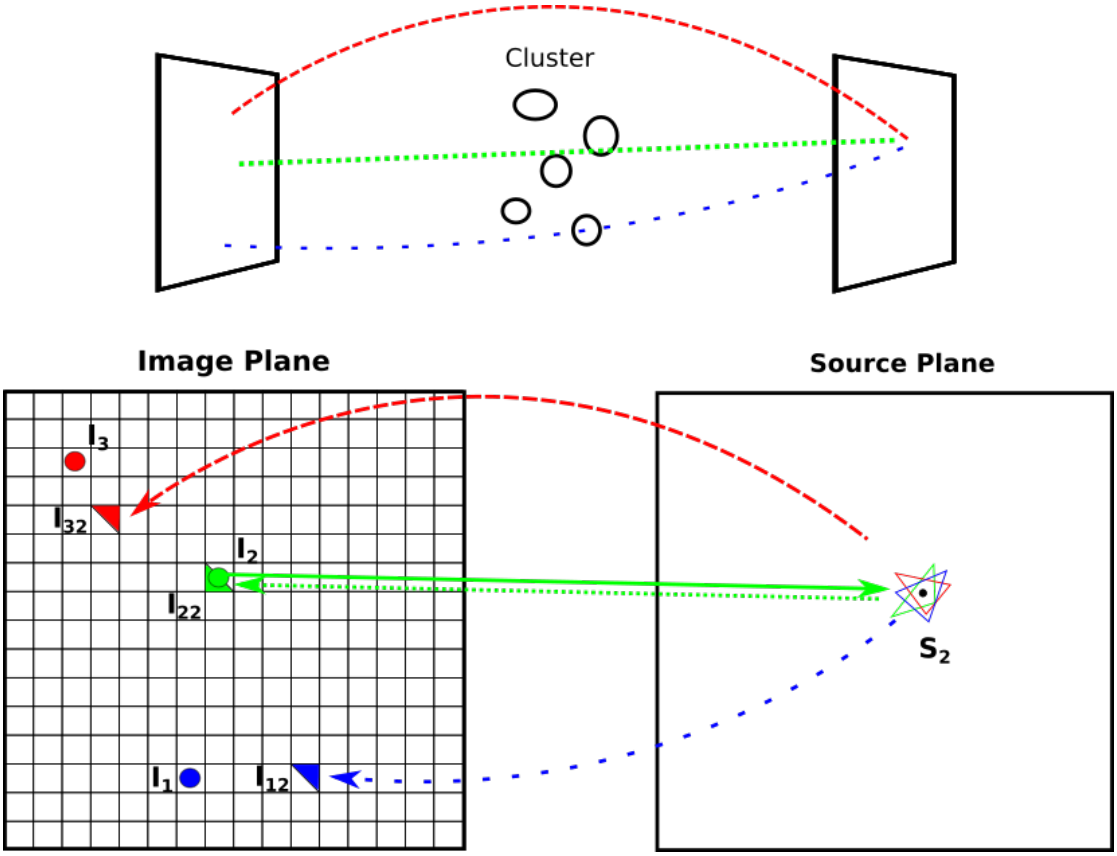


Figure 5.2 – Brute force lens equation resolution. Using the lens equation, we project one of the images I_2 onto the source plane, which gives us a source position. Then every cell of a quadractic grid is unlensed unto the source plane and checked for the presence of the source. Typically, each cell is divided into two triangles for more efficient source checking. If an unlensed cell contains the source, its lensed counterpart is kept as a solution. The barycenter of that triangle is considered the position of a predicted multiple image.

Brute force

The brute force method is the algorithmically simplest way we have to solve the lens equation. Since the lens equation is non invertible, it is easy to compute the source position from the images but to do the inverse is analytically impossible. In this direction the lens equation has multiple solutions, each corresponding to the different multiply lensed image positions. To solve this, we typically trace a grid in the image plane and unlens that grid unto the source plane.

Supposing we have one family of three multiple images I_1, I_2, I_3, all generated by the same source. Using the lens equation and the to be tested mass distribution, we project one of the images onto the source plane giving us a source position. We then unlens each cell of the quadratic grid onto the source plane and check for the presence of a source. A cell found containing the source is retained as a solution. For an ideal mass distribution, these solutions will be identical with the constraints that generated them. Since usually the tested mass distribution deviates from reality, the computed theoretical images will not correspond exactly to the other images (Fig.5.2). This deviation is the χ^2 error metric mentioned in Eq. 5.2 used to evaluate the mass distribution.

This method is extremely stable but computationally slow because the size of the quadratic grid determines the precision of the prediction. Since we want the prediction to be as precise as the observational material, for a Hubble Frontier Fields (HFF) cluster image of $200''$ the grid needs to have a size of 6000 by 6000 pixels for a precision of $0.03''$. The unlensing of this grid however is computationally intensive because for each corner one has to calculate the deflection gradient. This gradient is the sum of the deflection gradients of each single mass distribution of the cluster. With a cluster of 200 members to unlens one grid, one would have to do around 10^{10} computations of complex gradient operations.

Image transport

A popular alternative in Lenstool is the image transport method which proposes an iterative solution to the problem. Instead of unlensing an entire grid, the method starts by un-lensing a triangle around the constraints where it should find at least one source. It subdivides the triangle into four smaller triangles, checks which subtriangle has the source and then redoes the process, zooming in on the predicted image position. This method needs a lot less deflection computation than the brute force approach and is sequentially much faster. It suffers however from multiple problems that make it less stable and more tricky to use. It works for example only when the model is already reasonably close to reality because it does not handle missing images well. When the counterpart of a constraint does not exist with the current model, or if two images are close enough together that they are in one triangle the iterative loop is broken or simply ignores an image, falsifying the results.

Another problem is the zoom-in problem, inherent to any zoom approach in the source plane. The triangles in the image and the source plane are simply a set of three points. The subtriangles are created by taking the middle point between two points of the initial triangle (Fig. 5.3). However since any line in the image plane when unlensed unto the source plane is distorted into a curve, the area covered by the fours subtriangles in the source plane will not be the same as the initial triangle. Instead the sub triangles will either not cover some part of the initial triangle or check outside the triangle. This can lead to a missing image problem or even shifted predictions. These problems increase with magnification

[1] "All things being equal, the simplest solution tends to be the best one."
[2] https://www.inference.org.uk/bayesys/

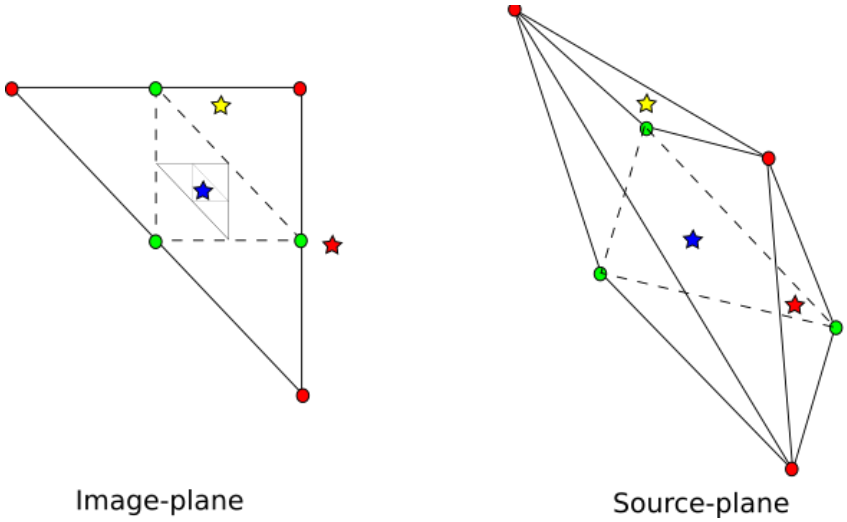


Figure 5.3 – Schematised zoom-in problem: The image transport method consists of iteratively subdividing a triangle centered around a constraint, zooming in to the necessary precision (blue star). While fast, it is not perfectly stable. Because of the lens effect distortion, the subdivided triangles do no cover the same area as the initial triangle. This can cause sources to be mistakenly found (red star) or lost (yellow star). While small initially, this problem increases with amplification, making the method break down around the critical lines.

and the method breaks down near the critical line. Lenstool handles these problems using checks and a more sophisticated version of the algorithm, capable of stepping back and adapting the triangle size, but the solution is not perfect. This makes the model fitting using image transport tricky to use, introducing possible errors. Typically, lens modellers will use image transport to fine-tune a model when the model is already well determined, to avoid most of these problems.

6 Lenstool-HPC

Lenstool-HPC is a remake of the original Lenstool code using modern HPC concepts. The goal was to keep it as close as possible to the original software while making it capable of running on modern supercomputers. This development of its concurrent capabilities allows the software to keep up with future computer architecture, ensuring its longevity while the essentially identical user interface allows an easier transition for active users from Lenstool to Lenstool-HPC.

This chapter gives an overview of the design process and presents the details in two papers written on the subject of Lenstool-HPC. The first of the following papers works on the question of numerical precision, presenting error propagation calculation through Lenstools equations with benchmarks to verify the theoretical calculations. It showcases the reduced energy consumption and higher speed-up one would gain from using single floating point format instead of double for storing numerical numbers. The second paper presents the first version of Lenstool-HPC, with benchmarks of its efficiency and speed-up compared to Lenstool.

6.1 Lenstool-HPC design

Before starting the development of an HPC code, it is necessary to set a rigorous design plan that answers the basic design questions. What algorithms do we use? What are the bottlenecks? What levels of parallelisation can we exploit? On what hardware do we run the software? What numerical precision do we need? Drafting an initial comprehensive design of these questions is necessary to avoid dead-ends and repeated work.

The algorithm choice is dominated by the scaling question. Depending on its parallelisable portion, a slower algorithm can quickly become faster when distributed over multiple cores than its faster, more serial counterpart. For this reason, for Lenstool-HPC we took the brute force approach as the main fitting algorithm. Despite being slower, it is eminently more parallelisable and more stable than the image transport method. Cluster lens modelling, at least in the case of Lenstool using the brute force method, suffers from four specific bottlenecks: the computation of one deflection angle gradient using parametric model, the computation of those gradients over a grid, the computation of the fit of the model once those gradients are computed and the sampling of the parameter space using an MCMC sampler. Each of these bottlenecks can be optimised at different levels of parallelisation shown in detail in Fig. 6.1 so as to get every possible computation potential out of the available hardware. The gradient computation and grid gradient computation represent a classic SIMD/SIMT problem which is ideal for

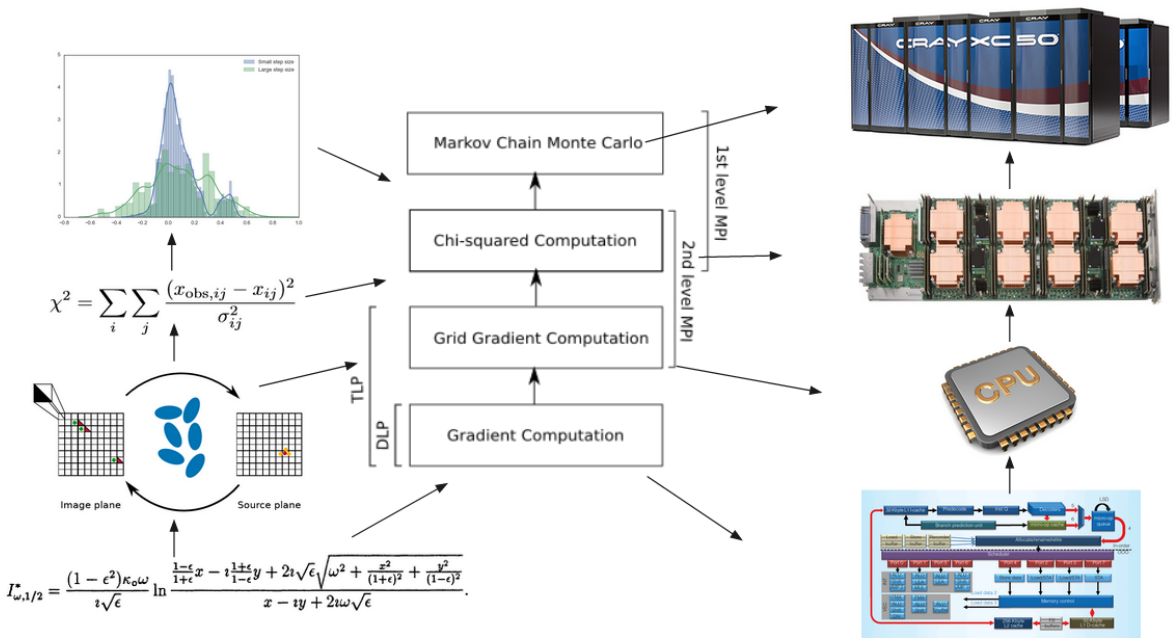


Figure 6.1 – Schematised design plan of Lenstool-HPC. The fit computation is an complex MIMD problem which distributes different tasks over multiple CPUs and GPUs using MPI. Of these tasks, gradient computation is the most time intensive and is accelerated at the DLP and TLP level using vectorisation, multi-core CPUs and GPUs.

a DLP and TLP implementation. It is a big workload with a lot of independent operations which can easily be distributed over multiple CPU cores and GPUs. The fit computation itself, which contains the gradient computations, is an MIMD problem, with multiple complex operations which can be distributed using MPI over multiples CPUs and GPUs. This allows the usage of GPUs for SIMD/SIMT problems like the gradient computation, and CPUs for the more sequential part of image/constraint association.

The question of what numerical precision is needed revealed itself to be a more complicated question, which we address in detail in the following paper.

6.2 High Performance Computing for gravitational lens modeling: single vs double precision on GPUs and CPUs

6.2.1 Preface

The question of whether to use single or double floating point numbers (better known as float and double) to store the numerical values during the computation is one that most scientist will never ask themselves and this with some justification. The underlying time-consuming error propagation calculation are difficult to justify in the current fast-paced publishing scientific lifestyle. Most scientist create codes to test some new concepts quickly. To avoid problems always using the highest machine precision available makes sense when you do not care about computational speed and want to avoid numerical errors.

As soon as the codes evolve however into more definite scientific software which become widely used, an error propagation analysis does become important. First the impact of finite machine precision is widely underestimated and the numerical error originating from even doubles could have an impact on observational data already suffering from experimental errors and observational biases. Secondly the cost of running softwares like Lenstool especially in the case of huge datasets can become prohibitive in terms of energy cost. Running a super-cluster is extremely expensive, something which is often underestimated especially by young scientists. The Pizdaint supercomputer for example takes 2,300 kW to run which is roughly the equivalent of 1850 households while the most powerful supercomputer, the Tian-He 2[1], has an energy consumption of 18,000 kW, the equivalent of 14450 households. With many of these supercomputers existing around the world and the growing demand for them, any attempt at reducing these costs can have a serious beneficial impact in financial and ecological terms. Thirdly, as we show in the paper, single precision floating point computations are in general faster than double, because the hardware has been built for it and it can do more computations simultaneously. For widely used scientific software, faster computation is always a boon.

6.2.2 Paper

This chapter is presented in the form of a published paper as Markus Rexroth, Christoph Schaefer, Gilles Fourestey, and Jean-Paul Kneib, High Performance Computing for gravitational lens modeling: single vs double precision on GPUs and CPUs, Astronomy and Computing, [Volume 30, January 2020, 100340].

[1] https://www.top500.org/lists/2019/11/

Astronomy and Computing 30 (2020) 100340

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

High Performance Computing for gravitational lens modeling: Single vs double precision on GPUs and CPUs

M. Rexroth [a], C. Schäfer [a,*], G. Fourestey [b], J.-P. Kneib [a,c]

[a] Institute of Physics, Laboratory of Astrophysics, Ecole Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, 1290 Versoix, Switzerland
[b] SCITAS, Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
[c] Aix Marseille Université, CNRS, LAM (Laboratoire d'Astrophysique de Marseille) UMR 7326, 13388, Marseille, France

ARTICLE INFO

Article history:
Received 9 February 2019
Accepted 19 October 2019
Available online 31 October 2019

Keywords:
Gravitational lensing
Computing methodologies: Parallel computing methodologies: Parallel algorithms: Massively parallel algorithms
Applied computing: Physical sciences and engineering: Astronomy
Galaxies: clusters: general
Galaxies: halos
Dark matter

ABSTRACT

Strong gravitational lensing is a powerful probe of cosmology and the dark matter distribution. Efficient lensing software is already a necessity to fully use its potential and the performance demands will only increase with the upcoming generation of telescopes. In this paper, we present a proof-of-concept study on the impact of High Performance Computing techniques on a performance-critical part of the widely used lens modeling software LENSTOOL. We implement the algorithm once as a highly optimized CPU version and once with graphics card acceleration for a simple parametric lens model. In addition, we study the impact of finite machine precision on the lensing algorithm. While double precision is the default choice for scientific applications, we find that single precision can be sufficiently accurate for our purposes and lead to a big speedup. Therefore we develop and present a mixed precision algorithm which only uses double precision when necessary. We measure the performance of the different implementations and find that the use of High Performance Computing Techniques dramatically improves the code performance both on CPUs and GPUs. Compared to the current LENSTOOL implementation on 12 CPU cores, we obtain speedup factors of up to 170. We achieve this optimal performance by using our mixed precision algorithm on a high-end GPU which is common in modern supercomputers. We also show that these techniques reduce the energy consumption by up to 98%. Furthermore, we demonstrate that a highly competitive speedup can be reached with consumer GPUs. While they are an order of magnitude cheaper than the high-end graphics cards, they are rarely used for scientific computations due to their low double precision performance. However, our mixed precision algorithm unlocks their full potential. Consequently, the consumer GPU delivers a speedup which is only a factor of four lower than the best speedup achieved by a high-end GPU.

1. Introduction

The ΛCDM cosmology standard model describes our universe with great precision, but it also introduces two unknown quantities, Dark Energy and Dark Matter. They dominate the energy density of the universe (e.g., Ade et al., 2016), but their physical nature has so far remained elusive. Consequently their study is one of the prime targets of cosmological research.

Strong gravitational lensing is a unique tool for cosmology, as it is sensitive to the total (baryonic and dark) matter density and thus it probes Dark Matter directly (see e.g. Kneib and Natarajan, 2011; Schneider et al., 2006, for reviews). Its application has led to constraints on cosmological parameters (e.g., Jullo et al., 2010; Bonvin et al., 2017) and the Dark Matter self-interaction cross-section (e.g., Randall et al., 2008; Bradač et al., 2008). In addition, the magnification effect of a strong gravitational lens can be used to study the early universe and to constrain its reionization (e.g., Atek et al., 2015; Ishigaki et al., 2015). Strong lensing requires deep, high-resolution data and indeed the field has prospered thanks to programs like the Cluster Lensing And Supernova survey with *Hubble* (CLASH, Postman et al., 2012) and the *Hubble* Frontier Fields (HFF, Lotz et al., 2017).

Future missions like Euclid, the *James Webb Space Telescope (JWST)*, the Large Synoptic Survey Telescope (LSST), and the *Wide Field Infrared Survey Telescope (WFIRST)* will provide a large amount of excellent data sets for lensing. These will enable the lensing community to further push the boundaries of cosmological knowledge. This, however, will only be feasible if we are able to efficiently harvest the wealth of information available in the data. This will be a challenge, e.g. due to the amount of data available or the high quality of the data, which permits the

* Corresponding author.
E-mail address: christophernstrerne.schaefer@epfl.ch (C. Schäfer).

https://doi.org/10.1016/j.ascom.2019.100340
2213-1337/

creation of lens models with a high level of detail and precision, but also requires more computing time. Gravitational lensing software and pipelines will have to be ready to process these data sets in a reasonable amount of time.

Therefore we are currently redesigning the strong lensing software LENSTOOL[1] (Jullo et al., 2007; Kneib et al., 1996). LENSTOOL has been successfully used to model many strong lensing galaxy clusters with high precision (see e.g. Jauzac et al., 2014, 2015; Limousin et al., 2016, for recent lens models) and has been serving the lensing community for more than two decades. In a recent comparison of strong lensing modeling software it has performed very well (Meneghetti et al., 2016). However, the HFF data sets provided the greatest number of lensing constraints so far and this posed a computing challenge for LENSTOOL. It took several weeks to compute a single HFF lens model and several different lens models from different priors are required to find the best fitting model.

The new version is designed to meet this computation challenge by using High Performance Computing (HPC) methods. The LENSTOOL algorithms are very well suited for massive parallelism and we employ this technique to accelerate the computations. While we focus on lensing by galaxy clusters, a recent publication by Tessore et al. (2016) has shown that massive parallelism holds also great promise for the modeling of galaxy lenses. In this paper, we discuss the central lensing algorithm of LENSTOOL and in particular the performance-critical computation of deflection potential gradients. We have implemented the gradient computation algorithm using two different hardware types in order to be able to compare performance. The first version is a highly optimized and parallelized CPU code and the second version uses Graphics Processing Unit (GPU) acceleration.

During the development phase, we have asked ourselves the question: Can we do even better by using single precision instead of the commonly used double precision? The computing power of both CPUs and GPUs is higher for single precision (see e.g. Eijkhout et al., 2016; Besl, 2013), so we can expect a significant performance improvement. The downside is that this might lead to an error in our results if single precision is not precise enough for our computations. Therefore we use error propagation to compute the impact of single precision on the results of the central lensing algorithm. In addition, we measure and compare the single and double precision performance of both CPU and GPU implementations.

The paper is organized as follows: Section 2 gives a concise introduction to strong gravitational lensing and the LENSTOOL algorithms. It also presents the CPU and GPU implementations. Section 3 introduces the single and double precision floating-point representations and investigates if single precision is precise enough for our computations. We present and compare the performance measurements of the single and double precision CPU and GPU implementations in Section 4. We discuss our results in Section 5 and conclude in Section 6.

2. Accelerating lensing with massive parallelism

2.1. Strong gravitational lensing

Galaxies and galaxy clusters are so dense that they locally deform space–time. As a result, they can act as a lens for background objects, which are magnified and distorted or even multiply imaged. Lensing also changes the locations at which we observe the lensed images on the sky so that they are typically not coincident with the locations at which we would observe the background sources in the absence of lensing. In practice, we can only observe the lensed images of a background source, but not the background source itself. However, the position of the background source on the sky can be calculated with the lens equation (see e.g. the reviews Kneib and Natarajan, 2011; Bartelmann and Schneider, 2001, for a derivation),

$$\boldsymbol{\beta} = \boldsymbol{\theta} - \boldsymbol{\alpha}(\boldsymbol{\theta}), \tag{1}$$

where the two dimensional vectors $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, and $\boldsymbol{\alpha}$ describe respectively the location of the source in the source plane, the location of the lensed image in the image plane, and the scaled deflection angle. Note that these quantities are angles. In the case of multiple images, the lens equation has more than one solution $\boldsymbol{\theta}$ for a fixed value of $\boldsymbol{\beta}$ (e.g., Bartelmann and Schneider, 2001). The lens equation is derived under the assumption that we have only one lens, that the gravitational field is weak enough so that the field equations of General Relativity can be linearized, that we can use the Born approximation, and that the physical extent of the lens is small compared to the angular diameter distances between observer and lens, D_{OL}, and lens and source, D_{LS}.

The background objects are typically extended sources like galaxies. The shape of the lensed images will differ from the shape of the source, since the light coming from the object at coordinate $\boldsymbol{\beta}'$ will be lensed slightly differently than the light coming from the object at coordinate $\boldsymbol{\beta}''$ (e.g., Bartelmann and Schneider, 2001). Therefore we can use the lens equation to compute $\boldsymbol{\theta}$ for each coordinate $\boldsymbol{\beta}$ of the object and thus the shape of the lensed image due to isotropic and anisotropic distortion.

The scaled deflection angle $\boldsymbol{\alpha}$ is the gradient of the deflection potential ψ,

$$\boldsymbol{\alpha} = \nabla\psi, \tag{2}$$

$$\psi(\boldsymbol{\theta}) = \frac{1}{\pi}\int_{\mathbb{R}^2} \mathrm{d}^2\theta'\ \kappa(\boldsymbol{\theta}')\ln|\boldsymbol{\theta} - \boldsymbol{\theta}'|, \tag{3}$$

and ψ depends on the dimensionless projected surface mass density κ,

$$\kappa(\boldsymbol{\theta}) = \frac{\Sigma(\boldsymbol{\theta})}{\Sigma_{\mathrm{crit}}}, \tag{4}$$

$$\Sigma_{\mathrm{crit}} = \frac{c^2}{4\pi G}\frac{D_{\mathrm{OS}}}{D_{\mathrm{OL}}D_{\mathrm{LS}}}, \tag{5}$$

where $\Sigma(\boldsymbol{\theta})$ is the projected surface mass density,

$$\Sigma(\boldsymbol{\theta}) = \int \mathrm{d}z\ \rho(\boldsymbol{\theta}, z), \tag{6}$$

and we defined the critical projected surface mass density Σ_{crit}. Here ρ is the mass density, c is the speed of light, G is the gravitational constant, and D_{OS} is the angular diameter distance between observer and source. We can see from these equations that $\boldsymbol{\alpha}$ and thus the strength of the lensing effect depend on the projected surface mass density. Therefore lensing probes the total surface mass density of the lens, including baryonic and Dark Matter components.

The value of κ is a good indicator to distinguish the so-called "weak" and "strong" lensing regimes. In the case of weak lensing, the lensed image is only slightly distorted, while in the case of strong lensing, the image is distorted strongly enough that multiple images appear. A mass distribution which has $\kappa \geq 1$ somewhere produces multiple images for some source positions $\boldsymbol{\beta}$ (e.g., Bartelmann and Schneider, 2001). In the case of cluster lensing, the strong lensing area and thus the multiple images are typically located in the central regions of the cluster, where the projected surface mass density is large enough (e.g., Kneib and Natarajan, 2011).

We will illustrate gravitational lensing with an example. We will look at a simple lens model, the Singular Isothermal Sphere

[1] Open source software publicly available at https://projets.lam.fr/projects/lenstool/wiki.

(SIS), which we will use for the remainder of this paper as it has a relatively simple mathematical expression and is thus very instructive. The projected surface mass density is

$$\Sigma(\boldsymbol{\theta}) = \frac{\sigma_v^2}{2GD_{\mathrm{OL}}|\boldsymbol{\theta}|}, \tag{7}$$

where σ_v is the line-of-sight velocity dispersion of the "particles" (e.g. galaxies in a galaxy cluster), which are assumed to be in virial equilibrium (e.g., Bartelmann and Schneider, 2001). Thus we have

$$\kappa(\boldsymbol{\theta}) = \frac{\theta_E}{2|\boldsymbol{\theta}|}, \tag{8}$$

$$\theta_{\mathrm{E}} = 4\pi\left(\frac{\sigma_v}{c}\right)^2 \frac{D_{\mathrm{LS}}}{D_{\mathrm{OS}}}, \tag{9}$$

where we defined the Einstein deflection angle θ_{E}. Using Eqs. (2) and (3), we find that the magnitude of the scaled deflection angle is constant,

$$|\boldsymbol{\alpha}| = \theta_{\mathrm{E}}. \tag{10}$$

We see that the lens equation has infinitely many solutions for $\boldsymbol{\beta} = \mathbf{0}$, namely each point on the circle with radius θ_{E}. Therefore a background source at this location will be strongly lensed into a perfect Einstein ring.

2.2. Strong lensing algorithm

2.2.1. Overview

LENSTOOL models strong lensing galaxy clusters by using parametric models of the large-scale cluster halos and the galaxy-scale halos. In a typical merging cluster, we have two large-scale halos and hundreds of galaxy halos. Depending on the chosen parametric model, we have several free parameters such as x and y position, velocity dispersion, etc. for each halo. It is possible to constrain the range of the free parameters or to reduce their number, e.g. by assuming a scaling relation like the Faber–Jackson relation (Faber and Jackson, 1976) for galaxy-scale halos (Natarajan et al., 1998). Nevertheless, the best lens model will still be hidden in a massive, high dimensional parameter space. LENSTOOL uses BayeSys3,[2] a Bayesian Markov Chain Monte Carlo (MCMC) software package, to sample this parameter space (see Jullo et al., 2007, for a detailed description). For each parameter combination probed by the MCMC, LENSTOOL computes the goodness of fit of the corresponding lens model given the observational data. It does this by modeling the lens with the given set of parameters and, using this model, lensing the observed multiple images into the source plane and subsequently back into the image plane, see Fig. 1. If the probed lens model is close to the true matter distribution, the re-lensed multiple image positions will be close to the observed multiple image positions and the goodness of fit parameter

$$\chi^2 = \sum_i \sum_j \frac{(x_{\mathrm{obs},ij} - x_{ij})^2}{\sigma_{ij}^2} \tag{11}$$

will be small (Jullo et al., 2007). We denote the observed position of multiple image j of multiple image system i with $x_{\mathrm{obs},ij}$, the re-lensed position with x_{ij}, and the error budget of the position with σ_{ij}. Since the parameter space probed by the MCMC is massive, it typically takes several weeks of computation time to find the best model for lenses with HFF-like data.

There are two ways to speed up the computation. The first is to speed up the MCMC, e.g. by parallelizing it. The second way is to speed up the χ^2 computation. In this paper, we will focus on accelerating a crucial part of it, the gradient computation. Since we will have to take a very precise look at the algorithm when we compute the impact of single and double precision on its result, we will now present a detailed description.

[2] Publicly available at http://www.inference.org.uk/bayesys/.

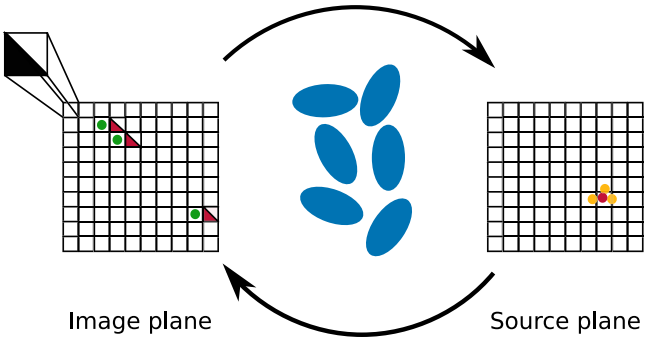


Fig. 1. Lenstool computes the multiple image positions predicted by a lens model (red triangles, image plane). In the first step, it lenses the observed multiple images (green dots, image plane) onto their respective predicted sources (yellow dots, source plane) and computes their barycenter (red dot, source plane). In the second step, it decomposes the image plane pixels into triangles and lenses each triangle into the source plane. Every time that the source plane triangle includes the barycenter, a predicted multiple image is found. If the lens model is close to the true model, these re-lensed images will be located very close to the observed images. Note that image plane pixels lensed into the source plane will typically be distorted due to the strong lensing effect. We do not show this effect to keep the figure simple. As a result of this distortion, squares are not always mapped onto squares and we thus have to partition the pixels into triangles (top left corner, image plane), which are always mapped onto triangles. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.2.2. Gradient computation in the χ^2 algorithm

Before we present the χ^2 algorithm, we reformulate the lens Eq. (1) by introducing

$$\Psi = \frac{D_{\mathrm{OS}}}{D_{\mathrm{LS}}}\psi \tag{12}$$

and making the gradient dependence explicit:

$$\begin{aligned}\beta_1 &= \theta_1 - \frac{D_{\mathrm{LS}}}{D_{\mathrm{OS}}}(\nabla\Psi(\boldsymbol{\theta}))_1,\\ \beta_2 &= \theta_2 - \frac{D_{\mathrm{LS}}}{D_{\mathrm{OS}}}(\nabla\Psi(\boldsymbol{\theta}))_2.\end{aligned} \tag{13}$$

As a result, we only have to compute the constant $D_{\mathrm{LS}}/D_{\mathrm{OS}}$ once instead of for every image pixel. Note that the deflection potential at position $\boldsymbol{\theta}$ is a superposition of all cluster-scale and galaxy-scale deflection potentials ψ_{cluster} and ψ_{galaxy},

$$\psi(\boldsymbol{\theta}) = \sum \psi_{\mathrm{cluster}}(\boldsymbol{\theta}) + \sum \psi_{\mathrm{galaxy}}(\boldsymbol{\theta}), \tag{14}$$

(see e.g. Jullo et al., 2007) and as a result we have

$$\nabla\Psi(\boldsymbol{\theta}) = \sum \nabla\Psi_{\mathrm{cluster}}(\boldsymbol{\theta}) + \sum \nabla\Psi_{\mathrm{galaxy}}(\boldsymbol{\theta}). \tag{15}$$

We see that the lens equation is computationally cheap to evaluate once the total gradient $\nabla\Psi$ is known. The computation of $\nabla\Psi$, however, involves potentially complicated gradient calculations for hundreds of potentials and as we will see in the next paragraph, it has to be computed for every pixel in our image. The *Hubble Space Telescope Advanced Camera for Surveys* (HST ACS) produces images with 4096 × 4096 pixels at a pixel scale of ≈ 0.05 arcsec/pixel (Avila et al., 2017), which we can typically upsample to 0.03 arcsec/pixel (Lotz et al., 2017), so that HFF images have a total of ≈ 6730 × 6730 pixels ≈ 45 million pixels. This shows that the computation of $\nabla\Psi$ is computationally expensive and an excellent target for speedup with HPC parallelism methods.

The χ^2 computation is now performed as follows. We compute $\nabla\Psi$ for each pixel of the image plane. Then we loop over each multiple image j in each multiple image system i. For each multiple image, we use Eq. (13) to compute the source coordinates, $\beta_{ij,1}$ and $\beta_{ij,2}$. Subsequently, we determine the barycenter of the sources of a given multiple image system i. If we are close to the true lens model, all multiple images will be mapped onto approximately the same source location, but in general the locations of the predicted sources can differ substantially, which makes it necessary to use the barycenter. In the next step, we re-lens the barycenter back into the image plane to obtain the locations of the multiple images predicted by the lens model. However, the lens equation cannot easily be inverted, so we have to find the locations in a different way. First, we divide each pixel in the image plane into two triangles, see Fig. 1. We do this because lensing always maps triangles onto triangles, but not squares onto squares. Second, we lens each triangle into the source plane by using Eq. (13) and we check if the barycenter is inside this triangle in the source plane. If it is, a predicted multiple image location in the image plane is found. Once we have found the locations of all predicted multiple images for all multiple image systems, we compute the χ^2 according to Eq. (11).

The gradient calculations will naturally differ for different chosen parametric models. As an example, we present the gradient computation for a generalized form of the SIS, the pseudo-elliptical SIS (henceforth called SIE), in Algorithm 1. It is necessary to generalize the parametric model, as we want to use this algorithm to model any SIS lens configuration by simply choosing the appropriate number of lenses and parameter values. We expand our treatment of the SIS in Section 2.1 by following the procedure in Golse and Kneib (2002). We introduce the pseudo-ellipticity of the deflection potential, ϵ, and the coordinate system

$$R = \sqrt{\theta_{1,\epsilon}^2 + \theta_{2,\epsilon}^2},$$
$$\phi = \arctan\left(\frac{\theta_{2,\epsilon}}{\theta_{1,\epsilon}}\right), \tag{16}$$

with

$$\theta_{1,\epsilon} = \sqrt{a_{1,\epsilon}}\ \theta_1,$$
$$\theta_{2,\epsilon} = \sqrt{a_{2,\epsilon}}\ \theta_2, \tag{17}$$
$$a_{1,\epsilon} = 1 - \epsilon,$$
$$a_{2,\epsilon} = 1 + \epsilon. \tag{18}$$

Note that we call ϵ a pseudo-ellipticity, because the resulting elliptical shapes will only correspond to ellipses with classical ellipticity $\epsilon' = 1 - b/a$, where a and b are the semi-major and semi-minor axes of the ellipse, for small values of ϵ (Golse and Kneib, 2002). Therefore we assume in the following $\epsilon \ll 1$. The advantage of using a pseudo-elliptical parametric model is that it leads to relatively simple analytic expressions of the derived lensing quantities (Golse and Kneib, 2002). Now we can simply calculate the values of the pseudo-elliptical deflection potential ψ_ϵ at location $\boldsymbol{\theta}$ by using the relation (Golse and Kneib, 2002)

$$\psi_\epsilon(\boldsymbol{\theta}) = \psi(R, \phi), \tag{19}$$

and analogous for Ψ_ϵ. The resulting pseudo-elliptical shape is stretched along the θ_1-axis, so that we have $\Phi = 0$, where Φ is the counter-clockwise angle between the semi-major-axis and the θ_1-axis. Algorithm 1 extends this approach to potentials with $\Phi \neq 0$ by using rotations. We obtain the following equations for the scaled deflection angle (Golse and Kneib, 2002),

$$\alpha_{1,\epsilon}(\boldsymbol{\theta}) = |\boldsymbol{\alpha}(R)|\ \sqrt{a_{1,\epsilon}}\ \cos(\phi),$$
$$\alpha_{2,\epsilon}(\boldsymbol{\theta}) = |\boldsymbol{\alpha}(R)|\ \sqrt{a_{2,\epsilon}}\ \sin(\phi). \tag{20}$$

We can now combine Eqs. (2), (9), (10), (12), and (20) to derive the gradient expressions for the SIE,

$$\left(\nabla\Psi_\epsilon\right)_1 = (1-\epsilon)\ b_0\ \frac{\theta_1}{R},$$
$$\left(\nabla\Psi_\epsilon\right)_2 = (1+\epsilon)\ b_0\ \frac{\theta_2}{R}, \tag{21}$$

where we introduced the parameter

$$b_0 = 4\pi\left(\frac{\sigma_v}{c}\right)^2. \tag{22}$$

The presented equations for the SIE always reduce to the previously presented equations for the spherical SIS for $\epsilon = 0$.

Algorithm 1 Compute $\nabla\Psi_\epsilon$ in each image pixel for a SIE

Require: $\boldsymbol{\theta}_{\text{center}}$, b_0, ϵ, Φ $\forall$ SIE lenses, image **I**
Output: $\nabla\Psi_\epsilon$ $\forall$ pixels $(\theta_1, \theta_2) \in$ **I**
Procedure GRADIENT(**I**,{$\boldsymbol{\theta}_{\text{center},i}$, $b_{0,i}$, ϵ_i, Φ_i}):
for all $(\theta_1, \theta_2) \in$ **I** **do**
  for $i \in$ SIE lenses **do**
    $\Delta\theta_{1,i} \leftarrow \theta_1 - \theta_{\text{center},i,1}$
    $\Delta\theta_{2,i} \leftarrow \theta_2 - \theta_{\text{center},i,2}$
    $\Delta\theta'_{1,i} \leftarrow \Delta\theta_{1,i}\cos(\Phi_i) + \Delta\theta_{2,i}\sin(\Phi_i)$
    $\Delta\theta'_{2,i} \leftarrow \Delta\theta_{2,i}\cos(\Phi_i) - \Delta\theta_{1,i}\sin(\Phi_i)$
    $R_i \leftarrow$ **sqrt**$\left((\Delta\theta'_{1,i})^2(1-\epsilon_i) + (\Delta\theta'_{2,i})^2(1+\epsilon_i)\right)$
    $(\nabla\Psi_\epsilon)_{1,i} \leftarrow (1-\epsilon_i)\ b_{0,i}\ \Delta\theta'_{1,i}/R_i$
    $(\nabla\Psi_\epsilon)_{2,i} \leftarrow (1+\epsilon_i)\ b_{0,i}\ \Delta\theta'_{2,i}/R_i$
    $(\nabla\Psi_\epsilon)'_{1,i} \leftarrow (\nabla\Psi_\epsilon)_{1,i}\cos(-\Phi_i) + (\nabla\Psi_\epsilon)_{2,i}\sin(-\Phi_i)$
    $(\nabla\Psi_\epsilon)'_{2,i} \leftarrow (\nabla\Psi_\epsilon)_{2,i}\cos(-\Phi_i) - (\nabla\Psi_\epsilon)_{1,i}\sin(-\Phi_i)$
  end for
  $(\nabla\Psi_\epsilon)_1 \leftarrow \sum_i (\nabla\Psi_\epsilon)'_{1,i}$
  $(\nabla\Psi_\epsilon)_2 \leftarrow \sum_i (\nabla\Psi_\epsilon)'_{2,i}$
end for
return $\{\nabla\Psi_\epsilon\}$

2.3. CPU and GPU implementations

We developed two versions of the gradient computation for Benchmark and comparison purposes, one for CPUs and one for GPUs.

2.3.1. CPU

The performance-optimized CPU version of the gradient computation is coded in C++[3] using the following techniques. First, we structure our data in the Structures of Arrays (SoA) format instead of the Arrays of Structures (AoS) format.

In the SoA format the data is stored as a structure of arrays, each array possessing the same type of information. The more human intuitive AoS format stores the data as an array comprised of multiple different data structures. Using the SoA format results in the data occupying contiguous parts of the memory, which is usually beneficial for vectorized computations (e.g., Eijkhout et al., 2016; Besl, 2013). Second, we use Advanced Vector Extensions (AVX) technology available on the latest CPU generations to harvest their built-in vectorization potential. AVX (Advanced Vector Extensions) is an instruction set architecture that implements SIMD (Same Instruction Multiple Data, Eijkhout et al., 2016) Data Level parallelism on x86 CPUs. It allows compatible CPUs to execute the same operations on multiple data paths

[3] C++ is a programming language standardized by the International Organization for Standardization, public website: https://isocpp.org/.

(vectors) at the same time, thus increasing arithmetic throughput accordingly. For instance, AVX can operate on 4 DP numbers at once and AVX512 on 8. The main disadvantages of AVX (and SIMD in general) is that not all algorithms are a priori vectorizable, and vectorizable codes are required to use specific data structures (e.g. Structure Of Array) in order to be fully efficient (Fig. 2). Implementing these data structures can be cumbersome in the overall code design. Third, we parallelize the computation using Open Multi-Processing (OpenMP)[4] on the outermost loop of Algorithm 1. Each core of the multi-core CPU will now work on computing the total gradient for its assigned pixel and thus we compute the gradients for several pixels in parallel.

2.3.2. GPU

We implement the GPU version of the algorithm with CUDA.[5] First, we structure our data again in the SoA format. Second, we use the massively parallel architecture of GPUs to parallelize the gradient computation. Modern GPUs have many Streaming Multiprocessors (SM), which in turn consist of many Streaming Processors (SP), so the total amount of processor cores is computed by multiplying the two (e.g., Eijkhout et al., 2016). The number of cores available depends on the GPU model, for example the Nvidia Tesla P100 (henceforth called P100) possesses 3584 cores for single precision computations (Nvidia Corporation, 2016). In addition, GPUs are designed to be extremely efficient at switching between threads, where all threads in a single block of threads execute the same instruction (Eijkhout et al., 2016). Therefore we can effectively use many more threads than we have GPU cores. Different blocks of threads can be processed independently. This GPU parallelism is called Single Instruction Multiple Thread (SIMT) (Eijkhout et al., 2016). We use GPU threads to parallelize the outermost loop of Algorithm 1. Each GPU thread computes the total gradient for its assigned pixel. Therefore we can compute the gradients for thousands of pixels simultaneously.

3. Finite machine precision errors in strong lensing

3.1. Single and double precision

Modern computers usually store real numbers in the IEEE 754 single precision floating-point representation (henceforth called SP) or the IEEE 754 double precision floating-point representation (henceforth called DP) (Institute of Electrical and Electronics Engineers, 2008, see e.g. Goldberg (1991) for an overview of floating-point arithmetic). A real number $x \in \mathbb{R}$ in decimal representation is thus stored in a binary format,

$$x = \sigma \times \bar{x}_2 \times 2^e, \tag{23}$$

where the integer e is the exponent, the sign σ equals $+1$ or -1, and $\bar{x}_2$ is a binary number satisfying $(1)_2 \leq \bar{x}_2 < (10)_2$ (Institute of Electrical and Electronics Engineers, 2008). Note that the binary number $\bar{x}_2$ consists of several integer digits $d \in \{0, 1\}$, i.e. $\bar{x}_2 = d_0.d_1d_2 \ldots d_{p-1}$. Left of the decimal point, d_0 corresponds to the binary integer representation. To the right d_i for $i > 0$ are used for the binary fraction representation with $d_i = 2^{-i}$. In the remainder of this paper we will denote the binary format by using the subscript 2, so $(1)_2$ and $(10)_2$ correspond to the numbers 1 and 2 in decimal representation. For example, the number 2.25 would correspond to $\sigma = +1$, $\bar{x}_2 = (1.001)_2$, and $e = (1)_2$.[6] The number of digits in $\bar{x}_2$ is called the precision p of the representation. According to IEEE 754, SP has a precision of $p = 24$ digits and an exponent $-126 \leq e \leq 127$, while DP has $p = 53$ and $-1022 \leq e \leq 1023$. SP values are stored using 4 bytes (= 32 bits) and DP values using 8 bytes (= 64 bits) (Institute of Electrical and Electronics Engineers, 2008). As a result, DP can store a number x with higher accuracy than SP, but this comes at the price of increased memory consumption and usually also reduced computing performance (e.g., Besl, 2013; Eijkhout et al., 2016).

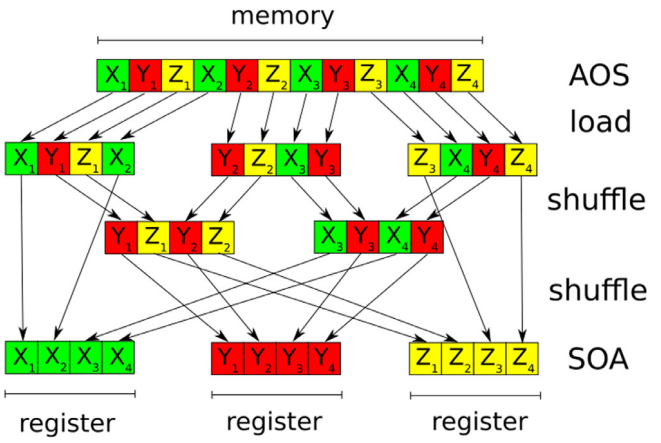


Fig. 2. This illustration shows how data stored in the Structures of Arrays (SoA) and Arrays of Structures (AoS) formats is loaded into registers. The parameters X, Y, and Z are part of their respective data sets T_1, T_2, T_3, and T_4. A CPU core with AVX technology uses registers to process 4 parameters simultaneously, but this requires a homogeneous memory layout. Data stored in the SoA format provides this homogeneous memory layout without any additional operation and can be processed after being loaded into the registers. Data stored in the AoS format is first loaded into the registers and subsequently rearranged by shuffling the data between the registers. These shuffle operations consume time and thus lead to lower performance.

Both DP and SP have only a limited amount of memory available and thus their accuracy is limited. We define the machine epsilon ϵ as the difference between 1 and the next larger number that can be stored using the given representation (Eijkhout et al., 2016). For SP and DP we thus have respectively $\epsilon = 2^{-23} \approx 1.2 \times 10^{-7}$ and $\epsilon = 2^{-52} \approx 2.2 \times 10^{-16}$. These errors are so small that they might seem unimportant at first, but they will be magnified by the different computing operations performed in the course of an algorithm, so that they can become very large and relevant once the final result is obtained.

To illustrate this point, we now look at a hypothetical calculator.[7] For simplicity, it does not use SP or DP, but a decimal number representation with 6 digits precision and no exponent. We compute a relatively simple function, $f(x) = x \times (\sqrt{x+1} - \sqrt{x})$. For $x = 50{,}000$, the result from the hypothetical calculator is 100, while the true result is 111.8, so we have a relative error of more than 10%. To understand this behavior, we take a look at the different steps which the calculator has to perform. It computes $\sqrt{50001}$ and rounds the result to 6 digits (result: 223.609) and then it repeats these steps for $\sqrt{50000}$ (result: 223.607). Therefore we have two rounding errors, but they are very small. However, now the calculator subtracts two almost

[4] OpenMP is an application programming interface managed by the non-profit OpenMP Architecture Review Board, public website: http://www.openmp.org.

[5] CUDA is a parallel computing platform and programming model for general computing on GPUs managed by Nvidia Corporation, public website: https://developer.nvidia.com/cuda-zone.

[6] In practice, the leading bit of $\bar{x}_2$ would be implicit and e would be stored as a biased exponent, but we can ignore such intricacies here to simplify the presentation.

[7] This illustration is inspired by an example in the lecture notes of Catalin Trenchea, available online at http://www.math.pitt.edu/~trenchea/math1070/MATH1070_2_Error_and_Computer_Arithmetic.pdf.

Table 1
Theoretical maximum computing performance for our used CPU and GPU models. These values can only serve as a rough indicator of expected performance, as the real application performance will depend on many parameters such as the used algorithm and its implementation. We list the base frequency for the CPU while we use the boost frequency for the GPUs, as the CPU typically reaches the boost frequency only on a few cores and not on all cores simultaneously. We compute the CPU maximum computing performance using the following formula: Two operations per cycle ×frequency ×AVX vectorization ×number of cores (Besl, 2013). Note that the AVX factor for SP is two times larger than for DP. We use the same formula, but without the AVX factor, for GPUs. Graphics cards have a different number of cores for SP and DP computations and thus a different maximum performance. The number of GPU cores listed in the table is for SP computations. Due to the lower number of DP cores, the Nvidia GTX 1080 Ti's GP102 GPU has thirty-two times less performance in DP computations than in SP (e.g., Harris, 2016), while the P100's GP100 GPU and the V100's GV100 GPU are two times slower (Nvidia Corporation, 2016, 2017b). The number of cores and the frequencies are taken from Intel Corporation (2014) and Nvidia Corporation (2017a,b, 2016).

	Intel Xeon E5-2680 v3 12 cores, 2.50 GHz	Nvidia GTX 1080 Ti 3584 cores, 1582 MHz	Nvidia P100 3584 cores, 1480 MHz	Nvidia V100 5120 cores, 1530 MHz
Double precision	240 GFLOPS	354 GFLOPS	5304 GFLOPS	7833 GFLOPS
Single precision	480 GFLOPS	11340 GFLOPS	10609 GFLOPS	15667 GFLOPS

equal numbers to obtain 000.002, so only the last number of the result is a significant digit. We have lost a lot of accuracy which we cannot recover. The subsequent multiplication does not increase the error, but it propagates it into the final result. This example illustrates that even with the high precision available in modern computers, the result of a sufficiently long and complex algorithm can be significantly affected by the chosen number representation.

DP permits a much higher accuracy than SP and therefore it is tempting to simply use it for all computations. However, this accuracy comes at the price of computing performance. As shown in Table 1, this is particularly true for GPUs. While the theoretical maximum computing performance of a modern CPU decreases by a factor of two, the peak performance of a consumer GPU like the Nvidia GeForce GTX 1080 Ti (henceforth called GTX) drops by two orders of magnitude. This is a significant problem for GPU-accelerated scientific software, where SP is often not accurate enough. To ameliorate this issue, graphics card manufacturers introduced new hardware specifically designed to improve the DP performance. The P100 and its recently released successor, the Nvidia Tesla V100 (henceforth called V100) achieve half of their SP performance when using DP. However, these special purpose GPUs are much more expensive than regular consumer GPUs like the GTX, typically by an order of magnitude. Table 1 shows that the SP performance of a high-end consumer GPU is comparable to the SP power of the special purpose GPUs. Thus, if it is possible to use SP instead of DP in our lensing algorithm, we would not only significantly increase the code performance on both CPUs and GPUs, but we might also be able to achieve a close to optimal performance with relatively cheap hardware.

3.2. Computing finite precision errors for strong lensing

We will now show that using SP in Algorithm 1 is accurate enough for a large fraction of the image pixels. We restrict ourselves again to the SIE model. It is possible to generalize these results to other parametric models, but the fraction of the image for which SP is accurate enough will vary and has to be computed for each model independently.

The lens Eq. (13) maps the pixels in the image plane onto pixels in the source plane. We assume a HFF pixel size of 0.03 arcsec and we maximize the lensing effect by using $D_{\mathrm{LS}}/D_{\mathrm{OS}} = 1$. As a result, the lens equation is now a simple subtraction of $\nabla\Psi_\epsilon(\theta_1, \theta_2)$. We now look at an observed multiple image in the image plane. Note that our ability to locate the multiple image is observationally constrained by the size of the image pixels, so there is an observational error budget on the image location of half a pixel. In addition, the algorithm lenses both the triangular pixel and the image position into the source plane. It is possible that their respective errors due to machine precision have the same magnitude but the opposite sign, and therefore the error budget shrinks by another factor of two. As a result, the value of $\nabla\Psi_\epsilon$ can be considered accurate enough if the error E is smaller than a quarter of a pixel. Thus our upper limit for the gradient error is $E_i \leq 7.5 \times 10^{-3}$ arcsec, where $i = 1, 2$.

However, this error budget does not yet account for the magnification effect of strong lensing. Background sources and distance scales appear magnified when they are strongly lensed and consequently distance scales in the image plane like pixel sizes will be de-magnified when they are mapped into the source plane. The resulting error budget for $\nabla\Psi_\epsilon$ becomes thus $E_i \leq 7.5 \times 10^{-3}$ arcsec$/M_i$, where M_i is the magnification along the θ_i-axis.

In Appendix we derive an upper bound for the error of $\nabla\Psi_\epsilon$ due to finite machine precision. We assume that the lens is a strong lensing cluster modeled with two cluster-scale SIE halos. The SP upper error bound along the θ_i-axis is $\Delta(\nabla\Psi_\epsilon) \leq 2.3 \times 10^{-3}$ arcsec if we use the following approach. As discussed in Appendix, we compute the gradients with SP except in pixel grids of 400×400 pixels around cluster-scale halos and 20×20 pixels around galaxy-scale halos, where we use DP. In these regions the SP error would be so large that it could alter the result. This corresponds to approximately 1% of all image pixels. As a result, SP is accurate enough for each of the remaining 99% of the image pixels if the respective magnification along both θ_i-axes is $M_i \leq 3.26$. In strong lensing, we typically measure the magnification of the area of a multiple image and not the magnification along an axis. The measured values are typically single digits (see e.g. Jauzac et al., 2015, for magnification values of a HFF cluster). While these values cannot easily be converted to axis-magnifications due to the typically arc-like shape of strongly magnified images, they strongly suggest that SP will be accurate enough for a large fraction of the image. However, strong lensing clusters have critical lines where the magnification diverges. In the case of the SIS, this critical line is the Einstein ring. While the magnification does not become infinite in practice (see e.g. Bartelmann and Schneider, 2001, for a detailed discussion), it can become very large and thus SP will no longer be accurate enough. Consequently, we can use SP for a large fraction of the image, but we also need to implement a mechanism which ensures that we compute the gradients with DP whenever SP is not enough due to high magnification.

3.3. Fixing the missing accuracy close to critical lines

We add the missing accuracy close to critical lines as follows. First, we compute $\nabla\Psi_\epsilon$ for each pixel in the image using the approach presented in the previous subsection. Second, we compute for each pixel

$$\delta_1(\theta_1, \theta_2) = (\nabla\Psi_\epsilon)_1(\theta_1, \theta_2) - (\nabla\Psi_\epsilon)_1(\theta_1 - \Delta x, \theta_2),$$
$$\delta_2(\theta_1, \theta_2) = (\nabla\Psi_\epsilon)_2(\theta_1, \theta_2) - (\nabla\Psi_\epsilon)_2(\theta_1, \theta_2 - \Delta x), \tag{24}$$

which is computationally cheap because we have already computed the gradient values for all pixels. For the HST ACS, we

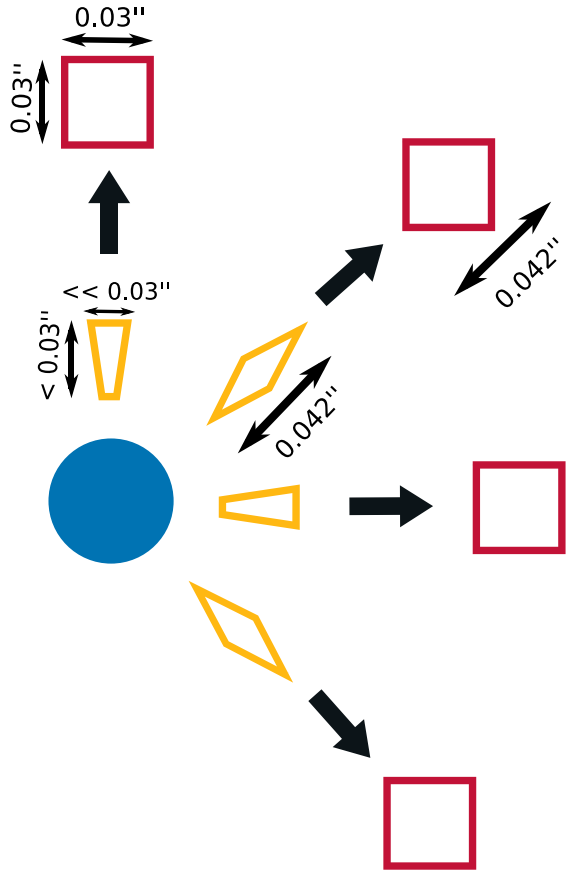


Fig. 3. The source plane pixels (yellow) are greatly distorted with respect to the corresponding regular image plane pixels (red). This example shows the distortions caused by a single symmetrical SIS lens (blue) for angles of −45, 0, 45, and 90 degrees. Note that the greatest distortion occurs perpendicular to the lensing direction, but small lensing effects can also occur alongside this direction, as the example for the 90° angle shows. The magnitude of this effect is typically negligible compared to the perpendicular distortion. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

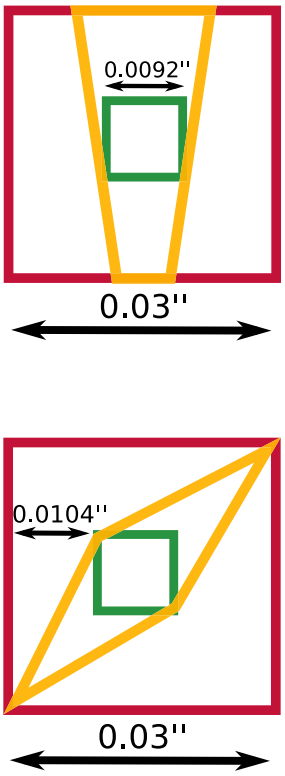


Fig. 4. The values of $\nabla\Psi_\epsilon$ must be recomputed in DP if the distorted and de-magnified source plane pixels (yellow) become smaller than the error due to finite machine precision (green box). The regular image plane pixel is overplotted in red. The top figure illustrates the lensing example with an angle of 90 degrees shown in Fig. 3 and the bottom figure demonstrates the example with an angle of 45 degrees. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

have a pixel height and width $\Delta x = 0.03$ arcsec. Note that δ_i corresponds to the change of the pixel length along the θ_i-axis due to lensing into the source plane. Third, we recompute $\nabla\Psi_\epsilon$ in DP for all pixels where

$$|0.03 \text{ arcsec} - \delta_i(\theta_1, \theta_2)| < 0.0092 \text{ arcsec} \tag{25}$$

for $i = 1, 2$, which implies that $M_i > 3.26$. We derive this condition by computing the pixel length in the source plane Δx_{source} along the β_1-axis,

$$\begin{aligned} |\Delta x_{\text{source},1}(\theta_1, \theta_2)| &= |\beta_1(\theta_1, \theta_2) - \beta_1(\theta_1 - \Delta x, \theta_2)| \\ &= |\Delta x - \delta_1(\theta_1, \theta_2)|, \end{aligned} \tag{26}$$

where we used the lens equation. An analogous relation holds for the β_2-axis. Note that taking the absolute value of Δx_{source} is necessary because lensing can change the image parity (see e.g. the review Kneib and Natarajan, 2011). For the assumed HFF pixel scale, the condition that $M_i > 3.26$ translates into $\Delta x_{\text{source},i} < 0.0092$ arcsec. The lensing effect along the θ_1- and θ_2-axis is shown in Fig. 3 and the criterion in Eq. (25) is illustrated in the top part of Fig. 4.

We can assume that each source is lensed along a chosen θ_i-axis, as this can be achieved by a simple change of the image plane coordinate system. However, the shape of the image plane pixels is not invariant under such a transformation, as Fig. 3 illustrates. Therefore we need two additional criteria. We define

$$\begin{aligned} \delta_3(\theta_1, \theta_2) &= (\nabla\Psi_\epsilon)_1(\theta_1, \theta_2) - (\nabla\Psi_\epsilon)_1(\theta_1, \theta_2 - \Delta x), \\ \delta_4(\theta_1, \theta_2) &= (\nabla\Psi_\epsilon)_2(\theta_1, \theta_2) - (\nabla\Psi_\epsilon)_2(\theta_1 - \Delta x, \theta_2), \end{aligned} \tag{27}$$

and we recompute $\nabla\Psi_\epsilon$ in DP if

$$|\delta_i(\theta_1, \theta_2)| > 0.0104 \text{ arcsec} \tag{28}$$

for $i = 3$ or $i = 4$. This case is illustrated in the bottom part of Fig. 4.

In summary, we compute $\nabla\Psi_\epsilon$ in SP everywhere except in small patches centered on the origin of each lens as described in Appendix and for the pixels where the criteria defined in Eqs. (25) and (28) hold. This is illustrated in Fig. 5.

4. Performance measurements

We implement both the GPU and the CPU version of the gradient computation twice, once in SP and once in DP. The respective versions are identical up to the change in precision. In addition, we implement the mixed precision algorithm for both types of hardware. In the first step, this algorithm computes the result for each pixel in SP. In the second step, it checks which results are not accurate enough and recomputes these with DP. For this purpose, the algorithm uses the criterions developed in the previous section. The GPU implementation of the mixed precision algorithm uses asynchronous computations and load balancing

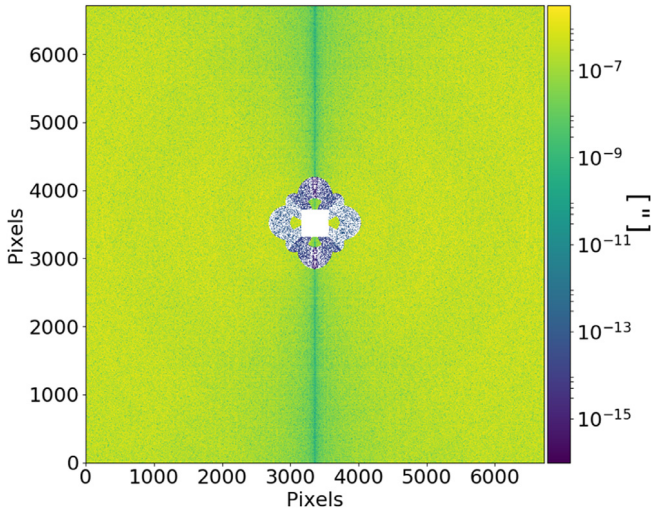


Fig. 5. Difference between the values of $(\nabla\Psi_\epsilon)_1$ computed with our mixed precision and DP algorithms for a single spherical cluster-scale SIS lens. The color white indicates a difference of zero. The green pixels are calculated with SP and the white and blue pixels are re-computed with DP. The error for every pixel is within the allowed error bounds. The rectangular patch around the lens center in which we use DP is clearly visible. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

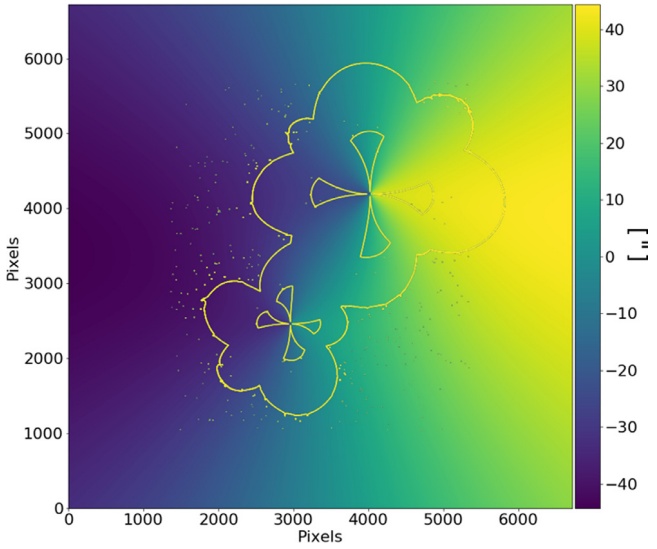


Fig. 6. Values of $(\nabla\Psi_\epsilon)_1$ computed in DP for the HFF-like galaxy cluster lens used for the performance benchmark. The yellow contours indicate the area in which at least one of the conditions shown in Eqs. (25) and (28) is triggered. The patches for the halo centers are not displayed. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

for the second step, i.e. the computation of the most expensive DP calculations can be dispatched asynchronously between the CPU and the GPU. We adjust the load balancing for the different GPU models. As Table 1 shows, this is particularly interesting for the GTX: The use of a hybrid CPU/GPU approach alleviates the very low DP performance of this card and drastically reduces the impact of the DP computations on the overall run time. From an implementation point of view, the load balancing split between CPU and GPU is performed manually according to an input parameter that depends on the floating point throughput ratio between each component. Auto-tuning this parameter would require some form of work-adjusting strategy that is beyond the scope of this paper.

In the next step, we want to measure the performance gain of using HPC methods in strong lensing. For this purpose, we measure the time which the different software implementations require to compute the gradient of a HFF-like cluster lens for each pixel of a *Hubble* image. We have repeated this measurement several times and find that the benchmark results are stable, i.e. they do not significantly vary in different runs with the same setup. We also compute the gradients with the current LENSTOOL software, which serves as a reference. We assume a ΛCDM cosmology with $H_0 = 70$ km/(s Mpc), $\Omega_m = 0.3$, and $\Omega_\Lambda = 0.7$. We use an image with 6730×6730 pixels and a pixel scale of 0.03 arcsec/pixel to simulate images from the HST ACS. The galaxy cluster consists of two cluster-scale and 700 galaxy-scale halos like in the HFF cluster Abell 2744 (e.g., Jauzac et al., 2015). We model the lens using SIE halos. The lens redshift is 0.3 and all sources are at the same redshift $z_{\text{source}} = 2.0$. For a realistic velocity dispersion we use as a reference the ones determined by Jauzac et al. (2015) for one cluster-scale halo approximately 1200 km/s and for a galaxy halo roughly 150 km/s. These values correspond to the velocity dispersion parameter of the parametric lens model chosen in Jauzac et al. (2015), which is not identical to measured line-of-sight velocity dispersions of galaxies. The exact conversion must be computed numerically, but for our purposes a rough agreement is enough, so we can use a conversion factor of 0.85 (Elíasdóttir et al., 2007). This leads to $\sigma_v \approx 1000$ km/s and we choose this value and a pseudo-ellipticity of the potential $\epsilon = 0.05$ for the first cluster scale halo. For the second large-scale halo, we use $\sigma_v = 700$ km/s and $\epsilon = 0.04$. We model the galaxy halos by allowing σ_v and ϵ to randomly vary between 10 and 15 km/s and 0 and 0.15, respectively. Note that the galaxy halo velocity dispersions are approximately a factor ten smaller than the ones used in Jauzac et al. (2015), because the magnitude of the scaled deflection angle for a SIE does not decrease with distance from the lens center, as it does for more realistic parametric lens models. Thus we need to decrease the velocity dispersion to limit the lensing effect of individual galaxies at large separations from the galaxy.

Fig. 6 presents the gradient values in θ_1 direction for the cluster lens. Fig. 7 shows that the error resulting from our mixed precision algorithm is within the allowed limit for each pixel. Fig. 8 illustrates the hypothetical error of a pure SP algorithm close to a cluster-scale halo. We see that the area in which we re-compute $\nabla\Psi_\epsilon$ shown in Fig. 5 covers nicely the area in which the SP error is largest.

Table 2 and Fig. 9 present the benchmark performance of the different gradient computation implementations. Each Benchmark was run 10 times and the minimum result was kept so as to minimize the impact of Operating system/network jitters. Standard deviation of the runs were at most at 10%. The speedup of the HPC-optimized codes with respect to the current LENSTOOL software is considerable. The indicated LENSTOOL performance is obtained by using all 12 CPU cores. It is thus the best currently achievable speed, as LENSTOOL cannot be run on multiple computer nodes and its performance is thus limited by the number of CPUs available on a single node. In addition, the mixed precision implementations are consistently faster than the DP ones, which validates our approach. The HPC CPU version reduces the run time of the benchmark by one order of magnitude and the GPU implementations by up to two while keeping the error within the allowed bounds. The consumer-grade GTX card displays the largest performance gain with respect to the DP computation. Fig. 10 demonstrates that our HPC-optimized CPU software scales almost perfectly with the number of cores available on a single

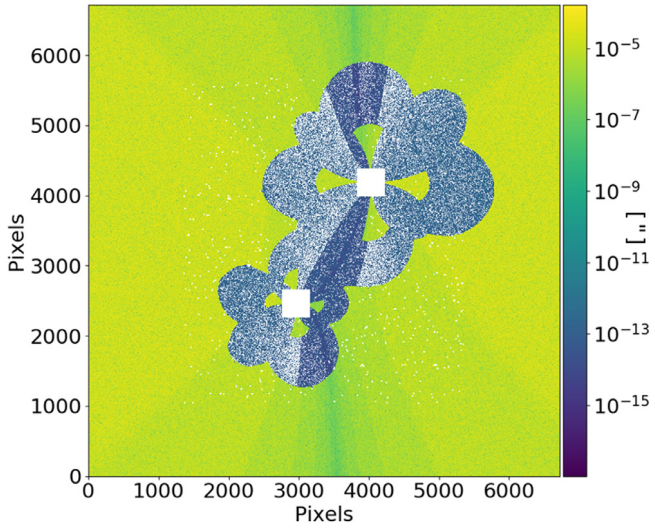


Fig. 7. Difference between the values of $(\nabla\Psi_\epsilon)_1$ computed with our mixed precision and DP algorithms for the HFF-like galaxy cluster lens used for the performance benchmark. The color white indicates a difference of zero. The green pixels are calculated with SP and the white and blue pixels are re-computed with DP. The error for every pixel is within the allowed error bounds. The small rectangular patches around the 700 galaxy halo centers in which we use DP are clearly visible. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

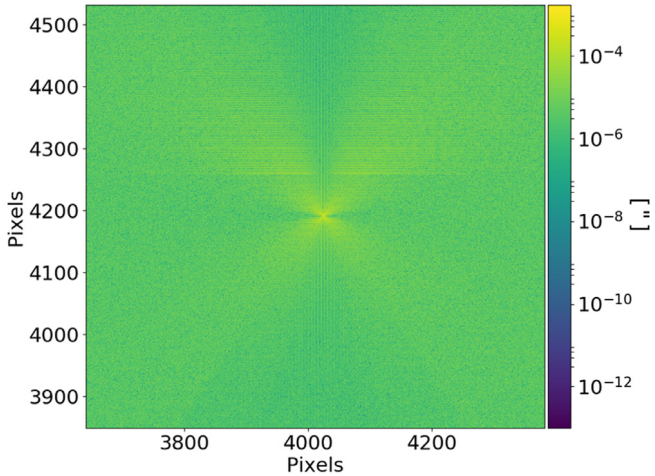


Fig. 8. Difference between the values of $(\nabla\Psi_\epsilon)_1$ computed with a pure SP algorithm and a DP algorithm. The figure shows a zoom-in on a cluster-scale SIE halo of the HFF-like galaxy cluster used in the performance benchmark. The areas with the largest errors follow clearly the pattern shown in Fig. 5. Therefore our mixed precision algorithm would re-compute these pixels with DP and thus ensure the accuracy of the result.

node using multi-threading. Fig. 11 compares the benchmark performance of the Nvidia GTX with the high-end GPU based on the same Pascal GPU architecture (Nvidia P100). The P100 is an order of magnitude more expensive than the GTX. The P100 is considerably faster when only DP is used, which is expected as it was designed for this purpose. However, as soon as the mixed precision algorithm is used, the GTX reduces its run time dramatically and the performance comes close to the P100.

5. Discussion

The performance measurements in the last section demonstrate clearly the value of HPC methods for strong lensing software. They lead to a speedup of one to two orders of magnitude,

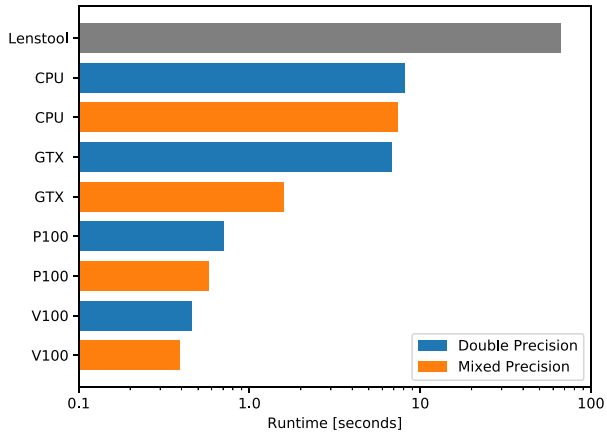


Fig. 9. Logarithmic plot of the benchmark performance for the different gradient computation implementations. We calculate the gradient for each pixel in the simulated HST image of a HFF-like galaxy cluster lens. The current LENSTOOL software serves as performance reference. The HPC-optimized implementation on the same Intel CPU with 12 cores is called CPU. It is already an order of magnitude faster. The GPU implementations can reduce the run time by another order of magnitude. Note that the mixed precision GPU algorithm uses a hybrid CPU/GPU approach. The mixed precision algorithm is faster than the DP implementation for each of the different hardware devices.

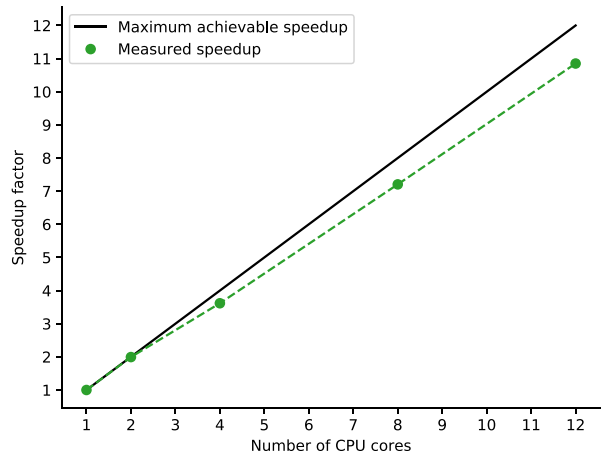


Fig. 10. The performance of the HPC-optimized CPU code scales almost perfectly with the number of used CPU cores on a single computing cluster node.

depending on the chosen hardware. In addition, our measurements show that GPUs are perfectly suited for the massively parallel lensing calculations. As expected, the high-end GPUs have a big performance advantage in DP computations, but our mixed precision algorithm and the hybrid CPU/GPU approach bring the consumer GPU's performance very close to its more expensive siblings. Note that the use of mixed precision also benefits the benchmark performance of the high-end GPUs and of the CPU, but not on the same scale. Mixed precision thus leads to a performance benefit regardless of used hardware while also delivering accurate results.

Furthermore, Table 3 demonstrates that the use of HPC methods dramatically reduces the energy consumption. We estimate the required energy to solution of the respective gradient computation implementations by multiplying the Thermal Design Power (TDP) of the used hardware with the time to solution. Note that we use only the TDP of the CPU and the GPUs for the energy to solution computations and we neglect the power consumption of other components which is typically considerably lower (see e.g. Cumming et al., 2014, for a detailed energy-efficiency study

Table 2
Benchmark results of the gradient computation implementations for a HFF-like lens. We show the double and mixed precision run times for the HPC-optimized CPU version using 12 cores and the three GPU models. The mixed precision algorithm for the GPUs uses a hybrid CPU/GPU implementation. Column three presents the measured run time advantage of mixed precision over double precision. Note that the mixed precision algorithm requires a substantial amount of additional computations compared to the double precision algorithm, as it must check for which pixels single precision is accurate enough and for which ones the gradient must be recomputed in double precision. Despite this overhead, the mixed precision implementation is the fastest for all four hardware devices. The fourth column shows the speedup of the mixed precision implementation with respect to the best currently achievable speed of LENSTOOL.

	Run time Double precision (seconds)	Run time Mixed precision (seconds)	Run time reduction Double → Mixed precision (%)	Speedup factor compared to LENSTOOL
CPU	8.1	7.40	9	9
GTX	6.8	1.58	77	42
P100	0.71	0.58	18	115
V100	0.46	0.39	15	171

Table 3
Energy comparison of the different gradient computation implementations for one run of the benchmark. We estimate the energy to solution by multiplying the Thermal Design Power (TDP) of the different hardware devices with the respective benchmark run times. In the case of the mixed precision GPU implementations, which use a hybrid CPU/GPU approach, we add the TDPs of the GPU and the CPU. The TDP values are taken from Intel Corporation (2014) and Nvidia Corporation (2016, 2017a,b). The last column shows the percentage of energy saved by using the double or mixed precision algorithm instead of the current LENSTOOL software.

	Hardware TDP (Watt)	Energy to solution Double precision (Joule)	Energy to solution Mixed precision (Joule)	Energy saved Double → Mixed precision (%)	Energy saved compared to LENSTOOL (%)
LENSTOOL	120	8016	–	–	–
CPU	120	972	888	9	88/89
GTX	250	1700	585	66	79/93
P100	300	213	244	−15	97/97
V100	300	138	164	−19	98/98

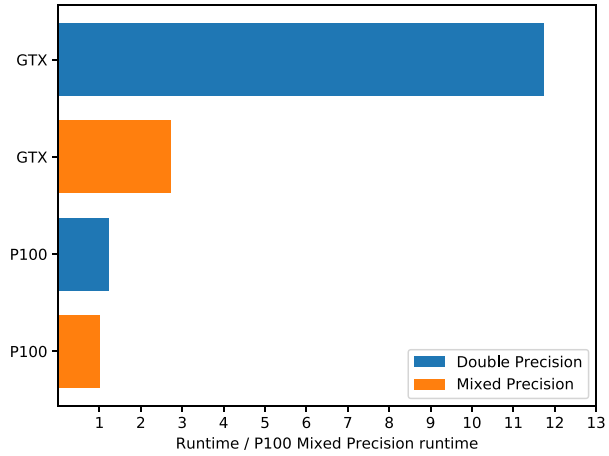


Fig. 11. Benchmark performance for a high-end GPU (Nvidia P100) and the consumer graphics card (Nvidia GTX). Both GPUs are based on the Pascal GPU architecture. The Nvidia V100 is based on the more recent Volta architecture and thus not shown in this comparison. The P100 is an order of magnitude more expensive than the GTX and especially designed for DP performance in scientific applications. Consequently it clearly outperforms the GTX when only DP is used. However, the mixed precision algorithm in combination with the hybrid CPU/GPU approach greatly accelerates the performance of the GTX and its run time comes close to the P100.

of a computing cluster). Energy savings of up to 98% are possible compared to the current LENSTOOL software. The HPC techniques are thus friendly to the environment and lower the electricity bill of the computing cluster. While the use of the mixed precision algorithm further reduces the energy consumption in the case of the CPU and the GTX, its use increases the required energy for the P100 and the V100. This is due to the hybrid CPU/GPU approach in the mixed precision implementation for the GPUs. In the case of the GTX, the decrease in run time can handily offset the additional power consumption of the CPU, while this is not the case for the high-end GPUs.

It is possible to generalize this approach to other commonly used parametric lens models like the Navarro–Frenk–White (NFW) profile (Navarro et al., 1997, 1996) or the dual Pseudo Isothermal Elliptical mass distribution (dPIE) (e.g., Elíasdóttir et al., 2007; Kassiola and Kovner, 1993). However, this will lead to much more complicated and longer gradient computation algorithms than the one for the SIE studied in this paper. As a result, it is possible that the resulting error due to finite machine precision will become bigger, too. While the exact fraction of the image for which SP is accurate enough must be explicitly calculated and studied for the respective model, it is possible that it will be considerably lower than for the SIE. Therefore the performance gain from using mixed precision might shrink accordingly. A separate study will be necessary to determine whether the expected performance gain is worth the effort.

Finally, it is necessary to discuss legal aspects of the GPU drivers provided by the graphics chip manufacturer, Nvidia. The drivers are required to use the hardware and their use is subject to certain terms and conditions, which Nvidia has recently updated. These new terms might be interpreted to prohibit the use of consumer-grade graphics cards like the GTX in computing clusters, which is the typical deployment in the scientific community. Thus the customers would be effectively forced to buy the much more expensive high-end GPUs, even though the cheaper graphics cards might be fully sufficient for the intended application. The authors of this paper are strongly concerned about this development, in particular given the limited financial budgets of academic research worldwide. Therefore they have contacted Nvidia and were informed in writing that Nvidia has no intentions to prohibit the use of the cheaper consumer-grade cards for the non-commercial purposes of researchers. The authors urge Nvidia to formalize this permission for scientific use by including it in the terms and conditions or preferably to remove this restriction altogether, thus allowing everyone to use the GPU which best suits their respective needs.

6. Conclusion

In this paper, we demonstrate the value of High Performance Computing techniques for strong lensing software. We study

a performance-critical part of the widely used LENSTOOL lens modeling software, namely the deflection potential gradient computations of the χ^2 calculation algorithm. We present and discuss an optimized CPU version with Advanced Vector Extensions and OpenMP and a GPU implementation in CUDA for a SIE lens model.

In addition, we calculate the impact of finite machine precision on the strong lensing algorithm. We demonstrate for the SIE model that single precision is accurate enough for a large part of the image. We develop a mixed precision algorithm which allows us to use single precision for performance while computing critical parts of the image in double precision.

Finally, we measure the computing performance for a galaxy cluster lens similar to the *Hubble Frontier Fields*. We find that our HPC techniques accelerate the computation by an order of magnitude on CPUs and by up to two orders of magnitude on GPUs. In addition, they reduce the energy consumption by up to 98%. The mixed precision approach delivers the best performance for every type of hardware while providing accurate results. It also permits to harness the full potential of a consumer-grade GPU, which can achieve a competitive benchmark performance for a small fraction of the monetary cost of a high-end GPU.

Acknowledgments

MR thanks Yves Revaz for fruitful discussions of GPU-accelerated computing. GF gratefully acknowledges support from the EPFL Faculté des Sciences de Base. This work was supported by EPFL, Switzerland through the use of the facilities of its Scientific IT and Application Support Center. The authors gratefully acknowledge the use of facilities of the Swiss National Supercomputing Centre (CSCS) and they thank Colin McMurtrie and Hussein Harake for their continued support. This research made use of matplotlib (Hunter, 2007), Inkscape, Astropy (Robitaille et al., 2013), TeX Live, Wolfram Alpha, and NASA's Astrophysics Data System.

Appendix. Finite machine precision error in $\nabla\Psi_\epsilon$ computation for SIE

We compute the error for $\nabla\Psi_\epsilon$ for a SIE parametric model due to finite machine precision. To do so, we investigate each line of the $\nabla\Psi_\epsilon$ Algorithm 1, we compute the respective error due to finite machine precision, and we propagate the resulting errors into the next line of the algorithm. We use ϵ to denote the machine epsilon as defined in Section 3.1 and we have $\epsilon \approx 10^{-7}$ and $\epsilon \approx 10^{-16}$ for single and double precision, respectively. Thus we can neglect terms of the order $\mathcal{O}(\epsilon^2)$ and higher. We make the assumption that the computer stores the result of one line of the algorithm, which typically corresponds to one line of code, in regular registers or memory. In addition, we assume that intermediate results, which occur while processing one line of the algorithm, are stored in extended precision registers, which are e.g. typically present in x87 Floating-Point Units (FPUs). As a result, we can neglect error contributions due to machine precision for these intermediate results. Note that this is no longer the case if we use Streaming SIMD Extension (SSE) or AVX registers, as these do not use extended precision.

The Appendix is organized as follows: In Appendix A.1, we present the error propagation rules, in Appendix A.2 we derive a mathematical expression for the error of $\nabla\Psi_\epsilon$ and we write it in a compact form by defining appropriate error variables, and in Appendix A.3 we compute upper bounds for the error.

A.1. Error propagation rules

We use the following error propagation rules which give upper limits on the propagated error:

Addition:

$$x \pm \epsilon a + y \pm \epsilon b = x + y \pm \epsilon(|a| + |b|) \tag{A.1}$$

Subtraction:

$$x \pm \epsilon a - y \pm \epsilon b = x - y \pm \epsilon(|a| + |b|) \tag{A.2}$$

Multiplication:

$$\begin{aligned}(x \pm \epsilon a)(y \pm \epsilon b) &= xy \pm \epsilon|ay| \pm \epsilon|xb| \pm \mathcal{O}(\epsilon^2)\\ &= xy \pm \epsilon(|ay| + |xb|)\end{aligned} \tag{A.3}$$

Division:

$$\frac{x \pm \epsilon a}{y \pm \epsilon b} = \frac{x}{y} \pm \epsilon\frac{|ay| + |bx|}{y^2 \pm \epsilon by} \tag{A.4}$$

Proof.

$$\begin{aligned}\frac{x \pm \epsilon a}{y \pm \epsilon b} - \frac{x}{y} &= \frac{(x \pm \epsilon a)y - x(y \pm \epsilon b)}{y^2 \pm \epsilon by}\\ &= \frac{xy \pm \epsilon ay - xy \pm \epsilon bx}{y^2 \pm \epsilon by}\\ &= \pm\epsilon\frac{|ay| + |bx|}{y^2 \pm \epsilon by}\end{aligned}$$

General, infinitely differentiable function $f(x)$:

We can use the Taylor expansion to first order,

$$f(x \pm \epsilon a) = f(x) \pm \epsilon a f'(x), \tag{A.5}$$

if the contribution from higher order terms is negligible:

$$\frac{f^n(x)a^n\epsilon^n}{f'(x)a\epsilon n!} \approx 0 \quad \forall n > 1,$$

where $f^n(x)$ denotes the nth derivative.

A.2. Error computation

We will denote a result x stored in a regular register or memory with $\texttt{stored}(x)$. We want to derive an upper limit on the final error, so we will assume that each of these storage operations produces an error, $\texttt{stored}(x) = x \pm \epsilon x$, and we propagate these errors. Note that in practice the storing of results does not necessarily produce an error and, since the storing error is basically due to a rounding operation, errors from different storing operations can cancel each other.

We compute now the machine precision error for one SIE lens at pixel (θ_1, θ_2):

$$\begin{aligned}\Delta\theta_1 &= \theta_1 \pm \epsilon\theta_1 - (\theta_{\text{center},1} \pm \epsilon\theta_{\text{center},1})\\ &= \theta_1 - \theta_{\text{center},1} \pm \epsilon(|\theta_1| + |\theta_{\text{center},1}|).\end{aligned} \tag{A.6}$$

$$\begin{aligned}\texttt{stored}(\Delta\theta_1) &= \theta_1 - \theta_{\text{center},1} \pm \epsilon(|\theta_1| + |\theta_{\text{center},1}|)\\ &\quad \pm \epsilon(|\theta_1 - \theta_{\text{center},1}|) \pm \mathcal{O}(\epsilon^2)\\ &= \theta_1 - \theta_{\text{center},1} \pm \epsilon(|\theta_1|\\ &\quad + |\theta_{\text{center},1}| + |\theta_1 - \theta_{\text{center},1}|)\\ &= \Delta\theta_{1,t} \pm \epsilon A_1.\end{aligned} \tag{A.7}$$

In the last line, we introduced the true, error-free value of $\Delta\theta_1$, $\Delta\theta_{1,t} = \theta_1 - \theta_{\text{center},1}$. In addition, we implicitly defined the error variable A_1, which contains all the terms which contribute to the error.

$$\begin{aligned}\Delta\theta_2 &= \theta_2 \pm \epsilon\theta_2 - (\theta_{\text{center},2} \pm \epsilon\theta_{\text{center},2})\\ &= \theta_2 - \theta_{\text{center},2} \pm \epsilon(|\theta_2| + |\theta_{\text{center},2}|).\end{aligned} \tag{A.8}$$

$$\begin{aligned}\texttt{stored}(\Delta\theta_2) &= \theta_2 - \theta_{\text{center},2} \pm \epsilon(|\theta_2| + |\theta_{\text{center},2}|)\\ &\quad \pm \epsilon(|\theta_2 - \theta_{\text{center},2}|) \pm \mathcal{O}(\epsilon^2)\\ &= \theta_2 - \theta_{\text{center},2} \pm \epsilon(|\theta_2|\\ &\quad + |\theta_{\text{center},2}| + |\theta_2 - \theta_{\text{center},2}|)\\ &= \Delta\theta_{2,t} \pm \epsilon A_2, \end{aligned} \tag{A.9}$$

where we again implicitly defined $\Delta\theta_{2,t}$ and A_2.

$$\begin{aligned}\Delta\theta_1' &= (\Delta\theta_{1,t} \pm \epsilon A_1)\cos(\Phi \pm \epsilon\Phi)\\ &\quad + (\Delta\theta_{2,t} \pm \epsilon A_2)\sin(\Phi \pm \epsilon\Phi)\\ &= (\Delta\theta_{1,t} \pm \epsilon A_1)[\cos(\Phi) \pm \sin(\Phi)\epsilon\Phi]\\ &\quad + (\Delta\theta_{2,t} \pm \epsilon A_2)[\sin(\Phi) \pm \cos(\Phi)\epsilon\Phi]. \end{aligned} \tag{A.10}$$

As we have $\sin(\Phi) \leq 1$ and $\cos(\Phi) \leq 1$, we can obtain an upper bound on the error by replacing the respective sine and cosine expressions in the parts which contribute to the error with 1:

$$\begin{aligned}\Delta\theta_1' &= (\Delta\theta_{1,t} \pm \epsilon A_1)[\cos(\Phi) \pm \epsilon\Phi]\\ &\quad + (\Delta\theta_{2,t} \pm \epsilon A_2)[\sin(\Phi) \pm \epsilon\Phi]\\ &= \Delta\theta_{1,t}\cos(\Phi) \pm \epsilon(|A_1\cos(\Phi)| + |\Delta\theta_{1,t}\Phi|)\\ &\quad + \Delta\theta_{2,t}\sin(\Phi) \pm \epsilon(|A_2\sin(\Phi)| + |\Delta\theta_{2,t}\Phi|) + \mathcal{O}(\epsilon^2)\\ &= \Delta\theta_{1,t}\cos(\Phi) \pm \epsilon(|A_1| + |\Delta\theta_{1,t}\Phi|)\\ &\quad + \Delta\theta_{2,t}\sin(\Phi) \pm \epsilon(|A_2| + |\Delta\theta_{2,t}\Phi|)\\ &= \Delta\theta_{1,r} \pm \epsilon(|A_1| + |A_2| + |\Delta\theta_{1,t}\Phi| + |\Delta\theta_{2,t}\Phi|). \end{aligned} \tag{A.11}$$

In the last line we implicitly defined the true value after rotation, $\Delta\theta_{1,r}$.

$$\begin{aligned}\texttt{stored}(\Delta\theta_1') &= \Delta\theta_{1,r} \pm \epsilon(|A_1| + |A_2| + |\Delta\theta_{1,t}\Phi| + |\Delta\theta_{2,t}\Phi|\\ &\quad + |\Delta\theta_{1,t}\cos(\Phi)| + |\Delta\theta_{2,t}\sin(\Phi)|) + \mathcal{O}(\epsilon^2)\\ &= \Delta\theta_{1,r} \pm \epsilon(|A_1| + |A_2| + |\Delta\theta_{1,t}\Phi|\\ &\quad + |\Delta\theta_{2,t}\Phi| + |\Delta\theta_{1,t}| + |\Delta\theta_{2,t}|)\\ &= \Delta\theta_{1,r} \pm \epsilon B, \end{aligned} \tag{A.12}$$

where we again replaced sine and cosine with 1 and implicitly defined the error variable B.

Similarly, we obtain for $\Delta\theta_2'$:

$$\begin{aligned}\Delta\theta_2' &= (\Delta\theta_{2,t} \pm \epsilon A_2)[\cos(\Phi) \pm \epsilon\Phi]\\ &\quad - (\Delta\theta_{1,t} \pm \epsilon A_1)[\sin(\Phi) \pm \epsilon\Phi]\\ &= \Delta\theta_{2,t}\cos(\Phi) \pm \epsilon(|A_2| + |\Delta\theta_{2,t}\Phi|)\\ &\quad - \Delta\theta_{1,t}\sin(\Phi) \pm \epsilon(|A_1| + |\Delta\theta_{1,t}\Phi|) + \mathcal{O}(\epsilon^2)\\ &= \Delta\theta_{2,r} \pm \epsilon(|A_1| + |A_2| + |\Delta\theta_{1,t}\Phi| + |\Delta\theta_{2,t}\Phi|). \end{aligned} \tag{A.13}$$

In the last line we implicitly defined the true value after rotation, $\Delta\theta_{2,r}$.

$$\texttt{stored}(\Delta\theta_2') = \Delta\theta_{2,r} \pm \epsilon B. \tag{A.14}$$

In this Appendix, we denote the pseudo-ellipticity of the deflection potential with p instead of ϵ to avoid confusion with the machine epsilon. We further define $p_\star = 1 - p$, $p_\dagger = 1 + p$ and we obtain: (see Eq. (A.15) given in Box I)
We define the true value of R,

$$R_t = \sqrt{\Delta\theta_{1,r}^2(1-p) + \Delta\theta_{2,r}^2(1+p)}, \tag{A.16}$$

and use a Taylor expansion to obtain (see Eqs. (A.17) and (A.18) given in Box II
where we implicitly defined the error variable C.

$$\begin{aligned}\nabla\Psi_{\epsilon,1} &= (1 - p \pm \epsilon p_\star)(b_0 \pm \epsilon b_0)\frac{\Delta\theta_{1,r} \pm \epsilon B}{R_t \pm \epsilon C}\\ &= [(1-p)b_0 \pm \epsilon(2b_0 p_\star)]\frac{\Delta\theta_{1,r} \pm \epsilon B}{R_t \pm \epsilon C} + \mathcal{O}(\epsilon^2)\\ &= [(1-p)b_0 \pm \epsilon(2b_0 p_\star)]\left[\frac{\Delta\theta_{1,r}}{R_t} \pm \epsilon\frac{|BR_t| + |C\Delta\theta_{1,r}|}{R_t^2 \pm \epsilon|CR_t|}\right]\\ &= (1-p)b_0\frac{\Delta\theta_{1,r}}{R_t} \pm \epsilon\left[\left|2b_0 p_\star\frac{\Delta\theta_{1,r}}{R_t}\right|\right.\\ &\quad \left. + \left|(1-p)b_0\frac{|BR_t| + |C\Delta\theta_{1,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right] + \mathcal{O}(\epsilon^2)\\ &= \nabla\Psi_{\epsilon,t,1} \pm \epsilon\left[|2\nabla\Psi_{\epsilon,t,1}|\right.\\ &\quad \left. + \left|(1-p)b_0\frac{|BR_t| + |C\Delta\theta_{1,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right], \end{aligned} \tag{A.19}$$

where we implicitly defined the true value of the gradient, $\nabla\Psi_{\epsilon,t,1}$.

$$\begin{aligned}\texttt{stored}(\nabla\Psi_{\epsilon,1}) &= \nabla\Psi_{\epsilon,t,1} \pm \epsilon\left[|3\nabla\Psi_{\epsilon,t,1}|\right.\\ &\quad \left. + \left|(1-p)b_0\frac{|BR_t| + |C\Delta\theta_{1,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right] + \mathcal{O}(\epsilon^2)\\ &= \nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1, \end{aligned} \tag{A.20}$$

where we implicitly defined the error variable D_1.

$$\begin{aligned}\nabla\Psi_{\epsilon,2} &= (1 + p \pm \epsilon p_\dagger)(b_0 \pm \epsilon b_0)\frac{\Delta\theta_{2,r} \pm \epsilon B}{R_t \pm \epsilon C}\\ &= [(1+p)b_0 \pm \epsilon(2b_0 p_\dagger)]\frac{\Delta\theta_{2,r} \pm \epsilon B}{R_t \pm \epsilon C} + \mathcal{O}(\epsilon^2)\\ &= [(1+p)b_0 \pm \epsilon(2b_0 p_\dagger)]\left[\frac{\Delta\theta_{2,r}}{R_t} \pm \epsilon\frac{|BR_t| + |C\Delta\theta_{2,r}|}{R_t^2 \pm \epsilon|CR_t|}\right]\\ &= (1+p)b_0\frac{\Delta\theta_{2,r}}{R_t} \pm \epsilon\left[\left|2b_0 p_\dagger\frac{\Delta\theta_{2,r}}{R_t}\right|\right.\\ &\quad \left. + \left|(1+p)b_0\frac{|BR_t| + |C\Delta\theta_{2,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right] + \mathcal{O}(\epsilon^2)\\ &= \nabla\Psi_{\epsilon,t,2} \pm \epsilon\left[|2\nabla\Psi_{\epsilon,t,2}|\right.\\ &\quad \left. + \left|(1+p)b_0\frac{|BR_t| + |C\Delta\theta_{2,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right], \end{aligned} \tag{A.21}$$

where we implicitly defined the true value of the gradient, $\nabla\Psi_{\epsilon,t,2}$.

$$\begin{aligned}\texttt{stored}(\nabla\Psi_{\epsilon,2}) &= \nabla\Psi_{\epsilon,t,2} \pm \epsilon\left[|3\nabla\Psi_{\epsilon,t,2}|\right.\\ &\quad \left. + \left|(1+p)b_0\frac{|BR_t| + |C\Delta\theta_{2,r}|}{R_t^2 \pm \epsilon|CR_t|}\right|\right] + \mathcal{O}(\epsilon^2)\\ &= \nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2, \end{aligned} \tag{A.22}$$

where we implicitly defined the error variable D_2.

$$\begin{aligned}\nabla\Psi_{\epsilon,1}' &= (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)\cos(-\Phi \pm \epsilon\Phi)\\ &\quad + (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)\sin(-\Phi \pm \epsilon\Phi)\\ &= (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)[\cos(-\Phi) \pm \sin(-\Phi)\epsilon\Phi]\\ &\quad + (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)[\sin(-\Phi) \pm \cos(-\Phi)\epsilon\Phi]. \end{aligned} \tag{A.23}$$

As we have $\sin(\Phi) \leq 1$ and $\cos(\Phi) \leq 1$, we can obtain an upper bound on the error by replacing the respective sine and cosine expressions in the parts which contribute to the error with 1:

$$\begin{aligned}\nabla\Psi_{\epsilon,1}' &= (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)[\cos(-\Phi) \pm \epsilon\Phi]\\ &\quad + (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)[\sin(-\Phi) \pm \epsilon\Phi]\\ &= \nabla\Psi_{\epsilon,t,1}\cos(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi|\\ &\quad + |D_1\cos(-\Phi)|) + \nabla\Psi_{\epsilon,t,2}\sin(-\Phi)\\ &\quad \pm \epsilon(|\nabla\Psi_{\epsilon,t,2}\Phi| + |D_2\sin(-\Phi)|) + \mathcal{O}(\epsilon^2) \end{aligned}$$

$$
\begin{aligned}
R &= \sqrt{(\Delta\theta_{1,r} \pm \epsilon B)^2(1 - p \pm \epsilon p_\star) + (\Delta\theta_{2,r} \pm \epsilon B)^2(1 + p \pm \epsilon p_\dagger)} \\
&= \sqrt{(\Delta\theta_{1,r}^2 \pm \epsilon|2\Delta\theta_{1,r}B|)(1 - p \pm \epsilon p_\star) + (\Delta\theta_{2,r}^2 \pm \epsilon|2\Delta\theta_{2,r}B|)(1 + p \pm \epsilon p_\dagger) + \mathcal{O}(\epsilon^2)} \\
&= \sqrt{\Delta\theta_{1,r}^2(1-p) \pm \epsilon(|\Delta\theta_{1,r}^2 p_\star| + |2\Delta\theta_{1,r}B(1-p)|) + \Delta\theta_{2,r}^2(1+p) \pm \epsilon(|\Delta\theta_{2,r}^2 p_\dagger| + |2\Delta\theta_{2,r}B(1+p)|) + \mathcal{O}(\epsilon^2)} \\
&= \sqrt{\Delta\theta_{1,r}^2(1-p) + \Delta\theta_{2,r}^2(1+p) \pm \epsilon(|\Delta\theta_{1,r}^2 p_\star| + |\Delta\theta_{2,r}^2 p_\dagger| + |2\Delta\theta_{1,r}B(1-p)| + |2\Delta\theta_{2,r}B(1+p)|)}. \qquad \text{(A.15)}
\end{aligned}
$$

Box I.

$$
R = R_t \pm \epsilon \frac{|\Delta\theta_{1,r}^2 p_\star| + |\Delta\theta_{2,r}^2 p_\dagger| + |2\Delta\theta_{1,r}B(1-p)| + |2\Delta\theta_{2,r}B(1+p)|}{2|R_t|}. \qquad \text{(A.17)}
$$

$$
\begin{aligned}
\texttt{stored}(R) &= R_t \pm \epsilon\left(\frac{|\Delta\theta_{1,r}^2 p_\star| + |\Delta\theta_{2,r}^2 p_\dagger| + |2\Delta\theta_{1,r}B(1-p)| + |2\Delta\theta_{2,r}B(1+p)|}{2|R_t|} + |R_t|\right) + \mathcal{O}(\epsilon^2) \\
&= R_t \pm \epsilon C, \qquad \text{(A.18)}
\end{aligned}
$$

Box II.

$$
\begin{aligned}
&= \nabla\Psi_{\epsilon,t,1}\cos(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| \\
&\quad + |D_1|) + \nabla\Psi_{\epsilon,t,2}\sin(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,2}\Phi| + |D_2|) \\
&= \nabla\Psi_{\epsilon,r,1} \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |\nabla\Psi_{\epsilon,t,2}\Phi| + |D_1| + |D_2|), \qquad \text{(A.24)}
\end{aligned}
$$

where we implicitly defined the true value of the first gradient component after rotation, $\nabla\Psi_{\epsilon,r,1}$.

We use the relation

$$
\begin{aligned}
&|\nabla\Psi_{\epsilon,t,1}\cos(-\Phi) + \nabla\Psi_{\epsilon,t,2}\sin(-\Phi)| \\
&\quad \le |\nabla\Psi_{\epsilon,t,1}\cos(-\Phi)| + |\nabla\Psi_{\epsilon,t,2}\sin(-\Phi)| \\
&\quad \le |\nabla\Psi_{\epsilon,t,1}| + |\nabla\Psi_{\epsilon,t,2}| \qquad \text{(A.25)}
\end{aligned}
$$

to obtain:

$$
\begin{aligned}
\texttt{stored}(\nabla\Psi'_{\epsilon,1}) &= \nabla\Psi_{\epsilon,r,1} \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |\nabla\Psi_{\epsilon,t,2}\Phi| + |D_1| \\
&\quad + |D_2| + |\nabla\Psi_{\epsilon,t,1}| + |\nabla\Psi_{\epsilon,t,2}|) + \mathcal{O}(\epsilon^2) \\
&= \nabla\Psi_{\epsilon,r,1} \pm \epsilon F, \qquad \text{(A.26)}
\end{aligned}
$$

where we implicitly defined the error variable F.

$$
\begin{aligned}
\nabla\Psi'_{\epsilon,2} &= (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)\cos(-\Phi \pm \epsilon\Phi) \\
&\quad - (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)\sin(-\Phi \pm \epsilon\Phi) \\
&= (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)[\cos(-\Phi) \pm \sin(-\Phi)\epsilon\Phi] \\
&\quad - (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)[\sin(-\Phi) \pm \cos(-\Phi)\epsilon\Phi]. \qquad \text{(A.27)}
\end{aligned}
$$

As we have $\sin(\Phi) \le 1$ and $\cos(\Phi) \le 1$, we can obtain an upper bound on the error by replacing the respective sine and cosine expressions in the parts which contribute to the error with 1:

$$
\begin{aligned}
\nabla\Psi'_{\epsilon,2} &= (\nabla\Psi_{\epsilon,t,2} \pm \epsilon D_2)[\cos(-\Phi) \pm \epsilon\Phi] \\
&\quad - (\nabla\Psi_{\epsilon,t,1} \pm \epsilon D_1)[\sin(-\Phi) \pm \epsilon\Phi] \\
&= \nabla\Psi_{\epsilon,t,2}\cos(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,2}\Phi| \\
&\quad + |D_2\cos(-\Phi)|) - \nabla\Psi_{\epsilon,t,1}\sin(-\Phi) \\
&\quad \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |D_1\sin(-\Phi)|) + \mathcal{O}(\epsilon^2) \\
&= \nabla\Psi_{\epsilon,t,2}\cos(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,2}\Phi| + |D_2|) \\
&\quad - \nabla\Psi_{\epsilon,t,1}\sin(-\Phi) \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |D_1|) \\
&= \nabla\Psi_{\epsilon,r,2} \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |\nabla\Psi_{\epsilon,t,2}\Phi| + |D_1| + |D_2|), \qquad \text{(A.28)}
\end{aligned}
$$

where we implicitly defined the true value of the second gradient component after rotation, $\nabla\Psi_{\epsilon,r,2}$.

We use the relation

$$
\begin{aligned}
&|\nabla\Psi_{\epsilon,t,2}\cos(-\Phi) - \nabla\Psi_{\epsilon,t,1}\sin(-\Phi)| \\
&\quad \le |\nabla\Psi_{\epsilon,t,2}\cos(-\Phi)| + |\nabla\Psi_{\epsilon,t,1}\sin(-\Phi)| \\
&\quad \le |\nabla\Psi_{\epsilon,t,2}| + |\nabla\Psi_{\epsilon,t,1}| \qquad \text{(A.29)}
\end{aligned}
$$

to obtain:

$$
\begin{aligned}
\texttt{stored}(\nabla\Psi'_{\epsilon,2}) &= \nabla\Psi_{\epsilon,r,2} \pm \epsilon(|\nabla\Psi_{\epsilon,t,1}\Phi| + |\nabla\Psi_{\epsilon,t,2}\Phi| + |D_1| \\
&\quad + |D_2| + |\nabla\Psi_{\epsilon,t,1}| + |\nabla\Psi_{\epsilon,t,2}|) + \mathcal{O}(\epsilon^2) \\
&= \nabla\Psi_{\epsilon,r,2} \pm \epsilon F. \qquad \text{(A.30)}
\end{aligned}
$$

As a result, the total error of one computed $\nabla\Psi_\epsilon$ for one pixel (θ_1, θ_2) due to finite machine precision is ϵF for both gradient components.

A.3. Upper error bounds for cluster- and galaxy-scale SIE lenses

A.3.1. Centered SIE lenses

Let us consider a single lens at the origin of a two dimensional image plane coordinate system,

$$
(\theta_{\text{center},1}, \theta_{\text{center},2}) = (0, 0). \qquad \text{(A.31)}
$$

We assume that the point (θ_1, θ_2) for which we compute $\nabla\Psi_\epsilon$ lies on the θ_1-axis. We assume that we can do so without loss of generality, as this can be achieved by a simple rotation of the coordinate system. This simplifies the expression for the A terms to

$$
\begin{aligned}
A_1 &= 2|\theta_1|, \\
A_2 &= 0. \qquad \text{(A.32)}
\end{aligned}
$$

We want to maximize the errors to obtain an upper bound. Therefore we maximize the angle Φ, which always appears as an error increasing factor in the error variables. Due to the symmetry of an ellipse, the largest value is $\Phi = \pi$. As a result, we have

$$
B = 2|\theta_1| + \pi|\theta_1| + |\theta_1| = (3 + \pi)|\theta_1|. \qquad \text{(A.33)}
$$

Note that our coordinate system is now rotated by 180 degrees, so we have

$$
\Delta\theta_1 \to -\Delta\theta'_1,
$$

$$\Delta\theta_2 \to -\Delta\theta_2'. \tag{A.34}$$

Next, we note that the pseudo-ellipticity p is typically small and that it appears in the error variables in connection with $\Delta\theta_1'$ as a factor $1-p$ and in connection with $\Delta\theta_2'$ as a factor $1+p$. Therefore we will minimize it and assume $p = 0$. Thus we have

$$C = \frac{|\theta_1^2| + (6+2\pi)|\theta_1^2|}{2|\theta_1|} + |\theta_1| = (4.5+\pi)|\theta_1|. \tag{A.35}$$

The lensing effect will be maximal for a source at high redshift, so we assume $D_{OS}/D_{LS} = 1$ and thus we have

$$\begin{aligned} |\nabla\Psi_{\epsilon,1}| &= |(\nabla\psi)_1| = \theta_E, \\ |\nabla\Psi_{\epsilon,2}| &= |(\nabla\psi)_2| = 0, \qquad (A.36) \\ b_0 &= \theta_E, \qquad (A.37) \end{aligned}$$

and thus

$$\begin{aligned} D_1 &\approx 3\theta_E + \theta_E \frac{(3+\pi)|\theta_1|^2 + (4.5+\pi)|\theta_1|^2}{|\theta_1|^2} = (10.5+2\pi)\theta_E, \\ D_2 &\approx \theta_E \frac{(3+\pi)|\theta_1|^2}{|\theta_1|^2} = (3+\pi)\theta_E. \qquad (A.38) \end{aligned}$$

We now rotate the coordinate system by -180 degrees,

$$\begin{aligned} \nabla\Psi_{\epsilon,1} &\to -\nabla\Psi'_{\epsilon,1}, \\ \nabla\Psi_{\epsilon,2} &\to -\nabla\Psi'_{\epsilon,2}, \qquad (A.39) \end{aligned}$$

and we obtain

$$F = \pi\theta_E + (10.5+2\pi)\theta_E + (3+\pi)\theta_E + \theta_E \approx 27\theta_E. \tag{A.40}$$

As a result, we have for a cluster-scale halo with $\theta_E = 20$ arcsec

$$F_{\text{cluster-scale}} = 540 \text{ arcsec} \tag{A.41}$$

and for a galaxy-scale halo with $\theta_E = 0.2$ arcsec

$$F_{\text{galaxy-scale}} = 5.4 \text{ arcsec}. \tag{A.42}$$

For single and double precision, we have respectively $\epsilon_{SP} \approx 1.2 \times 10^{-7}$ and $\epsilon_{DP} \approx 2.2 \times 10^{-16}$, and thus the upper error bounds

$$\begin{aligned} \epsilon_{SP} F_{\text{cluster-scale}} &\approx 6.5 \times 10^{-5} \text{ arcsec}, \\ \epsilon_{SP} F_{\text{galaxy-scale}} &\approx 6.5 \times 10^{-7} \text{ arcsec}, \qquad (A.43) \\ \epsilon_{DP} F_{\text{cluster-scale}} &\approx 1.2 \times 10^{-13} \text{ arcsec}, \\ \epsilon_{DP} F_{\text{galaxy-scale}} &\approx 1.2 \times 10^{-15} \text{ arcsec}. \qquad (A.44) \end{aligned}$$

The computed gradients for each halo are finally added up to obtain the total gradient,

$$\nabla\Psi_{\epsilon,i} = \sum_k \nabla\Psi'_{\epsilon,i,k}, \tag{A.45}$$

and as a result, the respective errors are combined as well. However, the respective errors can have different signs and magnitudes, so we expect to see some error cancellation. We estimate the total gradient error in the following way: We neglect the contribution from the galaxy-scale halos and we add the respective upper error bounds of the cluster-scale halos. Neglecting the galaxy-scale lenses is justified, because first, their absolute errors are two orders of magnitude smaller than those of the cluster-scale halos, and second, we add many of these halos which are typically scattered throughout the image, so we expect significant error cancellation effects. We are left with typically two cluster-scale halos. The error contribution from these halos will depend on their respective parameters. To obtain an upper bound, we will add up the respective upper bounds on the gradient, so we have

$$\begin{aligned} \Delta(\nabla\Psi_{\epsilon,i})_{SP} &\approx 1.3 \times 10^{-4} \text{ arcsec}, \\ \Delta(\nabla\Psi_{\epsilon,i})_{DP} &\approx 2.4 \times 10^{-13} \text{ arcsec}. \qquad (A.46) \end{aligned}$$

A.3.2. General SIE lenses

In the previous part, we implicitly assumed that the finite machine precision error is invariant under translations and rotations of the coordinate system. As a result, it was sufficient to compute the error for a single centered SIE lens and we could use the result to derive the total error for the cluster lens system. However, we now show that this assumed invariance only holds approximately and only far away from the lens center.

Let us consider a HST ACS image. We let the origin of the coordinate system coincide with the first pixel of the image in the lower left corner. Consequently all pixel values are positive, so we have

$$|\theta_i| \le |\theta_i - \theta_{\text{center},i}| + |\theta_{\text{center},i}| \tag{A.47}$$

and thus the upper bounds for the A terms are

$$\begin{aligned} A_1 &= 2|\Delta\theta_1| + 2|\theta_{\text{center},1}|, \\ A_2 &= 2|\Delta\theta_2| + 2|\theta_{\text{center},2}|, \qquad (A.48) \end{aligned}$$

where

$$|\Delta\theta_i| = |\theta_i - \theta_{\text{center},i}|. \tag{A.49}$$

We want to maximize the errors to obtain an upper bound. Therefore we maximize the angle Φ, which always appears as an error increasing factor in the error variables. Due to the symmetry of an ellipse, the largest value is $\Phi = \pi$. As a result, we have

$$\begin{aligned} B &= 2|\Delta\theta_1| + 2|\theta_{\text{center},1}| + 2|\Delta\theta_2| + 2|\theta_{\text{center},2}| \\ &\quad + \pi|\Delta\theta_1| + \pi|\Delta\theta_2| + |\Delta\theta_1| + |\Delta\theta_2| \\ &= (3+\pi)|\Delta\theta_1| + (3+\pi)|\Delta\theta_2| + 2|\theta_{\text{center},1}| + 2|\theta_{\text{center},2}|. \end{aligned} \tag{A.50}$$

Note that our coordinate system is now rotated by 180 degrees, so we have

$$\begin{aligned} \Delta\theta_1 &\to -\Delta\theta_1', \\ \Delta\theta_2 &\to -\Delta\theta_2'. \qquad (A.51) \end{aligned}$$

Next, we note that the pseudo-ellipticity p is typically small and that it appears in the error variables in connection with $\Delta\theta_1'$ as a factor $1-p$ and in connection with $\Delta\theta_2'$ as a factor $1+p$. Therefore we will minimize it and assume $p = 0$. Thus we have Eq. (A.52) in Box III. The lensing effect will be maximal for a source at high redshift, so we assume $D_{OS}/D_{LS} = 1$ and thus we have

$$\begin{aligned} |\nabla\Psi_{\epsilon,1}| &= |(\nabla\psi)_1| \le \theta_E, \\ |\nabla\Psi_{\epsilon,2}| &= |(\nabla\psi)_2| \le \theta_E, \qquad (A.53) \\ b_0 &= \theta_E, \qquad (A.54) \end{aligned}$$

and thus, see Eq. (A.55) in Box IV and see Eq. (A.56) in Box V. We now rotate the coordinate system by -180 degrees,

$$\begin{aligned} \nabla\Psi_{\epsilon,1} &\to -\nabla\Psi'_{\epsilon,1}, \\ \nabla\Psi_{\epsilon,2} &\to -\nabla\Psi'_{\epsilon,2}. \qquad (A.57) \end{aligned}$$

We first compute one part of the error variable F, namely Eq. (A.58) in Box VI.

Each term in this equation is maximized if we simultaneously maximize $\Delta\theta_1$ and $\Delta\theta_2$ for a fixed radius $\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}$. We can see this by rewriting the following relations in polar coordinates,

$$\begin{aligned} &\frac{|\Delta\theta_1| + |\Delta\theta_2|}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}} \\ &= \frac{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}(|\cos(\phi')| + |\sin(\phi')|)}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}} \le \sqrt{2}, \end{aligned} \tag{A.59}$$

$$C = \frac{|\Delta\theta_1^2| + |\Delta\theta_2^2| + (6+2\pi)|\Delta\theta_1^2| + (6+2\pi)|\Delta\theta_2^2| + (12+4\pi)|\Delta\theta_1||\Delta\theta_2| + 4(|\Delta\theta_1| + |\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{2\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}$$
$$+ \left|\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}\right|$$
$$= \frac{(12+4\pi)|\Delta\theta_1||\Delta\theta_2| + 4(|\Delta\theta_1| + |\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{2\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}} + (4.5+\pi)\left|\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}\right|. \tag{A.52}$$

Box III.

$$D_1 \approx 3\theta_E + \theta_E \frac{(3+\pi)(|\Delta\theta_1| + |\Delta\theta_2|)\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|} + 2(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}{|\Delta\theta_1|^2 + |\Delta\theta_2|^2}$$
$$+ \theta_E \frac{(4.5+\pi)|\Delta\theta_1|\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}{|\Delta\theta_1|^2 + |\Delta\theta_2|^2} + \theta_E \frac{(6+2\pi)|\Delta\theta_1^2||\Delta\theta_2| + 2(|\Delta\theta_1^2| + |\Delta\theta_1||\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}}$$
$$= \theta_E\Bigg[3 + \frac{(3+\pi)(|\Delta\theta_1| + |\Delta\theta_2|) + 2(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|) + (4.5+\pi)|\Delta\theta_1|}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}$$
$$+ \frac{(6+2\pi)|\Delta\theta_1^2||\Delta\theta_2| + 2(|\Delta\theta_1^2| + |\Delta\theta_1||\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}}\Bigg], \tag{A.55}$$

Box IV.

$$D_2 \approx 3\theta_E + \theta_E \frac{(3+\pi)(|\Delta\theta_1| + |\Delta\theta_2|)\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|} + 2(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}{|\Delta\theta_1|^2 + |\Delta\theta_2|^2}$$
$$+ \theta_E \frac{(4.5+\pi)|\Delta\theta_2|\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}{|\Delta\theta_1|^2 + |\Delta\theta_2|^2} + \theta_E \frac{(6+2\pi)|\Delta\theta_1||\Delta\theta_2^2| + 2(|\Delta\theta_2^2| + |\Delta\theta_1||\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}}$$
$$= \theta_E\Bigg[3 + \frac{(3+\pi)(|\Delta\theta_1| + |\Delta\theta_2|) + 2(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|) + (4.5+\pi)|\Delta\theta_2|}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}$$
$$+ \frac{(6+2\pi)|\Delta\theta_1||\Delta\theta_2^2| + 2(|\Delta\theta_2^2| + |\Delta\theta_1||\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}}\Bigg]. \tag{A.56}$$

Box V.

$$|D_1| + |D_2| = \theta_E\Bigg[6 + \frac{(6+2\pi)(|\Delta\theta_1| + |\Delta\theta_2|) + 4(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|) + (4.5+\pi)(|\Delta\theta_1| + |\Delta\theta_2|)}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}$$
$$+ \frac{(6+2\pi)(|\Delta\theta_1^2||\Delta\theta_2| + |\Delta\theta_1||\Delta\theta_2^2|) + 2(|\Delta\theta_1^2| + |\Delta\theta_2^2| + 2|\Delta\theta_1||\Delta\theta_2|)(|\theta_{\text{center},1}| + |\theta_{\text{center},2}|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}}\Bigg]. \tag{A.58}$$

Box VI.

$$\frac{|\Delta\theta_1^2||\Delta\theta_2| + |\Delta\theta_1||\Delta\theta_2^2|}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}} = \frac{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}(\cos^2(\phi')|\sin(\phi')| + |\cos(\phi')|\sin^2(\phi'))}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}} \leq \frac{1}{\sqrt{2}}, \tag{A.60}$$

$$\frac{|\Delta\theta_1^2| + |\Delta\theta_2^2| + 2|\Delta\theta_1||\Delta\theta_2|}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}} = \frac{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)(\cos^2(\phi') + \sin^2(\phi') + 2|\cos(\phi')||\sin(\phi')|)}{(|\Delta\theta_1^2| + |\Delta\theta_2^2|)^{\frac{3}{2}}} \leq \frac{2}{\sqrt{|\Delta\theta_1^2| + |\Delta\theta_2^2|}}, \tag{A.61}$$

so we choose $\phi' = \pi/4$. In addition, the error due to the lens center is also maximized if we maximize both center coordinates simultaneously. As a result, we have

$$|\Delta\theta_1| = |\Delta\theta_2|,$$
$$|\theta_{\text{center},1}| = |\theta_{\text{center},2}|. \tag{A.62}$$

This choice also fixes the absolute value of the gradient in $\Delta\theta_1$ and $\Delta\theta_2$ direction,

$$|\nabla\Psi'_{\epsilon,1}| = |(\nabla\psi)_1| = \frac{\theta_E}{\sqrt{2}},$$
$$|\nabla\Psi'_{\epsilon,2}| = |(\nabla\psi)_2| = \frac{\theta_E}{\sqrt{2}}. \tag{A.63}$$

We have used an upper limit of θ_E for the gradient value in the D error variables which we can now replace with $\theta_E/\sqrt{2}$. Inserting these results into Eq. (A.58), we have

$$|D_1| + |D_2| = \theta_E\left[\frac{6}{\sqrt{2}} + (6+2\pi)\sqrt{2} + \frac{16|\theta_{\text{center},1}|}{\sqrt{2}|\Delta\theta_1|} + \sqrt{2}(4.5+\pi) + \frac{6+2\pi}{\sqrt{2}}\right] = \theta_E\left(16.5\sqrt{2} + 4\pi\sqrt{2} + \frac{8\sqrt{2}|\theta_{\text{center},1}|}{|\Delta\theta_1|}\right), \tag{A.64}$$

and we obtain

$$F = 2\pi\frac{\sqrt{2}\theta_E}{2} + \theta_E\left(16.5\sqrt{2} + 4\pi\sqrt{2} + \frac{8\sqrt{2}|\theta_{\text{center},1}|}{|\Delta\theta_1|}\right) + 2\frac{\sqrt{2}\theta_E}{2} \approx 47\theta_E + \frac{8\sqrt{2}|\theta_{\text{center},1}|}{|\Delta\theta_1|}\theta_E. \tag{A.65}$$

We see that the assumption that the machine precision error is invariant under translations and rotations of the coordinate system is approximately correct far away from the lens center. The computed correction factor is less than two. However, close to the center we obtain a divergent correction term. This divergence is not a problem, as the SIE lens model itself has a divergence at the center, see e.g. Eq. (8). This unrealistic property of the SIE model is well known and other, more realistic parametric lens models do not suffer from this divergence at the center. Therefore neither single nor double precision are accurate enough at the center, but since the SIE is not a realistic lens model at its center, this is perfectly acceptable.

First, we compute the error variable F' without the $1/|\Delta\theta_1|$ term for a cluster-scale halo with $\theta_E = 20$ arcsec,

$$F'_{\text{cluster-scale}} = 940 \text{ arcsec}, \tag{A.66}$$

and for a galaxy-scale halo with $\theta_E = 0.2$ arcsec,

$$F'_{\text{galaxy-scale}} = 9.4 \text{ arcsec}. \tag{A.67}$$

For single and double precision, we have respectively $\epsilon_{SP} \approx 1.2 \times 10^{-7}$ and $\epsilon_{DP} \approx 2.2 \times 10^{-16}$, and thus the upper error bounds

$$\epsilon_{SP}F'_{\text{cluster-scale}} \approx 1.1 \times 10^{-4} \text{ arcsec},$$
$$\epsilon_{SP}F'_{\text{galaxy-scale}} \approx 1.1 \times 10^{-6} \text{ arcsec}, \tag{A.68}$$
$$\epsilon_{DP}F'_{\text{cluster-scale}} \approx 2.1 \times 10^{-13} \text{ arcsec},$$
$$\epsilon_{DP}F'_{\text{galaxy-scale}} \approx 2.1 \times 10^{-15} \text{ arcsec}. \tag{A.69}$$

Now we maximize the magnitude of the $1/|\Delta\theta_1|$ correction term by assuming a lens center

$$(\theta_{\text{center},1}, \theta_{\text{center},2}) = (200 \text{ arcsec}, 200 \text{ arcsec}). \tag{A.70}$$

The smallest non-divergent separation from the lens center is one pixel and for a HST ACS image with a pixel size of 0.03 arcsec we obtain a correction

$$F^C_{\text{cluster-scale}} \approx 1.5 \times 10^6 \text{ arcsec},$$
$$F^C_{\text{galaxy-scale}} \approx 1.5 \times 10^4 \text{ arcsec}, \tag{A.71}$$

and thus

$$\epsilon_{SP}F^C_{\text{cluster-scale}} \approx 1.8 \times 10^{-1} \text{ arcsec},$$
$$\epsilon_{SP}F^C_{\text{galaxy-scale}} \approx 1.8 \times 10^{-3} \text{ arcsec}, \tag{A.72}$$
$$\epsilon_{DP}F^C_{\text{cluster-scale}} \approx 3.3 \times 10^{-10} \text{ arcsec},$$
$$\epsilon_{DP}F^C_{\text{galaxy-scale}} \approx 3.3 \times 10^{-12} \text{ arcsec}. \tag{A.73}$$

The accuracy requirement computed in Section 3 shows that single precision is not accurate enough very close to the center of an isolated cluster lens, even in the absence of a magnification M_i. For an isolated galaxy lens, it is sufficient close to the center as long as $M_i \leq 4$. Therefore we will use double precision to compute the gradients in a pixel grid of 400×400 pixels centered on the respective cluster lens halos and in a grid of 20×20 pixels centered on the respective galaxy lens halos. For a HFF-like lens with 700 galaxy-scale halos and two cluster-scale halos we thus have to use double precision for 6×10^5 pixels out of a total of 45×10^6 pixels. This corresponds to approximately 1% of all image pixels. The correction terms for cluster-scale and galaxy-scale halos at a separation of 201 pixels and 11 pixels are respectively

$$F^C_{\text{cluster-scale}} \approx 7505 \text{ arcsec},$$
$$F^C_{\text{galaxy-scale}} \approx 1371 \text{ arcsec}, \tag{A.74}$$

and thus we have

$$F_{\text{cluster-scale}} \approx 8445 \text{ arcsec},$$
$$F_{\text{galaxy-scale}} \approx 1381 \text{ arcsec}, \tag{A.75}$$

and

$$\epsilon_{SP}F_{\text{cluster-scale}} \approx 1.0 \times 10^{-3} \text{ arcsec},$$
$$\epsilon_{SP}F_{\text{galaxy-scale}} \approx 1.7 \times 10^{-4} \text{ arcsec}, \tag{A.76}$$
$$\epsilon_{DP}F_{\text{cluster-scale}} \approx 1.9 \times 10^{-12} \text{ arcsec},$$
$$\epsilon_{DP}F_{\text{galaxy-scale}} \approx 3.0 \times 10^{-13} \text{ arcsec}. \tag{A.77}$$

The computed gradients for each halo are finally added up to obtain the total gradient,

$$\nabla\Psi_{\epsilon,i} = \sum_k \nabla\Psi'_{\epsilon,i,k}, \tag{A.78}$$

and as a result, the respective errors are combined as well. However, the respective errors can have different signs and magnitudes, so we expect to see some error cancellation. We estimate the total gradient error in the following way: We add the error contributions of two galaxy-scale lenses including upper bounds on the correction terms, but we neglect the remaining galaxy-scale halos and we add the respective upper error bounds of the cluster-scale halos. Neglecting the remaining galaxy-scale lenses is justified, because the dominating correction term decreases quickly with separation from the lens center and we expect only very few galaxies to be so close to each other that their respective correction terms are non-negligible and add up. The errors without correction term are three orders of magnitude smaller than those of the cluster-scale halos and we add many of these lenses, which are usually scattered throughout the image, so we expect significant error cancellation effects. The error contribution from the typically two cluster-scale halos will depend on their respective parameters. To obtain an upper bound, we will add up the respective upper bounds on the gradient. In total, we have

$$\begin{aligned} \Delta(\nabla\Psi_{\epsilon,i})_{SP} &\approx 2.3 \times 10^{-3} \text{ arcsec}, \\ \Delta(\nabla\Psi_{\epsilon,i})_{DP} &\approx 4.4 \times 10^{-12} \text{ arcsec}. \end{aligned} \tag{A.79}$$

References

Planck Collaboration, Ade, P.A.R., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Banday, A.J., Barreiro, R.B., Bartlett, J.G., et al., 2016. Planck 2015 results. XIII. Cosmological parameters. Astron. Astrophys. 594, A13. doi:10.1051/0004-6361/201525830, arXiv:1502.01589.

Atek, H., Richard, J., Jauzac, M., Kneib, J.P., Natarajan, P., Limousin, M., Schaerer, D., Jullo, E., Ebeling, H., Egami, E., Clement, B., 2015. Are ultra-faint galaxies at z = 6-8 re-sponsible for cosmic reionization? Agron. J. 814, 69. doi:10.1088/0004-637X/814/1/69, arXiv:1509.06764.

Avila, R., Grogin, N., Anderson, J., Bellini, A., Bohlin, R., Borncamp, D., Chiaberge, M., Coe, D., Hoffman, S., Kozhurina-Platais, V., Lucas, R., Maybhate, A., McMaster, M., Miles, N., Ryon, J., 2017. Advanced Camera for Surveys Instrument Handbook, Version 16.0. STScI, Baltimore, URL http://adsabs.harvard.edu/abs/2017acsi.book..A.

Bartelmann, M., Schneider, P., 2001. Weak gravitational lensing. Phys. Rep. 340, 291–472. doi:10.1016/S0370-1573(00)00082-X, arXiv:astro-ph/9912508, URL http://adsabs.harvard.edu/abs/2001PhR...340..291B.

Besl, P., 2013. A case study comparing arrays of structures and structures of arrays data layouts for a compute-intensive loop run on intel xeon processors and intel xeon phi product family coprocessors. (Online; Accessed 29 September 2017). URL https://software.intel.com/sites/default/files/article/392271/aos-to-soa-optimizations-using-iterative-closest-point-mini-app.pdf.

Bonvin, V., Courbin, F., Suyu, S.H., Marshall, P.J., Rusu, C.E., Sluse, D., Tewes, M., Wong, K.C., Collett, T., Fassnacht, C.D., Treu, T., Auger, M.W., Hilbert, S., Koopmans, L.V.E., Meylan, G., Rumbaugh, N., Sonnenfeld, A., Spiniello, C., 2017. H0LiCOW - V. New COSMOGRAIL time delays of HE 0435-1223: H_0 to 38 per cent precision from strong lensing in a flat Λcdm model. Mon. Not. R. Astron. Soc. 465, 4914–4930. doi:10.1093/mnras/stw3006, arXiv:1607.01790.

Bradač, M., Allen, S.W., Treu, T., Ebeling, H., Massey, R., Morris, R.G., von der Linden, A., Applegate, D., 2008. Revealing the properties of dark matter in the merging cluster MACS J0025.4-1222. Agron. J. 687, 959–967. doi:10.1086/591246, arXiv:0806.2320.

Cumming, B., Fourestey, G., Fuhrer, O., Gysi, T., Fatica, M., Schulthess, T.C., 2014. Application centric energy-efficiency study of distributed multi-core and hybrid CPU–GPU systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Press, pp. 819–829. doi:10.1109/SC.2014.72.

Eijkhout, V., van de Geijn, R., Chow, E., 2016. Introduction To High Performance Scientific Computing. doi:10.5281/zenodo.49897, lulu.com..

Elíasdóttir, Á., Limousin, M., Richard, J., Hjorth, J., Kneib, J.P., Natarajan, P., Pedersen, K., Jullo, E., Paraficz, D., 2007. Where is the matter in the merging cluster abell 2218? arXiv:0710.5636, ArXiv e-prints URL http://adsabs.harvard.edu/abs/2007arXiv0710.5636E.

Faber, S.M., Jackson, R.E., 1976. Velocity dispersions and mass-to-light ratios for elliptical galaxies. Agron. J. 204, 668–683. doi:10.1086/154215.

Goldberg, D., 1991. What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. 23, 5–48. doi:10.1145/103162.103163, URL http://doi.acm.org/10.1145/103162.103163.

Golse, G., Kneib, J.P., 2002. Pseudo elliptical lensing mass model: Application to the NFW mass distribution. Astron. Astrophys. 390, 821–827. doi:10.1051/0004-6361:20020639, arXiv:astro-ph/0112138.

Harris, M., 2016. Mixed-precision programming with CUDA 8. (Online; Accessed 2 October 2017). URL https://devblogs.nvidia.com/parallelforall/mixed-precision-programming-cuda-8/.

Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Comput. Sci. Eng. 9, 90–95. doi:10.1109/MCSE.2007.55.

Institute of Electrical and Electronics Engineers, 2008. IEEE Standard 754-2008 - IEEE Standard for Floating-Point Arithmetic. IEEE, doi:10.1109/IEEESTD.2008.4610935.

Intel Corporation, 2014. Intel xeon processor E5-2680 v3 specifications. (Online; Accessed 2 October 2017). URL https://ark.intel.com/products/81908/Intel-Xeon-Processor-E5-2680-v3-30M-Cache-2_50-GHz.

Ishigaki, M., Kawamata, R., Ouchi, M., Oguri, M., Shimasaku, K., Ono, Y., 2015. Hubble Frontier fields first complete cluster data: Faint galaxies at z $\tilde{5}$-10 for UV luminosity functions and cosmic reionization. Agron. J. 799, 12. doi:10.1088/0004-637X/799/1/12, arXiv:1408.6903.

Jauzac, M., Clément, B., Limousin, M., Richard, J., Jullo, E., Ebeling, H., Atek, H., Kneib, J.P., Knowles, K., Natarajan, P., Eckert, D., Egami, E., Massey, R., Rexroth, M., 2014. Hubble Frontier fields: a high-precision strong-lensing analysis of galaxy cluster MACSJ0416.1-2403 using ∼200 multiple images. Mon. Not. R. Astron. Soc. 443, 1549–1554. doi:10.1093/mnras/stu1355, arXiv:1405.3582.

Jauzac, M., Richard, J., Jullo, E., Clément, B., Limousin, M., Kneib, J.P., Ebeling, H., Natarajan, P., Rodney, S., Atek, H., Massey, R., Eckert, D., Egami, E., Rexroth, M., 2015. Hubble Frontier fields: a high-precision strong-lensing analysis of the massive galaxy cluster Abell 2744 using ∼180 multiple images. Mon. Not. R. Astron. Soc. 452, 1437–1446. doi:10.1093/mnras/stv1402, arXiv:1409.8663.

Jullo, E., Kneib, J.P., Limousin, M., Elíasdóttir, Á., Marshall, P.J., Verdugo, T., 2007. A Bayesian approach to strong lensing modelling of galaxy clusters. New J. Phys. 9, 447. doi:10.1088/1367-2630/9/12/447, arXiv:0706.0048, URL http://adsabs.harvard.edu/abs/2007NJP....9..447J.

Jullo, E., Natarajan, P., Kneib, J.P., D'Aloisio, A., Limousin, M., Richard, J., Schimd, C., 2010. Cosmological constraints from strong gravitational lensing in clusters of galaxies. Science 329, 924–927. doi:10.1126/science.1185759, arXiv:1008.4802.

Kassiola, A., Kovner, I., 1993. Elliptic mass distributions versus elliptic potentials in gravitational lenses. Agron. J. 417, 450. doi:10.1086/173325, URL http://adsabs.harvard.edu/abs/1993ApJ.417.450K.

Kneib, J.P., Ellis, R.S., Smail, I., Couch, W.J., Sharples, R.M., 1996. Hubble space telescope observations of the lensing cluster Abell 2218. Agron. J. 471, 643. doi:10.1086/177995, arXiv:astro-ph/9511015.

Kneib, J.P., Natarajan, P., 2011. Cluster lenses. Astron. Astrophys. Rev. 19, 47. doi:10.1007/s00159-011-0047-3, arXiv:1202.0185, URL http://adsabs.harvard.edu/abs/2011A%26ARv..19...47K.

Limousin, M., Richard, J., Jullo, E., Jauzac, M., Ebeling, H., Bonamigo, M., Alavi, A., Clément, B., Giocoli, C., Kneib, J.P., Verdugo, T., Natarajan, P., Siana, B., Atek, H., Rexroth, M., 2016. Strong-lensing analysis of MACS j0717.5+3745 from hubble frontier fields observations: How well can the mass distribution be constrained? Astron. Astrophys. 588, A99. doi:10.1051/0004-6361/201527638, arXiv:1510.08077.

Lotz, J.M., Koekemoer, A., Coe, D., Grogin, N., Capak, P., Mack, J., Anderson, J., Avila, R., Barker, E.A., Borncamp, D., Brammer, G., Durbin, M., Gunning, H., Hilbert, B., Jenkner, H., Khandrika, H., Levay, Z., Lucas, R.A., MacKenty, J., Ogaz, S., Porterfield, B., Reid, N., Robberto, M., Royle, P., Smith, L.J., Storrie-Lombardi, L.J., Sunnquist, B., Surace, J., Taylor, D.C., Williams, R., Bullock, J., Dickinson, M., Finkelstein, S., Natarajan, P., Richard, J., Robertson, B., Tumlinson, J., Zitrin, A., Flanagan, K., Sembach, K., Soifer, B.T., Mountain, M., 2017. The Frontier fields: Survey design and initial results. Agron. J. 837, 97. doi:10.3847/1538-4357/837/1/97, arXiv:1605.06567.

Meneghetti, M., Natarajan, P., Coe, D., Contini, E., De Lucia, G., Giocoli, C., Acebron, A., Borgani, S., Bradac, M., Diego, J.M., Hoag, A., Ishigaki, M., Johnson, T.L., Jullo, E., Kawamata, R., Lam, D., Limousin, M., Liesenborgs, J., Oguri, M., Sebesta, K., Sharon, K., Williams, L.L.R., Zitrin, A., 2016. The Frontier fields lens modeling comparison project. arXiv:1606.04548, ArXiv e-prints.

Natarajan, P., Kneib, J.P., Smail, I., Ellis, R.S., 1998. The mass-to-light ratio of early-type galaxies: Constraints from gravitational lensing in the rich cluster AC 114. Agron. J. 499, 600–607, arXiv:astro-ph/9706129.

Navarro, J.F., Frenk, C.S., White, S.D.M., 1996. The structure of cold dark matter halos. Agron. J. 462, 563. doi:10.1086/177173, arXiv:astro-ph/9508025, URL http://adsabs.harvard.edu/abs/1996ApJ...462..563N.

Navarro, J.F., Frenk, C.S., White, S.D.M., 1997. A universal density profile from hierarchical clustering. Agron. J. 490, 493. doi:10.1086/304888, arXiv:astro-ph/9611107, URL http://adsabs.harvard.edu/abs/1997ApJ...490..493N.

Nvidia Corporation, 2016. Nvidia Tesla P100 whitepaper. (Online; Accessed 2 October 2017). URL https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf.

Nvidia Corporation, 2017a. Nvidia GeForce GTX 1080 Ti specifications. (Online; Accessed 2 October 2017). URL https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/.

Nvidia Corporation, 2017b. Nvidia tesla V100 GPU architecture whitepaper. (Online; 8 Accessed May 2018). URL https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

Postman, M., Coe, D., Benítez, N., Bradley, L., Broadhurst, T., Donahue, M., Ford, H., Graur, O., Graves, G., Jouvel, S., Koekemoer, A., Lemze, D., Medezinski, E., Molino, A., Moustakas, L., Ogaz, S., Riess, A., Rodney, S., Rosati, P., Umetsu, K., Zheng, W., Zitrin, A., Bartelmann, M., Bouwens, R., Czakon, N., Golwala, S., Host, O., Infante, L., Jha, S., Jimenez-Teja, Y., Kelson, D., Lahav, O., Lazkoz, R., Maoz, D., McCully, C., Melchior, P., Meneghetti, M., Merten, J., Moustakas, J., Nonino, M., Patel, B., Regös, E., Sayers, J., Seitz, S., Van der Wel, A., 2012. The cluster lensing and supernova survey with Hubble: An overview. Agron. J. 199, 25. doi:10.1088/0067-0049/199/2/25, arXiv:1106.3328.

Randall, S.W., Markevitch, M., Clowe, D., Gonzalez, A.H., Bradač, M., 2008. Constraints on the self-interaction cross section of dark matter from numerical simulations of the merging galaxy cluster 1E 0657-56. Agron. J. 679, 1173–1180. doi:10.1086/587859, arXiv:0704.0261.

Astropy Collaboration, Robitaille, T.P., Tollerud, E.J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., Davis, M., Ginsburg, A., Price-Whelan, A.M., Kerzendorf, W.E., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M.M., Nair, P.H., Unther, H.M., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J.E.H., Singer, L., Fox, R., Weaver, B.A., Zabalza, V., Edwards, Z.I., Azalee Bostroem, K., Burke, D.J., Casey, A.R., Crawford, S.M., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P.L., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., Streicher, O., 2013. Astropy: A community python package for astronomy. Astron. Astrophys. 558, A33. doi:10.1051/0004-6361/201322068, arXiv:1307.6212.

Schneider, P., Kochanek, C.S., Wambsganss, J., 2006. Gravitational Lensing: Strong, Weak and Micro. Saas-Fee Advanced Course 33. Springer Berlin Heidelberg, doi:10.1007/978-3-540-30310-7.

Tessore, N., Bellagamba, F., Metcalf, R.B., 2016. LENSED: a code for the forward reconstruction of lenses and sources from strong lensing observations. Mon. Not. R. Astron. Soc. 463, 3115–3128. doi:10.1093/mnras/stw2212, arXiv:1505.07674.

6.3 Lenstool-HPC: A High Performance Computing based mass modelling tool for cluster-scale gravitational lenses

6.3.1 Preface

The following paper is the presentation of the first version of Lenstool-HPC. We showcase the computational capabilities of Lenstool-HPC's deflection angle gradient and fit computation using extensive benchmarks run on the Swiss national computing center (CSCS) clusters in Lugano. It gives details on how we achieve these speed-ups using vectorisation, efficient use of hardware and an hybrid GPU-CPU implementation.

6.3.2 Paper

This chapter is presented in the form of a published paper as Christoph Schaefer, Gilles Fourestey, and Jean-Paul Kneib, "Lenstool-HPC: A High Performance Computing based mass modelling tool for cluster-scale gravitational lenses", Astronomy and Computing, [Volume 30, January 2020, 100360].

Astronomy and Computing 30 (2020) 100360

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

Lenstool-HPC: A High Performance Computing based mass modelling tool for cluster-scale gravitational lenses

C. Schäfer [a,*], G. Fourestey [b], J.-P. Kneib [a,c]

[a] *Institute of Physics, Laboratory of Astrophysics, Ecole Polytechnique Fédérale de Lausanne (EPFL), Observatoire de Sauverny, 1290 Versoix, Switzerland*
[b] *SCITAS, Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland*
[c] *Aix Marseille Université, CNRS, LAM (Laboratoire d'Astrophysique de Marseille) UMR 7326, 13388, Marseille, France*

ARTICLE INFO

Article history:
Received 4 September 2019
Accepted 16 December 2019
Available online 24 December 2019

Keywords:
Gravitational lensing software
High performance computing algorithms
Applied computing: astronomy
Galaxies: clusters
Galaxies: halos
Lenstool

ABSTRACT

With the upcoming generation of telescopes, cluster scale strong gravitational lenses will act as an increasingly relevant probe of cosmology and dark matter. The better resolved data produced by current and future facilities requires faster and more efficient lens modelling software. Consequently, we present *Lenstool-HPC*, a strong gravitational lens modelling and map generation tool based on High Performance Computing (HPC) techniques and the renowned *Lenstool* software. We also showcase the HPC concepts needed for astronomers to increase computation speed through massively parallel execution on supercomputers. *Lenstool-HPC* was developed using lens modelling algorithms with high amounts of parallelism. Each algorithm was implemented as a highly optimised CPU, GPU and Hybrid CPU–GPU version. The software was deployed and tested on the Piz Daint cluster of the Swiss National Supercomputing Centre (CSCS). *Lenstool-HPC* perfectly parallel lens map generation and derivative computation achieves a factor 30 speed-up using only 1 GPU compared to *Lenstool*. *Lenstool-HPC* hybrid Lens-model fit generation tested at Hubble Space Telescope precision is scalable up to 200 CPU–GPU nodes and is faster than *Lenstool* using only 4 CPU–GPU nodes.

1. Introduction

With the advent of high-precision astronomy and big data, high performance computing (HPC) has reached a critical importance for astrophysicists. Astrophysical codes developed over 10 years ago are not able to keep up with the amount of data that new instruments are bringing in. To handle these new challenges it is now necessary to implement HPC techniques and alternative thinking to speed up these softwares. One such example is *Lenstool*, a mass modelling tool for strong gravitational lenses. These lenses are rare astrophysical phenomena where a distant light-source is aligned so closely with a foreground galaxy or cluster that its images appear to an Earth observer multiple times. The images appear distorted and magnified similar to objects seen through an unfocused lens. They take the shape of distorted arcs, multiple images and Einstein rings. These distortions are due solely to the gravitational potential of the foreground galaxies or cluster which acts as a lens. This allows specialised mass-modelling software like *Lenstool*[1] (Jullo et al., 2007; Kneib et al., 1996) to create precise mass-models of the lenses by fitting parametric mass-models (Limousin et al., 2008; Richard et al., 2011; Jauzac et al., 2015) using Bayesian MCMC samplers (see Fig. 1 in Jauzac et al. (2014)).

The astrophysical interests are multiple. They are used to study the dark matter profile of lensing galaxies (Jauzac et al., 2015) and calculate the dark-baryonic matter ratio (Jiang and Kochanek, 2007; More et al., 2011; Sonnenfeld et al., 2015). Lensed Quasars are used for time-delay studies which constrain the Hubble constant (Bonvin et al., 2016; Suyu et al., 2017) and the magnification effect of gravitational lenses allows for the study of high-redshift background objects (Kneib et al., 2004; Richard et al., 2011; Atek et al., 2015).

These precise mass-models are obtained by observers through an iterative process using *Lenstool*'s modelling capabilities repeatedly, adding new observational constraints. Using *Lenstool* however, especially on deep *Hubble* Space Telescope (HST) observations, is becoming extremely time-consuming possibly taking up to one month for one iteration. Beyond slowing down the release of precise mass-models, it severely limits the capability of observers to test new theories for the assembly of mass in galaxy-clusters.

To tackle this problem, we developed *Lenstool-HPC*, a new parallelism aware library which uses High Performance Computing (HPC) techniques to increase computation-speed by orders of

* Corresponding author.
E-mail address: christophernstrerne.schaefer@epfl.ch (C. Schäfer).

[1] Publicly available at https://projets.lam.fr/projects/lenstool/wiki.

https://doi.org/10.1016/j.ascom.2019.100360
2213-1337/

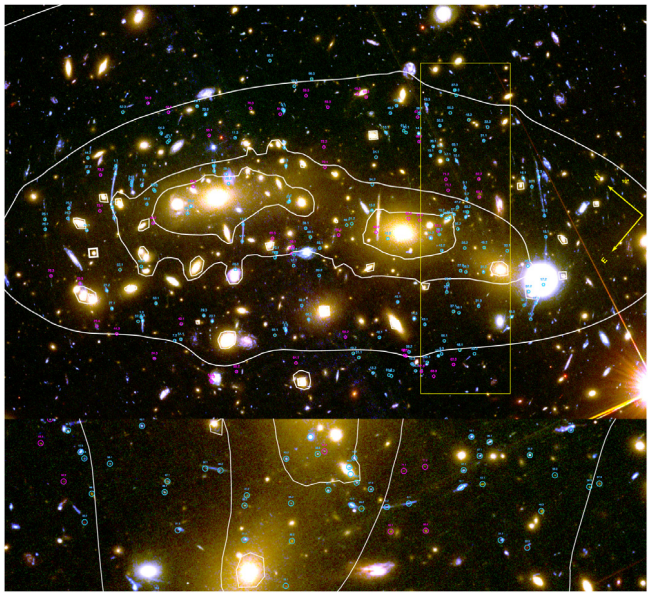


Fig. 1. MACSJ0416-2403: The cluster has 68 confirmed multiple lensed background sources. The isolines trace the distribution of matter in the cluster which were computed using *Lenstool*. The highlighted (green) rectangle represents a zoomframe of the cluster showing the fainter multiple images. Credit. Jauzac et al. (2014). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

magnitude through parallelisation. *Lenstool-HPC* was developed for CPUs and GPUs using CUDA and C++ in a collaboration between HPC experts and astrophysicists. The first section presents a brief overview of the theory behind gravitational lensing and *Lenstool* mass modelling algorithm and the computational challenge it poses. This is followed by a section summarising the HPC notions that defined the development of the library before presenting the library itself. We finish this paper with detailed benchmark results of the library, studying in particular the speed-up and scaling of the lensing map generation and mass modelling fit computation compared to *Lenstool* on modern CPU and GPU clusters.

2. Gravitational lens mass-modelling

2.1. Gravitational lens theory overview

A gravitational lens system of first order can be represented by projecting the lens and the source respectively on an image and source plane. We usually can assume that the size of the lens in the line-of-sight direction is negligible compared to the distance between observer, lens and source, this is the "thin-lens" approximation. Then the gravitational lensing phenomenon can be summarised by a simple trigonometric equation called the lens-equation:

$$\vec{\beta} = \vec{\theta} - \vec{\alpha}(\vec{\theta}) \quad , \tag{1}$$

where $\vec{\beta}$ is the angular position of the source in the source-plane and $\vec{\theta}$ the angular position of the image in the lens-plane. The deflection angle $\vec{\alpha}$ is the gradient of the lensing potential:

$$\psi(\vec{\theta}) = \frac{1}{\pi} \int_{\mathbb{R}^2} d^2\theta' \kappa(\vec{\theta}) \ln|\vec{\theta} - \vec{\theta}'| \quad , \tag{2}$$

where $\kappa(\vec{\theta})$ is the surface mass density of the lens-plane defined as

$$\kappa(\vec{\theta}) = \frac{\Sigma(D_d\vec{\theta})}{\Sigma_{crit}} \quad \text{with} \quad \Sigma_{\text{crit}} = \frac{c^2}{4\pi G} \frac{D_s}{D_l D_{ls}} \quad , \tag{3}$$

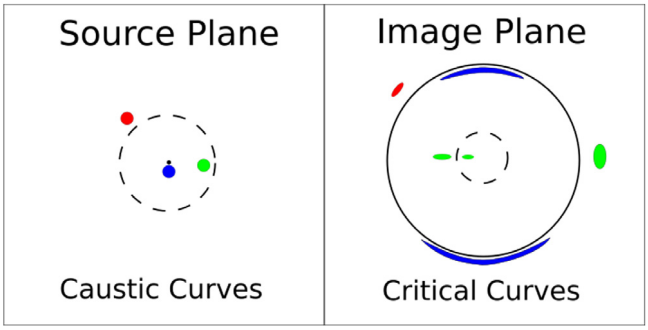


Fig. 2. Schematic of the lensed images formed by three sources due to the gravitation potential of a non singular isothermal sphere. The red source is outside the caustic lines therefore has only lensed image. The green source is inside a caustic line and is lensed three times. The blue source is almost perfectly aligned with the centre of the lens and is starting to form an Einstein ring.

and Σ_{crit} is the critical surface mass density. D_s, D_l and D_{ls} are respectively the distance from the observer to the source, to the lens and between lens and source. The lensing potential $\psi(\vec{\theta})$ is the normalised Newtonian gravitational potential, satisfying the relations $\vec{\alpha} = \nabla\psi$ and $\kappa = \nabla^2\psi$.

The distortion of the images described by the following Jacobian matrix (the magnification matrix) is derived from the lens equation:

$$\vec{A}^{-1}(\vec{\theta}) = \frac{\partial\vec{\beta}}{\partial\vec{\theta}} = (\delta_{ij} - \frac{\partial^2\psi(\vec{\theta})}{\partial\theta_i\partial\theta_j}) = \begin{pmatrix} 1-\kappa-\gamma_1 & -\gamma_2 \\ -\gamma_2 & 1-\kappa+\gamma_1 \end{pmatrix} \quad , \tag{4}$$

where γ_1 and γ_2 are the shear components, quantifying the amount and direction of the gravitational shear.

The magnification value is related to the determinant of the Jacobian matrix as

$$\mu(\vec{\theta_0}) = \frac{1}{\det(\vec{A}^{-1})}. \tag{5}$$

The points where $\det(\vec{A}^{-1}) = 0$ form the critical lines where the magnification is theoretically infinite. In practice the wave-nature of light leads to finite amplification. Their unlensed counter-part in the source plane are called caustics. These caustics set the boundaries of areas where the image of a source is not just distorted but also multiplied. Every source which moves across will have two more or less lensed images (Fig. 2). More details on lensing theory can be found in Bartelmann and Schneider (2001).

2.2. Mass-modelling

Lenstool (Kneib et al., 1996; Jullo et al., 2007; Jullo and Kneib, 2009) creates mass-models for lensing cluster by fitting parametric mass-models to each individual cluster sub-halos. Depending on the parametric model used, the free parameters can vary. They include generally the position of the centre of the sub-halo, its dispersion velocity, its ellipticity and orientation, as well as other free parameters specific to the model (Elíasdóttir et al., 2007) in particular related to the mass profile. In clusters of galaxies, the main constraints used for fitting are the position of identified multiple lensed images.

Each image is unlensed onto the source-plane using the mass-model to be tested. In the case of a perfect model all images should end up at the same point in a source plane. However, in practice the model is off, so the corresponding sources of the multiple-images are at slightly different positions. Sending back

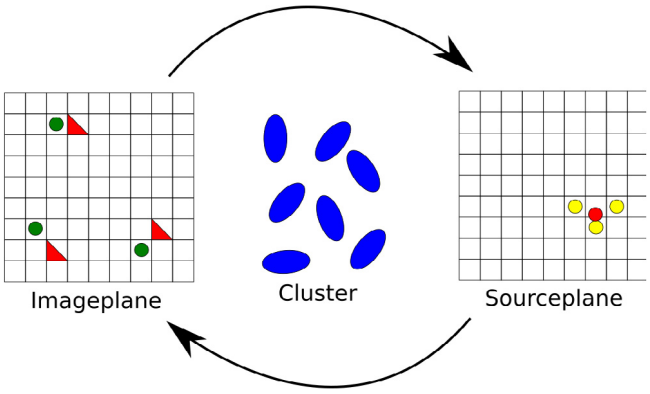


Fig. 3. *Lenstool* Mass modelling: Multiply imaged sources (green dots) work as constraints. Each image position is lensed onto the source-plane (yellow dots) using the current mass-model. The barycentre of these constraints (red dot) is taken as the best approximation of the source position and then lensed back into the lens-plane (red triangles). The difference between the constraints and lensed back source approximation gives an approximation of the fit of the mass-model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

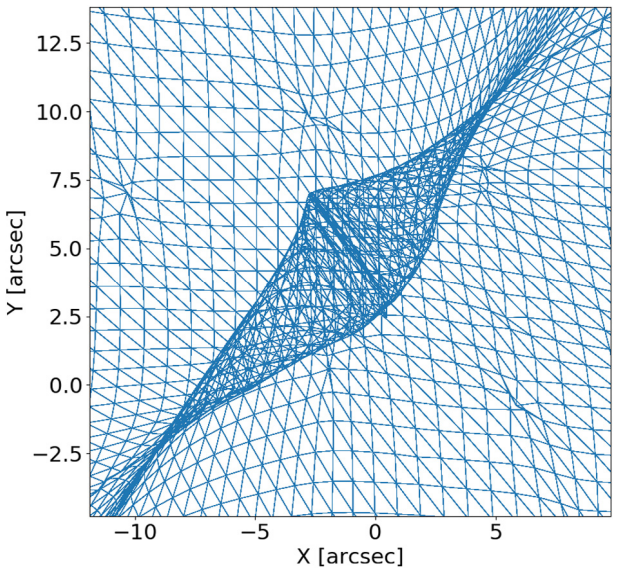


Fig. 4. Graphical representation of the unlensing of the quadratic triangular grid from the image-plane unto the source plane. The lines which delimit the area where the grid folds unto itself (where therefore multiple images can be found) are the caustic lines.

the barycentre of these positions to the image plane (see Fig. 3), we can define a cost-function that *Lenstool* will try to minimise:

$$\chi^2 = \sum_i^N \chi_i^2 = \sum_i^N \sum_j^{M_i} \frac{(c_{ij} - x_{ij})^2}{\sigma_{ij}^2} \tag{6}$$

where N is the number of lensed sources, M_i the multiplicity of those sources, c_{ij} the multiple-image constraints, x_{ij} the back and forth lensed constraints and σ_{ij}^2 the error-budget. The exploration of the parameter space and of the optimum solution is done using a Bayesian Markov Chain Monte Carlo Algorithm (MCMC). More details on the procedure can be found in Jullo et al. (2007) and Kneib et al. (1996).

The number of optimised free parameters depends on the parametric model used but can range up to a thousand for a typical cluster-lens model (see Jauzac et al. (2014, 2015)). In the high dimensionality of the problem lies the first computational challenge from *Lenstool*. Even using a Bayesian MCMC algorithm, *Lenstool* has to try an enormous amount of parameter-combinations to find solutions that minimise the cost-function.

2.3. *Chi*^{2} Computation

The second computational challenge is the *Chi*^{2} computation based on the unlensing and relensing of multiple imaged sources. Unlensing a point into the source-plane is a simple but non revertible application of the lens-equation (Eq. (1)). The multiple solutions for the relensing problem can as a consequence not be computed analytically. To compute predicted multiple images of a source, the "brute force" approach is to unlens a image-plane grid unto the source-plane and check each quadrant for the presence of the source (see Fig. 4). *Lenstool* avoids this computationally costly approach by using a variant of the "image-transport" method (Schneider et al., 1992).

The method works as follows: It defines a triangle around the constraint that does not contain any other constraint but is likely to contain the source. The triangle is then subdivided into 4 smaller triangles. Each triangle is checked for the source. If a triangle containing the source is not found, the immediate environment of the triangle is searched. The subdividing process is continued recursively until a precision of 10^{-4} arcsec is reached. While a lot faster than the brute force approach, this method is not fully stable. Extremely strong amplification near the critical lines can degrade the zoom-in process sufficiently to lose images which can complicate the modelling process.

2.4. Lensing maps

The third computational challenge we address is the computation of lensing maps. Lensing maps are used to visualise crucial information of the lens-system. Each map is organised into a rectangular grid defined on the image-plane, each grid cell usually being the size of a pixel of image data. Information that can be visualised are the projected surface mass density κ of the lenses, the projected shear γ (norm, direction, individual components), the amplification μ or its inverse, the lens deflection field, the lensing potential φ, the time delay surface and variations thereof. More information on these maps can be found at: https://projets.lam.fr/projects/lenstool/wiki.

Lensing-maps are also used to calculate the statistical error inherent to the Bayesian process. In order to compute the map variances, full-resolution lensing maps have to be generated for each tested parameter-combination which is a time-consuming task.

The critical part of the lensing map computation is the second order derivatives of the gravitational potentials of each cluster member, which allow to compute κ, γ and μ:

$$\kappa(x,y) = \sum_i^{N_h} \frac{1}{2} \left(\partial_{xx}\phi(x,y) + \partial_{yy}\phi(x,y)\right) \tag{7}$$

$$\gamma^2(x,y) = \sum_i^{N_h} \frac{1}{4} \left(\left(\partial_{xx}\phi(x,y) - \partial_{yy}\phi(x,y)\right)^2 + \left(\partial_{xy}\phi(x,y)\right)^2\right) \tag{8}$$

$$\mu = ((1-\kappa)^2 + \gamma^2)^{-1} \tag{9}$$

with N_h the number of halos, which includes the large scale components and the sub-halos (attached to each cluster galaxy). At each grid-point the second-order derivative contribution of each cluster member is added up to compute the total derivative. The advantage of *Lenstool*'s parametric mass-models is that their

single and double derivative can be explicitly calculated through analytical function rather that through a numerical calculation. This makes the computation of the various lensing properties fast, as analytically calculated gradient are faster to compute and do not suffer the numerical errors introduced by numerical derivation and interpolation.

Despite this advantage, the computational challenge is impressive. For Abell 2744 (Jauzac et al., 2015) error calculation (one of the Hubble Frontier Field Cluster [HFF]), 10 014 maps with 6000×6000 pixels had to be generated. With 258 parametric potentials this corresponds to 10^9 derivatives per map for a grand total of 10^{14} derivative computations. The total process adds up to a total of 300 CPU hours using *Lenstool* just for the map generation.

3. High Performance Computing (HPC)

Due to the impossibility of increasing much further the clock-frequency of processors (Mudge, 2001), hardware development focus has gone into integrating multiple cores capable of multiple simultaneous operations. This was motivated by Little's Law (Bailey, 1997), which states that the performance of computation can be increased through parallel execution. In other words performance can be improved by distributing the work on multiple computation units. Multicore CPUs and GPUs are the consequences of this design choice. This increasingly parallel execution orientated development does not work well with parallelism unaware (often serial) algorithms like *Lenstool*'s Image transport method which lack the necessary concurrency for parallel execution. This creates large performance gaps called the Ninja Gap (Satish et al., 2012).

The following section introduces a few essential concepts of High Performance Computing (HPC) necessary to understand how to remove this gap: (1) how performance for software is defined and can be improved and (2) how to implement the different parallelism strategies on CPU and GPUs.

3.1. Software performance and parallelism

The performance of a software, better known as its throughput, is defined as the number of Floating-point operation per second [flop/s] it is capable of performing. Little's Law states that the throughput ([flop/s]) of a computation is equal to the level of parallel computation instances divided by the latency ([s]). Latency is defined as the time of a single computation instance to process and store an operation. The amount of parallelism that a software can reach is directly related to the level of concurrency the underlying algorithm possesses where concurrency refers to the ability of an algorithm to execute parts of itself out of order without affecting the final outcome.

$$\text{Throughput} = \frac{\text{Parallelism}}{\text{Latency}}$$

The obvious consequence of Little's Law is that it is possible for software with high parallelism but also higher latency to achieve a higher performance than non parallel low latency software. To achieve optimum computation speed it is therefore necessary to choose carefully the underlining algorithms so as to be "parallelism aware" meaning balancing a high level of concurrency with low latency.

Increasing performance can therefore either be done by reducing latency or increasing concurrency to fully use the available parallel computation capabilities of the hardware. HPC tends to focus on the latter.

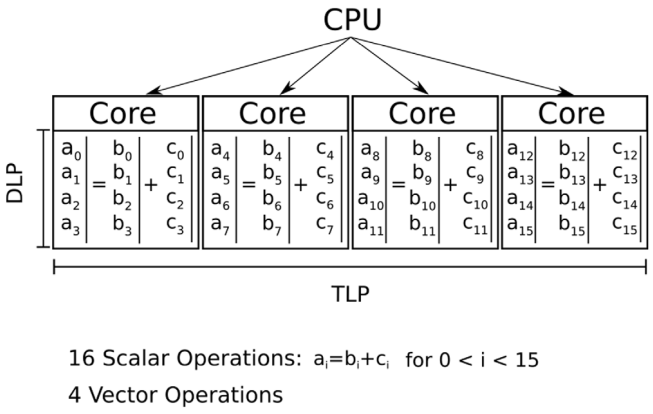


Fig. 5. Scaling at a node level: This example CPU is AVX capable, meaning each of his cores is capable computing 4 scalar operations in parallel. This CPU can compute 16 scalar operations distributed over 4 cores (TLP) in vectors of size 4 (DLP) simultaneously.

3.2. Hardware

Software computation speed is extremely dependent on the hardware it runs on. CPUs and GPU rely on different parallelism strategies to achieve an optimum throughput which need to be taken into account in the development. Multicore CPUs are mainly designed for single thread performance (Hölzle, 2010). Their lower latencies make them ideal for less parallelisable applications that use irregular patterns or data structures. GPUs in contrast are designed for massively parallel software. Their individual threads are slow but the much higher number of them allows to hide their high latency and achieve a high throughput on problems with a high number of simple and parallelisable computation.

3.2.1. Parallelism on CPUs

A CPU consists of multiple cores sharing memory, each capable of executing different independent tasks. Each core can also execute multiple operations simultaneously for the same task by generalising scalar operations to vectors and matrix operations. At a single CPU (node) level, parallelism is typically divided into three levels: Thread-level parallelism (TLP), Data-level parallelism (DLP) and Instruction-level parallelism (ILP).

TLP optimises the concurrent execution of tasks (threads) between the different cores, handled by libraries such as OpenMP, Intel's TBB or POSIX pthreads. It mainly handles the problems that come from sharing resources like the memory.

DLP handles the vectorisation of scalar operations on a single CPU core using Advanced Vector Extension (AVX). AVX2 and AVX-512 (Advanced Vectorisation Extension) capable CPUs can vectorise respectively 4 to 8 scalar operations through the SIMD (Single Instruction Multiple Data) programming model (Kreinin, 2011) (Fig. 5). This additional parallelism level comes theoretically at a low development cost. Most compilers are capable of doing implicit vectorisation without developer input by identifying vector operations in the algorithm (Intel, 2015). Vector operations however require that the AVX registers are loaded homogeneously with the necessary information using Data structures of type Structure of Array (SOA). Data structures of SOA type stores data of the same type into one parallel array contrary to the more conventional Array of structure (AOS) which interleaves the information (see Fig. 6) (Besl, 2013; Intel, 2013).

ILP leverage's the superscalar capabilities of modern CPUs, allowing multiple independent instructions to be handled at once. This is mainly handled by the compiler and falls outside of the scope of this paper (Intel, 2019).

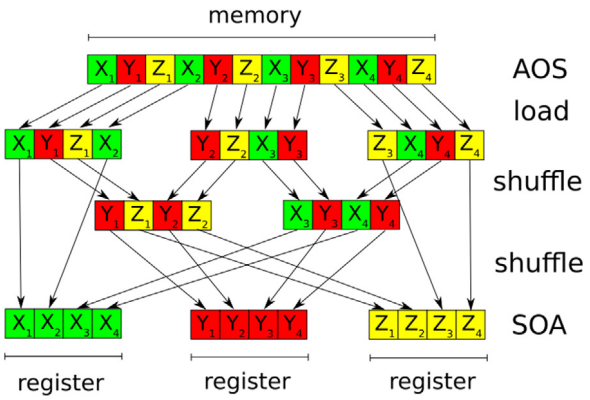


Fig. 6. Preparation for vectorisation with a heterogeneous memory and AOS structures: The CPU core first loads from the main memory the needed information into AVX registers. Those registers have to be shuffled multiple times to achieve the needed homogeneous layout. Once the computations are done, they have to be reshuffled back into the AOS structure. Beyond the obvious time loss, the compiler is not able to vectorise these operations automatically. If developers still wish to implement AOS structures, SIMD pragmas have to be used to vectorise the operations manually. X_i, Y_i and Z_i represent fictional position information.

3.2.2. Parallelism on GPUs

Originally developed for gaming, GPUs are composed of multiple Streaming Multiprocessors (SM) each consisting of multiple Streaming Processors (SP). SP are capable of computing arithmetic operations and are grouped together into warps which share instruction sets.

The important difference between GPUs and CPUs is that GPUs are not designed for single thread performance (Hölzle, 2010). GPU threads have much higher latencies than CPUs for floating point operations and memory transfer. To maximise throughput, GPUs are designed to be massively multithreaded. Using the SIMT (Single Instruction, Multiple Threads) programming model, GPUs have hardware threading support that allows hundreds of threads to be active simultaneously, each computing operations in parallel (Lindholm et al., 2008; Nvidia, 2012).

The downside of this approach is that if an algorithm has a low level of concurrency, its GPU throughput will be dominated by the high latency (Valkov, 2010; Liang et al., 2013). If the problem does not propose enough parallel computation to hide the high latency, computation speed will be extremely slow. This makes GPUs compared to CPUs limited in their choice of problems.

Another important aspect of GPU optimisation is paying attention to the ILP problems like divergent execution paths. CPU compilers tend to extract ILP more efficiently than GPUs using modern techniques like Out-Of-Order or speculative execution (Intel, 2019) without any developer input needed. While DLP is implicitly optimised by the SIMT model (Kreinin, 2011), ILP for GPUs has to be explicitly coded.

4. *Lenstool-HPC*

Lenstool is comprised of three crucial computations which constitute a bottleneck and can be parallelised: the computation of the deflection potential gradient over a grid, the computation of the χ^2 and the MCMC sampler. *Lenstool-HPC* has to date fully optimised the first two of those computations. The gradient computation over a grid is a trivially parallelisable problem with no need for communication between the different parallel tasks for which *Lenstool-HPC* proposes a CPU-OpenMP and a GPU based solution. It is vectorisable and has enough parallel computation to hide the GPU latencies. In contrast the computation of the χ^2 is a typical example of a non trivially parallelisable algorithm. It possesses divergent execution paths controlled by the presence of a source in a triangle, atomic operations which cannot be parallelised and imposes a certain amount of communication between the different tasks. For this *Lenstool-HPC* proposes a pure-CPU based and a mixed CPU–GPU implementation of the brute-force approach.

4.1. Gradient computation

Computing the various lensing maps or the brute force computation of the χ^2 necessitates the computation of the deflection potential gradient over the whole image. This is done by defining a rectangular grid over the image. For each point the gradient can be calculated analytically from the deflection potentials modelled by multiple parametric potentials. The total gradient in a certain point is simply the sum of the first order derivative of all parametric potentials at that specific point:

$$\nabla\phi(x,y) = \sum^{N_h} \phi_i(x,y) \tag{10}$$

where N_h is the number of parametric potentials.

The gradient computation of different points is independent of each other. Both the CPU-OpenMP and GPU implementation can therefore distribute the task of computing the gradient of a single point to separate computation units. The CPU version uses the implicit vectorisation capability of the intel compiler to additionally vectorise the gradient computation of a single point.

4.2. χ^2 Computation

In contrast to computing deflection gradients, the image transport method based χ^2 computation is extremely difficult to parallelise. To be able to efficiently distribute the work over multiple computation units, *Lenstool-HPC* therefore uses the more computationally intensive but less serial brute-force approach algorithm with GPUs. The algorithm can be subdivided into four main stages: The gradient computation of a grid, the source computation based of the constraints, finding images by delensing and checking the grid, and computing the χ^2 based on the found images.

The distribution of these tasks in *Lenstool-HPC* Hybrid CPU–GPU implementation is summarised in Fig. 7. The gradient computations are divided among the available GPUs. During that time the CPU computes the positions of the sources in the source-plane and sends the information to the GPUs. Once the gradients and the sources are known, the GPUs can start searching for the images by delensing and checking the grid for sources, again by subdividing the grid. Each image found is stored temporarily and at the end of the computation send to the CPU. This operation possesses a divergent execution path, based on if an image is found or not. As a consequence the computation incurs an overhead based on the different amount of found images in each GPUs operational territory. Once all images have been found and received, the master CPU assigns them to their closest constraint and then computes the χ^2.

The purely CPU-based implementation distributes the work similarly to the Hybrid CPU–GPU version with the exception that all CPU cores calculate the sources positions.

4.3. Implementation

We developed *Lenstool-HPC* to be similar to *Lenstool* to assure continuity for *Lenstool* users. *Lenstool-HPC* can be compiled as a library with the above mentioned functions and as an executable with the same image-plane mapping capabilities as *Lenstool*. All mapping methods have been tested against the corresponding

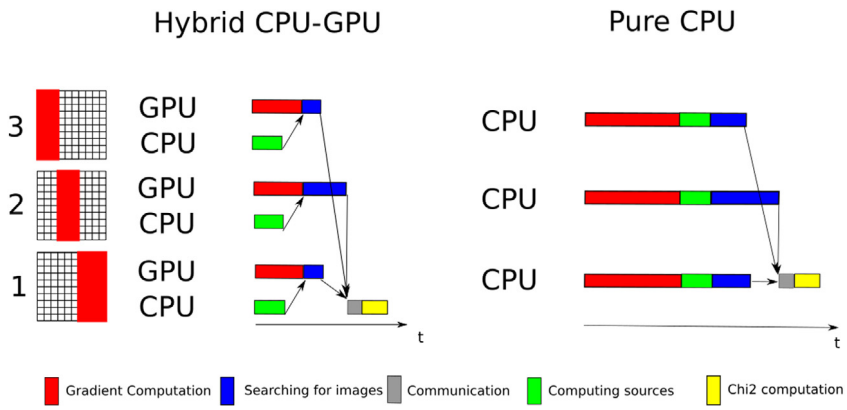


Fig. 7. Each GPU is assigned a part of the grid where it computes the gradient. During that time the CPU computes the positions of the sources in the source-plane and sends the information to the GPUs. Once the gradients and the sources are known, the GPUs can start searching for the images by delensing and checking the grid for sources, again by subdividing the grid. Each image found is stored temporarily and at the end of the computation send to the CPU. This operation possesses a divergent execution path, based on if an image is found or not. As a consequence the computation incurs an overhead based on the different amount of found images in each GPUs operational territory. Once all images have been found and received, the master CPU assigns them to their closest constraint and then computes the χ^2. The purely CPU-based implementation distributes the work similarly to the Hybrid CPU–GPU version with the exception that all CPU cores calculate the same sources positions.

Table 1
Characteristics and sustained performance of Computing Cluster used for the *Lenstool-HPC* benchmarks.

Name	Piz Daint CPU	Piz Daint GPU	Tave	Helvetios	Fidis
CPU type	E5-2695 v4	E5-2690 v3	Xeon Phi 7230	Xeon Gold 6140	E5-2690 v4
Microarchitecture	Broadwell	Haswell	Knight's Landing	Skylake	Broadwell
Number of cores	36	12	64	36	24
Frequency (GHz)	2.1	2.6	1.3	2.3	2.6
Memory size (GB)	64	64	112/16[a]	192	128
FP Peak (Gflops/s)	1200	488	1785	2136	1068
stream copy (GB/s)	116	59.7	87/465[a]	164	120
GPU type		P100			
Frequency (GHz)		1.126			
Memory size (GB)		16			
FP Peak (Gflops/s)		4546			
Stream copy (GB/s)		489			

[a]DDR4/MCDRAM.

Lenstool-maps and found correct inside the boundaries of numerical errors. The χ^2 computation is for the moment only available as a function of the library for future MCMC development. The executable works in exactly the same way as *Lenstool*, with a master parameter file, and separate file for constraints and mass-modelling potentials as described in the *Lenstool* wiki.[2] The χ^2 computation is also resistant to missing image problem near caustic lines because of the brute-force approach used. The software and installation instruction can be found at https://git-cral.univ-lyon1.fr/lenstool/LENSTOOL-HPC.

5. Results and benchmarks

This result and benchmark section is organised as follows: First an analysis of the effects of CPU vectorisation and GPUs on the gradient computation. Second a study on the scaling of the CPU and GPU implementation of the χ^2 computation. The scaling studied here is the strong scaling, meaning the same amount of operations distributed over more Nodes.

The Benchmark configurations were taken from an example strong lensing model of MACS J1149.5+2223 from here on named M1149. It is made of 217 different potentials, modelling the cluster. To constrain the model 80 sources have been generated, adding to a total of 227 multiple images. The grid spans over 150 by 150 arcseconds and has 5000 by 5000 pixels for a typical Hubble sampling of 0.03 arcseconds (resolution of ∼0.1 arcsec or better). The Benchmarks were run on five different clusters summarised in Table 1. We have chosen to concentrate on the Helvetios CPU cluster and the Piz Daint hybrid CPU–GPU cluster. Both are comprised of the most modern CPU and GPUs on the market we had access to at the writing of this paper. This will allow us to compare the peak performance of the CPU and GPU version of *Lenstool-HPC*. To enable a fair comparison of the single-map generation algorithm, we upgraded *Lenstool*'s algorithm to support multicore parallelism, distributing the computational operation in the same way as *Lenstool-HPC*'s CPU version over the multiple cores. *Lenstool* has already OpenMP parallelisation in its native code but it is only implemented in its multi-map generation algorithm.

5.1. Core scaling analysis

First we studied the scaling at a single processor level, meaning how well it scaled on multiple cores. The benchmark task was to compute one full 5000 × 5000 gradient map for the M1149 model. Compared were *Lenstool*, *Lenstool-HPC* using AOS structures, *Lenstool-HPC* using SOA structures and no vectorisation and *Lenstool-HPC* using SOA structures with vectorisation. The results are summarised in Fig. 8 and are detailed in Tables A.2 and A.3 . They were run ten times each on Helvetios with AVX 512 capable machines and on Fidis with AVX2 capable machines and no significant standard deviation was observed. An additional Benchmark

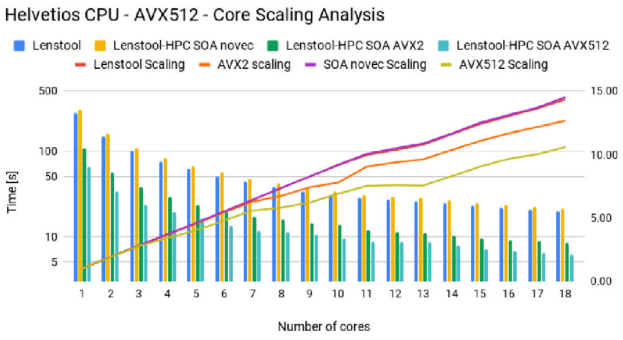
[2] https://projets.lam.fr/projects/lenstool/wiki.

Table A.2
Core scaling analysis results on Helvetios on AVX512 capable machines. The results shown are the mean of 10 runs.

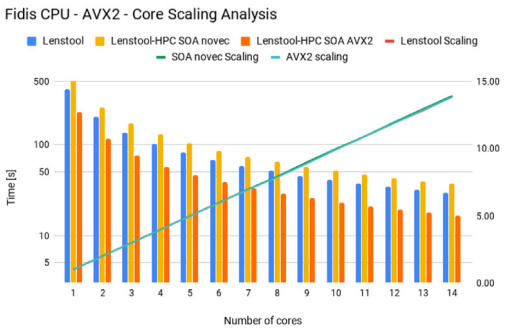
Core scaling results Helvetios AVX512										
Cores	*Lenstool*		*Lenstool-HPC*							
	Lenstool [s]	Scaling	AOS [s]	Scaling	SOA novec [s]	Scaling	SOA AVX2 [s]	Scaling	SOA AVX512 [s]	Scaling
1	281.92 ± 0.051	1.0	235.47 ± 0.040	1.0	307.80 ± 0.069	1.0	107.04 ± 0.067	1.0	65.09 ± 0.068	1.0
2	146.79 ± 0.428	1.9	122.71 ± 0.539	1.9	160.21 ± 0.504	1.9	55.90 ± 0.091	1.9	34.07 ± 0.051	1.9
3	99.68 ± 0.155	2.8	83.43 ± 0.178	2.8	108.74 ± 0.085	2.8	38.20 ± 0.027	2.8	23.35 ± 0.246	2.8
4	76.04 ± 0.103	3.7	63.88 ± 0.097	3.7	83.25 ± 0.055	3.7	29.23 ± 0.059	3.7	19.10 ± 0.262	3.4
5	62.07 ± 0.002	4.5	51.32 ± 0.001	4.6	66.85 ± 0.001	4.6	23.54 ± 0.003	4.6	16.21 ± 0.020	4.0
6	51.16 ± 0.003	5.5	43.16 ± 0.002	5.5	55.79 ± 0.002	5.5	19.66 ± 0.004	5.4	13.56 ± 0.005	4.8
7	43.86 ± 0.001	6.4	36.72 ± 0.003	6.4	47.88 ± 0.002	6.4	17.03 ± 0.045	6.3	11.68 ± 0.048	5.6
8	38.31 ± 0.002	7.4	32.37 ± 0.002	7.3	41.83 ± 0.002	7.4	15.90 ± 0.127	6.7	11.21 ± 0.059	5.8
9	34.12 ± 0.006	8.3	28.60 ± 0.002	8.2	37.23 ± 0.003	8.3	14.44 ± 0.001	7.4	10.48 ± 0.037	6.2
10	30.81 ± 0.030	9.1	25.75 ± 0.016	9.1	33.48 ± 0.010	9.2	13.76 ± 0.009	7.8	9.44 ± 0.008	6.9
11	28.32 ± 0.019	10.0	23.57 ± 0.032	10.0	30.62 ± 0.032	10.1	11.84 ± 0.032	9.0	8.65 ± 0.019	7.5
12	27.23 ± 0.153	10.4	22.50 ± 0.146	10.5	29.36 ± 0.147	10.5	11.42 ± 0.072	9.4	8.59 ± 0.057	7.6
13	26.13 ± 0.001	10.8	21.71 ± 0.001	10.8	28.27 ± 0.001	10.9	11.13 ± 0.001	9.6	8.62 ± 0.001	7.5
14	24.31 ± 0.002	11.6	20.20 ± 0.005	11.7	26.40 ± 0.002	11.7	10.33 ± 0.001	10.4	7.85 ± 0.002	8.3
15	22.71 ± 0.004	12.4	18.88 ± 0.004	12.5	24.57 ± 0.009	12.5	9.66 ± 0.004	11.1	7.20 ± 0.001	9.0
16	21.64 ± 0.057	13.0	17.97 ± 0.035	13.1	23.46 ± 0.028	13.1	9.18 ± 0.020	11.7	6.74 ± 0.001	9.7
17	20.69 ± 0.003	13.6	17.19 ± 0.000	13.7	22.46 ± 0.007	13.7	8.79 ± 0.001	12.2	6.50 ± 0.002	10.0
18	19.64 ± 0.012	14.4	16.25 ± 0.021	14.5	21.19 ± 0.025	14.5	8.44 ± 0.001	12.7	6.15 ± 0.002	10.6

Table A.3
Core scaling analysis results on FIDIS on AVX2 capable machines. The results shown are the mean of 10 runs.

Core scaling results FIDIS AVX2								
Cores	*Lenstool*		*Lenstool-HPC*					
	Lenstool [s]	Scaling	AOS [s]	Scaling	SOA novec [s]	Scaling	SOA AVX2 [s]	Scaling
1	406.87 ± 0.003	1.0	374.36 ± 0.005	1.0	515.32 ± 0.003	1.0	228.90 ± 0.001	1.0
2	203.61 ± 0.010	2.0	187.25 ± 0.002	2.0	257.95 ± 0.011	2.0	115.46 ± 0.001	2.0
3	135.92 ± 0.002	3.0	124.80 ± 0.005	3.0	172.01 ± 0.012	3.0	76.38 ± 0.001	3.0
4	101.83 ± 0.008	4.0	93.58 ± 0.001	4.0	129.87 ± 0.007	4.0	57.51 ± 0.000	4.0
5	81.47 ± 0.001	5.0	74.85 ± 0.002	5.0	103.23 ± 0.005	5.0	45.90 ± 0.001	5.0
6	67.96 ± 0.001	6.0	62.46 ± 0.002	6.0	86.08 ± 0.003	6.0	38.36 ± 0.000	6.0
7	58.29 ± 0.001	7.0	53.57 ± 0.001	7.0	73.84 ± 0.001	7.0	32.92 ± 0.001	7.0
8	51.40 ± 0.002	7.9	46.80 ± 0.001	8.0	64.98 ± 0.002	7.9	29.09 ± 0.001	7.9
9	45.34 ± 0.002	9.0	41.68 ± 0.001	9.0	57.37 ± 0.001	9.0	25.81 ± 0.000	8.9
10	40.94 ± 0.001	9.9	37.47 ± 0.001	10.0	51.62 ± 0.001	10.0	23.09 ± 0.000	9.9
11	37.23 ± 0.002	10.9	34.11 ± 0.001	11.0	47.24 ± 0.002	10.9	21.02 ± 0.000	10.9
12	34.16 ± 0.000	11.9	31.28 ± 0.001	12.0	43.05 ± 0.001	12.0	19.26 ± 0.000	11.9
13	31.72 ± 0.001	12.8	28.87 ± 0.001	13.0	39.74 ± 0.001	13.0	17.79 ± 0.000	12.9
14	29.29 ± 0.002	13.9	26.87 ± 0.001	13.9	36.97 ± 0.001	13.9	16.54 ± 0.000	13.8



(a) Core Scaling analysis results on AVX512 capable Helvetios cluster.

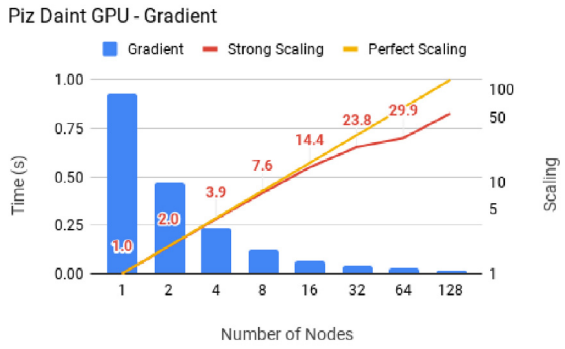


(b) Core Scaling analysis results on AVX2 capable Fidis cluster.

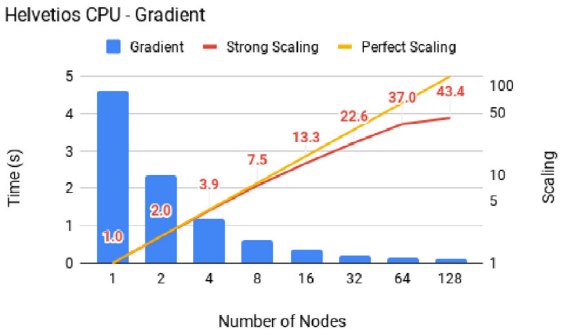
Fig. 8. Core Scaling analysis results: In histograms are compared *Lenstool*, *Lenstool-HPC* with SOA layout without vectorisation (novec) and with vectorisation (SIMD). We observe a speedup of factor 4 for AVX512 machines and factor 2 for AVX2 machines compared to *Lenstool*. Without vectorisation *Lenstool-HPC* with SOA layout is slightly slower than *Lenstool*, indicating that for gradient computation AOS layouts allow for faster memory access than the SOA layout. *Lenstool-HPC* AVX scaling diminishing in function of cores (orange and green line) on Helvetios also indicates that memory overheads are getting significant and that we are hitting hardware limits. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

was run on Helvetios with the amount of zmm-registers limited to AVX2 levels.

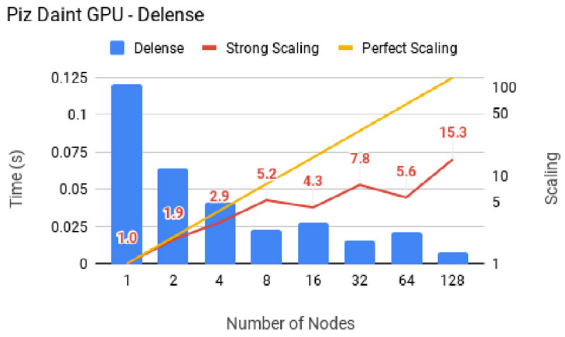
It is immediately obvious that on Helvetios, *Lenstool-HPC* with vectorisation is indeed faster than *Lenstool* by approximately a factor 4. This does not correspond to our theoretical expectations for AVX 512 capable machines which use vectors of size 8 for double-precision floating point operations. This almost twice slower behaviour seems to be due to Intel limiting the frequency of the cores depending on the workload. According to Intel (2017, 2019), the AVX512 and AVX2 top frequency is limited at a lower rate than the non AVX one because of the differing thermal and electrical requirements. The results on the slower AVX-2



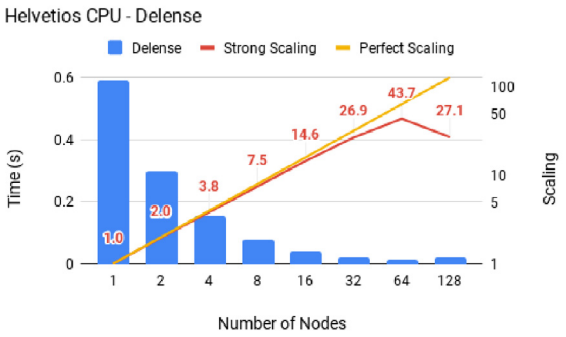
(a) Benchmark results for the deflection gradient computation using the hybrid GPU-CPU version.



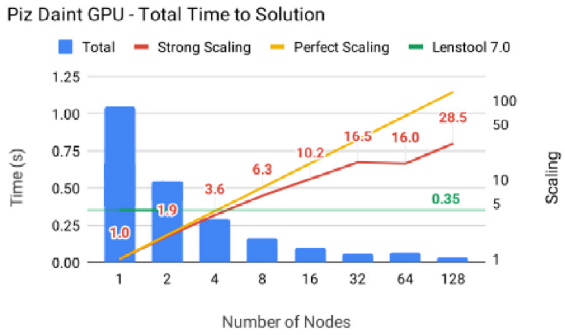
(b) Benchmark results for the deflection gradient computation using the CPU version.



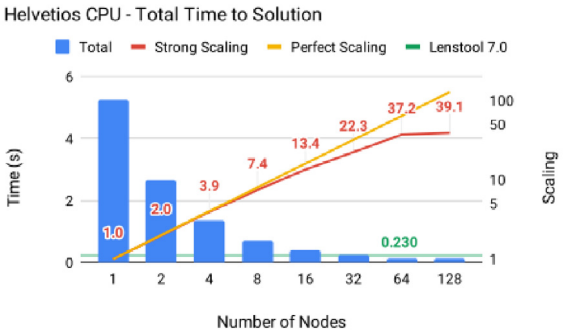
(c) Benchmark results for the constraint delensing operation using the hybrid GPU-CPU version.



(d) Benchmark results for the constraint delensing operation using the CPU version.



(e) Benchmark results for the total computation time using the hybrid GPU-CPU version.



(f) Benchmark results for the total computation time using the CPU version.

Fig. 9. Benchmark and Scaling results of *Lenstool-HPC* on Pizdaint GPU (CSCS) and Helvetios (EPFL): the blue histogram shows the time results in function of the number of computation units (nodes) used. The yellow and red lines indicate respectively the ideal and actual scaling of Lenstool-HPC computation time in function of nodes used. The horizontal green line shows *Lenstool*'s best computation time. When the blue histogram is below the green line is the point when *Lenstool-HPC* brute force approach to lensing beats *Lenstool*'s image transport method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

capable Fidis machine and the AVX2-limited Helvetios run seem to confirm this. They show the same tendencies as on Helvetios with a speed-up gained by vectorisation around 1.77 for Fidis and 2.22 for AVX2 limited Helvetios which corresponds roughly to half the theoretically expected factor 4.

It is interesting that, when deactivating vectorisation with the compiler flag no-vec, *Lenstool* actually performs better than the *Lenstool-HPC* SOA version. For comparison purposes, we created a *Lenstool-HPC* AOS version with the results shown in Tables A.2 and A.3 which improves on the *Lenstool* results. The speed-up due to vectorisation is however still significant enough to beat our own AOS version. This lower performance by the non vectorised SOA version could suggest that the memory access using SOA layout is not optimised for the gradient computation but more in detailed tests would be necessary to be certain.

In the Helvetios results, we also observe a decrease in parallelism efficiency the more cores are used. This is probably due to bandwidth saturation (Intel, 2010) because of the increased amount of information used by AVX operations. AVX512 operations use 8 times more information than non vectorised operations and 2 times more than AVX2. The decrease in efficiency over 18 cores is not too important but it does show that we are starting to approach the hardware limits of actual CPUs. FIDIS does not show the same trend because even with a 4 times increase in speed due to AVX2, bandwidth saturation will not be significant compared to the total operation time.

Table A.4
Distributed scaling analysis results. The benchmarks were run on the CSCS pizdaint CPU and GPU machines and the EPFL Helvetios and Grand Tave CPU clusters. All results shown are in seconds.

Piz Daint@CSCS-GPU								
Nodes	Gradient	Source	Delense	Comm	χ^2	Total	Strong scaling	*Lenstool* 7.0
1	0.93	0.0025	0.1210	1.40×10^{-5}	2.1×10^{-5}	1.05	1.00	0.35
2	0.47	0.0155	0.0643	7.92×10^{-3}	2.3×10^{-5}	0.54	1.93	0.35
4	0.24	0.0205	0.0412	7.87×10^{-3}	2.5×10^{-5}	0.29	3.56	0.35
8	0.12	0.0195	0.0230	8.51×10^{-3}	2.8×10^{-5}	0.17	6.31	0.35
16	0.06	0.0282	0.0279	8.13×10^{-3}	3.2×10^{-5}	0.10	10.19	0.35
32	0.04	0.0279	0.0155	8.64×10^{-3}	3.2×10^{-5}	0.06	16.54	0.35
64	0.03	0.0310	0.0215	9.08×10^{-3}	3.3×10^{-5}	0.07	15.97	0.35
128	0.02	0.0148	0.0079	9.29×10^{-3}	3.6×10^{-5}	0.04	28.52	0.35
Piz Daint@CSCS-CPU								
Nodes	Gradient	Source	Delense	Comm	χ^2	Total	Strong scaling	*Lenstool* 7.0
1	5.99	0.0022	0.7999	2.70×10^{-5}	2.1×10^{-5}	6.82	1.00	0.301
2	3.04	0.0022	0.4247	2.06×10^{-4}	2.3×10^{-5}	3.48	1.96	0.301
4	1.51	0.0030	0.2069	3.01×10^{-4}	2.5×10^{-5}	1.75	3.90	0.301
8	0.80	0.0030	0.1101	1.46×10^{-3}	2.8×10^{-5}	0.91	7.47	0.301
16	0.44	0.0030	0.0616	2.14×10^{-3}	3.2×10^{-5}	0.51	13.42	0.301
32	0.28	0.0030	0.0421	4.85×10^{-3}	3.2×10^{-5}	0.33	20.67	0.301
64	0.25	0.0031	0.0268	8.18×10^{-3}	3.3×10^{-5}	0.28	24.33	0.301
128	0.17	0.0030	0.0320	8.55×10^{-3}	3.6×10^{-5}	0.23	29.96	0.301
Helvetios@EPFL								
Nodes	Gradient	Source	Delense	Comm	χ^2	Total	Strong scaling	*Lenstool* 7.0
1	4.63	0.0011	0.5894	1.50×10^{-5}	1.2×10^{-5}	5.25	1.00	0.230
2	2.34	0.0017	0.2989	3.53×10^{-4}	1.3×10^{-5}	2.65	1.98	0.230
4	1.18	0.0013	0.1534	4.50×10^{-4}	1.1×10^{-5}	1.34	3.92	0.230
8	0.62	0.0015	0.0787	2.47×10^{-4}	9.0×10^{-6}	0.71	7.44	0.230
16	0.35	0.0015	0.0405	2.95×10^{-4}	9.0×10^{-6}	0.39	13.42	0.230
32	0.20	0.0015	0.0219	8.25×10^{-4}	9.0×10^{-6}	0.24	22.33	0.230
64	0.13	0.0015	0.0135	1.28×10^{-3}	1.0×10^{-5}	0.14	37.15	0.230
128	0.11	0.0015	0.0217	1.68×10^{-3}	1.0×10^{-5}	0.13	39.15	0.230
Tave@CSCS								
Nodes	Gradient	Source	Delense	Comm	χ^2	Total	Strong scaling	*Lenstool* 7.0
1	2.45	0.0054	1.1759	1.0×10^{-4}	1.73×10^{-4}	3.78	1.0	2.8
2	1.23	0.0060	0.5921	2.1×10^{-4}	1.05×10^{-4}	1.90	2.0	2.8
4	0.62	0.0061	0.3013	3.1×10^{-4}	1.06×10^{-4}	0.96	3.9	2.8
8	0.38	0.0061	0.1828	1.12×10^{-3}	1.08×10^{-4}	0.57	6.6	2.8
16	0.25	0.0061	0.1219	6.33×10^{-3}	1.10×10^{-4}	0.40	9.6	2.8
32	0.13	0.0062	0.0642	1.13×10^{-2}	1.14×10^{-4}	0.22	17.5	2.8
64	0.14	0.0062	0.0633	2.29×10^{-2}	1.14×10^{-4}	0.23	16.5	2.8
128	0.14	0.0062	0.0632	4.67×10^{-2}	1.13×10^{-4}	0.25	15.0	2.8

5.2. Distributed scaling

The *Chi*^{2} benchmarks time the full *Chi*^{2} computation and its four main stages: The gradient computation of a grid, the source computation based of the constraints, finding images by delensing and checking the grid and computing the *Chi*^{2} based on the found images. The benchmark was distributed and scaled over 128 nodes which was our maximum available number of test nodes. In contrast to the core scaling analysis, due to time-constraint on the allotted server time we could not rerun them multiple times to study the standard deviation. The results are summarised in Fig. 9 and more details can be found in the appendix. The two main stages to pay attention to are the gradient computation and the delensing stage.

5.2.1. Gradient computation

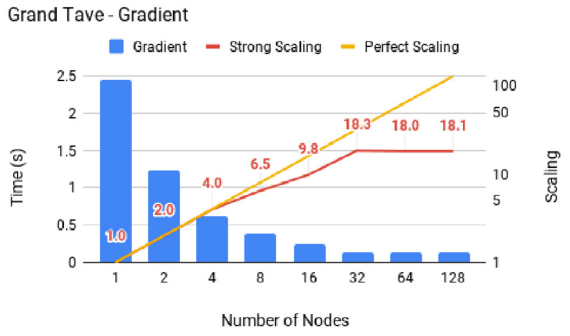
The computation of a 5000 × 5000 lensing map on one Pizdaint P100 GPU takes only 0.93 s. At a single node level, compared to a Helvetios node with 36 cores, the single GPU version outperforms *Lenstool-HPC*s CPU version by a factor 5 and *Lenstool* by a factor 10. Since we upgraded *Lenstool*s single map generation to be distributable at a node level, for the common user the P100 version actually outperforms it by a factor 360.

Figs. 8a and 9b show the scaling of the gradient computation for CPUs and GPUs up 128 Nodes. Up to 32 nodes the software scales well with a parallelism efficiency of 0.75. Around 64, for both GPU and CPUs, the scaling worsens with a parallelism efficiency of around 0.5. This is mainly because the shrinking amount of work per node is starting to be insufficient to hide the latencies of the computation. The parallelism efficiency should rise the more complex the problem, but the inverse is also true. The gradient computation could still be distributed over more than 128 nodes for slight gain but we were limited here by the available hardware.

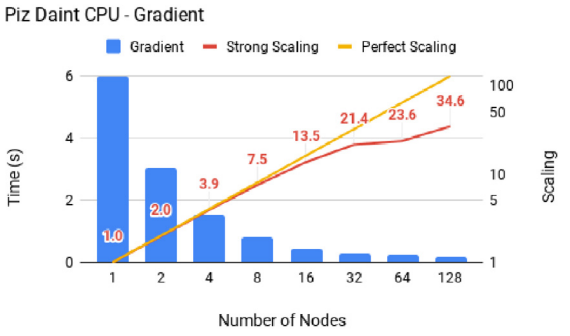
At 128 nodes, *Lenstool-HPC* is 55.3 times faster than its single node GPU version, meaning approximately 500 times faster than *Lenstool*'s single map gradient computation. The parallelised CPU version is roughly 4 times slower than its parallelised GPU counter part. It remains competitive enough that even users who do not have access to GPU cluster can generate lensing map efficiently. For cost-conscious users who wish a reasonably high efficiency, with 32 P100 GPUs at one map every 0.04 s we can do the Abell 2744 error computation (Jauzac et al., 2015) with 10014 in 166 min, a bit less than 3 h. This is 25 times faster than the *Lenstool* version especially tuned for the Benchmarks.

5.2.2. Delensing and searching for images

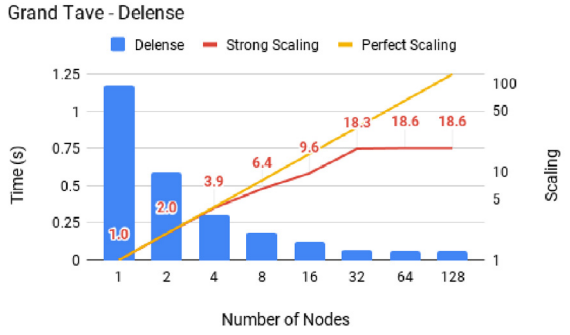
As stated above, this task is not trivially parallelisable. While the work can distributed over the different GPUs, the divergent execution path that appears when an image is found, impacts



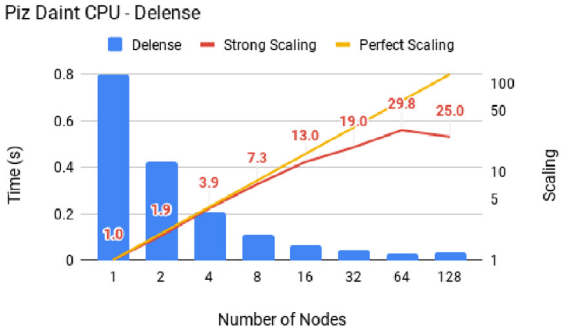
(a) Benchmark results for the deflection gradient computation using the CPU version on the Grand Tave cluster.



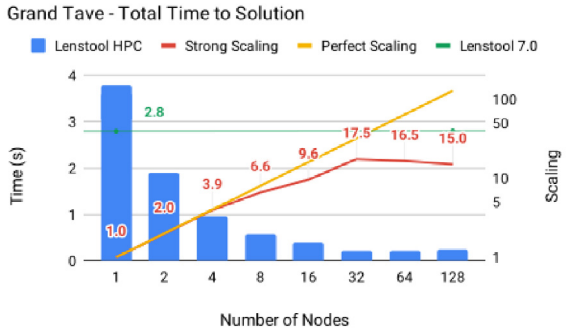
(b) Benchmark results for the deflection gradient computation using the CPU version on the Pizdaint cluster.



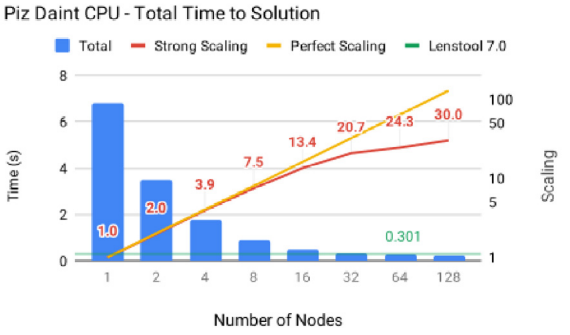
(c) Benchmark results for the constraint delensing operation using the CPU version on the Grand Tave cluster.



(d) Benchmark results for the constraint delensing operation using the CPU version on the Pizdaint cluster.



(e) Benchmark results for the total computation time using the hybrid GPU-CPU version on the Grand Tave cluster.



(f) Benchmark results for the total computation time using the hybrid GPU-CPU version on the Pizdaint cluster.

Fig. A.10. Benchmark and Scaling results of *Lenstool-HPC* on Pizdaint CPU (CSCS) and Grand Tave (CSCS): the blue histogram shows the time results in function of the number of computation units (nodes) used. The yellow and red lines indicate respectively the ideal and actual scaling of Lenstool-HPC computation time in function of nodes used. The horizontal green line shows *Lenstool*'s best computation time. When the blue histogram is below the green line is the point when *Lenstool-HPC* brute force approach to lensing beats *Lenstool*'s image transport method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the parallelism efficiency quickly. Already at 4 GPU nodes (see Fig. 9c), we are at an efficiency of 0.73 and seem to saturate around 8 nodes. At a single node level this does not impact us much (see Fig. 9e). The task takes only 11% of the total runtime, with the rest going to the gradient computation. However, since the gradient is scaling well, already on 16 cores the delensing task takes 27% of the total runtime with noticeable effects. The parallelism efficiency of the total χ^2 computation starts to drop around 8 nodes by the delensing task before it saturates completely around 32 nodes. This final saturation is not only due to worsening of the gradient computation efficiency. The computation time of the gradients over 32 to 128 GPUs simply has reached the same level as the computing of the sources on the CPUs around 0.04 to 0.02 s. Since the delensing task depended on both gradient computation and source computation, it cannot start without both having finished running, creating the observed saturation.

The CPU version in contrast shows a lot less degradation to its parallelism efficiency, at least up to 64 nodes. This corresponds to our expectation since CPUs have less but faster computation units than GPUs. The amount of work per core never reaches a stage when it is insufficient to hide the latencies of the divergent

execution paths. CPUs compilers have also been already heavily optimised to handle these complex operations.

With this in mind, the *Lenstool-HPC*'s brute force GPU version manages to beat *Lenstool* fully recurrent image transport with only 4 GPUs (see Fig. 9e) and can still scale up 32 for a total speedup of 5.9. *Lenstool-HPC* brute force CPUs version is less successful, managing to beat *Lenstool* only with 64 nodes for a speed-up of 1.7 but also demonstrates more parallel efficiency. Depending on the hardware developments of the future, they could become an extremely credible option.

6. Conclusions

We have shown that it is possible to use modern HPC based programming to greatly speed up conventional gravitational lens mass modelling software. On P100 GPUs et SLK CPUs the new *Lenstool-HPC* GPU based library has shown to be 360 times faster than *Lenstool* on single map computation and 10 times faster on multi-map computation with only a single GPU. The necessary gradient computation has shown to scale extremely well up to 64 nodes with a Hubble Frontier Fields' size problem, generating an additional corresponding speed-up. The brute force implementation proposed for the mass-model χ^2 computation beats *Lenstool* recursive but tricky to use image-transport implementation with only 4 GPUs and scales reasonably well up to 32 nodes. Additionally *Lenstool-HPC* non recursive HPC implementation of lens-modelling tools will scale with future hardware developments, ensuring future speed-ups that recursive options will not have. Future development will go towards the full integration of the library into *Lenstool* and optimisation of the last bottle necks, in particular the (MCMC) optimisation process. This will be combined with a thorough comparison to other GPU and non GPU based Lens-modelling tools to assess and further improve Lenstool-HPCs Lens-modelling process.

The achieved speed-up is key to continue using *Lenstool* for clusters with many constraints, and to allow a fast evaluation (through the lensing maps) of the quality and properties of the lensing mass models computed. As an example, having a fast lensing maps computation allows quick evaluation of the lensing model and the identification of where the fit is good or bad, allowing us to focus on the modelling. Ultimately, a fast code will allow to address the "bad RMS" of models (typically larger than 10× the Hubble image resolution) and understand its origin.

The C++ and CUDA based library is publicly available on Github https://git-cral.univ-lyon1.fr/lenstool/LENSTOOL-HPC .

Acknowledgements

CS thanks Mathilde Jauzac for fruitful discussions on Lens-modelling. CS also acknowledges support from the ESA-NPI grant 4000120530/17/NL/MH, and the SNF Sinergia "Euclid" FNS, Switzerland CRSII5_173716. GF gratefully acknowledges support from the EPFL Faculté des Sciences de Base, Switzerland. This work was supported by EPFL, Switzerland through the use of the facilities of its Scientific IT and Application Support Center. The authors gratefully acknowledge the use of facilities of the Swiss National Supercomputing Centre (CSCS), in particular Colin McMurtrie and Hussein Nasser El-Harake for their constant support on the Greina test cluster where most of the GPU development was performed. This research made use of matplotlib (Hunter, 2007), Inkscape, TeX Live, and NASA's Astrophysics Data System.

Appendix. Benchmark results

Tables A.2–A.4 and Fig. A.10 contain all information of the Benchmarks we did for *Lenstool-HPC* run on the clusters summarised in Table 1.

References

Atek, H., Richard, J., Kneib, J.P., Jauzac, M., Schaerer, D., Clement, B., Limousin, M., Jullo, E., Natarajan, P., Egami, E., Ebeling, H., 2015. New constraints on the faint end of the UV luminosity function at z 7-8 using the gravitational lensing of the hubble frontier fields cluster A2744. Astrophys. J. 800, 18. doi:10.1088/0004-637X/800/1/18, arXiv:1409.0512.

Bailey, D.H., 1997. Little's Law and High Performance Computing. Technical Report, URL: http://www.tera.com/arpa95/architecture.html.

Bartelmann, M., Schneider, P., 2001. Weak gravitational lensing. Phys. Rep. 340, 291. doi:10.1016/S0370-1573(00)00082-X, URL: https://arxiv.org/pdf/astro-ph/9912508.pdf http://adsabs.harvard.edu/cgi-bin/nph-data_query?bibcode=2001PhR...340..291B&link_type=ABSTRACT%5Cnpapers://dcc533b5-8613-47b7-b88c-2b0c0d39c33f/Paper/p5672 arXiv:0509252.

Besl, P., 2013. A case study comparing Arrays of Structures and Structures of Arrays data layouts for a compute-intensive loop run on Intel Xeon processors and Intel Xeon Phi product family coprocessors.

Bonvin, V., Tewes, M., Courbin, F., Kuntzer, T., Sluse, D., Meylan, G., 2016. COSMOGRAIL: the COSmological MOnitoring of GRAvItational Lenses. XV. Assessing the achievability and precision of time-delay measurements. Astron. Astrophys. 585, A88. doi:10.1051/0004-6361/201526704, arXiv:1506.07524.

Elíasdóttir, Á., Limousin, M., Richard, J., Hjorth, J., Kneib, J.P., Natarajan, P., Pedersen, K., Jullo, E., Paraficz, D., 2007. Where is the matter in the Merging Cluster Abell 2218?. ArXiv e-prints arXiv:0710.5636.

Hölzle, U., 2010. Brawny Cores Still Beat Wimpy Cores, Most of the Time. Technical Report, URL: http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36448.pdf.

Hunter, J.D., 2007. Matplotlib: A 2d graphics environment. Comput. Sci. Eng. 9, 90–95. doi:10.1109/MCSE.2007.55.

Intel, 2010. Detecting memory bandwidth saturation in threaded applications. URL: https://software.intel.com/en-us/articles/detecting-memory-bandwidth-saturation-in-threaded-applications.

Intel, 2013. Compiler methodology for intel MIC architecture. URL: https://software.intel.com/en-us/articles/memory-layout-transformations.

Intel, 2015. Opencl developer guide for intel® processor graphics. URL: https://software.intel.com/en-us/node/540482.

Intel, 2017. Frequency behavior - intel. URL: https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf.

Intel, 2019. Intel®64 and IA-32 architectures software developer's manual. URL: https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf.

Jauzac, M., Clément, B., Limousin, M., Richard, J., Jullo, E., Ebeling, H., Atek, H., Kneib, J.P., Knowles, K., Natarajan, P., Eckert, D., Egami, E., Massey, R., Rexroth, M., 2014. Hubble Frontier Fields: a high-precision strong-lensing analysis of galaxy cluster MACSJ0416.1-2403 using 200 multiple images. Mon. Not. Roy. Astron. Soc. 443, 1549–1554. doi:10.1093/Mon.Not.Roy.Astron.Soc./stu1355, arXiv:1405.3582.

Jauzac, M., Richard, J., Jullo, E., Clément, B., Limousin, M., Kneib, J.P., Ebeling, H., Natarajan, P., Rodney, S., Atek, H., Massey, R., Eckert, D., Egami, E., Rexroth, M., 2015. Hubble frontier fields: a high-precision strong-lensing analysis of the massive galaxy cluster abell 2744 using 180 multiple images. Mon. Not. Roy. Astron. Soc. 452, 1437–1446. doi:10.1093/Mon.Not.Roy.Astron.Soc./stv1402, arXiv:1409.8663.

Jiang, G., Kochanek, C.S., 2007. The baryon fractions and mass-to-light ratios of early-type galaxies. Astrophys. J. 671, 1568–1578. doi:10.1086/522580, arXiv:0705.3647.

Jullo, E., Kneib, J.P., 2009. Multiscale cluster lens mass mapping - I. Strong lensing modelling. Mon. Not. Roy. Astron. Soc. 395, 1319–1332. doi:10.1111/j.1365-2966.2009.14654.x, arXiv:0901.3792.

Jullo, E., Kneib, J.P., Limousin, M., Elíasdóttir, Á., Marshall, P.J., Verdugo, T., 2007. A Bayesian approach to strong lensing modelling of galaxy clusters. New J. Phys. 9, 447. doi:10.1088/1367-2630/9/12/447, arXiv:0706.0048.

Kneib, J.P., Ellis, R.S., Santos, M.R., Richard, J., 2004. A probable z 7 galaxy strongly lensed by the rich cluster A2218: Exploring the dark ages. Astrophys. J. 607, 697–703. doi:10.1086/386281, arXiv:astro-ph/0402319.

Kneib, J.P., Ellis, R.S., Smail, I., Couch, W.J., Sharples, R.M., 1996. Hubble space telescope observations of the lensing cluster abell 2218. Astrophys. J. 471 (643), doi:10.1086/177995, arXiv:astro-ph/9511015.

Kreinin, Y., 2011. SIMD < SIMT < SMT: parallelism in NVIDIA GPUs. URL: https://yosefk.com/blog/simd-simt-smt-parallelism-in-nvidia-gpus.html.

Liang, X., Nguyen, M., Che, H., 2013. Wimpy or brawny cores: A throughput perspective. J. Parallel Distrib. Comput. 73, 1351–1361. doi:10.1016/j.jpdc.2013.06.001.

Limousin, M., Richard, J., Kneib, J.P., Brink, H., Pelló, R., Jullo, E., Tu, H., Sommer-Larsen, J., Egami, E., Michałowski, M.J., Cabanac, R., Stark, D.P., 2008. Strong lensing in Abell 1703: constraints on the slope of the inner dark matter distribution. Astron. Astrophys. 489, 23–35. doi:10.1051/0004-6361:200809646, arXiv:0802.4292.

Lindholm, E., Nickolls, J., Oberman, S., Montrym, J., 2008. Nvidia tesla: A unified graphics and computing architecture. IEEE Micro 28, 39–55. doi:10.1109/MM.2008.31.

More, S., van den Bosch, F.C., Cacciato, M., Skibba, R., Mo, H.J., Yang, X., 2011. Satellite kinematics - III. Halo masses of central galaxies in SDSS. Mon. Not. Roy. Astron. Soc. 410, 210–226. doi:10.1111/j.1365-2966.2010.17436.x, arXiv:1003.3203.

Mudge, T., 2001. Power: A first-class architectural design constraint. Computer 34, 52–58. doi:10.1109/2.917539.

Nvidia, 2012. Version 4.2 NVIDIA CUDA NVIDIA CUDA C Programming Guide. Technical Report, URL: http://developer.nvidia.com/cuda-gpus.

Richard, J., Jones, T., Ellis, R., Stark, D.P., Livermore, R., Swinbank, M., 2011. The emission line properties of gravitationally lensed $1.5 < z < 5$ galaxies. Mon. Not. Roy. Astron. Soc. 413, 643–658. doi:10.1111/j.1365-2966.2010.18161.x, arXiv:1011.6413.

Satish, N., Kim, C., Chhugani, J., Saito, H., Krishnaiyer, R., Smelyanskiy, M., Girkar, M., Dubey, P., 2012. Can traditional programming bridge the ninja performance gap for parallel computing applications? SIGARCH comput. Archit. News 40, 440–451. doi:10.1145/2366231.2337210, URL: http://doi.acm.org/10.1145/2366231.2337210.

Schneider, P., Ehlers, J., Falco, E.E., 1992. Gravitational lenses. http://dx.doi.org/101007/978-3-662-03758-4.

Sonnenfeld, A., Treu, T., Marshall, P.J., Suyu, S.H., Gavazzi, R., Auger, M.W., Nipoti, C., 2015. The SL2S galaxy-scale lens sample. V. Dark matter halos and stellar IMF of massive early-type galaxies out to redshift 0.8. Astrophys. J. 800 (94), doi:10.1088/0004-637X/800/2/94, arXiv:1410.1881.

Suyu, S.H., Bonvin, V., Courbin, F., Fassnacht, C.D., Rusu, C.E., Sluse, D., Treu, T., Wong, K.C., Auger, M.W., Ding, X., Hilbert, S., Marshall, P.J., Rumbaugh, N., Sonnenfeld, A., Tewes, M., Tihhonova, O., Agnello, A., Blandford, R.D., Chen, G.C.F., Collett, T., Koopmans, L.V.E., Liao, K., Meylan, G., Spiniello, C., 2017. H0licow - I. H_0 lenses in COSMOGRAIL's wellspring: program overview. Mon. Not. Roy. Astron. Soc. 468, 2590–2604. doi:10.1093/Mon.Not.Roy.Astron.Soc./stx483, arXiv:1607.00017.

Valkov, V., 2010. Better performance at low occupancy.

6.3.3 Outlook

Lenstool-HPC has, at the time of writing of this thesis, implemented methods solving and optimising two of the three main bottlenecks of Lenstool's modelisation algorithm. It also contains all of the necessary infrastructure for easier future development. While the gradient computation runs at peak parallel efficiency, the fit computation can be still further improved. For example by defining a smaller search area around specific constraints instead of checking the whole grid, one can reduce the total amount of cell-checking done when looking for images. This would significantly improve the overall fit computation. The utility of Lenstool-HPC at the moment is mainly based on its capacity to generate gradient maps almost instantaneously. It can be used to significantly speed up any application using them, including the error computation of the bayesian parameter space exploration.

The third bottleneck, the parameter space exploration using an MCMC algorithm, remains for the moment an unfinished task. Once a working implementation of the bayesys3 package by John Skilling is added, Lenstool-HPC can be used for lens modelling but it will not run at optimum throughput.

MCMC samplers are not trivially parallelisable algorithms since the concept behind them is inherently serial. The main idea, as stated in chapter 5, is that the consecutive exploration of the parameter space allows for a more informed distribution of the parameters. The fewer steps are used, the less informed the distribution is. The literature of the bayesys3 package states that it is parallelisable up to 10 concurrent MCMC chains but to the best of our knowledge, based on discussions with Eric Jullo, any concurrent application of the bayesys3 package significantly increases the probability of it converging to local minima. This reduces the confidence that astronomers can have in the models proposed by the method, which is not acceptable.

The overlying Nested Sampling concept introduced by John Skilling (Skilling, 2006) however does not necessarily require the use of MCMCs. Other variations exist and have been implemented like MultiNest (Feroz et al., 2009), PolyChord (Handley et al., 2015) and DNest4 (Diffusive Nested Sampling) (Brewer and Foreman-Mackey, 2016). Each of these use different sampling techniques that are not based on MCMCs even if the main concept behind them remains bayesian and as such serial. The logical next step would therefore be to do an in-depth test study of these bayesian based samplers to find a replacement for the current bayesys3 method. What future studies should particularly focus on is evaluating the concurrency capabilities of the different methods, i.e. their parallel computation potential, to ascertain that a future HPC implementation really is as efficient as it can be. If no good replacement exists, creating a bayesian based model selection method that is truly compatible with HPC infrastructure will be crucial. Without it, anyone using bayesian statistics in the future HPC-based world will be hampered by low computation speed due to less efficient parallelism. Due to the multiple domains involved, I believe that to properly perform this task a team combining at least astrophysical, high performance computing and statistical skills is needed.

Bibliography

Abell, G. O., Corwin, Harold G., J., and Olowin, R. P. (1989). A Catalog of Rich Clusters of Galaxies. , 70:1.

Ahn, K. and Shapiro, P. R. (2005). Formation and evolution of self-interacting dark matter haloes. , 363(4):1092–1110.

Altieri, B., Berta, S., Lutz, D., Kneib, J. P., Metcalfe, L., Andreani, P., Aussel, H., Bongiovanni, A., Cava, A., Cepa, J., Ciesla, L., Cimatti, A., Daddi, E., Dominguez, H., Elbaz, D., Förster Schreiber, N. M., Genzel, R., Gruppioni, C., Magnelli, B., Magdis, G., Maiolino, R., Nordon, R., Pérez García, A. M., Poglitsch, A., Popesso, P., Pozzi, F., Richard, J., Riguccini, L., Rodighiero, G., Saintonge, A., Santini, P., Sanchez-Portal, M., Shao, L., Sturm, E., Tacconi, L. J., Valtchanov, I., Wetzstein, M., and Wieprecht, E. (2010). Herschel deep far-infrared counts through Abell 2218 cluster-lens. , 518:L17.

Altieri, B., Metcalfe, L., Kneib, J. P., McBreen, B., Aussel, H., Biviano, A., Delaney, M., Elbaz, D., Leech, K., Lémonon, L., Okumura, K., Pelló, R., and Schulz, B. (1999). An ultra-deep ISOCAM observation through a cluster-lens. , 343:L65–L69.

Auger, M. W., Treu, T., Bolton, A. S., Gavazzi, R., Koopmans, L. V. E., Marshall, P. J., Bundy, K., and Moustakas, L. A. (2009). The Sloan Lens ACS Survey. IX. Colors, Lensing, and Stellar Masses of Early-Type Galaxies. , 705(2):1099–1115.

Barnabè, M., Czoske, O., Koopmans, L. V. E., Treu, T., Bolton, A. S., and Gavazzi, R. (2009). Two-dimensional kinematics of SLACS lenses - II. Combined lensing and dynamics analysis of early-type galaxies at z = 0.08-0.33. , 399(1):21–36.

Barshan, E. and Fieguth, P. (2015). Stage-wise training: An improved feature learning strategy for deep models. In *Feature Extraction: Modern Questions and Challenges*, pages 49–59.

Bastian, N., Covey, K. R., and Meyer, M. R. (2010). A Universal Stellar Initial Mass Function? A Critical Look at Variations. , 48:339–389.

Blain, A. W., Smail, I., Ivison, R. J., Kneib, J. P., and Frayer, D. T. (2002). Submillimeter galaxies. , 369(2):111–176.

Blandford, R. D. and Narayan, R. (1992). Cosmological applications of gravitational lensing. *Annual Review of Astronomy and Astrophysics*, 30(1):311–358.

Bolton, A. S., Burles, S., Treu, T., Koopmans, L. V. E., and Moustakas, L. A. (2007). A More Fundamental Plane. , 665:L105–L108.

Bolton, A. S., Treu, T., Koopmans, L. V. E., Gavazzi, R., Moustakas, L. A., Burles, S., Schlegel, D. J., and Wayth, R. (2008). The Sloan Lens ACS Survey. VII. Elliptical Galaxy Scaling Laws from Direct Observational Mass Measurements. , 684(1):248–259.

Bonvin, V., Courbin, F., Suyu, S. H., Marshall, P. J., Rusu, C. E., Sluse, D., Tewes, M., Wong, K. C., Collett, T., Fassnacht, C. D., Treu, T., Auger, M. W., Hilbert, S., Koopmans, L. V. E., Meylan, G., Rumbaugh, N., Sonnenfeld, A., and Spiniello, C. (2017). H0LiCOW - V. New COSMOGRAIL time delays of HE 0435-1223: H_0 to 3.8 per cent precision from strong lensing in a flat ΛCDM model. , 465(4):4914–4930.

Brewer, B. J. and Foreman-Mackey, D. (2016). DNest4: Diffusive Nested Sampling in C++ and Python. *arXiv e-prints*, page arXiv:1606.03757.

Browne, I. W. A., Wilkinson, P. N., Jackson, N. J. F., Myers, S. T., Fassnacht, C. D., Koopmans, L. V. E., Marlow, D. R., Norbury, M., Rusin, D., Sykes, C. M., Biggs, A. D., Blandford, R. D., de Bruyn, A. G., Chae, K. H., Helbig, P., King, L. J., McKean, J. P., Pearson, T. J., Phillips, P. M., Readhead, A. C. S., Xanthopoulos, E., and York, T. (2003). The Cosmic Lens All-Sky Survey - II. Gravitational lens candidate selection and follow-up. , 341(1):13–32.

Chan, J. H. H., Suyu, S. H., Sonnenfeld, A., Jaelani, A. T., More, A., Yonehara, A., Kubota, Y., Coupon, J., Lee, C.-H., Oguri, M., Rusu, C. E., and Wong, K. C. (2019). Survey of Gravitationally-lensed Objects in HSC Imaging (SuGOHI). IV. Lensed quasar search in the HSC survey. *arXiv e-prints*, page arXiv:1911.02587.

Clément, B., Cuby, J. G., Courbin, F., Fontana, A., Freudling, W., Fynbo, J., Gallego, J., Hibon, P., Kneib, J. P., Le Fèvre, O., Lidman, C., McMahon, R., Milvang-Jensen, B., Moller, P., Nilsson, K. K., Pentericci, L., Venemans, B., Villar, V., and Willis, J. (2011). Evolution of the Observed Ly-alpha Luminosity Function from z = 6.5 to z = 7.7: Evidence for the Epoch of Re-ionisation? *The Messenger*, 146:31–34.

Collett, T. E. (2015). The Population of Galaxy-Galaxy Strong Lenses in Forthcoming Optical Imaging Surveys. , 811(1):20.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. math cont sig syst (mcss) 2:303-314. *Mathematics of Control, Signals, and Systems*, 2:303–314.

Diehl, H. T., Buckley-Geer, E. J., Lindgren, K. A., Nord, B., Gaitsch, H., Gaitsch, S., Lin, H., Allam, S., Collett, T. E., Furlanetto, C., Gill, M. S. S., More, A., Nightingale, J., Odden, C., Pellico, A., Tucker, D. L., da Costa, L. N., Fausti Neto, A., Kuropatkin, N., Soares-Santos, M., Welch, B., Zhang, Y., Frieman, J. A., Abdalla, F. B., Annis, J., Benoit-Lévy, A., Bertin, E., Brooks, D., Burke, D. L., Carnero Rosell, A., Carrasco Kind, M., Carretero, J., Cunha, C. E., D'Andrea, C. B., Desai, S., Dietrich, J. P., Drlica-Wagner, A., Evrard, A. E., Finley, D. A., Flaugher, B., García-Bellido, J., Gerdes, D. W., Goldstein, D. A., Gruen, D., Gruendl, R. A., Gschwend, J., Gutierrez, G., James, D. J., Kuehn, K., Kuhlmann, S., Lahav, O., Li, T. S., Lima, M., Maia, M. A. G., Marshall, J. L., Menanteau, F., Miquel, R., Nichol, R. C., Nugent, P., Ogand o, R. L. C., Plazas, A. A., Reil, K., Romer, A. K., Sako, M., Sanchez, E., Santiago, B., Scarpine, V., Schindler, R., Schubnell, M., Sevilla-Noarbe, I., Sheldon, E., Smith, M., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Thomas, D., Walker, A. R., and DES Collaboration (2017). The DES Bright Arcs Survey: Hundreds of Candidate Strongly Lensed Galaxy Systems from the Dark Energy Survey Science Verification and Year 1 Observations. , 232(1):15.

Ebeling, H., Edge, A. C., and Henry, J. P. (2001). MACS: A Quest for the Most Massive Galaxy Clusters in the Universe. , 553(2):668–676.

Eddington, A. S. (1920). *Space, time and gravitation. an outline of the general relativity theory.*

Einstein, A. (1911). Über den Einfluß der Schwerkraft auf die Ausbreitung des Lichtes. *Annalen der Physik*, 340(10):898–908.

Einstein, A. (1936). Lens-Like Action of a Star by the Deviation of Light in the Gravitational Field. *Science*, 84(2188):506–507.

Elíasdóttir, Á., Limousin, M., Richard, J., Hjorth, J., Kneib, J.-P., Natarajan, P., Pedersen, K., Jullo, E., and Paraficz, D. (2007). Where is the matter in the Merging Cluster Abell 2218? *arXiv e-prints*, page arXiv:0710.5636.

Feroz, F., Hobson, M. P., and Bridges, M. (2009). MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. , 398(4):1601–1614.

Flynn, M. (1967). Very high-speed computing systems. *Proceedings of the IEEE*, 54:1901 – 1909.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goodfellow, I., Bengio, Y., and Courville, A. (2018). *Deep Learning*. MITP.

Handley, W. J., Hobson, M. P., and Lasenby, A. N. (2015). POLYCHORD: next-generation nested sampling. , 453(4):4384–4398.

Harley, A. W. (2015). An interactive node-link visualization of convolutional neural networks. In *ISVC*, pages 867–877.

Hartley, P., Flamary, R., Jackson, N., Tagore, A. S., and Metcalf, R. B. (2017). Support vector machine classification of strong gravitational lenses. , 471(3):3378–3397.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257.

Jacobs, C., Collett, T., Glazebrook, K., McCarthy, C., Qin, A. K., Abbott, T. M. C., Abdalla, F. B., Annis, J., Avila, S., Bechtol, K., Bertin, E., Brooks, D., Buckley-Geer, E., Burke, D. L., Carnero Rosell, A., Carrasco Kind, M., Carretero, J., da Costa, L. N., Davis, C., De Vicente, J., Desai, S., Diehl, H. T., Doel, P., Eifler, T. F., Flaugher, B., Frieman, J., García-Bellido, J., Gaztanaga, E., Gerdes, D. W., Goldstein, D. A., Gruen, D., Gruendl, R. A., Gschwend, J., Gutierrez, G., Hartley, W. G., Hollowood, D. L., Honscheid, K., Hoyle, B., James, D. J., Kuehn, K., Kuropatkin, N., Lahav, O., Li, T. S., Lima, M., Lin, H., Maia, M. A. G., Martini, P., Miller, C. J., Miquel, R., Nord, B., Plazas, A. A., Sanchez, E., Scarpine, V., Schubnell, M., Serrano, S., Sevilla-Noarbe, I., Smith, M., Soares-Santos, M., Sobreira, F., Suchyta, E., Swanson, M. E. C., Tarle, G., Vikram, V., Walker, A. R., Zhang, Y., Zuntz, J., and DES Collaboration (2019). Finding high-redshift strong lenses in DES using convolutional neural networks. , 484(4):5330–5349.

Jacobs, C., Glazebrook, K., Collett, T., More, A., and McCarthy, C. (2017). Finding strong lenses in CFHTLS using convolutional neural networks. , 471(1):167–181.

Jauzac, M., Clément, B., Limousin, M., Richard, J., Jullo, E., Ebeling, H., Atek, H., Kneib, J. P., Knowles, K., Natarajan, P., Eckert, D., Egami, E., Massey, R., and Rexroth, M. (2014). Hubble Frontier Fields: a high-precision strong-lensing analysis of galaxy cluster MACSJ0416.1-2403 using ~200 multiple images. , 443(2):1549–1554.

Jauzac, M., Eckert, D., Schwinn, J., Harvey, D., Baugh, C. M., Robertson, A., Bose, S., Massey, R., Owers, M., Ebeling, H., Shan, H. Y., Jullo, E., Kneib, J.-P., Richard, J., Atek, H., Clément, B., Egami, E., Israel, H., Knowles, K., Limousin, M., Natarajan, P., Rexroth, M., Taylor, P., and Tchernin, C. (2016). The extraordinary amount of substructure in the Hubble Frontier Fields cluster Abell 2744. *Monthly Notices of the Royal Astronomical Society*, 463(4):3876–3893.

Jauzac, M., Harvey, D., and Massey, R. (2018). The shape of galaxy dark matter haloes in massive galaxy clusters: insights from strong gravitational lensing. , 477(3):4046–4051.

Jiang, G. and Kochanek, C. S. (2007). The Baryon Fractions and Mass-to-Light Ratios of Early-Type Galaxies. , 671(2):1568–1578.

Jullo, E. and Kneib, J. P. (2009). Multiscale cluster lens mass mapping - I. Strong lensing modelling. , 395(3):1319–1332.

Jullo, E., Kneib, J. P., Limousin, M., Elíasdóttir, Á., Marshall, P. J., and Verdugo, T. (2007). A Bayesian approach to strong lensing modelling of galaxy clusters. *New Journal of Physics*, 9(12):447.

Kassiola, A. and Kovner, I. (1993). Elliptic Mass Distributions versus Elliptic Potentials in Gravitational Lenses. , 417:450.

Kneib, J.-P., Ellis, R. S., Santos, M. R., and Richard, J. (2004). A Probable z~7 Galaxy Strongly Lensed by the Rich Cluster A2218: Exploring the Dark Ages. , 607(2):697–703.

Kneib, J. P., Ellis, R. S., Smail, I., Couch, W. J., and Sharples, R. M. (1996). Hubble Space Telescope Observations of the Lensing Cluster Abell 2218. , 471:643.

Kneib, J.-P. and Natarajan, P. (2011). Cluster lenses. , 19:47.

Koopmans, L. V. E., Bolton, A., Treu, T., Czoske, O., Auger, M. W., Barnabè, M., Vegetti, S., Gavazzi, R., Moustakas, L. A., and Burles, S. (2009). The Structure and Dynamics of Massive Early-Type Galaxies: On Homology, Isothermality, and Isotropy Inside One Effective Radius. , 703(1):L51–L54.

Kravtsov, A. (2010). The Dark Matter Annihilation Signal from Dwarf Galaxies and Subhalos. *Advances in Astronomy*, 2010:281913.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Lagattuta, D. J., Richard, J., Bauer, F. E., Clément, B., Mahler, G., Soucail, G., Carton, D., Kneib, J.-P., Laporte, N., Martinez, J., Patrício, V., Payne, A. V., Pelló, R., Schmidt, K. B., and de la Vieuville, G. (2019). Probing 3D structure with a large MUSE mosaic: extending the mass model of Frontier Field Abell 370. , 485(3):3738–3760.

Lagattuta, D. J., Richard, J., Clément, B., Mahler, G., Patrício, V., Pelló, R., Soucail, G., Schmidt, K. B., Wisotzki, L., Martinez, J., and Bina, D. (2017). Lens modelling Abell 370: crowning the final frontier field with MUSE. , 469(4):3946–3964.

Laplace, P. S. (1799). Beweis des Satzes, dass die anziehende Kraft bey einem Weltkörper so groß seyn könne, dass das Licht davon nicht ausströmen kann. *Allgemeine Geographische Ephemeriden*, 4(1):1–6.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Limousin, M., Richard, J., Kneib, J. P., Brink, H., Pelló, R., Jullo, E., Tu, H., Sommer-Larsen, J., Egami, E., Michałowski, M. J., Cabanac, R., and Stark, D. P. (2008). Strong lensing in Abell 1703: constraints on the slope of the inner dark matter distribution. , 489(1):23–35.

Little, J. D. C. (1961). A proof for the queuing formula: L = w. *Oper. Res.*, 9(3):383–387.

Lotz, J. M., Koekemoer, A., Coe, D., Grogin, N., Capak, P., Mack, J., Anderson, J., Avila, R., Barker, E. A., Borncamp, D., Brammer, G., Durbin, M., Gunning, H., Hilbert, B., Jenkner, H., Khandrika, H., Levay, Z., Lucas, R. A., MacKenty, J., Ogaz, S., Porterfield, B., Reid, N., Robberto, M., Royle, P., Smith, L. J., Storrie-Lombardi, L. J., Sunnquist, B., Surace, J., Taylor, D. C., Williams, R., Bullock, J., Dickinson, M., Finkelstein, S., Natarajan, P., Richard, J., Robertson, B., Tumlinson, J., Zitrin, A., Flanagan, K., Sembach, K., Soifer, B. T., and Mountain, M. (2017). The Frontier Fields: Survey Design and Initial Results. , 837(1):97.

Mahler, G., Richard, J., Clément, B., Lagattuta, D., Schmidt, K., Patrício, V., Soucail, G., Bacon, R., Pello, R., Bouwens, R., Maseda, M., Martinez, J., Carollo, M., Inami, H., Leclercq, F., and Wisotzki, L. (2018). Strong-lensing analysis of A2744 with MUSE and Hubble Frontier Fields images. , 473(1):663–692.

Marshall, P. J., Treu, T., Melbourne, J., Gavazzi, R., Bundy, K., Ammons, S. M., Bolton, A. S., Burles, S., Larkin, J. E., Le Mignant, D., Koo, D. C., Koopmans, L. V. E., Max, C. E., Moustakas, L. A., Steinbring, E., and Wright, S. A. (2007). Superresolving Distant Galaxies with Gravitational Telescopes: Keck Laser Guide Star Adaptive Optics and Hubble Space Telescope Imaging of the Lens System SDSS J0737+3216. , 671(2):1196–1211.

Metcalf, R. B., Meneghetti, M., Avestruz, C., Bellagamba, F., Bom, C. R., Bertin, E., Cabanac, R., Courbin, F., Davies, A., Decencière, E., Flamary, R., Gavazzi, R., Geiger, M., Hartley, P., Huertas-Company, M., Jackson, N., Jacobs, C., Jullo, E., Kneib, J. P., Koopmans, L. V. E., Lanusse, F., Li, C. L., Ma, Q., Makler, M., Li, N., Lightman, M., Petrillo, C. E., Serjeant, S., Schäfer, C., Sonnenfeld, A., Tagore, A., Tortora, C., Tuccillo, D., Valentín, M. B., Velasco-Forero, S., Verdoes Kleijn, G. A., and Vernardos, G. (2019). The strong gravitational lens finding challenge. , 625:A119.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.

More, A., Cabanac, R., More, S., Alard, C., Limousin, M., Kneib, J. P., Gavazzi, R., and Motta, V. (2012). The CFHTLS-Strong Lensing Legacy Survey (SL2S): Investigating the Group-scale Lenses with the SARCS Sample. , 749(1):38.

Natarajan, P. and Kneib, J.-P. (1997). Lensing by galaxy haloes in clusters of galaxies. , 287(4):833–847.

Natarajan, P., Kneib, J.-P., Smail, I., and Ellis, R. S. (1998). The Mass-to-Light Ratio of Early-Type Galaxies: Constraints from Gravitational Lensing in the Rich Cluster AC 114. , 499(2):600–607.

Navarro, J. F., Eke, V. R., and Frenk, C. S. (1996). The cores of dwarf galaxy haloes. , 283(3):L72–L78.

Navarro, J. F., Frenk, C. S., and White, S. D. M. (1997). A Universal Density Profile from Hierarchical Clustering. , 490(2):493–508.

Osawa, K., Swaroop, S., Jain, A., Eschenhagen, R., Turner, R. E., Yokota, R., and Emtiyaz Khan, M. (2019). Practical Deep Learning with Bayesian Principles. *arXiv e-prints*, page arXiv:1906.02506.

Paraficz, D., Courbin, F., Tramacere, A., Joseph, R., Metcalf, R. B., Kneib, J.-P., Dubath, P., Droz, D., Filleul, F., Ringeisen, D., and Schäfer, C. (2016). The pca lens-finder: application to cfhtls. *A&A*, 592:A75.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1211.5063.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *ArXiv*, abs/1211.5063.

Peirani, S., Dubois, Y., Volonteri, M., Devriendt, J., Bundy, K., Silk, J., Pichon, C., Kaviraj, S., Gavazzi, R., and Habouzit, M. (2017). Density profile of dark matter haloes and galaxies in the HORIZON-AGN simulation: the impact of AGN feedback. , 472(2):2153–2169.

Petrillo, C. E., Tortora, C., Chatterjee, S., Vernardos, G., Koopmans, L. V. E., Verdoes Kleijn, G., Napolitano, N. R., Covone, G., Schneider, P., Grado, A., and McFarland, J. (2017). Finding strong gravitational lenses in the Kilo Degree Survey with Convolutional Neural Networks. , 472(1):1129–1150.

Postman, M., Coe, D., Benítez, N., Bradley, L., Broadhurst, T., Donahue, M., Ford, H., Graur, O., Graves, G., Jouvel, S., Koekemoer, A., Lemze, D., Medezinski, E., Molino, A., Moustakas, L., Ogaz, S., Riess, A., Rodney, S., Rosati, P., Umetsu, K., Zheng, W., Zitrin, A., Bartelmann, M., Bouwens, R., Czakon, N., Golwala, S., Host, O., Infante, L., Jha, S., Jimenez-Teja, Y., Kelson, D., Lahav, O., Lazkoz, R., Maoz, D., McCully, C., Melchior, P., Meneghetti, M., Merten, J., Moustakas, J., Nonino, M., Patel, B., Regös, E., Sayers, J., Seitz, S., and Van der Wel, A. (2012). The Cluster Lensing and Supernova Survey with Hubble: An Overview. , 199(2):25.

Quider, A. M., Pettini, M., Shapley, A. E., and Steidel, C. C. (2009). The ultraviolet spectrum of the gravitationally lensed galaxy 'the Cosmic Horseshoe': a close-up of a star-forming galaxy at z ~2. , 398(3):1263–1278.

Refsdal, S. (1964). On the possibility of determining Hubble's parameter and the masses of galaxies from the gravitational lens effect. , 128:307.

Richard, J., Kneib, J.-P., Ebeling, H., Stark, D. P., Egami, E., and Fiedler, A. K. (2011). Discovery of a possibly old galaxy at z= 6.027, multiply imaged by the massive cluster Abell 383. , 414(1):L31–L35.

Richard, J., Kneib, J.-P., Jullo, E., Covone, G., Limousin, M., Ellis, R., Stark, D., Bundy, K., Czoske, O., Ebeling, H., and Soucail, G. (2007). A Statistical Study of Multiply Imaged Systems in the Lensing Cluster Abell 68. , 662(2):781–796.

Richard, J., Pei, L., Limousin, M., Jullo, E., and Kneib, J. P. (2009). Keck spectroscopic survey of strongly lensed galaxies in Abell 1703: further evidence of a relaxed, unimodal cluster. , 498(1):37–47.

Riechers, D. A., Walter, F., Brewer, B. J., Carilli, C. L., Lewis, G. F., Bertoldi, F., and Cox, P. (2008). A Molecular Einstein Ring at z = 4.12: Imaging the Dynamics of a Quasar Host Galaxy Through a Cosmic Lens. , 686(2):851–858.

Rosenblatt, F. (1957). *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory.

Ruff, A. J., Gavazzi, R., Marshall, P. J., Treu, T., Auger, M. W., and Brault, F. (2011). The SL2S Galaxy-scale Lens Sample. II. Cosmic Evolution of Dark and Luminous Mass in Early-type Galaxies. , 727(2):96.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Rusin, D. and Kochanek, C. S. (2005). The Evolution and Structure of Early-Type Field Galaxies: A Combined Statistical Analysis of Gravitational Lenses. , 623(2):666–682.

Sand, D. J., Treu, T., Ellis, R. S., Smith, G. P., and Kneib, J.-P. (2008). Separating Baryons and Dark Matter in Cluster Cores: A Full Two-dimensional Lensing and Dynamic Analysis of Abell 383 and MS 2137-23. , 674(2):711–727.

Satish, N., Kim, C., Chhugani, J., Saito, H., Krishnaiyer, R., Smelyanskiy, M., Girkar, M., and Dubey, P. (2012). Can traditional programming bridge the ninja performance gap for parallel computing applications? *SIGARCH Comput. Archit. News*, 40(3):440–451.

Schaerer, D., Hempel, A., Egami, E., Pelló, R., Richard, J., Le Borgne, J. F., Kneib, J. P., Wise, M., and Boone, F. (2007). EROs found behind lensing clusters. I. Stellar populations and dust properties of optical dropout EROs and comparison with related objects. , 469(1):47–60.

Schneider, P., Ehlers, J., and Falco, E. E. (1992). *Gravitational Lenses*.

Shankar, F., Sonnenfeld, A., Grylls, P., Zanisi, L., Nipoti, C., Chae, K.-H., Bernardi, M., Petrillo, C. E., Huertas-Company, M., Mamon, G. A., and Buchan, S. (2018). Revisiting the bulge-halo conspiracy - II. Towards explaining its puzzling dependence on redshift. , 475(3):2878–2890.

Shankar, F., Sonnenfeld, A., Mamon, G. A., Chae, K.-H., Gavazzi, R., Treu, T., Diemer, B., Nipoti, C., Buchan, S., Bernardi, M., Sheth, R., and Huertas-Company, M. (2017). Revisiting the Bulge-Halo Conspiracy. I. Dependence on Galaxy Properties and Halo Mass. , 840(1):34.

Sharon, C. E., Tagore, A. S., Baker, A. J., Rivera, J., Keeton, C. R., Lutz, D., Genzel, R., Wilner, D. J., Hicks, E. K. S., Allam, S. S., and Tucker, D. L. (2019). Resolved Molecular Gas and Star Formation Properties of the Strongly Lensed z = 2.26 Galaxy SDSS J0901+1814. , 879(1):52.

Shu, Y., Brownstein, J. R., Bolton, A. S., Koopmans, L. V. E., Treu, T., Montero-Dorta, A. D., Auger, M. W., Czoske, O., Gavazzi, R., Marshall, P. J., and Moustakas, L. A. (2017). The Sloan Lens ACS Survey. XIII. Discovery of 40 New Galaxy-scale Strong Lenses. , 851(1):48.

Skilling, J. (2006). Nested sampling for general bayesian computation. *Bayesian Anal.*, 1(4):833–859.

Smail, I., Ivison, R. J., Blain, A. W., and Kneib, J. P. (1998). Faint Submillimeter Galaxies: Hubble Space Telescope Morphologies and Colors. , 507(1):L21–L24.

Soldner, J. (1801). Über die Ablenkung eines Lichtstrahls von seiner geradlinigen Bewegung durch die Attraktion eines Weltkörpers, an welchem er nähe vorbeigeht; von J. Soldner, 1801. (German) [On the deviation of a light ray from its straight-line motion due to the attraction of a heavenly body that it passes by closely]. 65(15):593–604. With a foreword, and considerable editing and abridging, by Philipp Lenard. An English translation of the complete original 1801 article is available in Wikisource.

Sonnenfeld, A., Treu, T., Marshall, P. J., Suyu, S. H., Gavazzi, R., Auger, M. W., and Nipoti, C. (2015). The SL2S Galaxy-scale Lens Sample. V. Dark Matter Halos and Stellar IMF of Massive Early-type Galaxies Out to Redshift 0.8. , 800(2):94.

Sonnenfeld, A., Verma, A., More, A., Allen, C., Baeten, E., Chan, J. H. H., Hutchings, R., Jaelani, A. T., Lee, C.-H., Macmillan, C., Marshall, P. J., O' Donnell, J., Oguri, M., Rusu, C. E., Veldthuis, M., Wong, K. C., Cornen, C., Davis, C., McMaster, A., Trouille, L., Lintott, C., and Miller, G. (2020). Survey of Gravitationally-lensed Objects in HSC Imaging (SuGOHI). VI. Crowdsourced lens finding with Space Warps. *arXiv e-prints*, page arXiv:2004.00634.

Stark, D. P., Ellis, R. S., Richard, J., Kneib, J.-P., Smith, G. P., and Santos, M. R. (2007). A Keck Survey for Gravitationally Lensed Lyα Emitters in the Redshift Range 8.5<z<10.4: New Constraints on the Contribution of Low-Luminosity Sources to Cosmic Reionization. , 663(1):10–28.

Steinhardt, C., Jauzac, M., Capak, P., Koekemoer, A., Oesch, P., Richard, J., Sharon, K. q., and BUFFALO (2018). The BUFFALO HST Survey. In *American Astronomical Society Meeting Abstracts #231*, volume 231 of *American Astronomical Society Meeting Abstracts*, page 332.04.

Suyu, S. H., Bonvin, V., Courbin, F., Fassnacht, C. D., Rusu, C. E., Sluse, D., Treu, T., Wong, K. C., Auger, M. W., Ding, X., Hilbert, S., Marshall, P. J., Rumbaugh, N., Sonnenfeld, A., Tewes, M., Tihhonova, O., Agnello, A., Blandford, R. D., Chen, G. C. F., Collett, T., Koopmans, L. V. E., Liao, K., Meylan, G., and Spiniello, C. (2017). H0LiCOW - I. H_0 Lenses in COSMOGRAIL's Wellspring: program overview. , 468(3):2590–2604.

Swinbank, A. M., Smail, I., Longmore, S., Harris, A. I., Baker, A. J., De Breuck, C., Richard, J., Edge, A. C., Ivison, R. J., Blundell, R., Coppin, K. E. K., Cox, P., Gurwell, M., Hainline, L. J., Krips, M., Lundgren, A., Neri, R., Siana, B., Siringo, G., Stark, D. P., Wilner, D., and Younger, J. D. (2010). Intense star formation within resolved compact regions in a galaxy at z = 2.3. , 464(7289):733–736.

Talbot, M. S., Brownstein, J. R., Bolton, A. S., Bundy, K., Andrews, B. H., Cherinka, B., Collett, T. E., More, A., More, S., Sonnenfeld, A., Vegetti, S., Wake, D. A., Weijmans, A.-M., and Westfall, K. B. (2018). SDSS-IV MaNGA: the spectroscopic discovery of strongly lensed galaxies. , 477(1):195–209.

Teyssier, R., Pontzen, A., Dubois, Y., and Read, J. I. (2013). Cusp-core transformations in dwarf galaxies: observational predictions. , 429(4):3068–3078.

Thomas, J., Saglia, R. P., Bender, R., Thomas, D., Gebhardt, K., Magorrian, J., Corsini, E. M., Wegner, G., and Seitz, S. (2011). Dynamical masses of early-type galaxies: a comparison to lensing results and implications for the stellar initial mass function and the distribution of dark matter. , 415:545–562.

Treu, T. (2010). Strong lensing by galaxies. *Annual Review of Astronomy and Astrophysics*, 48(1):87–125.

Treu, T. and Koopmans, L. V. E. (2004). Massive Dark Matter Halos and Evolution of Early-Type Galaxies to z $\sim$1. , 611(2):739–760.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg.

Vieira, J. D., Marrone, D. P., Chapman, S. C., De Breuck, C., Hezaveh, Y. D., Weiβ, A., Aguirre, J. E., Aird, K. A., Aravena, M., Ashby, M. L. N., Bayliss, M., Benson, B. A., Biggs, A. D., Bleem, L. E., Bock, J. J., Bothwell, M., Bradford, C. M., Brodwin, M., Carlstrom, J. E., Chang, C. L., Crawford, T. M., Crites, A. T., de Haan, T., Dobbs, M. A., Fomalont, E. B., Fassnacht, C. D., George, E. M., Gladders, M. D., Gonzalez, A. H., Greve, T. R., Gullberg, B., Halverson, N. W., High, F. W., Holder, G. P., Holzapfel, W. L., Hoover, S., Hrubes, J. D., Hunter, T. R., Keisler, R., Lee, A. T., Leitch, E. M., Lueker, M., Luong-van, D., Malkan, M., McIntyre, V., McMahon, J. J., Mehl, J., Menten, K. M., Meyer, S. S., Mocanu, L. M., Murphy, E. J., Natoli, T., Padin, S., Plagge, T., Reichardt, C. L., Rest, A., Ruel, J., Ruhl, J. E., Sharon, K.,

Schaffer, K. K., Shaw, L., Shirokoff, E., Spilker, J. S., Stalder, B., Staniszewski, Z., Stark, A. A., Story, K., Vand erlinde, K., Welikala, N., and Williamson, R. (2013). Dusty starburst galaxies in the early Universe as revealed by gravitational lensing. , 495(7441):344–347.

Walsh, D., Carswell, R. F., and Weymann, R. J. (1979). 0957+561 A, B: twin quasistellar objects or gravitational lens? , 279:381–384.

Wechsler, R. H. and Tinker, J. L. (2018). The connection between galaxies and their dark matter halos. *Annual Review of Astronomy and Astrophysics*, 56(1):435–487.

Yoshida, N., Springel, V., White, S. D. M., and Tormen, G. (2000). Weakly Self-interacting Dark Matter and the Structure of Dark Halos. , 544(2):L87–L90.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv e-prints*, page arXiv:1411.1792.

Zitrin, A., Fabris, A., Merten, J., Melchior, P., Meneghetti, M., Koekemoer, A., Coe, D., Maturi, M., Bartelmann, M., Postman, M., Umetsu, K., Seidel, G., Sendra, I., Broadhurst, T., Balestra, I., Biviano, A., Grillo, C., Mercurio, A., Nonino, M., Rosati, P., Bradley, L., Carrasco, M., Donahue, M., Ford, H., Frye, B. L., and Moustakas, J. (2015). Hubble Space Telescope Combined Strong and Weak Lensing Analysis of the CLASH Sample: Mass and Magnification Models and Systematic Uncertainties. , 801(1):44.

Zwicky, F. (1937). On the Probability of Detecting Nebulae Which Act as Gravitational Lenses. *Physical Review*, 51(8):679–679.

Christoph Schaefer

Chemin de l'Orio 4, 1032 Romanel-sur-Lausanne
0041 78 964 32 23
schaefer.christoph@outlook.com
German & French national with residence permit B

Summary

PhD Student at EPFL under Jean-Paul Kneib in Astroinformatics, with a focus on Convolutional Neural Networks (CNN) and High Performance Computing (HPC) based software. Experience in Research & Development and project management with versatile informatics and physics background. German & French national having worked, lived and studied in Germany, Argentina and Switzerland.

Projects

Lenstool-HPC: HPC-based Mass Modelling Software — September 2016 – December 2019
EPFL, LASTRO, Lausanne
Development of Lenstool-HPC, a gravitational lensing software for modelling mass distribution of galaxies and clusters, using C++ and Cuda. Collaboration with the Swiss National Computing Center (CSCS) for large scale deployment.

Lensfinder: CNN-based Classifier Pipeline — September 2016 – September 2018
European Space agency, Madrid
Development of a competition winning CNN based classifier for the European Space Agency. Subsequent collaborative integration into data product pipeline for EUCLID, an ESA space based satellite.

Work Experience

Project Supervisor — September 2016 – August 2019
EPFL, LASTRO, Lausanne
Creation and supervision of machine-learning based master projects for improvement of astrophysical software, data processing and data analysis. Methods used included deep learning (LSTM, CNN, Resnets) and Bayesian statistics.

Research Assistant — July 2014 – September 2014
Observatoire de Sauvergny, Geneva
Development of a PCA based recognition software capable of reliably detecting gravitational lenses in low resolution astronomy catalogues. Programming and calibrating in python an alternative recognition software with the same goal based on simple image analysis.

Education

2016 – Present — **Ph.D**
Astrophysics
Lastro / Ecole Polytechnique Fédérale de Lausanne

2010 – 2016 — **Bachelor & Master of Science**
Physics
Ecole Polytechnique Fédérale de Lausanne

Skills

Computer Skills

Advanced: python, C++, Cuda, C
Intermediate: Java, LaTeX
Familiar with : Data processing, Data analysis, Agile Development, Database (Django), App development (Android)

Languages

Fluent: English, German, French
Intermediate: Spanish

Soft Skills

Versatility, Independence, Networking, Mentoring, Presentation, Team work, International Outlook