Computational Aesthetics and Image Enhancements using Deep Neural Networks

THÈSE N^{O} 8420 (2018)

PRÉSENTÉE LE 27 AVRIL 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'IMAGES ET REPRÉSENTATION VISUELLE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Bin JIN

acceptée sur proposition du jury:

Prof. P. Fua, président du jury
Prof. S. Süsstrunk, directrice de thèse
Prof. F. Dufaux, rapporteur
Dr S. Winkler, rapporteur
Prof. P. Frossard, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2018

Acknowledgements

The five and half years at Switzerland have been a wonderful time in my life. This experience has significantly changed my life and shaped my personalities in so many positive ways, which would not be possible without all the great people accompanied me. Here I would like to thank them for their support.

First of all, I would like to thank my supervisor, Prof Sabine Süsstrunk, for her guidance and support during my doctoral study. I started working with her since the first semester at EPFL, when I just graduated from bachelor study and had no skills in research. Her patience and encouragement always help me out of difficult times and lead me moving forward, contributing to the tremendous improvement of my technical and social skills. The freedom she provided for research and the flexibility of working time are also extremely valuable for me to enjoy the years as a PHD student. I am really grateful for the wonderful opportunity to work with her.

I also would like to thank Maria V. Ortiz Segovia, who was my supervisor from Océ during my 2nd to 4th years. I am very grateful for the useful insights and comments she provided during our weekly meeting. I also enjoyed the cooperation and communication with colleagues in this company, in particular Christophe Leynadier. I acknowledge Océ for providing fundings for my PHD, for my occasional visit to Paris, and for the three-month internship.

It was a great honor to have these distinguished jury members for my thesis: Prof. Pascal Fua, Prof. Pascal Frossard, Prof. Frédéric Dufaux, Dr. Stefan Winkler. I thank them for their time in reviewing my thesis and appreciate their useful comments.

I further want to thank my good friend and office-mate, Nikolaos Arvanitopoulos, who has been a great support during my ups and downs in life. I really enjoy the time we spend together both inside and outside EPFL. Besides, I am very grateful to meet the former and current colleagues in the lab: Achanta, Marjan, Sami, Majed, Ruofan, Fayez, Edo, Leo, Gökhan, Zahra, Cheryl, Appu, Neda, Albrecht, Damien, Dominic. They made the life in the lab interesting and colorful.

I also would like to thank my Chinese friends, Jian Zhang, Xifan Tang, Tian Guo, Jun Ma, Bin Ding, Jingjing Wang, Rong Hu, Hanjie Pan and many others, for the great moments we spend together in Switzerland. Without them, my life outside the office would be much less colorful. I am also very happy to meet my girl-friend during my PHD, Mengcheng Liu. I thank her for the support and accompany everyday.

Finally, I would like to thank my parents for their understanding and support all these

years. I have been away from them for many years. The time we spent together is so limited ever since I came to Switzerland. However, they always forgive me and support me to pursue my own interest in my study. Without them, it is impossible for me to complete this thesis. I thus dedicate this thesis to them.

Lausanne, December 4, 2017

Abstract

Imaging devices have become ubiquitous in modern life, and many of us capture an increasing number of images every day. When we choose to share or store some of these images, our primary selection criterion invariably is to choose the most visually pleasing ones. Yet, quantifying visual pleasantness is a challenge, as image aesthetics not only correlate with low-level image quality, such as contrast and blurriness, but also include mid-level and high-level visual processes, like composition and context. For most users, a considerable amount of manual effort and/or professional knowledge is required to get aesthetically pleasing images. Developing automatic solutions thus benefits a large community.

This thesis proposes several computational approaches to help users obtain the desired images. The first technique aims at automatically measuring the aesthetics, which benefits the users in selecting and ranking images. We form the aesthetics prediction problem as a regression task and train a deep neural network on a large image aesthetics dataset. The unbalanced distribution of aesthetics scores in the training set can result in bias of the trained model towards certain aesthetics levels. Therefore, we propose to add sample weights during training to overcome such bias. Moreover, we build a loss function on the histograms of user labels, thus enabling the network to predict not only the average aesthetics quality but also the difficulty of such predictions. Extensive experiments demonstrate that our model outperforms the previous state-of-the-art by a notable margin.

Additionally, we propose an image cropping technique that automatically outputs aesthetically pleasing crops. Given an input image and a certain template, we first extract a sufficient amount of candidate crops. These crops are later ranked according to the scores predicted by the pre-trained aesthetics network, after which the best crop is output to the users. We conduct psychophysical experiments to validate the performance.

Apart from cropping, we further present a keyword-based image color re-rendering algorithm. For this task, the colors in the input image are modified to be visually more appealing according to the keyword specified by users. Our algorithm applies local color re-rendering operations to achieve this goal. A novel weakly-supervised semantic segmentation algorithm is developed to locate the keyword-related regions where the color re-rendering operations are applied. The color re-rendering process benefits from the segmentation network in two aspects. Firstly, we achieve more accurate correlation measures between keywords and color characteristics, contributing to better re-rendering

results of the colors. Secondly, the artifacts caused by the color re-rendering operations are significantly reduced. The re-rendered images are visually more appealing according to our psychophysical experiments.

To avoid the need of keywords when enhancing image aesthetics, we explore generative adversarial networks (GANs) for automatic image enhancement. GANs are known for directly learning the transformations between images from the training data. To learn the image enhancement operations, we train the GANs on an aesthetics dataset with three different losses combined. The first two are standard generative losses that enforce the generated images to be natural and content-wise similar to the input images. We propose a third aesthetics loss that aims at improving the aesthetics quality of the generated images. Overall, the three losses together direct the GANs to apply appropriate image enhancement operations.

Keywords: aesthetics, image enhancement, color re-rendering, semantic segmentation, neural networks, deep learning, GANs

Résumé

Les appareils d'imagerie sont omniprésents dans la vie moderne, et nombreux d'entre nous prennent de plus en plus d'images chaque jour. Lorsque nous choisissons de partager ou d'enregistrer certaines de ces images, notre critère de sélection est invariablement de choisir les plus visuellement plaisantes. Cependant, la quantification de l'agrément visuel est un défi, comme l'esthétique de l'image est non seulement corrélée avec les simples mesures de qualité telles que le contraste et le flou, mais aussi avec des processus visuels plus complexes, comme la composition et le contexte. Pour la plupart des utilisateurs, un considérable effort et/ou une connaissance professionnelle est nécessaire pour obtenir des images esthétiquement plaisantes. Le développement de solutions automatiques bénéficie une large communauté.

Cette thèse propose plusieurs approches computationnelles pour aider les utilisateurs à obtenir les images souhaitées. La première technique vise à mesurer automatiquement la qualité esthétique, qui assiste les utilisateurs dans la sélection et le classement des images. On formule le problème de prédiction esthétique comme une tâche de régression et entraine un réseau de neurones à l'aide d'une large base de données d'esthétiques d'images. La répartition déséquilibrée des scores esthétiques dans l'ensemble d'entraînement peut créer un biais dans le model vers certains niveaux esthétiques. Par conséquent, on propose d'ajouter des poids d'échantillons pendant l'entraînement pour contrebalancer ce biais. En plus, on construit une fonction de perte sur les histogrammes des étiquettes des utilisateurs, permettant ainsi au réseau de mesurer non seulement la qualité esthétique mais aussi la difficulté dans ces prévisions. Des expériences approfondies démontrent que notre modèle surpasse l'état de l'art précédent par une marge notable.

Additionnellement, on propose une technique de découpage d'images qui génère automatiquement des rognages esthétiquement plaisants. Étant donné une image et un modèle, on extrait tout d'abord une quantité suffisante de rognages candidats. Ensuite, ceux-là sont arrangés selon les scores prédits par le réseau déjà entraîné et le meilleur résultat est produit pour l'utilisateur. On mène des tests psychophysiques pour valider la performance.

En plus, on présente encore un algorithme pour redéfinir les couleurs d'une image à partir d'un mot clé. Pour cette tâche, les couleurs de l'image d'entée sont modifiées pour devenir visuellement plus attrayantes selon le mot-clé spécifié par l'utilisateur. L'algorithme applique les opérations localement pour atteindre cet objectif. Un nouvel algorithme de segmentation, faiblement supervisé, est développé pour repérer les régions liées aux mots

clés où les opérations de re-rendu de couleur prennent place. Le processus de rendu de couleur bénéficie du réseau de segmentation en deux aspects. Premièrement, on obtient une corrélation plus précise entre le mot-clé et les caractéristiques de couleur, contribuant à une meilleure redéfinition des couleurs. Deuxièmement, les artéfacts causés par le re-rendu de couleur sont considérablement réduits. Selon les expériences psychophysiques, les images résultantes sont visuellement plus attrayantes que les images d'entrée.

Pour éviter le besoin de mots-clés dans l'amélioration des images, on explore les *Generative Adversarial Networks* (GANs) pour l'amélioration automatique des images. Les GANs sont connus leur capacité d'apprendre directement les transformations entre les images à partir des données d'entraînement. Pour apprendre les opérations d'amélioration des images, on entraine les GANs sur un ensemble de données esthétiques avec trois différentes fonctions de perte combinées. Les deux premières sont des pertes génératives normales qui forcent les images générées à être naturelles et avoir un contenu similaire aux images d'entrée. On propose une troisième fonction de perte esthétique qui vise à améliorer la qualité esthétique des images produites. Ensemble, les trois pertes dirigent les GANs pour appliquer les opérations d'amélioration adequates.

Mots-Clés : esthétiques, amélioration d'images, redéfinition des couleurs, segmentation sémantique, réseaux de neurones, GANs

Contents

List of Figures

List of Tables

1 Introduction

With the smartphone market thriving, imaging devices have become more prevalent than ever before. Using images to record and share daily life has become a trendy habit for many users, leading to a tremendous amount of images uploaded to social media platforms every day. Flickr, a long-standing image-sharing platform, reports around 1.68 million images uploaded per day in 2016 [1]. Instagram and Facebook claim even larger numbers, reaching 80 million and 350 million, respectively [2].

When uploading and sharing these images, the primary selection criterion for many users is to use visually pleasing images. The quality of imaging devices, such as smartphones or DSLR cameras, have improved dramatically over the years, resulting in better and better quality of the captured images. For instance, the resolution of images captured by smartphones has increased several times in the past decade: Nokia N90, a smartphone released in 2005, has a built-in camera with resolution of 2 million pixels, whereas the resolution of the latest IPhone 8 camera can reach 12 million pixels.

However, good imaging devices do not necessarily lead to visually pleasing images. Certain scene conditions, such as low lighting or high dynamic range scenes, can still cause unpleasant visual artifacts in the captured images. Image aesthetics is the concept that represents the overall visual appeal of images, which correlates with both low-level and high-level image characteristics. Low-level image characteristics, such as image resolution and noise level, are mainly determined by the imaging devices and the scene conditions; high-level characteristics are those that relate to semantic understanding of images, such as image composition and image content. As shown in Figure 1.1, two images with similar content but different composition are perceived as having different visual appeal. Since image aesthetics is determined by the combined set of both low-level and high-level characteristics, to get an aesthetically pleasing image is challenging for many users.

To get an aesthetically pleasing image often requires certain level of professional knowledge about photography, and/or a considerable amount of manual efforts. A common practice for many users is to take several images of the same scene with varying

(a) (b)

Figure 1.1: Two images with similar content but different composition. Image aesthetics can be affected by the composition.

compositions and different camera settings, after which they manually select the best one. Such scenarios necessitate a great deal of time in order to go through every candidate image and visually analyze its aesthetics quality. An automatic selection algorithm would hence be beneficial.

Additionally, many users tend to apply post-processing on the captured images to further enhance their aesthetics. Certain image-editing softwares, such as Photoshop and Lightroom, are widely used for this purpose. Figure 1.2 is the user interface for Photoshop. Although equipped with sophisticated editing functions, Photoshop is known to be complicated and tedious for non-professional users. It hence becomes increasingly important to develop computational techniques that can ease users' efforts in obtaining aesthetically pleasing images.

In this thesis, we develop several such algorithms. These algorithms can be summarized into two categories:

- **To build a computational approach that automatically judges the aesthetics quality (in terms of aesthetics scores) of an input image.** Image aesthetics is a non-deterministic concept. Users hold various understandings and preferences of image aesthetics. In this thesis, we focus on predicting the average aesthetics scores labeled by the crowd, not on modeling personal preferences. Our algorithm automates aesthetics judgment, which can be used in both image selection, image ranking, and image enhancement.
- **To develop several enhancement techniques that can automatically edit images for better aesthetics.** In this thesis, image enhancement is referred to as all operations that improve the overall aesthetics, including operations that focus on improving low-level image characteristics, such as color editing or super-

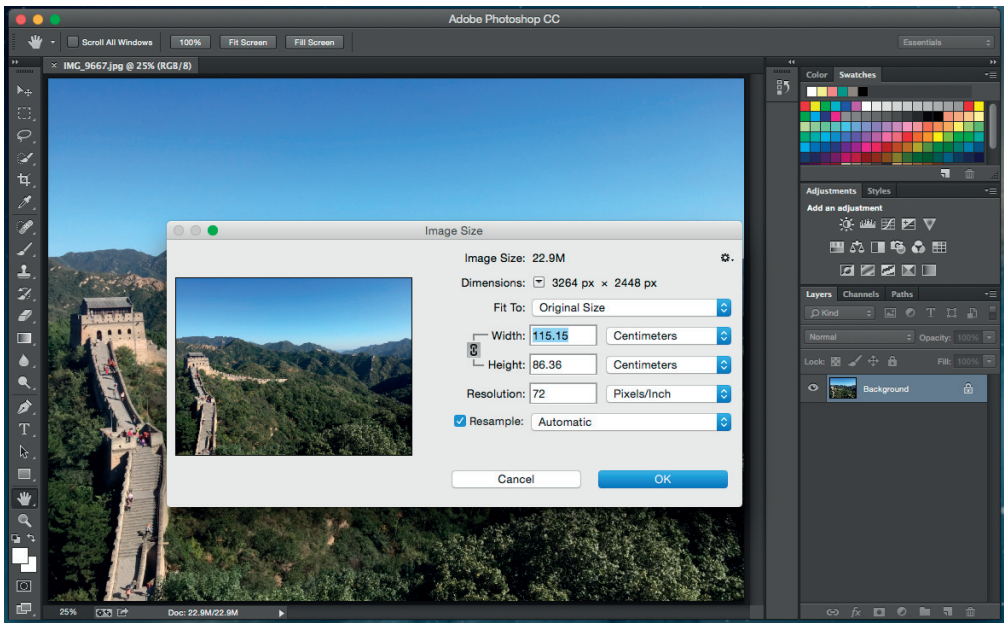


Figure 1.2: The user interface of Photoshop. As can be seen, the interface is filled with various operations, rendering it difficult for amateur users to properly edit images.

resolution, and those that modify high-level characteristics, such as cropping and image inpainting.

1.1 Computational Aesthetics

Currently, users perform image aesthetics judgment by visually evaluating the aesthetics quality according to personal preferences and past experiences, which is time-consuming. An automatic aesthetics predictor can thus benefit users for efficiently selecting images when they try to upload and share them. We formulate the aesthetics prediction problem as a regression task and solve it with a deep neural network. In recent years, neural networks have been widely adopted for many computer vision and image processing tasks, such as image classification [3, 4, 5], face recognition [6, 7], image denoising [8, 9], and image super-resolution [10, 11]. Although they achieve superior performance, deep neural networks are also criticized for the requirement of massive amount of training data. Thanks to the efforts by Murray *et al.* [12], a large aesthetics dataset (AVA dataset) was collected and made publicly available. 255,530 images are included in this dataset. Each image has aesthetics labels from approximately 200 different users. The collective aesthetics labels indicate the average judgment of aesthetics by the crowd.

We train a deep neural network on the AVA dataset in order to predict the aesthetics scores. Due to the unbalanced distribution of aesthetics scores in the AVA dataset, models trained on this dataset are often biased towards certain aesthetics levels. To overcome

this bias, we propose to use weighted samples during training. Two types of losses are tested to supervise the aesthetics network. The first loss is a mean square loss built on the average aesthetics scores, which directs the network to predict the average aesthetics quality of an input image. The second loss focuses on predicting the histogram of the collective aesthetics scores using a χ^2 loss. A histogram not only represents the average aesthetics judgment, but also reveals the difficulty level of such judgment. Extensive experiments demonstrate that our aesthetics networks produce state-of-the-art results for the aesthetics prediction task. Figure 1.3 shows sample results of our aesthetics networks.

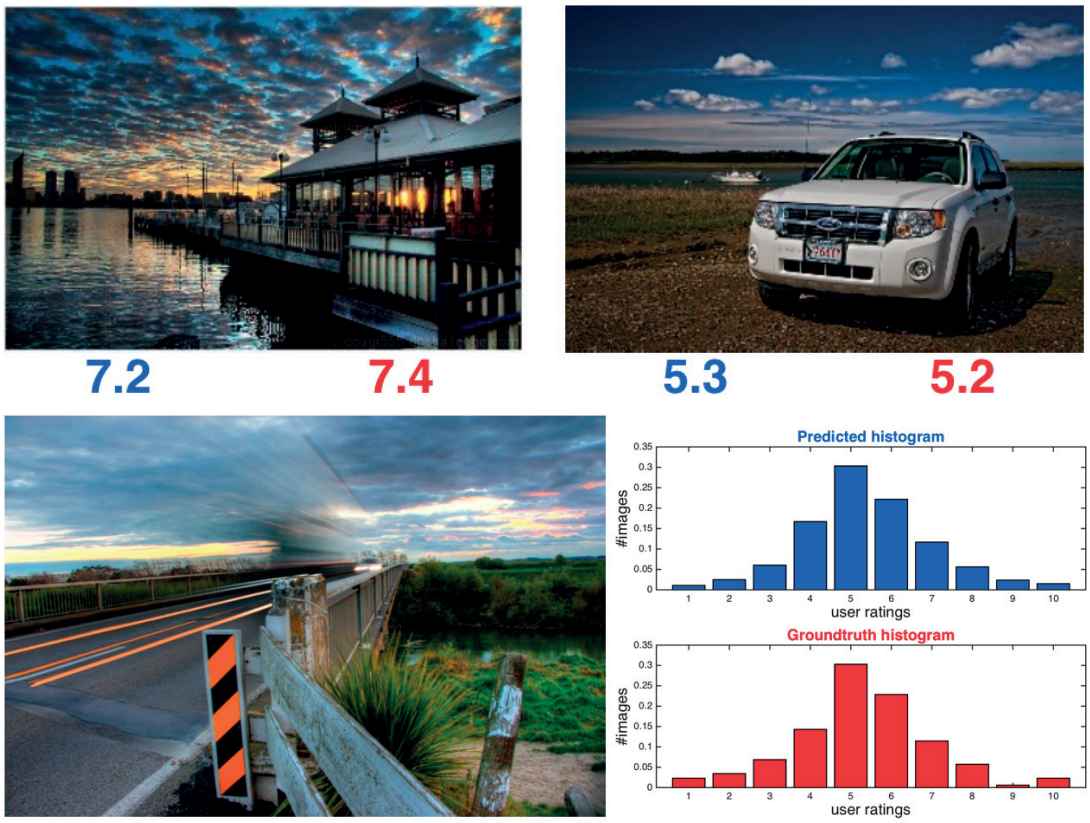


Figure 1.3: Sample results of our aesthetics networks. The aesthetics scores range from 1 to 10, where 1 represents the lowest aesthetics quality and 10 is the highest. In the first row, the values in blue are the predicted aesthetics scores from our regression network. The red values are the groundtruth scores provided by the AVA dataset. In the second row are the predicted histogram and the groundtruth histogram of user labels by our histogram prediction network.

1.2 Image Enhancement

As image aesthetics correlates with a wide range of image characteristics, there are numerous types of techniques to enhance the image aesthetics. The spectrum ranges from operations that enhance the low-level image characteristics, such as image denoising, to

those that focus on high-level image characteristics, such as cropping (modifies the image composition) and image inpainting (alters the content of the image).

Among all possible techniques, we concentrate on the following three aspects:

- image cropping: automatically generate an aesthetically pleasing crop.
- keyword-based color re-rendering: re-render the colors of an image according to the keywords specified by users.
- GANs-based image enhancement: automatically adjust image tones using Generative Adversarial Networks (GANs).

1.2.1 Image Cropping

When uploading or sharing images, users are often required to crop the image to fit into a fixed template. For example, when creating a profile for Facebook, users are asked to upload an image and crop it according to the Facebook template, as shown in Figure 1.4. For many blogs and magazines, images are also cropped according to certain templates for better composition of the whole page. Instead of using a random crop, users always prefer to adjust the crop manually for better visual appearance. Such scenarios call for an automatic cropping technique.

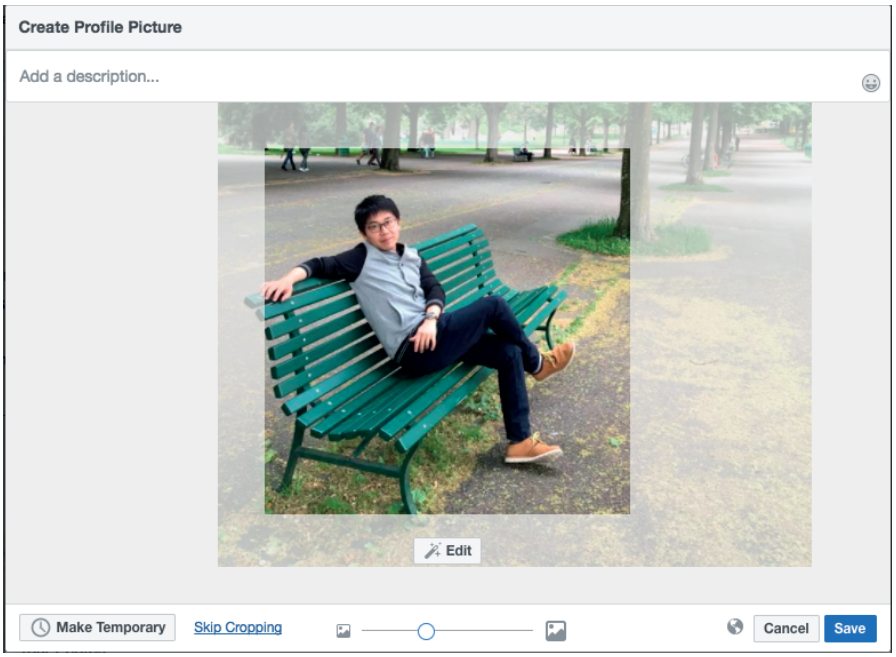


Figure 1.4: Cropping an image accord to a square template, in order to create a profile image for Facebook.

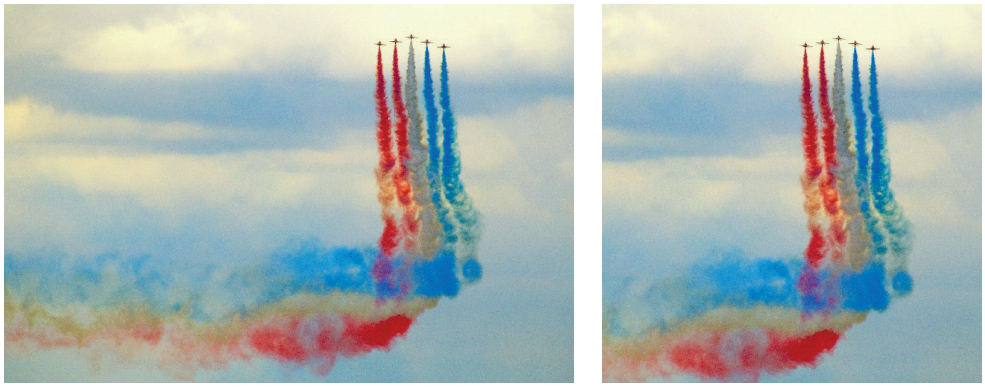
We propose a cropping algorithm that can automatically output a visually pleasing

crop to fit a template, as shown in Figure 1.5. Given an input image, our method first extracts a stack of candidate crops, respecting the shape constraint of the template. These candidate crops are then ranked according to the predicted aesthetics scores from our pre-trained aesthetics predictor. The crop with the highest aesthetics score is returned as the output crop. We conduct a psychophysical experiment to validate the performance of this method.

1.2.2 Keyword-based Color Re-rendering

With the dramatic progress of consumer cameras, sophisticated in-camera processing pipelines have been developed to improve the quality of the captured images. Certain operations, such as white-balancing and tone-mapping, are designed to ensure the appeal of the captured colors. However, due to the complex scenes or the misconfiguration of cameras, the output images might still have unfaithful or unappealing colors. We design a color re-rendering system to help users improve the colors if necessary.

When re-rendering colors, two problems need to be addressed: (1) What should the target colors be like? and (2) Where are the source colors that need to be re-rendered? Current algorithms [13,14,15,16] rely on supervision from users to solve these two problems. Several types of supervision are adopted, such as color strokes or color palettes [15, 16]. Among them, keywords serve as a simple and intuitive way for users to control the re-rendering operations. Albrecht *et al.* [17] propose a statistical framework to tackle the first problem. Their framework computes links between keywords and color characteristics, thus it can be used to find the target colors of a keyword. However, the localization problem (the second problem) is not fully considered in their approach, which leads to noticeable artifacts in many cases. We propose to incorporate a semantic segmentation algorithm into the pipeline. The segmentation masks can indicate the location of the source colors that need to be re-rendered.



(a) Input image (b) Generated crop

Figure 1.5: Our method automatically generates an aesthetically pleasing square crop. Here we use a square template.

Traditional semantic segmentation algorithms [18, 19, 20, 21] often require pixel-wise human annotations for training, as shown in Figure 1.6. Extending these algorithms to arbitrary keywords is thus expensive and tedious. We develop a novel weakly-supervised semantic segmentation algorithm that does not rely on pixel-wise human annotations for training. Instead, we directly query images from the internet. Our segmentation network progressively generates more and more accurate segmentation masks through a three-stage training pipeline.

Figure 1.6: Sample images with their pixel-wise segmentation annotations. These images are from the PASCAL VOC 2012 set [22].

Compared to the system in [17], we use the segmentation masks to achieve two benefits. First, from the segmentation masks, we obtain more accurate correlations between keywords and color characteristics, thus contributing to more pleasing target colors. Second, the segmentation masks are also adopted to locate the source colors, which results in locally applied re-rendering operations that reduce many artifacts. Through our algorithm, the colors of an input image are modified to be visually more appealing according to a keyword, as shown in Figure 1.7.

1.2.3 GANs for Image Enhancement

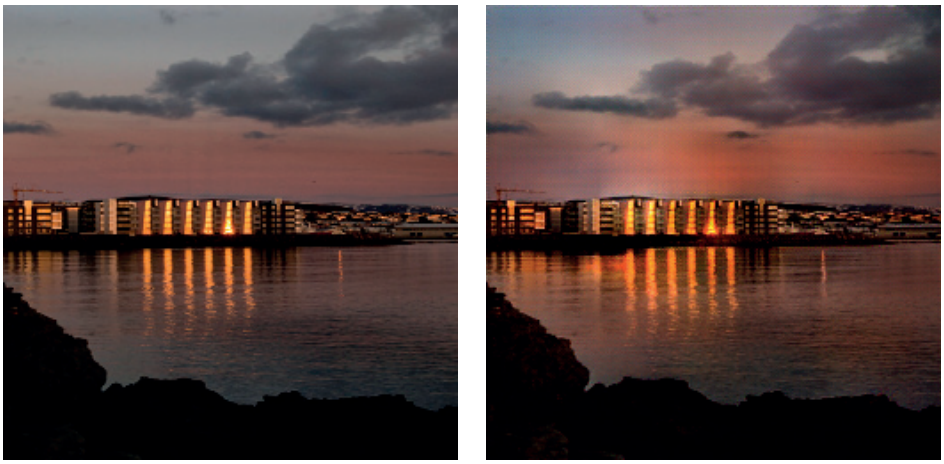
Although keywords are considered to be an easy source of supervision, it is still preferable to have a fully-automatic image enhancement system. In this thesis, we explore generative adversarial networks (GANs) for this purpose. GANs are known for automatically learning the appropriate operations from the training data. Therefore, GANs have been widely adopted for many image-enhancement applications, such as image super-resolution and image denoising. We use GANs for automatically enhancing image aesthetics by performing mainly tone adjustment.

Training GANs often requires pairs of images, so as to indicate the enhancement operations intuitively through the contrast between the pair of images. For image aesthetics enhancement, building a pair of training images means to edit one image to be

(a) Original image (b) Color re-rendering result

Figure 1.7: Sample result of our keyword-based color re-rendering system. Given a keyword ('strawberry'), the colors of the strawberries in the original image are re-rendered to be more visually appealing.

aesthetically more pleasing, which already requires a considerable amount of user effort. It is tedious and expensive to collect a sufficient amount of such pairs, with varying aesthetics qualities. To overcome this limitation, we propose a three-losses structure to train the GANs. The first one is the standard loss from the discriminative network, which forces the generated image to be natural. The second loss is the perceptual loss [23] that ensures content similarity between the input image and the enhanced image. For the third loss, we propose a novel aesthetics loss for improving the aesthetics of the generated image. Given an input image, our GANs enhance it to be visually more pleasing while still keeping the content, as shown in Figure 1.8.



(a) Original image (b) Enhanced result

Figure 1.8: Sample result for image aesthetics enhancement through GANs.

1.3 Thesis Outline

This thesis is structured as follows:

- Chapter 2: Related Work
  We discuss the work relevant to this thesis, summarized into four sub-fields: (1) neural networks, (2) computational aesthetics, (3) semantic segmentation, and (4) image enhancement
- Chapter 3: Computational Aesthetics
  In this chapter, we introduce our neural networks for automatically measuring image aesthetics. We also present the image cropping technique in this chapter, as it is a direct application of the aesthetics network.
- Chapter 4: Semantic Segmentation
  This chapter introduces the weakly supervised semantic segmentation algorithm, which is used in our color re-rendering system. We discuss the training pipelines of the segmentation network and validate its performance through qualitative and quantitative experiments.
- Chapter 5: Local Color Re-rendering
  We present the details of the color re-rendering system. The segmentation masks are employed in two steps: (1) obtain accurate correlations between keywords and colors, and (2) direct the color re-rendering operations to the related image regions.
- Chapter 6: GANs for Image Enhancement
  We present the architecture of our domain encoding GANs for image enhancement application. We compare this design with the traditional GANs and show the advantage of our approach.
- Chapter 7: Conclusion
  We conclude the thesis and discuss future work.

1.4 Summary of Contributions

Our contributions in this thesis can be summarized as follows:

- We train a deep neural network for the aesthetics prediction task. We propose to use sample weights during training to overcome the bias in the training set. Our algorithm achieves state-of-the-art performance.
- We present an image cropping technique that can automatically generate aesthetically pleasing crops.
- We propose a novel weakly supervised semantic segmentation algorithm that learns from web images. Our network is trained with a three-stage pipeline in order to progressively refine the segmentation masks. Our algorithm outperforms the related methods by a notable margin on the standard segmentation benchmark.
- We incorporate semantic segmentation into the color re-rendering pipeline. Our method achieves satisfactory results compared to the baseline approach.
- We explore GANs for automatic image enhancement. We propose a novel architecture, domain encoding GANs, that do not need image pairs for training.

2 Related Work

The computational algorithms developed in this thesis cover several research fields. As all the algorithms involve neural networks to some extent, we start with an introduction of neural networks in Section. 2.1. A great number of techniques have been proposed in this field. In this chapter, we review mainly three types of networks that are most related to this thesis: (1) Convolutional Neural Networks (CNNs), (2) Fully-Convolutional Neural Networks (FCNs), and (3) Generative Adversarial Networks (GANs).

In Section 2.2, we discuss the algorithms for aesthetics prediction, covering both the conventional approaches using handcrafted features and the recent ones based on neural networks. In Section 2.3, we specifically analyze semantic segmentation algorithms, because our color re-rendering technique relies on the segmentation masks. We first briefly explain the fully supervised semantic segmentation algorithms, then focus on discussing the weakly supervised approaches. In Section 2.4, we review the state-of-the-art image enhancement systems.

2.1 Neural Networks

The history of neural networks began in 1940s, when the functionality of neurons was discovered by neurophysiologist [24]. After that, researchers developed various models with multiple layers of neurons [25]. For many years, the performance of neural networks lagged far behind the requirements of real-world applications, mainly because of the limitation of datasets and the computing resources, However, due to recent progress in computing hardware and dataset collection, training very deep neural networks is now feasible. In 2012, Krizhevsky *et al.* proposed a deep neural network [3] for an image classification task that achieved breakthrough performance. Since then, deep neural networks have been widely adopted for many fields, such as computer vision [4, 5, 26], natural language processing [27,28,29] and robotics [30,31,32]. Various network architectures with increasing complexity and performance have been published [4, 5, 26, 33, 34, 35].

In Section 2.1.1, we first inspect the convolutional neural networks developed for classification and regression tasks. We then discuss, in Section 2.1.2, the fully convolutional neural networks that are mainly used for image segmentation. The details about generative adversarial networks are reviewed in Section 2.1.3. In the last section, we describe the useful techniques for training deep neural networks, including weights initialization, data augmentation, optimizers, and overfitting prevention.

There are a few concepts in neural networks that transcend the different models, as they are the basic blocks to build a deep neural network, such as convolution layer, pooling layer, and activation layer. For detailed definitions of these concepts, please refer to [36]. In this section, we focus on reviewing the network architectures.

2.1.1 Convolutional Neural Network

State-of-the-art Convulutional Neural Networks (CNNs) follow similar design rules. They consist mainly of several types of layers, including convolution layers, activation layers, pooling layers, normalization layers and fully connected layers. These layers are either concatenated or juxtaposed, forming complex neural networks. We briefly introduce four classic network architectures here: AlexNet [3], VGG [4], ResNet [5] and InceptionNet [26].

The reintroduction of neural networks since 2012 can be attributed to the success of AlexNet [3] for the ImageNet classification task [37]. Figure 2.1 shows the structure of AlexNet. This network takes an input image of $224 \times 224 \times 3$ and applies the first convolution with 96 kernels of size $11 \times 11 \times 3$ with stride 4. The output passes through another 4 layers of convolution before being fed into the fully connected layer. Relu activation [38] and max pooling layers are inserted to add non-linearity and to reduce dimensions. To compute the probability of each class, the output from the fully connected layer is followed by a 1000-way softmax. This network, with 60 million parameters in total, achieved top-1 and top-5 test-set error rate of 37.5% and 17.0%, which were more than 10% better than the previous state-of-the-art [39].

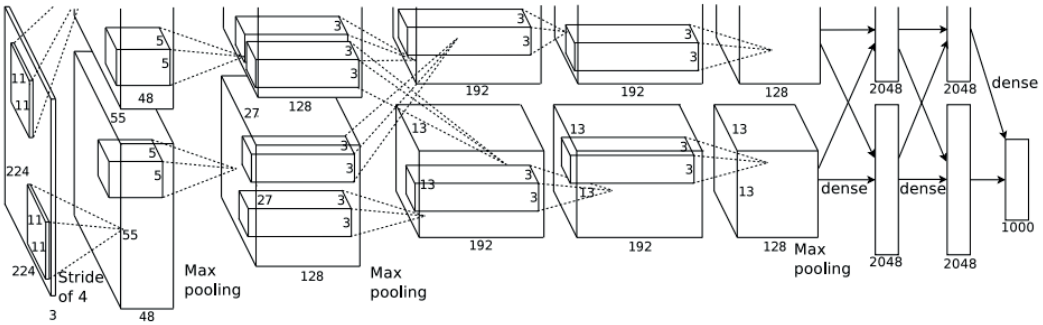


Figure 2.1: The structure of AlexNet [3].

After AlexNet, Simonyan *et al.* [4] propose a new architecture, called VGG, that further boosts the performance. In their paper, they propose several variations of VGG networks with different numbers of layers. The architecture of VGG16, the most widely used one of VGG family, is shown in Figure 2.2. VGG16 is designed with similar structure as AlexNet with one major difference. Instead of using large convolution kernels, such as 11×11 or 5×5 as in AlexNet, VGG16 only uses small convolution kernels of size 3×3. They use the concatenation of several small-kernel convolutions to achieve a receptive field that is similar to that of a large-kernel convolution. As a non-linear Relu activation function is applied after each convolution, such concatenation of multiple convolutions contributes to a better capability of modeling non-linear mappings. The VGG16 network achieves superior performance on the ImageNet classification task with the top-1 and top-5 test error of 25.6% and 8.1%. Due to its simple structure and great performance, the VGG16 model is used as the base model for many other applications [7, 40, 41, 42, 43]. Our aesthetics network in Chapter 3 is also based on the VGG16 model, with certain modifications to adapt to the aesthetics regression task.

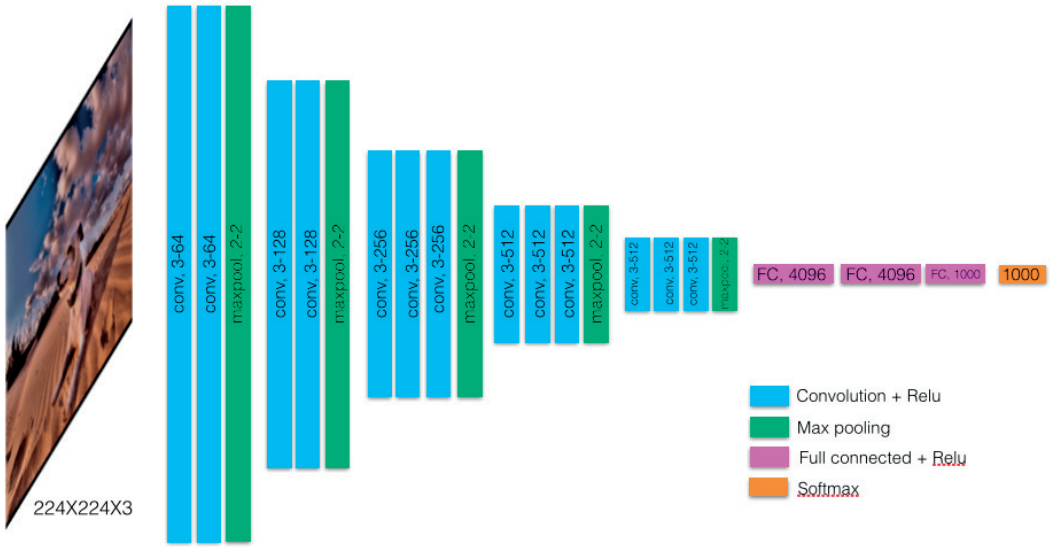
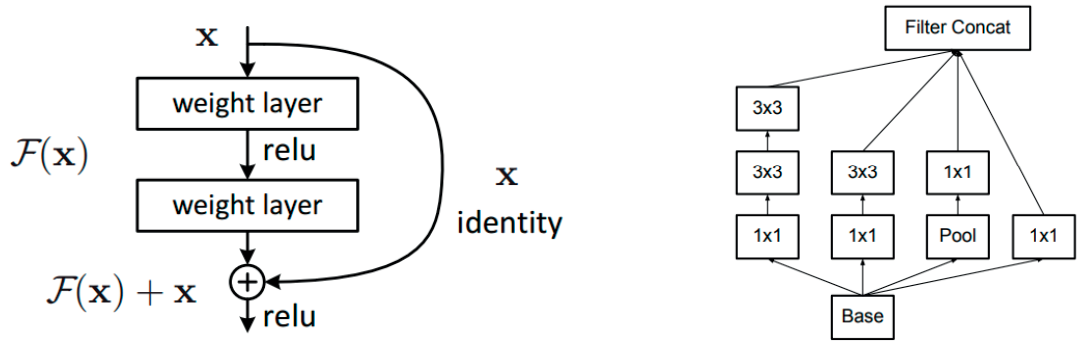


Figure 2.2: The structure of VGG16.

Endless efforts have been made to explore new architectures that are both faster and more powerful than VGG models. He *et al.* [5] propose to concatenate the residual blocks, as shown in Figure 2.3a, to build up a very deep neural network, called "ResNet". This design ensures that information can efficiently propagate through the network because its input is a direct part of the output, which enables training very deep networks. The residual network [5] achieves a top-5 error rate of 4.49% on the ImageNet classification task, already surpassing human performance of 5.1% [37].

Google develops the inception module [26], as shown in Figure 2.3b, for building networks. In one inception module, there are four parallel pathways for the input. Each pathway contains convolutions with different receptive fields. For instance, the

concatenation of two 3×3 convolutions contributes to an effective receptive field of 5×5. The concatenation of different pathways produces robust features that "model" the image in a multi-scale manner. Inception modules are sometimes combined with the residual block to build more powerful networks [44].



(a) Residual block (b) Inception module

Figure 2.3: Residual block [5] and inception module [26]. These two are basic blocks that can be concatenated to build a complex network.

2.1.2 Fully Convolutional Network

Many current CNNs are designed for image classification tasks. They model an input image into a 1D feature vector, rendering it difficult to maintain the spatial information (the local information at each position) from the input image to the feature vector. However, for some applications, spatial information is critical. For instance, image semantic segmentation aims at assigning a semantic label to each pixel in the input image. The label of a pixel is determined mainly by the local information at that specific position. Furthermore, the output of semantic segmentation is required to be a 3D image instead of a 1D vector. In such a scenario, traditional CNNs are therefore not suitable.

Long *et al.* [18] modify the VGG network into a fully convolutional style, which can handle inputs of various sizes and generate outputs of the same size as the inputs. Each fully connected layer is transformed into a convolution layer with a kernel that covers the entire input region. Transforming all the fully connected layers in a CNN leads to the fully convolutional network (FCN), shown in Figure 2.4. FCN effectively performs image classification on every patch of the input image in a sliding window manner, generating a probability value for each patch. The output from FCN is a 3D heatmap containing probabilities of each class. Bilinear interpolation is then applied onto the heatmap to generate a segmentation mask that is of the same size as the input image.

Spatial information is critical for segmentation. However, due to the pooling layers in CNNs, spatial information is gradually thrown away as the network goes deep. Therefore, it is beneficial to incorporate features from early layers of the network in the final

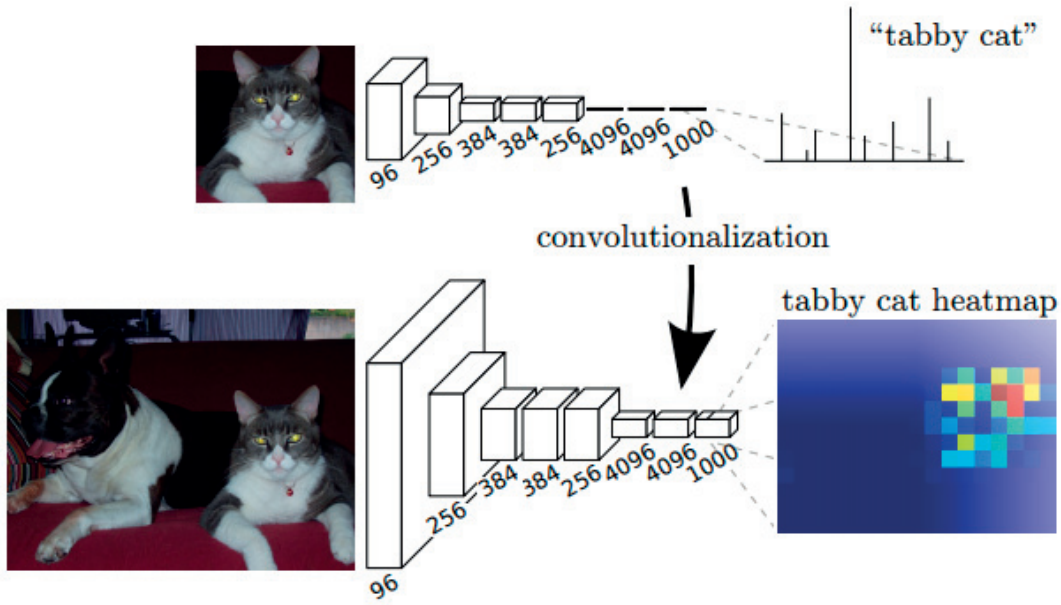


Figure 2.4: Transforming fully connected layers to convolution layers enables a classification net to output a heatmap [18].

classification stage, as these features better encode the spatial knowledge. Long *et al.* propose to concatenate the features from internal layers with those from the last layer, before being fed into the classification stage, as illustrated in Figure 2.5. Depending on which internal layer's activations are extracted, the last classification stage effectively performs classification on patches of different sizes. The output heatmap is upsampled 8, 16 or 32 times to get back to the input size. The best FCN model in [18] achieves 62.2 mean Intersection-over-Union (mIoU) value on PASCAL VOC 2012 segmentation test set [22], producing 20% relative improvement over the previous state of the art [45]. Our semantic segmentation algorithm in Chapter 4 is inspired by this design.

2.1.3 Generative Adversarial Networks

Generative models aim at modeling the underlying distribution of a training set. Some algorithms try to explicitly model the density of the distribution, such as Boltzmann machine [46] and variational autoencoder [47]. Other algorithms implicitly model the distribution by generating samples that are drawn from the same distribution as the training set. A typical case is using Generative Adversarial Networks (GANs) [34] to generate samples that follow the same distribution as the training set. Since their publication, GANs have been explored for many applications with various extensions [48, 49, 50, 51, 52].

The basic idea of GANs is to build up an adversarial game between two networks, as

illustrated in Figure 2.6. One of them is called generator: it creates samples from latent random variables with the intention of generating samples that are not distinguishable from the real samples. The other network is called discriminator: it aims at examining whether a sample is real or fake. The goal of the generator is to fool the discriminator, and the discriminator tries to classify every generated sample as fake and every real sample as real. During this counter game, both the generator and the discriminator are becoming more and more accurate, until the generated samples are indistinguishable from the real samples.

Formally, the generator defines a mapping function G from a latent variable z to a generated sample x, with parameters Θ^G. The discriminator defines a function D (with parameters Θ_D) that measures the probability of a sample x to be a real. For the generator, the aim is to fool the discriminator, albeit endure D to output high probability values for the generated samples. Therefore, the generator G tries to minimize the following loss L_G with respect to Θ_G:

$$L_G(\Theta_G) = \sum_z -\log D\Big(G(z)\Big) \tag{2.1}$$

Minimizing L_G effectively equals maximizing the log-likelihood of the generated samples $G(z)$ to be real. The loss used for the discriminator is:

$$L_D(\Theta_D) = \sum_z \sum_{\hat{x}} -\frac{1}{2}\log[D(\hat{x})] - \frac{1}{2}\log[1 - D\Big(G(z)\Big)] \tag{2.2}$$

Here $\hat{x}$ represents the real sample from the training set. This is the standard cross-entropy

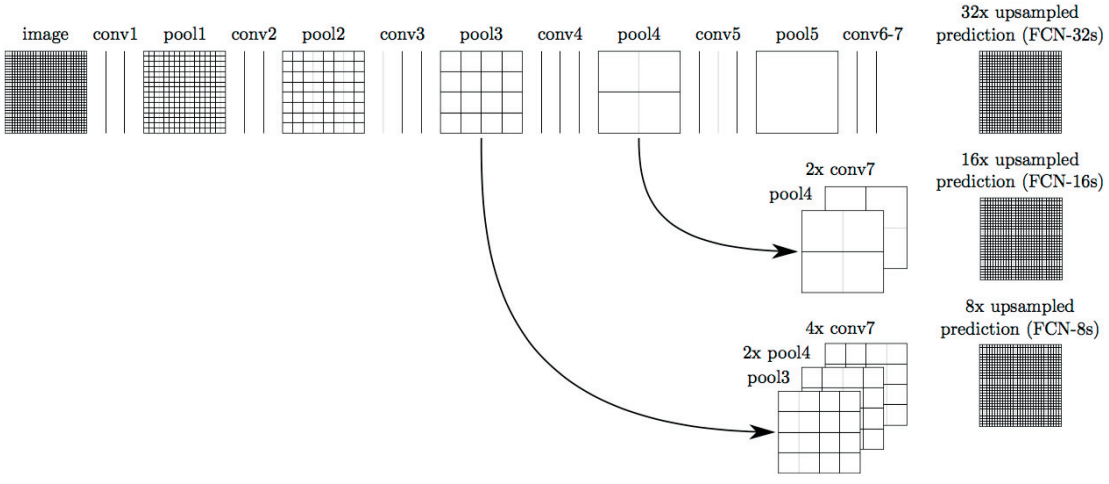


Figure 2.5: The Structure of FCN [18]. The outputs from the internal layers (pool3 or pool4) are combined with the output of the final layer (conv7) in order to generate heatmaps of different sizes. These heatmaps are then upsampled 8, 16, or 32 times to map to the original input size.

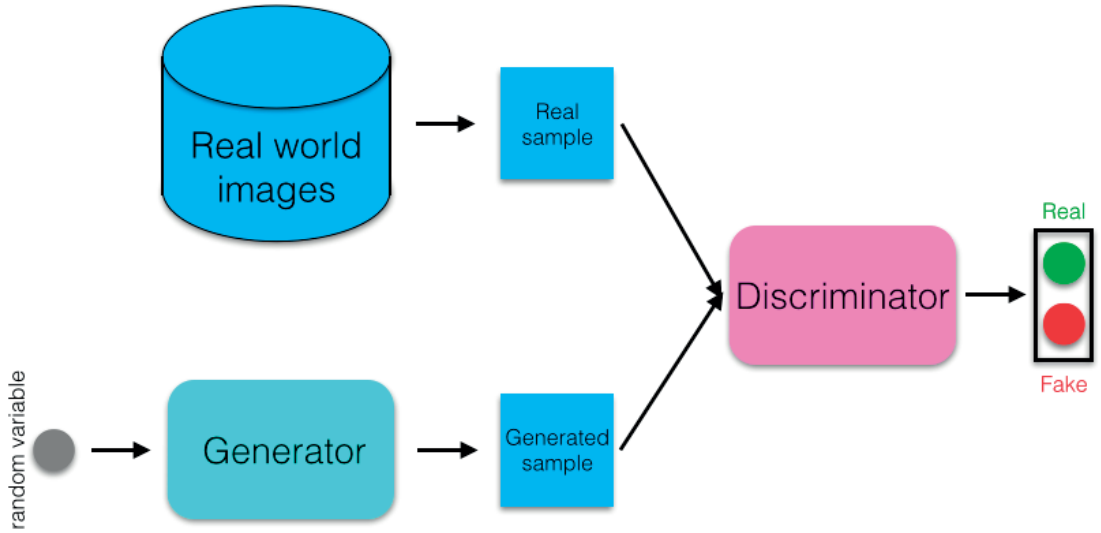


Figure 2.6: The structure of GANs that consists of a generator and a discriminator.

loss for binary classification problem.

Optimizing the generator and discriminator together is difficult. Goodfellow *et al.* [34] design an optimization procedure as explained in Algorithm 1. m is the size of each minibatch. k defines the number of steps to update the discriminator before updating one time the generator. Assuming k equals 1, the generator and the discriminator are then updated alternatively.

Algorithm 1: Training for GANs

for *number of iterations* **do**
  for k *steps* **do**
    Get: a minibatch of m noise samples $\{z^1, ..., z^m\}$ from latent noise space.
    Get: a minibatch of m samples $\{x^1, ..., x^m\}$ from the training set.
    Update: Update the parameters of the discriminator according to Eqn. 2.2
  end
  Get: a minibatch of m noise samples $\{z^1, ..., z^m\}$ from latent noise space.
  Update: Update the parameters of the generator according to Eqn. 2.1
end

The plain GANs in [34] show some difficulties in generating high-resolution images. In [53], the authors first decompose an image in a Laplacian pyramid, and then use multiple GANs to generate the details at different levels of the pyramid. The algorithm in [48], known as DCGANs, uses the concatenation of multiple convolutions to generate high-resolution images in one shot. Figure 2.7 shows bedroom images generated by DCGANs. Although they still have certain artifacts, the generated images are visually quite similar to real bedroom images.

Figure 2.7: Samples of bedroom images generated by DCGANs [48].

Later algorithms extend the designs of GANs and DCGANs to different applications. In [51], GANs are modified to be conditioned on some extra information y by feeding y as input to both the generator and the discriminator. Such structure can be used to generate images of a certain class by specifying the class information as y. The conditioned GANs are used in [54] for colorization of grayscale images. The authors use the grayscale image as the extra information y. DCGANs are explored for single image super-resolution in [49]. The authors modify the generator to be a convolutional network that takes in a low-resolution image and generates the corresponding high-resolution image. Isola *et al.* [55] use an autoencoder [56] style network as generator to tackle image translation problems. Their algorithm, named as pix2pix, is able to perform several image translation tasks, as shown in Figure 2.8 and Figure 2.9. Our DEGANs, proposed in Chapter 6, are based on the pix2pix work. We re-design the network architecture to adapt to the image enhancement problem.

Figure 2.8: Sample results of pix2pix [55] for edge → photo application. The network transforms edge images to photos. In each pair, the left one is the edge image and the right one is the generated photos.

Figure 2.9: Sample results of pix2pix [55] for map translation. Here pix2pix network learns to transform aerial photos to the corresponding satellite maps. The left are the input images and the right are the results.

2.1.4 Training Neural Networks

Deep neural networks are computational models with millions of parameters. For example, the VGG16 model [4] has 138 million parameters in total. Within such a high-dimensional parameter space, it is difficult to search for the optimal parameter set. Several useful techniques have been proposed to efficiently and effectively train deep neural networks.

2.1.4.1 Initialization

It is proven that a good initialization of parameters (also called weights) is a key factor to the convergence speed and the performance of neural networks [57]. For training networks from scratch, various methods [3, 57, 58, 59, 60, 61] have been tested to initialize the network. Apart from these, using the pre-trained weights from the ImageNet classification task [37] is another effective way for initializing networks, which often leads to better performance. Several classic networks achieve remarkable performance on ImageNet, as explained in Section 2.1.1. The weights of the convolution kernels in these models are shown to capture some meaningful patterns of natural images [62], thus being general to other imaging tasks. Especially for tasks that do not have enough training data, using the pre-trained weights for initialization would be beneficial. Our aesthetics network, in Chapter 3, therefore uses the VGG16 network with the pre-trained ImageNet weights.

2.1.4.2 Data Augmentation

Deep neural networks are known to be data hungry. Training a model with millions of parameters normally requires millions of training samples. This is expensive or even impossible for some tasks. A useful technique for increasing the amount of training data is to apply data augmentation. Various types of operations can be applied to augment the dataset, as long as those operations do not alter the groundtruth labels of the augmented results. For example, for the image classification task, image resizing, random cropping, random horizontal flipping, random noise injection, and random color modification are common augmentation methods. By applying one or several of these operations together, the quantity of the training data can be effectively increased by tens to hundreds times larger.

2.1.4.3 Optimizer

Training a deep neural network is essentially optimizing an extremely complex and non-convex loss function. Gradient descent is the most common optimization solution. Gradient descent methods compute the gradients of the loss function with respect to its parameters. These gradients are then used to update the parameters, based on different rules. Simple methods such as SGD directly remove the gradients, after being multiplied by a fixed learning rate, from the weights. Momentum [63] is a technique that can accelerate SGD during optimization. More advanced methods, such as Adagrad [64] and Adam [65], use adaptive learning rates for different parameters, thus often resulting in faster convergence speed and better performance. Figure 2.10 shows a comparison of different optimizers for the MNIST recognition task [66] with the same multilayer fully connected network. Although Adam shows the best performance for this task, the optimal optimizer might vary over different applications.

2.1.4.4 Overfitting Prevention

Deep neural networks often have significantly larger number of parameters than that of training samples, which can result in severe overfitting. Preventing these models from overfitting to the training data is hence very important. Overfitting is the phenomenon when the validation accuracy is notably lower than the training accuracy, as illustrated in Figure 2.11. Data augmentation is an effective way to prevent overfitting, which is already discussed in Section 2.1.4.2. Apart from this, there are also several other useful techniques. Srivastava *et al.* [67] introduce the dropout layer that randomly drops part of the activations during training. Dropout prevents the co-adaptation of neurons during training, thus performing well in practice. Another approach for reducing overfitting is to add a regularization term, such as L1 or L2 term, on the model weights. This is also known as weight decay. Additionally, in [68], the authors propose to apply early stopping during training. This method monitors training and validation losses after each epoch, and stops training when validation loss no longer decreases.

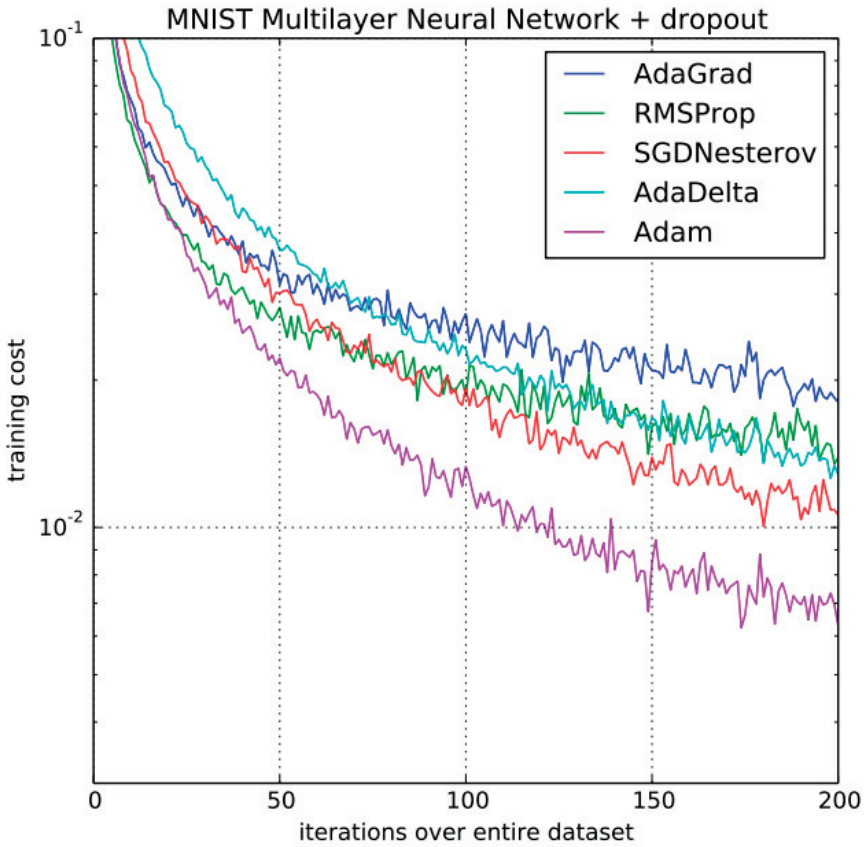


Figure 2.10: Comparison of different optimizers [65]. This plot shows the progress of the training losses for different optimizers with respect to the number of iterations.

2.2 Computational Aesthetics

Image aesthetics is the concept that represents the overall visual appeal of images. Automatically assessing image aesthetics is useful for many applications, which explains the dozens of algorithms published over the years. In this section, we first clarify the ambiguity between image aesthetics and image quality matrix. We then present the common aesthetics datasets used for aesthetics prediction. At last, we review the conventional methods using handcrafted features, followed by the recent deep learning approaches.

2.2.1 Image Aesthetics vs Image Quality

Image aesthetics is often associated with the concept of image quality, as both of them represent the perceived quality of an image. There exists a large body of research work on modeling and predicting image quality. Many successful image quality matrices, such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and Visual

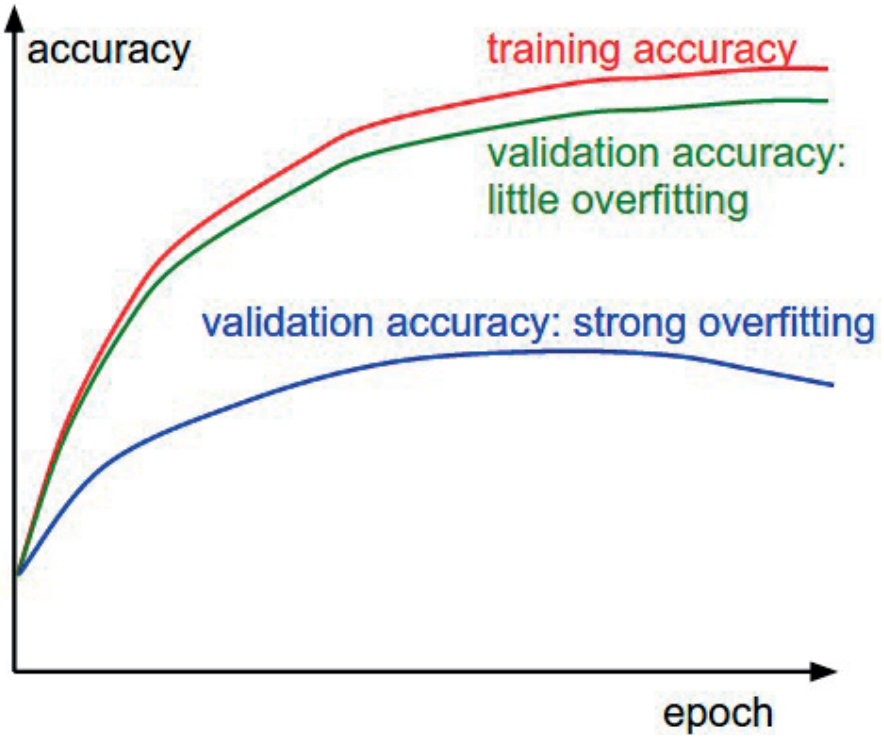


Figure 2.11: Overfitting phenomenon of neural networks. The gap between the training accuracy and the validation accuracy indicates the degree of overfitting. The blue line represents severe overfitting and the green line indicates slight overfitting. This figure is taken from the cs231n course from Stanford.

Information Fidelity (VIF), are widely used in many applications. Early approaches in this field use handcrafted features, such as natural scene statistics (NSS) [69], in the wavelet [69], sparse [70] or spatial domains [71] to derive a mathematical model for evaluating the image quality. Later methods employ deep neural networks to replace the need for handcrafted features [72, 73, 74].

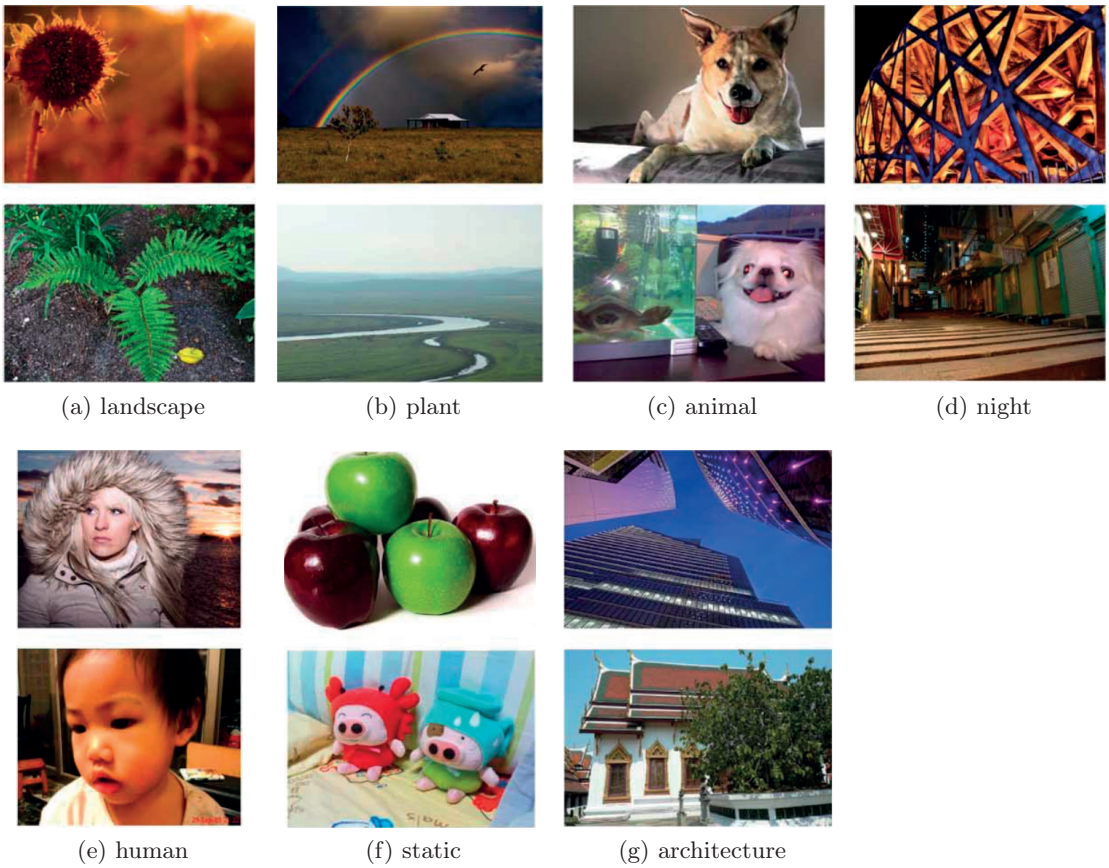
Although related, image aesthetics is also intrinsically different from image quality. Image quality measures the perceived quality of images with respect to certain distortions, such as blur or noise, whereas image aesthetics is the perceived appeal level of an image, which is more related to the understanding and appreciation of beauty and harmony. In this sense, a non-distorted image may have high values in the image quality assessment while not necessarily being an aesthetically pleasing image. Therefore, existing image quality matrices, such as PSNR and SSIM are not suitable for evaluating image aesthetics. In this thesis, we focus on building computational models for predicting image aesthetics. Note that, for simplicity, in the following chapters we use high/low quality to represent high/low *aesthetics* quality.

2.2.2 Aesthetics Datasets

Aesthetics datasets are the basis for many computational aesthetics algorithms. Some early work [75, 76] rely on their private collections of images, which are not made public to the community. Many efforts have been made to build publicly available large-scale aesthetics datasets.

2.2.2.1 CUHK-PQ Dataset

In [77], the authors release this CUHKPQ dataset containing 17,673 images, which are organized in 7 categories based on visual content: "animal", "plant", "static", "architecture", "landscape", "human", and "night". Each image is manually labeled with a binary label: high quality or low quality. Sample images from the CUHKPQ dataset are shown in Figure 2.12.



(a) landscape (b) plant (c) animal (d) night

(e) human (f) static (g) architecture

Figure 2.12: Sample images from CUHKPQ dataset [77]. For each category, the top one is a high-quality image and the bottom one is a low-quality image.

2.2.2.2 AADB Dataset

Kong *et al.* [78] build up the AADB dataset that contains not only aesthetics labels but also attributes labels. They select eleven attributes that are highly correlated with image aesthetics: interesting content, object emphasis, good lighting, color harmony, vivid color, shallow depth-of-field, motion blur, rule of thirds, balancing element, repetition, and symmetry. In total 10,000 images are collected from Flickr and labeled using Amazon Mechanical Turk[1]. Each image is labeled by 5 individual annotators for its overall aesthetics quality (using a score between 1 to 5), as well as each attribute quality (choosing a label between -1, 0, and 1). The annotator's ID is also released, which enables the study of personal preferences of aesthetics. We show some sample images in Figure 2.13.

Figure 2.13: Sample images of AADB dataset [78].

2.2.2.3 AVA Dataset

Previous aesthetics datasets all contain less than 100K images, rendering them insufficient for learning deep neural networks. Thanks to Murray *et al.* [12], a large-scale aesthetics dataset has been made available to the community. This dataset contains approximately 250,000 images natural and synthetic images obtained from the DPChallenge website[2]. Each of these image receives from 78 to 549 votes for its aesthetics quality, in the range of 1 to 10. 10 represents the highest aesthetics quality and 1 is the lowest. Figure 2.14 show some sample images from AVA dataset. Due to its large scale, AVA dataset is often used to train deep neural networks for aesthetics-related tasks.

[1] www.mturk.com
[2] www.dpchallenge.com

Figure 2.14: Sample images from AVA dataset [12]. The first row are images that have aesthetics scores larger than 5, the second row are images with aesthetics scores smaller than 5.

2.2.3 Conventional Methods

Conventional approaches for image aesthetics assessment follow a two-step framework. The first step is to build features to computationally model the photographic quality of the images. Some algorithms [77, 79, 80, 81, 82] design handcrafted features to explicitly model the photographic qualities that are related to image aesthetics, such as the lighting condition, contrast, color distribution and composition. Other algorithms [12, 83, 84, 85] use generic vision features, such as SIFT [86] descriptors, combined with Bag-of-Visual-Words (BOV) [87] or Fisher Vector (FV) [88]. These features are then passed through a learning step to generate the aesthetics labels, either as binary labels or continuous scores. Standard machine learning algorithms, such as Support Vector Machine (SVM) [89] or support vector regression [90], are employed at this stage.

2.2.4 Neural Network Based Methods

Neural networks are able to learn the proper feature representations from the data itself, thus significantly surpassing the capability of handcrafted features. Recent works propose to use deep neural networks for automatic aesthetics assessment. Generic features learned

from the ImageNet classification task are explored in [91] for predicting aesthetics. They use the features from AlexNet [3] and train a SVM on top. As AlexNet is not fine-tuned for aesthetics, its features are not adapted to the aesthetics prediction task, thus leading to unsatisfactory performance.

Wang *et al.* [92] modify the AlexNet architecture with 7 parallel convolution layers, where they collect images of seven different classes and train a convolution layer for each class. Lin *et al.* [93] propose to train a CNN to predict aesthetics for each randomly sampled image patch. A statistical aggregation structure is designed to aggregate the predictions from each patch. In [94], the authors propose to train a double column network. One column models the global view of the image and the other one models the local patch. Mai *et al.* [95] extend [94] by using 5-columns of VGG-type networks with an adaptive pooling layer. Each column has different receptive fields, hence modeling multi-scale image information. Kao *et al.* [96] propose to train three category-specific CNNs, aiming for three different classes: object, scene and texture. Depending on the category, the corresponding CNN takes different type of input, such as local patches or the whole images. A multi-task network is proposed in [97]: it learns to predict aesthetics, as well as the semantic category. The design of this network is based on the assumption that semantics information could help in the prediction of aesthetics. Depending on how to combine aesthetics labels with semantic labels, four types of architectures are proposed in [97], as shown in Figure 2.15.

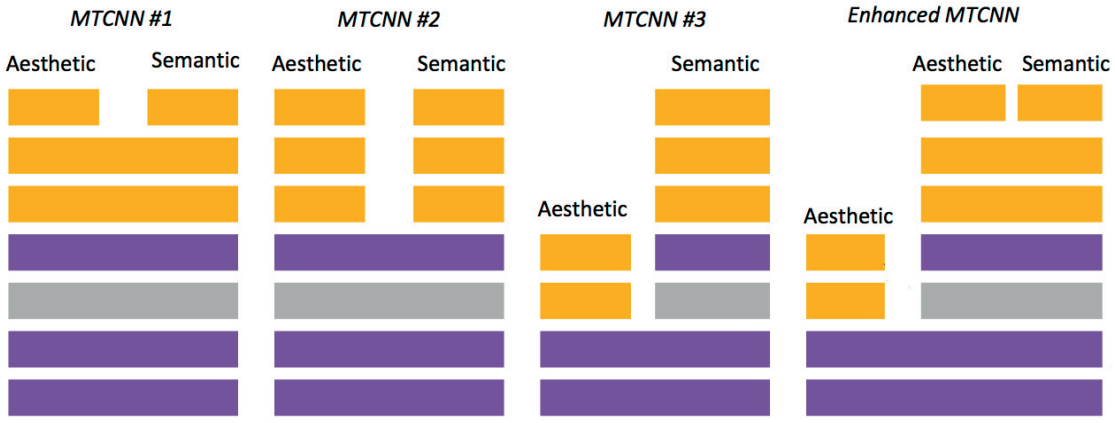


Figure 2.15: The structures of multi-task networks [97]. These networks try to predict aesthetic scores as well as semantic labels. Purple block: convolution layer + max pooling; gray block: convolution layer; yellow block: fully connected layer.

All the algorithms above model aesthetics assessment as a binary classification task. They split AVA dataset into two sets by thresholding the aesthetics scores. The network is trained to classify an image as either high quality or low quality. However, such setting discards the differences within a class, rendering these models less applicable for some real-world appliations. For instance, image ranking tasks need to measure the aesthetics on multiple levels, instead of on binary classes.

Kao et al. [98] propose a CNN regression model that provides continuous aesthetic scores. Following this line, our method also treats aesthetics prediction as a regression problem, which will be described in Chapter 3.

2.3 Semantic Segmentation

Semantic segmentation algorithms aim at generating pixel-wise masks that segment out the semantic objects in the images, as illustrated in Figure 2.16. It has been a long-standing problem in computer vision with multitudes of algorithms published. In this section, we focus on reviewing the neural-network-based algorithms, as the performance of these methods are significantly better than the traditional approaches. Depending on the level of supervision, semantic segmentation algorithms can be categorized as fully supervised, weakly supervised and unsupervised methods. We discuss the fully supervised and weakly supervised methods here, as our semantic segmentation method, which is presented in Chapter 4, can be viewed as a weakly supervised approach that borrows some techniques from fully supervised methods.

Figure 2.16: Sample results for semantic segmentation. These images are from the PASCAL VOC segmentation set [22]. In the segmentation masks, each color represents one class of object.

2.3.1 Fully Supervised Semantic Segmentation

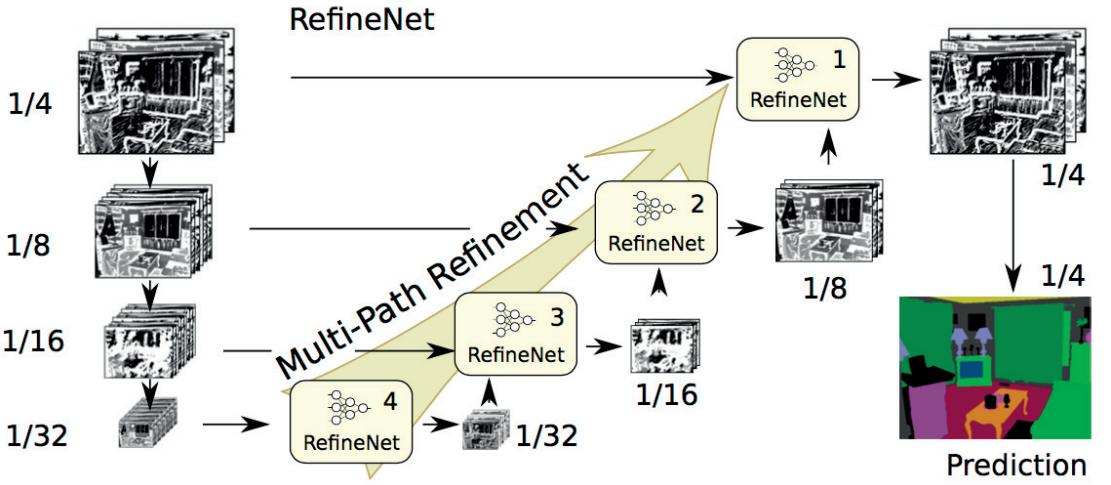
Before reviewing the algorithms, we first introduce the datasets used for this task. The most widely used dataset is the PASCAL VOC 2012 segmentation set [22]. There are a total of 9,993 images covering 20 classes: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, potted plant, sheep,

Figure 2.17: Sample images and annotations from Microsoft COCO dataset [99].

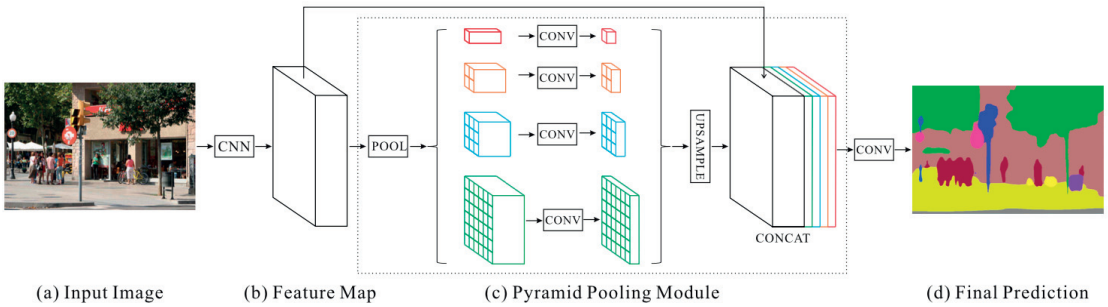
sofa, train, and tv/monitor. The authors in [100] further augment this dataset with extra annotations, resulting in an augmented training set of 10,582 images, a validation set of 1,449 images and a reserved test set of 1,456 images. Sample images with the corresponding segmentation masks are shown in Figure 2.16. Another useful set for training segmentation networks is the Microsoft COCO dataset [99] that covers 91 classes with 65k images. Figure 2.17 shows sample images and annotations from the Microsoft COCO dataset. The segmentation performance is often measured by the IoU (intersection over union) metric, *i.e.* the size of the intersection between the generated and the groundtruth segmentation masks divided by the union of the two. Therefore, IoU is a percentage value.

The breakthrough of semantic segmentation comes from the fully convolutional network (FCN), as discussed in Sec. 2.1.2. Following the design of FCN, later algorithms propose several techniques to improve the segmentation performance. Dense Conditional Random

Fields (CRFs) are adopted in [20] to refine the boundaries in the final segmentation output. To approximate the optimization of CRFs, the authors in [19] design a recurrent neural network that enables end-to-end training of FCN and CRFs together. Lin *et al.* [101] propose a pyramid structure with a RefineNet at each scale, as shown in Figure 2.18a. The RefineNet fuses the high-resolution features with the features from the low-resolution level, contributing to better details in the segmentation masks. Figure 2.18b illustrates the PSPNet proposed in [21]. This algorithm incorporates dilated convolution [102] into a Resnet architecture, followed by a pyramid pooling module that aggregates the context of different scales. Although achieving superior segmentation performance, fully supervised methods all require pixel-wise human annotations for training.



(a) RefineNet



(b) PSPNet

Figure 2.18: The architectures of RefineNet [101] and PSPNet [21].

2.3.2 Weakly Supervised Semantic Segmentation

Annotating images with pixel-wise labels is both time-consuming and expensive to obtain. As reported in [103], the average annotation time was 239.7 seconds per image in the PASCAL VOC 2012 dataset. Therefore, a considerable amount of research efforts

have shifted to weakly supervised algorithms, in which researchers explore other types of annotations that are easier and faster to collect, such as tags, points, scribbles or bounding boxes.

As no pixel-wise masks are available, the challenge for weakly supervised algorithms is to infer the location of each class from weak annotations. Bearman *et al.* [103] build a novel loss function that measures loss at the point level. They further incorporate objectness [104] into the loss to help stretch keypoints to the whole contour of objects. In [105], the authors combine scribbles and the segmentation network's output in a CRFs-type optimization problem, where they iteratively optimize the CRFs and the segmentation network. The authors of [106] replace the CRFs in [105] by a random-walk [107] label propagation algorithm, which enables learning the label propagator, jointly with the segmentation network. Some other work [108, 109] use bounding boxes as supervision. Grabcut algorithm [110] is employed to extract segments inside the boundary boxes.

Among all types of weak supervisions, an image tag is the easiest type of supervision to collect, but the most difficult one to use for supervising segmentation algorithms. This is because image tags reveal little information about the location and shape of the objects. Early works [111, 112, 113] extend the multiple-instance learning [114] framework for weakly supervised semantic segmentation. The loss functions are directly built on the image-tag level. Different approaches, *e.g.*, maximum pool [112] or log-sum-exp [111], are explored in order to pool pixel-level probability predictions into tag-level losses. No object-location information is considered in these frameworks, thus resulting in coarse segmentation masks. Recent methods [115, 116, 117, 118, 119, 120, 121] build up location priors by applying traditional computer vision algorithms or by investigating the pre-trained networks. [115] and [116] both build their location cues by backpropagating through the pre-trained classification network [4]. The difference lies in that [115] builds general objectness measure for all classes, whereas [116] focuses on class-specific saliency maps. Similar location priors are used in [117] in the form of the seed loss. They further integrate the seed loss with another two losses that encode more location information. The bottom-up segment proposals, such as (MCG) [122] and BING [123], are used in [118, 119] as another approach to obtain location information. [120] extracts rough estimations of object locations from a classification network. These estimations are then combined with saliency maps to find the contours of the whole objects. A novel technique is proposed in [121] where they first crawl videos according to the image tags, and then extract the common patterns across frames. The underlying assumption is that videos crawled according to a keyword should contain the corresponding object in a majority of frames.

We report the performance and the annotation types of different algorithms in Table 2.1. Although they do not require human interactions during testing, algorithms such as MCG [122] and BING [123] still use human annotations for training. In this sense, methods using these algorithms indirectly rely on additional supervision. In general,

algorithms with stronger supervision, such as scribbles or bounding boxes, perform better than algorithms that use weaker supervision, *e.g.*, only tags. However, tag-based algorithms have better extensibility to more classes, thus are more usable in real-world applications.

Table 2.1: Comparison of weakly supervised semantic segmentation algorithms on the PASCAL VOC 2012 validation and test set. The performance is measured as mIoU value.

Methods	*val*	*test*	Supervision
PointSup [103]	46.1	-	points
ScribbleSup [105]	63.1	-	scribbles
RAWKS [106]	61.4	-	scribbles
BoxSup [108]	62.0	64.6	boxes
SimpleSeg [109]	65.7	67.5	boxes
MIL-bb [111]	37.8	37.0	tags + BING [123]
MIL-seg [111]	42.0	40.6	tags + MCG [122]
MIL-FCN [112]	25.66	-	tags
DCSM [116]	44.1	45.1	tags
SEC [117]	50.7	51.7	tags
SN_B [119]	41.9	43.2	tags + MCG
CCCN-size [113]	42.4	45.1	tags + size estimation
CheckMask+CRF [115]	51.5	52.9	tags + mask selection
AFSeg [118]	54.34	55.5	tags + MCG
SSSeg [120]	55.7	56.7	tags + saliency
Webvideo [121]	58.1	58.7	tags + videos

2.4 Image Enhancement

This section presents an overview of different image enhancement methods that are categorized into five groups: (1) rule-based methods, (2) methods based on sample images, (3) interactive methods, (4) keyword-based methods, and (5) methods using neural networks.

2.4.1 Rule-Based Methods

This group of algorithms pre-define a set of rules or operations according to expert knowledge. For example, Hummel *et al.* [124] define a histogram equalization process that balances the color histogram of an image. This operation can improve the contrast of images. An image sharpening operation is defined in [125], where the high-frequency components of the input image is extracted and added back in order to enhance the details. [126, 127] define several harmonization rules to adjust the colors of the input image. [128, 129] modify the colors of an image according to predefined color palettes or color themes, as shown in Figure 2.19. In [130], the authors propose to detect objects

such as human eyes and sky, and associate a heuristic enhancement function with each object. Because these enhancement operations are manually designed, methods in this category lack the flexibility to adapt to different types of images.

Figure 2.19: Sample color re-rendering results from [129]. The bar at the bottom of each image indicates the color palette used for this image.

2.4.2 Methods Based on Sample Images

These algorithms require at least one sample image in order to supervise the enhancement process. [131, 132] propose to globally transfer the color of the sample image to the target image. [133, 134] extend the idea to local adjustment. Wang *et al.* [135] learn implicit color and tone adjustment rules from example images and apply those on the input image. Kang *et al.* [136] ask users to create personal example images, from which the parameters of the enhancement operation are learned. Figure 2.20 shows three different enhancement results created by three users.

(a) Input image (b) User #1 (c) User #2 (d) User #3

Figure 2.20: Results of the personal enhancement method [136]. Three users created different sample images that lead to different enhancement results.

2.4.3 Interactive Enhancement Methods

Methods [13, 14, 15, 16] in this category rely on the interactive inputs from users in order to direct the enhancement process. In [13], users are asked to indicate the regions that need to be enhanced. Laput *et al.* [14] combine two types of interaction together. They use speech recognition to associate tags with image regions and use sliders to enable users to adjust the colors. [15, 16] use stroke input from users to indicate the foreground and background regions, and to apply different processes accordingly, as shown in Figure 2.21.

Figure 2.21: Sample results from the interactive image enhancement method [16]. Users draw strokes in different color to indicate the foreground and background, and to assign the colors to the corresponding parts.

2.4.4 Keyword-Based Methods

Compared to the interactive methods, keyword-based approaches require much less human intervention and are more flexible. Wang *et al.* [137] first associate a color theme to each emotion keyword and apply color adjustment accordingly. Lindner *et al.* [17] generalize to more keywords by statistically analyzing the correlations between keywords and color characteristics, as shown in Figure 2.22. We will discuss the details of [17] in Chapter 5, as our keyword-based image enhancement algorithm is closely related to [17].

(a) Input image (b) Result

Figure 2.22: Sample result of the keyword-based image enhancement method [17]. The image is enhanced with keyword '*sky*'.

2.4.5 Neural Networks Based Methods

Since the breakthrough of AlexNet [3], neural networks have been applied to many tasks, including image enhancement. Xie *et al.* [138] propose an autoencoder style network for image denoising. [49, 139] adopt CNNs and GANs for single image super-resolution. Yan *et al.* [140] propose an automatic color adjustment operation, which is defined as a quadratic mapping function. The multi-layer neural network is used to compute the coefficients of the mapping function. They still use heuristic image features as input to the network. We show the example results of [140] in Figure 2.23. Compared to this method that still heuristically defines the enhancement functions, our GANs-based enhancement algorithm automatically learns the enhancement operations from the training data, as presented in Chapter 6.

Figure 2.23: Image enhancement result from Yan *et al.* [140]. The first row are the input images and the second row are the enhanced results.

3 Computational Aesthetics

In this chapter, we first present an aesthetics assessment algorithm based on neural networks. We also present the automatic cropping technique in Section 3.4, as it is a direct application of the aesthetics networks.

3.1 Introduction

Automatically assessing image aesthetics is useful for many applications. To name a few, aesthetics can be adopted as one of the ranking criteria for image retrieval systems or one of the objectives for image enhancement systems. Users can also manage their image collections based on aesthetics. Various algorithms [12, 77, 79, 81, 82, 83, 84, 85, 98, 141] have been proposed over the past few years for aesthetics assessment.

In this thesis, we train convolutional neural networks (CNNs) for aesthetics assessment. Our model is trained on the AVA dataset [12] that contains more than 250,000 images, as introduced in Chapter 2. Each image has around 200 user ratings about its aesthetics quality, with each rating being an integer between 1 and 10 (1 implies the lowest quality and 10 means the highest quality). We show two sample images and their corresponding histograms of user ratings in Figure 3.1. The average of user ratings is taken as the aesthetics score for each image. It represents the perceived aesthetics quality of that image.

The distribution of aesthetics scores in the AVA dataset is extremely unbalanced, as shown in Figure 3.3 (a), which introduces bias into all the previous CNN models that are trained on this dataset [98, 141]. As there are more images with medium aesthetics quality in the training set and few images with very high or low quality, the range of the predicted scores by these models are thus limited and biased towards the middle values. To reduce such bias, we propose to use sample weights during training. The sample weights are first computed according to the occurrences of the aesthetics scores and later incorporated into a weighted loss function for training. This loss function is balanced

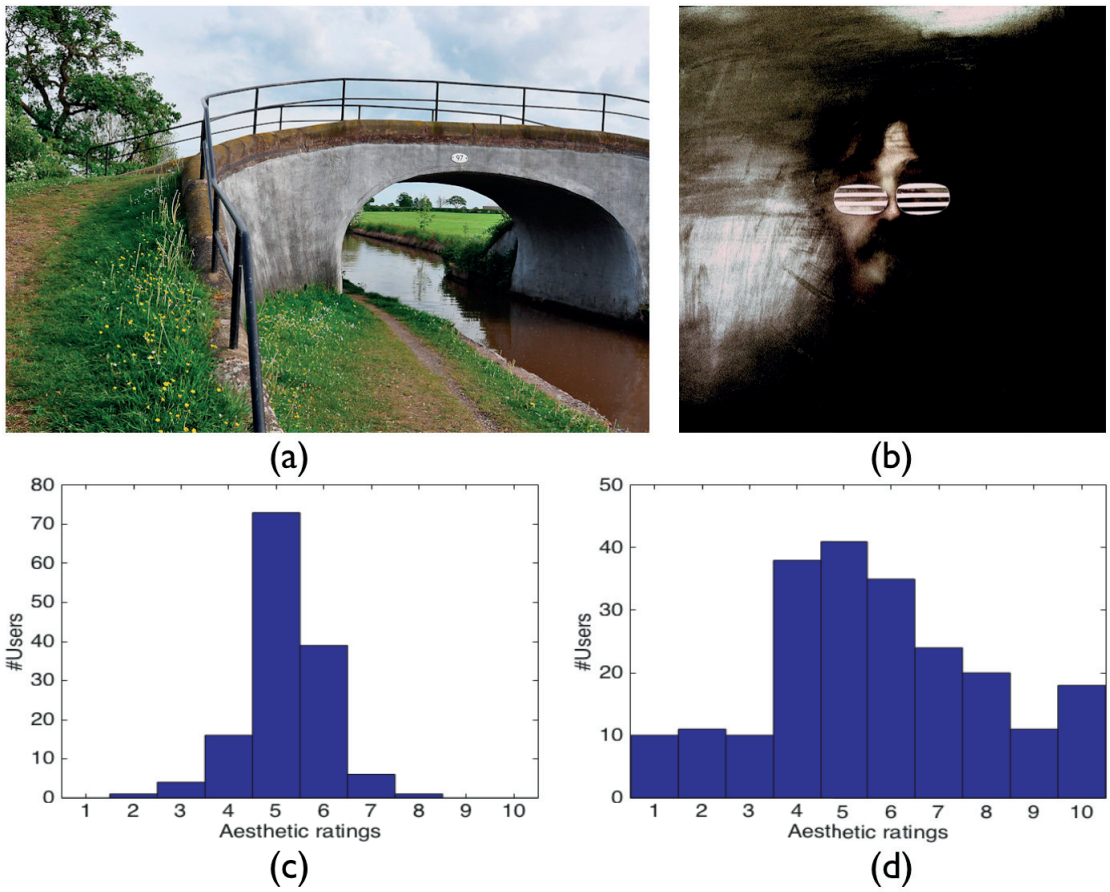


Figure 3.1: (a) and (b) are two images of the AVA dataset [12], (c) and (d) are their corresponding histograms of user ratings. In image (a), most of the user ratings fall within the middle aesthetics level (scores of 4, 5, and 6) while the ratings of image (b) is distributed over the whole range.

over images with different aesthetics scores, thus enabling the trained CNNs to work for images of different aesthetics quality. Using sample weights, we train a regression model that can achieve a larger prediction range and better accuracy than previous methods.

Additionaly, all previous methods [12, 82, 98, 141] directly use the average aesthetics scores for training while discarding the distribution of user ratings. As pointed in [142], the Mean Opinion Score (MOS) is only one statistic to describe the distribution, which implicitly assumes homogeneity among the annotators. In [143], the authors show that predicting the histogram of user ratings instead of the MOS can better model user preferences. As a matter of fact, the distribution of the ratings reveals not only the average aesthetics score, but also how much users agree with each other when aesthetically assessing the image. Therefore, the distribution is an indicator of the difficulty for performing an aesthetics assessment of a given image. For instance, the two histograms in Figure 3.1 clearly indicate that Figure 3.1(a) is agreed by the majority of users to be

of average quality, thus being easy to judge, whereas Figure 3.1(b) is less conclusive and more difficult to assess. To estimate the level of difficulty, we train a histogram prediction CNN model that can predict the normalized histogram of user ratings. Our experiments show that this model reliably estimate the histograms of user ratings.

Once we obtain CNN models to automatically assess the aesthetics quality of images, these models can then be directly used for many applications. In Section 3.4, we present an automatic cropping application that uses the trained aesthetics network. The cropping system can automatically generate a visually pleasing crop from the input image.

To summarize, our **contributions** in this chapter are:

- We use sample weights during training, which helps to overcome the bias in the training set of the AVA dataset and extend the prediction capability of the trained CNN models.
- We train a regression CNN model that achieves a larger prediction range and better accuracy than the state-of-the-art methods.
- We train a histogram prediction model that reliably estimates the aesthetics scores, as well as the difficulty of aesthetics assessment.
- We build an image cropping application that outputs an aesthetically pleasing crop of an input image by using the results of the trained CNN model.

3.2 Methods

In this section we first explain how we derive the sample weights for the training set, followed with two CNN models that we propose in order to predict aesthetics. We explain the regression model in Section 3.2.2 and the histogram prediction model in Section 3.2.3.

3.2.1 Sample Weights

Assume the histogram of the aesthetics scores in the training set is $\{b_i, i = 1, 2...B\}$. B is the number of bins that evenly cover the range of the aesthetics scores. We set B to 90 for the aesthetics score range of 1 to 10. b_i is the occurrence number of the ith bin, namely the number of images assigned with the aesthetics scores within the ith bin's range. The histogram for the training set is shown in Figure 3.3(b).

It is obvious that the distribution of aesthetics levels in the training set is extremely unbalanced. The medium quality images significantly outnumber the others. When forming the mini-batches to train neural networks, the medium quality images appear more frequently in each batch, contributing to the major part of the loss function.

Minimizing such loss function causes the predicted aesthetics scores center around the medium value. Therefore, the networks trained on this dataset tend to produce large errors for the very high quality or low quality images. However, the primary focus for many real-world applications, such as image ranking or image filtering, is to find these edge-case images accurately.

In this chapter, we propose to use weighted samples to tackle this problem. As the source of this problem is the uneven contribution to the loss function, we propose to add a weight to each sample to re-balance their contributions. The design of the weight function reveals the occurrence numbers of each aesthetics level, specifically assigning larger weights to the less frequent classes and vise versa. The sample weight w_i for the ith bin in the histogram is computed as:

$$b'_i = \frac{b_i}{\sum_{i=1}^{B} b_i}; \qquad w_i = \frac{1}{b'_i} \tag{3.1}$$

Images within the same bin share the same sample weights. The sample weight is inversely proportional to the normalized occurrence number. Consequently, images with rare scores are assigned larger sample weights than images with more frequent scores. Note that sample weights are only computed for the training set and only used during training, not during testing.

3.2.2 Regression Model

The architecture of our regression CNN model is the same as the VGG16 network [4], as this network architecture has shown superior performance on image classification. This network contains 13 convolutions layers, 5 max pooling layers followed by three fully connected layers. The last fully connected layer of the network is modified to have only one output neuron for predicting a single aesthetics score. We remove the last *softmax* activation function as the output is only one value. The network architecture is shown in Figure 3.2.

The training of this model is done by minimizing the following Weighted Mean Squared Error (WMSE) loss function:

$$WMSE = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \cdot (y_i - \hat{y}_i)^2 \tag{3.2}$$

Here w_i is the sample weight computed according to Equation 3.1. y_i is the predicted aesthetics score and $\hat{y}_i$ is the groundtruth aesthetics score. N is the number of images in the training set.

Note that images with large sample weights do not occur very often. As a result, the

overall contribution to the loss function is balanced across images with varying aesthetics scores. In this way, the sample weights help to reduce the bias in the training set.

3.2.3 Histogram Prediction Model

The histogram prediction model predicts the normalized histogram of user ratings for an input image. The output of the model is a vector with 10 bins, as user ratings are integers between 1 to 10. We adjust the last layer of VGG16 network [4] to have 10 output neurons, followed by a normalization step to output a normalized histogram. The structure is similar to that in Figure 3.2, except the last output layer. The loss function for training is the Weighted Mean χ^2 Error (WMCE):

$$WMCE = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \cdot \chi^2(\mathbf{h_i}, \hat{\mathbf{h}}_\mathbf{i}) \tag{3.3}$$

where w_i is the sample weight for image i. $\mathbf{h_i}$ is the output histogram from the network and $\hat{\mathbf{h}}_\mathbf{i}$ is the groundtruth normalized histogram. χ^2 represents the chi-square distance computed as in Equation 3.4.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3.4}$$

Based on the output histogram, two values are derived: the aesthetics score, which is the mean of user ratings, and the standard deviation (std) of user ratings. This std value represents the difficulty of aesthetics assessment. A small std means the consensus and simplicity of aesthetics assessment as user ratings concentrate around the average score. One image with a high std value indicates that users hold various opinions about its aesthetics quality, thus being more difficult for our model to predict the aesthetics scores. By comparing the std values, we can evaluate whether one image is more difficult to aesthetically assess than another.

3.3 Experiments

3.3.1 Training and Test Sets

We split the AVA dataset into three parts: training set, test set 1 (*RS-test*) and test set 2 (*ED-test*). The distributions of the aesthetics scores in these three sets are shown in Figure 3.3(b)-(d). *RS-test* contains 3000 *Random Sampled* images, which is similar to the test set in [98] that contains 5000 random sampled images. *ED-test* is built to have 3000 images *Evenly Distributed* among three categories: the low quality images (aesthetics

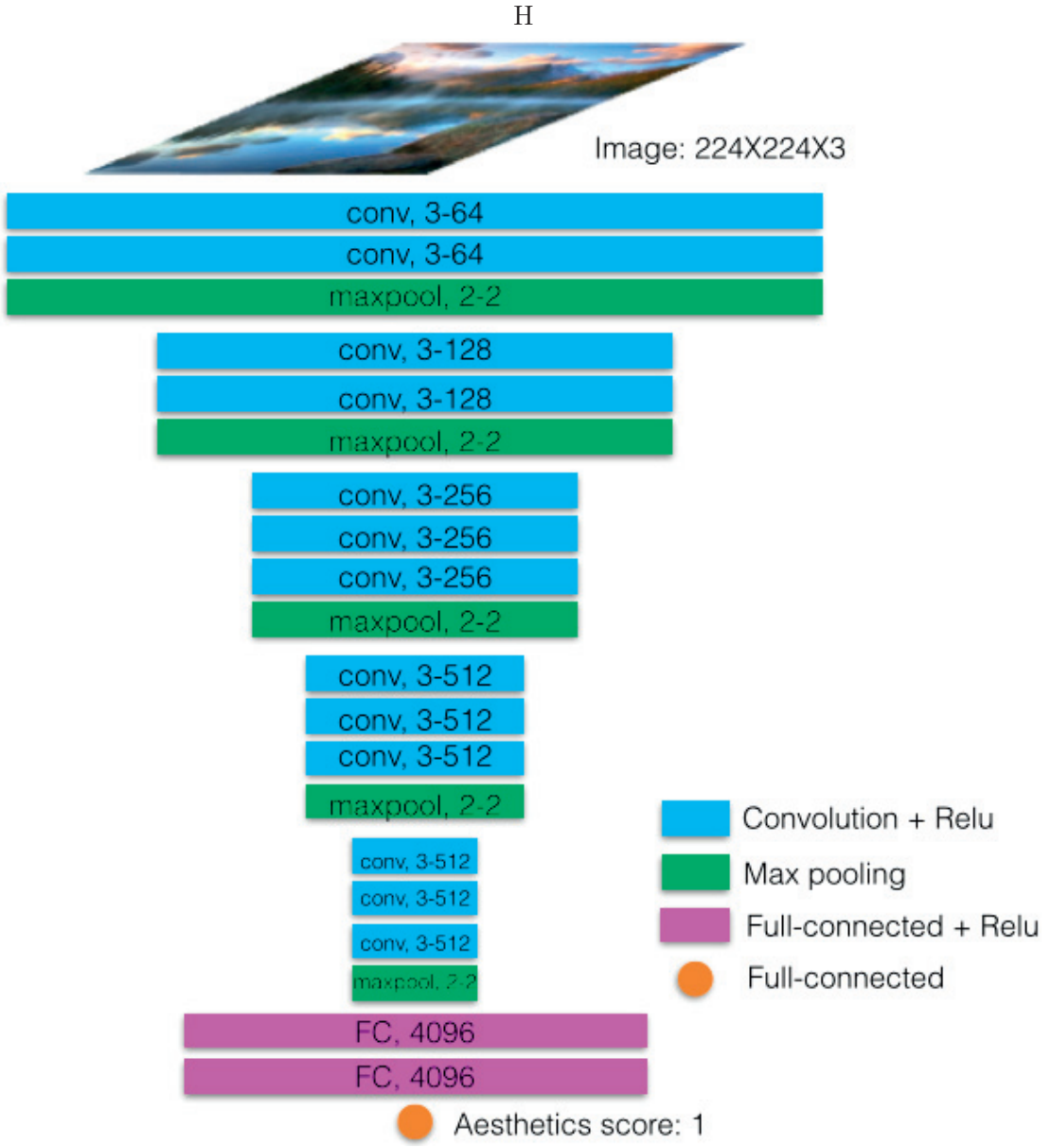


Figure 3.2: Architecture of our regression model for predicting average aesthetics scores. The network takes color images of 224 by 224 as input and outputs a single aesthetics score.

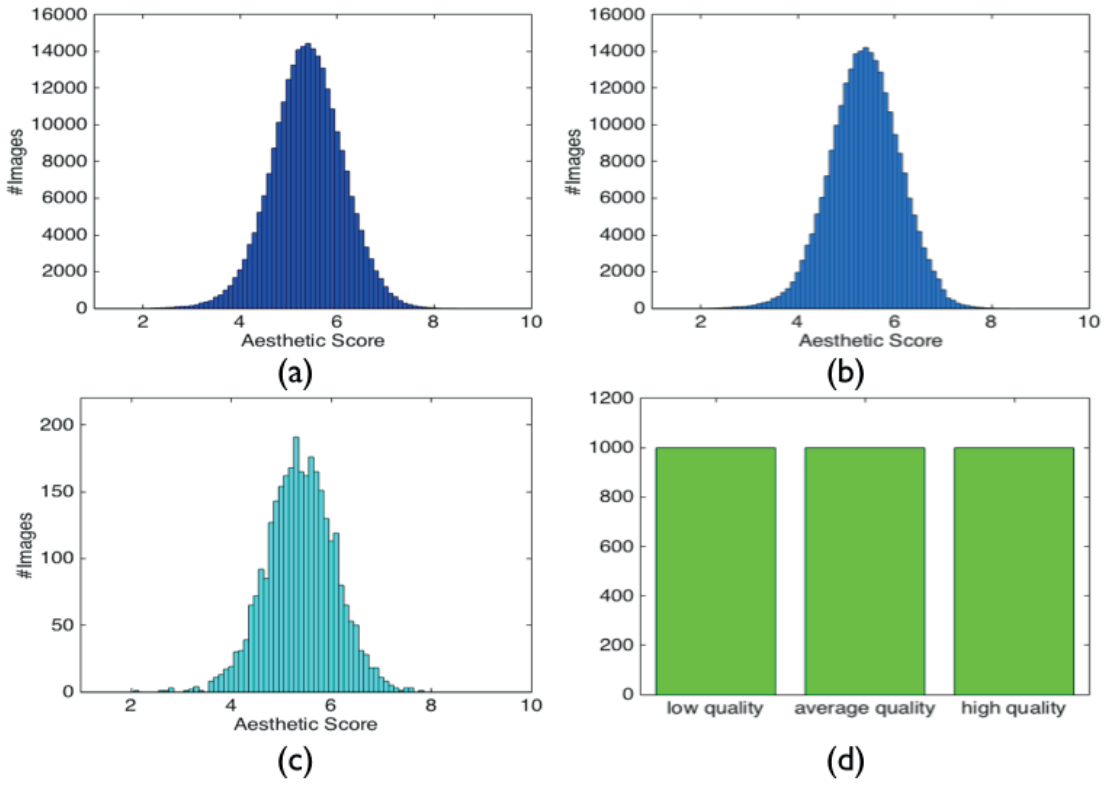


Figure 3.3: The distribution of the average aesthetics scores for (a) the whole AVA dataset (b) the training set, (c) the *Random Sampled* test set (RS-test), (d) the *Evenly Distributed* test set, which has an equal number of images from three categories: low, average, and high quality.

score < 4), the average quality images ($4 \leq$ aesthetics score ≤ 7) and the high quality images (aesthetics score > 7), as shown in Figure 3.3 (d). The other 249530 images of the AVA dataset are used for the training set.

3.3.2 Pre-process

As many aspects of an image can affect its aesthetics, such as composition and saturation, it is not recommended to apply data augmentation methods. During training, we only apply random horizontal flipping, which is believed to not significantly alter the perceived aesthetics. Since VGG16 network takes input images of 224 by 224, we directly resize the whole image to 224×224, which is then fed into the network. Although this operation might change the aspect ratio of the image, we have experimentally found that it produces the best results as opposed to cropping the images, which is corroborated in [141]. Both models are initialized with the pre-trained ImageNet weights [3], except the last layer which is randomly initialized. The CNNs are fine-tuned for 20 epochs on the whole training set. The learning rate is set to 0.00001, and divided by 10 when the training loss

stops decreasing. The weight decay is set as 0.0005. The algorithm is implemented using the *theano* package, and it took around 4 days for each model to finish 20 epochs on a single NVIDIA TITAN X GPU.

3.3.3 Regression Model Results

For the regression task, we use the Mean Squared Error (MSE) as the evaluation metric, which is the same as in [98]:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \tag{3.5}$$

Here, y_i and $\hat{y}_i$ are the predicted and the groundtruth aesthetics scores, respectively, for the ith image. M is the number of images in the test set. Note that sample weights are not applied in the evaluation metric.

Two regression CNN models with the same architecture are trained: a *Regression* model with *Sample Weights* (*SWR*) and a *Regression* model with *No Sample Weights* (*NSWR*). For the model without sample weights, we use the same network architecture and set the sample weights of all images to 1. The performance is shown in Table 3.1.

Table 3.1: MSE of different regression models for aesthetics prediction on the AVA dataset. Results of the top 5 methods are taken from [98].

	RS-test	*ED-test*
GIST linear-SVR	0.5222	NA
GIST rbf-SVR	0.5307	NA
BoVW SIFT linear-SVR	0.5401	NA
BoVW SIFT rbf-SVR	0.5513	NA
Kao et al. [98]	0.4510	NA
No SW regression (*NSWR*)	**0.3373**	1.3951
SW regression (*SWR*)	0.4847	**0.9754**

The top four methods in Table 3.1 combine the generic image descriptors, GIST [144], SIFT [86] and Bag-of-Visual-Words (BoVW) [87], together with the Support Vector Regression (SVR) with a linear or rbf kernel [145]. Refer to [85, 98] for the details of these methods. Note that none of the previous methods were evaluated on a test set with balanced distribution, namely the *ED-test* we created.

Our regression model without sample weights (*NSWR*) outperforms all the state-of-the-art methods on the *RS-test*, while the model with sample weights (*SWR*) further outperforms *NSWR* on the *ED-test*, demonstrating the effectiveness of our regression model to predict aesthetics for images of a variety of aesthetics qualities. Note that *SWR* produces larger MSE than *NSWR* and the method in [98] on the *RS-test*. This

is because the *RS-test* and training set have similar unbalanced distributions. Hence, the bias introduced by the training set actually benefits these two models with better performance on the *RS-test*.

However, such bias limits the prediction range of the models. Using sample weights during training enlarges this range, making it more practical for real-world applications. The minimum and maximum values of the aesthetics scores predicted by the *NSWR* model on both test sets combined are 3.54 and 6.46. For the *SWR* model, these two values are 2.06 and 7.53. We further illustrate this effect in Figure 3.4, which shows the mean MSE for different ranges of aesthetics scores. Using sample weights clearly contributes to reducing the MSE for images with aesthetics scores higher than 6 or lower than 4.

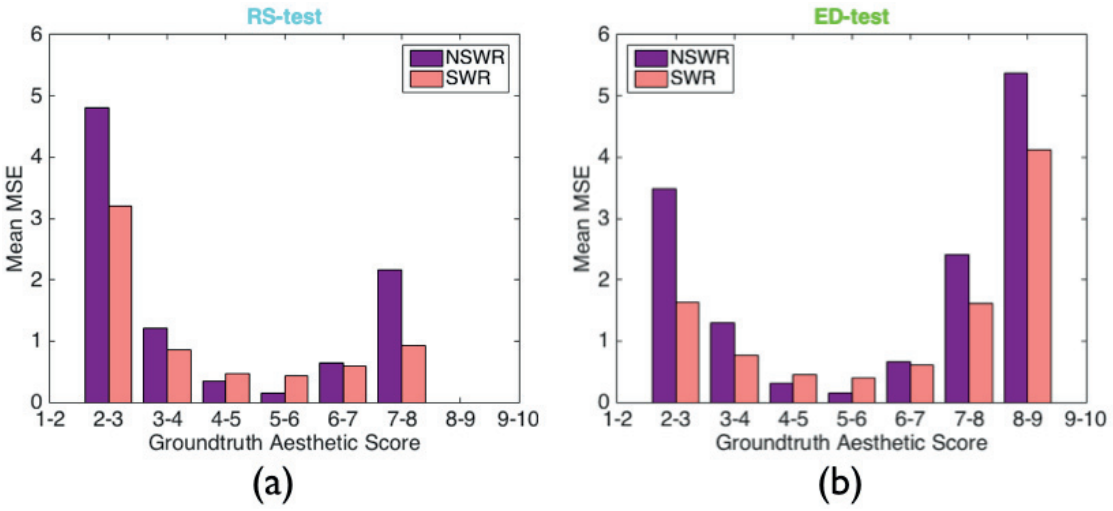


Figure 3.4: Mean MSE for different ranges of aesthetics scores on the (a) *RS-test*, the *Random Sampled* set (b) *ED-test*, the *Evenly Distributed* set where images are evenly distributed into three categories: low, average, and high quality. Note that there are no bars in the range of 1-2, 8-9, 9-10 in (a) and 1-2, 9-10 in (b). This is because the lack of image in those ranges in the two datasets. The number of images with these extreme scores is quite small in the whole AVA dataset.

Some algorithms model aesthetics estimation as a classification task. They predict one image as either high quality or low quality. For further evaluation, we also map our regression results to the classification task. The groundtruth labels for all test images are set as either high quality or low quality, by comparing the aesthetics scores with two thresholds, $5 - \delta/2$, $5 + \delta/2$, and discarding the images in between. Through increasing δ, we eliminate the ambiguous images and increase the gap between the two classes, thus simplifying the classification task. The predicted labels are created by thresholding at the score of 5 [98]. We show the accuracy in Table 3.2. The *NSWR* model achieves state-of-the-art accuracy on *RS-test* whereas the *SWR* model notably outperforms *NSWR* model on *ED-test*, again demonstrating the effectiveness of the sample weights for overcoming the bias of the training set.

Table 3.2: Accuracy (%) of different aesthetics prediction models for the classification task. δ indicates the aesthetics scores margin between the low quality class and the high quality class. The larger δ is, the easier the classification is.

	δ	0	0.1	0.5	1.0	1.5	2.0
RS-test	Lu et al. [141]	74.46	NA	NA	NA	NA	NA
	Kao et al. [98].	71.42	72.26	76.92	82.21	85.49	89.31
	NSWR	75.73	76.91	82.20	86.84	90.82	94.12
	SWR	72.4	73.22	77.66	82.28	85.41	89.87
ED-test	*NSWR*	82.73	83.18	85.11	86.62	87.21	86.95
	SWR	83.46	83.95	86.09	88.24	89.62	89.96

Figure 3.5 and Figure 3.6 show qualitative results of the *SWR* model. Our *SWR* model makes an accurate assessment for images in a wide range of aesthetics quality. In the last row of Figure 3.5 are typical failure cases where large gaps exist between the predicted scores and the groundtruth scores. Such failure cases happen mainly on images of very high quality or very low quality. This is because of the lack of such samples in the training set. Applying large sample weights on high quality or low quality images is effectively similar to duplicating these images multiple times in the training set. Therefore, the lack of variety in these images still results in less accuracy for these images. As shown in Figure 3.4, the mean errors for the two edge cases are larger than those in the middle bins.

3.3.4 Histogram Prediction Model Results

Two values can be extracted from the output of the histogram prediction model, the mean aesthetics score and the standard deviation (std) of the predicted user ratings. MSE in Equation 3.5 is used to evaluate the aesthetics score and the Root Mean Square Error Ratio (RMSER) is used for evaluating the standard deviation:

$$RMSER = \frac{\sqrt{\frac{1}{M}\sum_{i=1}^{M}(std_i - \hat{std}_i)^2}}{\frac{1}{M}\sum_{i=1}^{M}\hat{std}_i} \tag{3.6}$$

where std_i is the std of the predicted user ratings for image i and $\hat{std}_i$ is the std of the groundtruth histogram.

We train a *Histogram* prediction model with *Sample Weights* (*SWH*) and another model without sample weights (*NSWH*). Table 3.3 shows the results. The comparison between *SWH* and *NSWH* shows similar trends to that of the regression models. *SWH* achieves comparable performance to *SWR* for predicting the aesthetics scores on the *ED-test*, and produces less than 20% RMSER. Hence, the difficulty of aesthetics assessment for an image is also reliably estimated.

We use the Weighted Mean χ^2 Error as the loss function to train the histogram prediction model. There are other functions that can measure the distance between two distributions, such as the Eculidean distance, the cross-entropy, the Kullback-Leibler divergence [146] and the Jensen-Shannon divergence [147]. Although each has a different definition of distance, these functions all try to push the predicted distribution to be similar to the groundtruth one. Please refer to a recent work [148] for a complete analysis of different loss functions for aesthetics prediction problem.

Table 3.3: MSE and RMSER for the histogram prediction models *SWH* and *NSWH*.

	NSWH		*SWH*	
	MSE	RMSER	MSE	RMSER
RS-test	0.3730	16.62%	0.6358	26.75%
ED-test	1.6568	28.73%	1.0109	19.57%

Figure 3.5: Qualitative results of the aesthetics regression model *SWR* (using sample weights). The numbers in blue are the predicted average aesthetics scores, and the red numbers are the groundtruth scores.

Figure 3.6: Qualitative results of the aesthetics regression model *SWR* (using sample weights). The numbers in blue are the predicted average aesthetics scores, and the red numbers are the groundtruth scores.

Figure 3.7 and Figure 3.8 show samples results from the *SWH* model. We can see that the predicted histograms and the groundtruth histograms are quite similar, for both high quality images and low quality images. Images in the last row of Figure 3.8 have relatively large std values, indicating that users have less consensus on their aesthetics qualities, hence it is more difficult to determine their aesthetics scores.

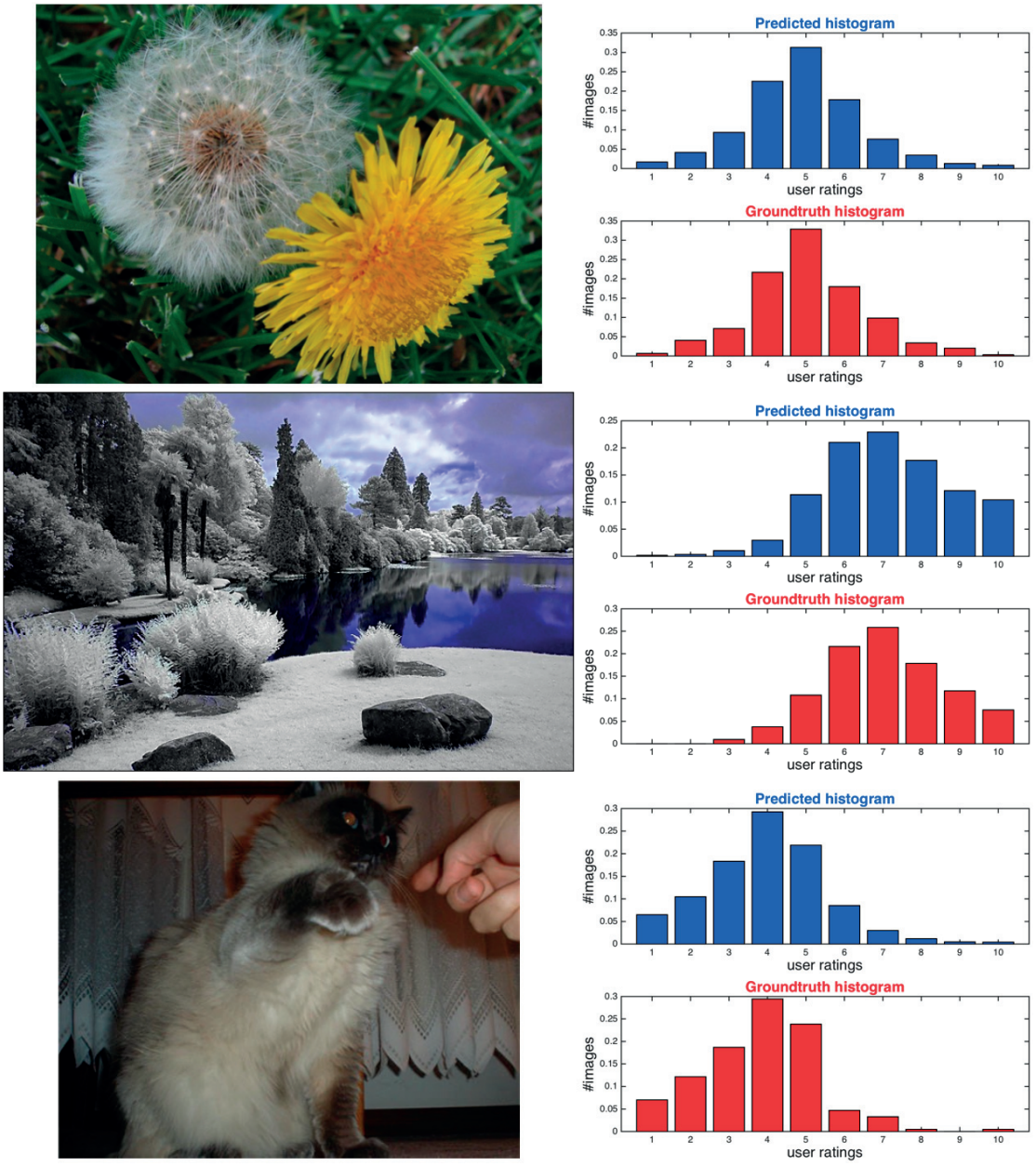


Figure 3.7: Qualitative results of the histogram prediction model *SWH* (using sample weights). On the right side of each image are the prediction histogram of user labels (blue) and the groundtruth histogram of user labels (red). The histograms have ten bins as users are asked to give an integer score between 1 to 10.

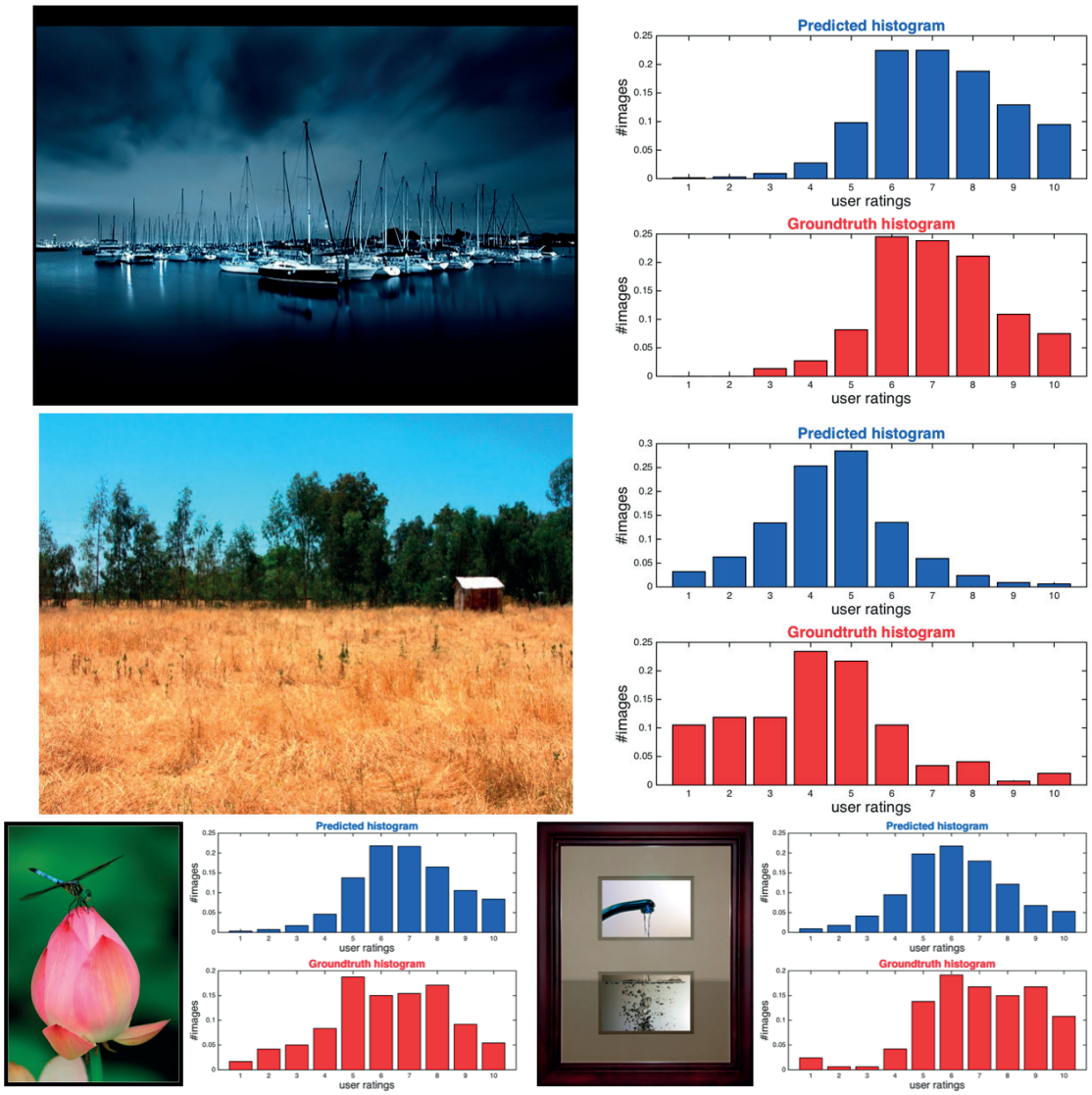


Figure 3.8: Qualitative results of the histogram prediction model *SWH* (using sample weights). On the right side of each image are the prediction histogram of user labels (blue) and the groundtruth histogram of user labels (red). The histograms have ten bins as users are asked to give an integer score between 1 to 10.

3.4 Cropping Application

The aesthetics prediction models can be used in many applications. We present a simple application where our *SWR* model is used to automatically choose an aesthetically pleasing crop from the input image in order to fit into a target window. For many applications, users are required to fit an image into a fixed-sized window, such as creating the profile for Facebook or arranging images in an album. An automatic cropping application is useful to save users effort.

For an input image, we randomly take 1000 fixed-sized crops, with respect to the window template. The selection of crops should respect two aspects so that the crops can encode the major content of the original image. First, the crop should cover at least 40% of the original image. Second, the center of the original image must be included in the crop. These crops are then fed into the *SWR* model for aesthetics assessment. Although all four models proposed in this chapter could accomplish this task, we choose the *SWR* model for its superior performance in predicting the mean aesthetics scores. The one with the highest score is chosen as the output.

Example cropping results are shown in Figure 3.9 and Figure 3.10, where we use a square template in this experiment. Other templates in different shapes are also possible. Our system automatically outputs pleasing square crops from the original images. One typical failure case is the last row in Figure 3.10, where the group of three people are not all included in the generated crop. Our cropping system models mainly image aesthetics while not incorporating the semantic understanding of the images. Consequently, the generated crops are not ensured to respect the semantic completeness of objects and might cut through them. Combining semantic understandings with our aesthetics rules could contribute to a better performance. Another possible solution is to teach the aesthetics engines to avoid half-cut images by including those half-cut images in the training set for aesthetics learning.

To quantitatively prove the effectiveness of this cropping approach, we conduct a psychophysical experiment on 50 images on a crowd-sourcing website[1], where we ask users to compare the crops chosen by our model with random crops. The interface of the crowd-sourcing experiment is shown in Figure 3.11. Among the three crops, one of them is the best crop predicted by our *SWR* model, another one is the worst crop by the model, and the last one is a random crop. On each page we show the original image as well as the three crops and ask users to choose the one they prefer. Note that we randomly adjust the order of the three crops so that no bias is introduced as the progressing of the experiment. In total, 40 users participated in the experiment, with each labeling all 50 images. The results show that for 31 out of 50 images, users prefer the best crops chosen by our model over the other crops.

[1] http://www.shorttask.com/

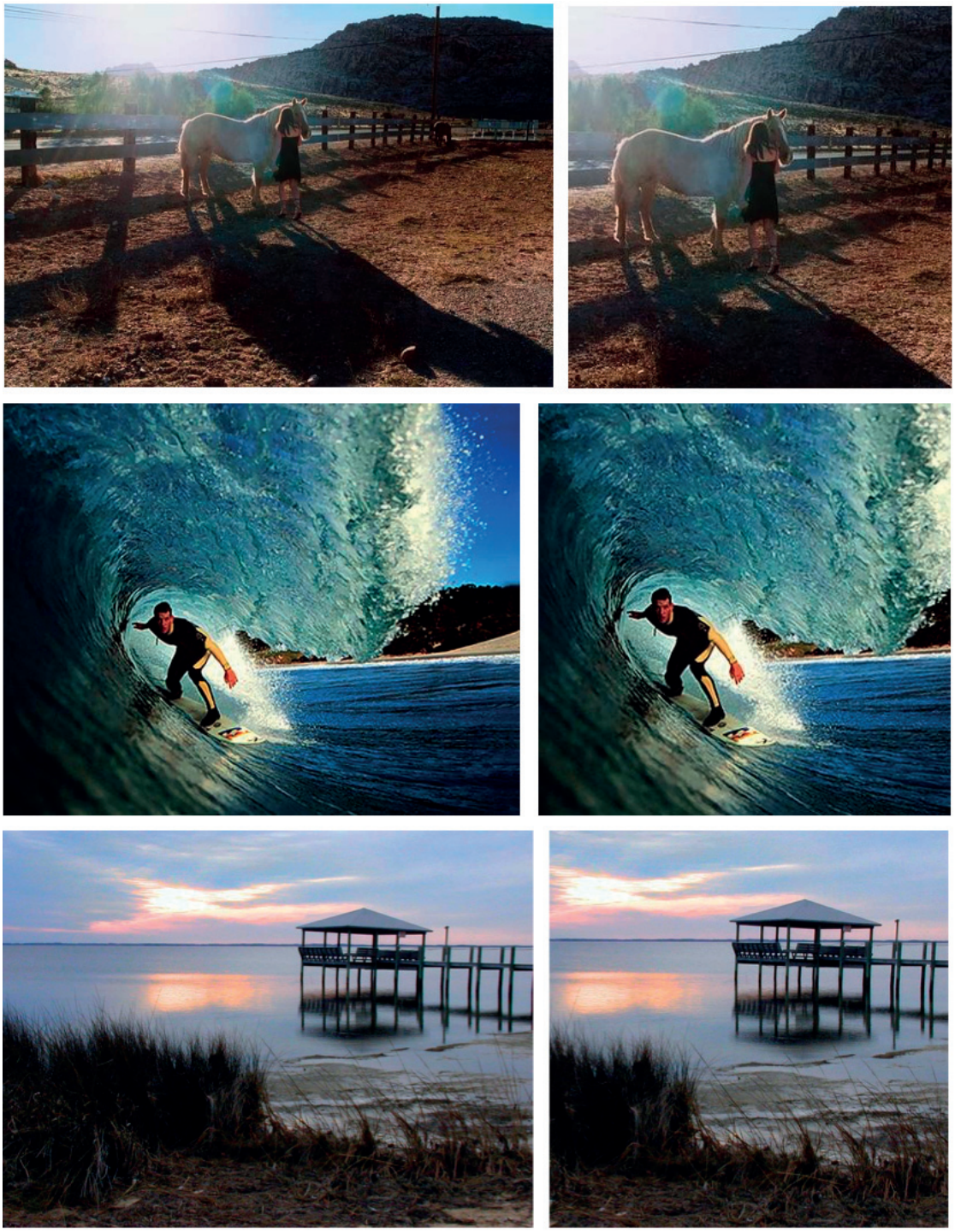


Figure 3.9: Outputs from our automatic cropping system. We assume a square crop for this experiment. In each row, the left image is the original and the right side is the square crop output from our algorithm.

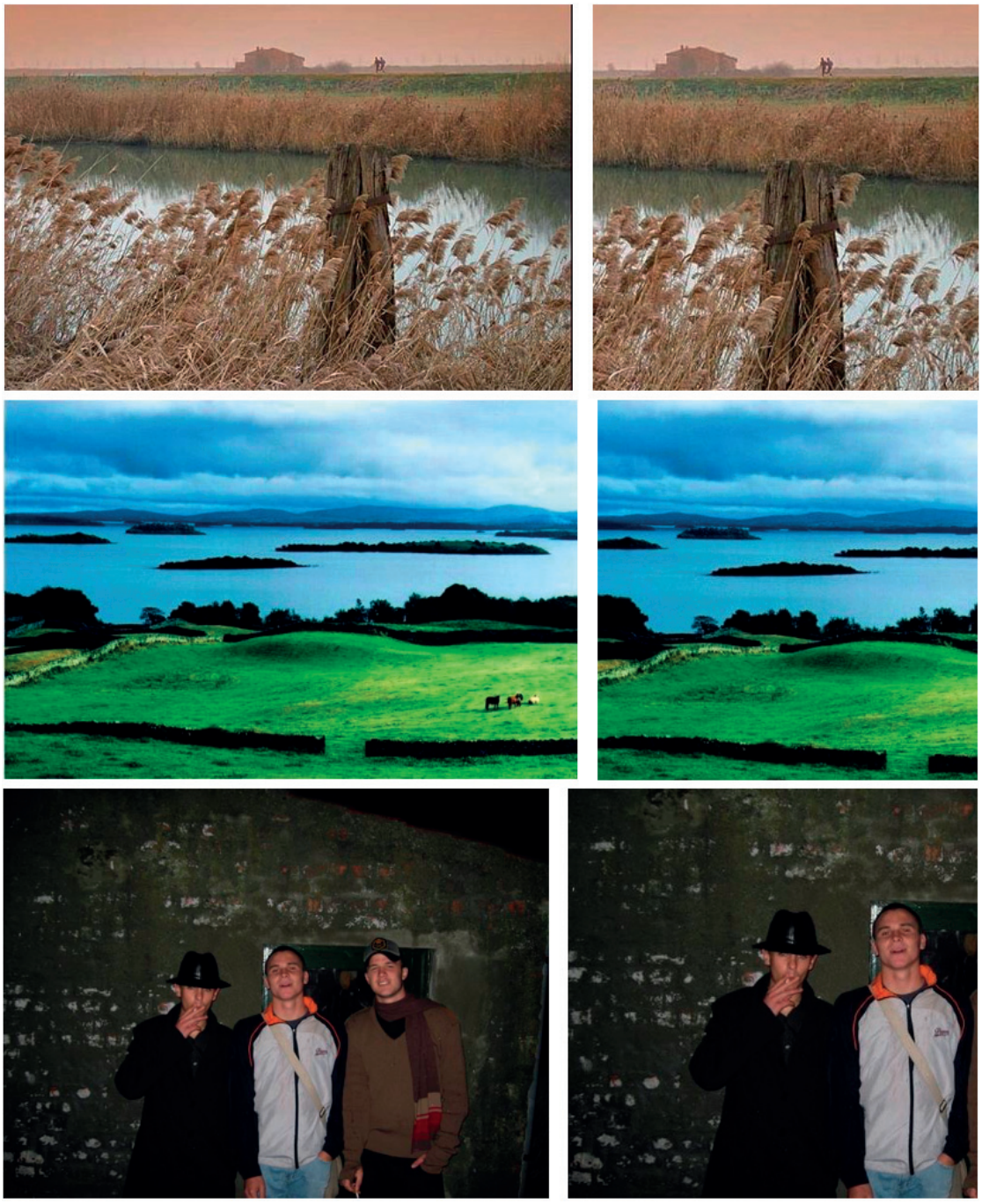


Figure 3.10: Outputs from our automatic cropping system. We assume a square crop for this experiment. In each row, the left image is the original and the right side is the square crop output from our algorithm.

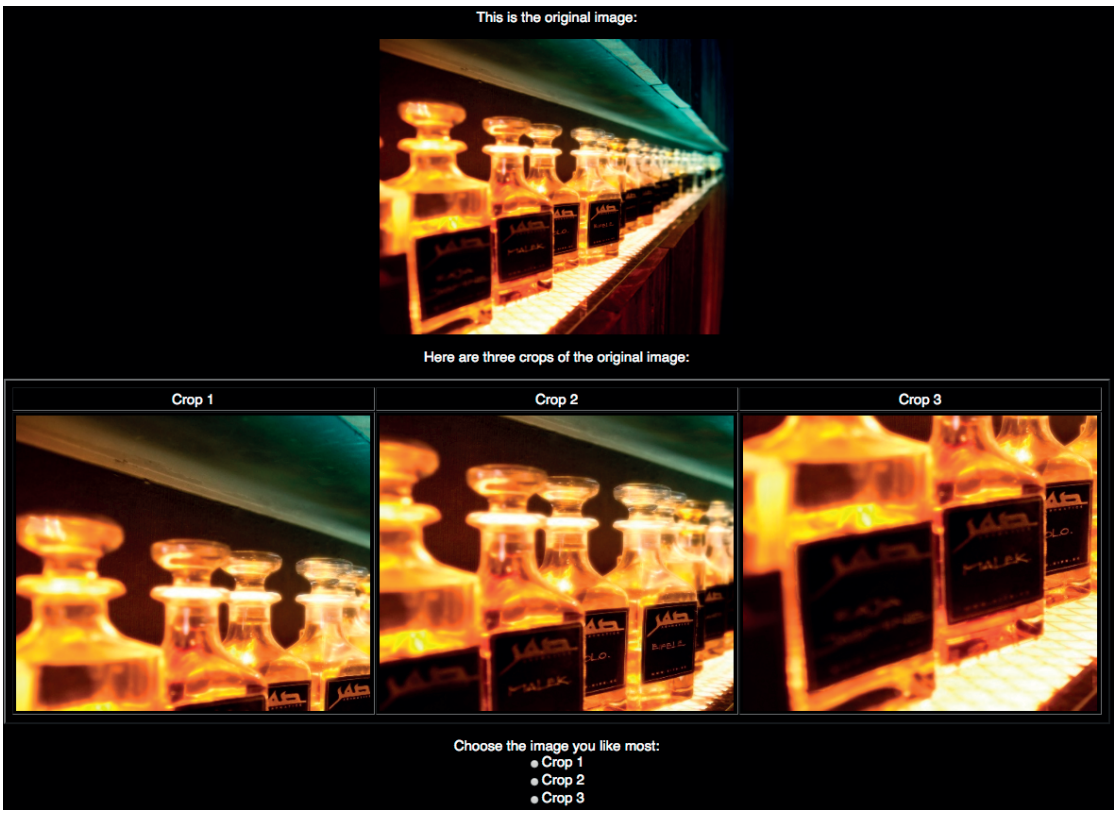


Figure 3.11: Interface for the psychophysical experiment to validate the automatic cropping technique.

3.5 Conclusion

In this chapter, we propose to use sample weights while training CNN models on the AVA dataset for aesthetics assessment. Our experiments demonstrate the effectiveness of the sample weights for reducing the bias in the training set. We train two CNN models with sample weights, a regression model and a histogram prediction model. Our CNN models can output not only accurate aesthetics scores, but also a reliable estimate of the difficulty of the aesthetics assessment. Using the results of our aesthetics prediction model, we further show an image enhancement application that automatically crops the input image for a better aesthetics quality.

Our algorithm focuses on reducing the bias caused by the unbalanced distribution of aesthetics scores, using the sample weights technique. This is essentially just one aspect of training CNNs for aesthetics prediction. Other techniques like multi-column or multi-task training framework [149] can be combined with ours to further improve the aesthetics prediction performance.

Moreover, after carefully analyzing the results of our CNNs, we discover that our networks mainly model low-level image features, such as colorfulness and blurriness, while

missing the semantics, which explains many failure cases in our cropping application, as discussed in Section 3.4. We attribute this to the lack of training samples, especially low quality and high quality images. Training CNNs to understand image tags already requires more than 1 million images [37]. We think image aesthetics is a more complex concept than image tags, as it relates to both low-level image features, such as colorfulness, and high-level image features, such as semantics and composition. Therefore, to fully model image aesthetics, it is necessary to build a much larger and more complete aesthetics dataset.

4 Semantic Segmentation

In this chapter and Chapter 5, we present an image enhancement system that re-renders image colors more appealing. This approach applies local color modifications based on keywords specified by users. We use semantic segmentation in the pipeline to locate the keyword-related regions, where the color modifications are then applied. We first discuss our weakly supervised semantic segmentation algorithm in this chapter, followed by the details of the color-rendering system in Chapter 5.

4.1 Introduction

Semantic segmentation, which refers to accurately assigning semantic labels to the corresponding pixels in an image, is a challenging task actively studied in computer vision. Recent breakthroughs [18, 19, 150, 151, 152, 153] in semantic segmentation are mainly due to *fully supervised* algorithms that apply Convolutional Neural Networks (CNNs) on datasets that contain images and their pixel-wise annotations, *e.g.*, PASCAL VOC [22] and Microsoft COCO [99]. These algorithms report excellent performance on the limited amount of classes covered by the datasets. The PASCAL VOC segmentation set contains 20 object classes with 500 images per class; the Microsoft COCO 91 object classes with 3.5K images per class, respectively, as introduced in Chapter 2. Extending fully supervised algorithms to more object classes, however, requires collecting massive amounts of pixel-wise annotations, which is both time-consuming and expensive. As reported in [103], the average annotation time was 239.7 seconds per image in the PASCAL VOC 2012 dataset. Therefore, other annotations that are less precise but faster to collect, such as points, scribbles, or bounding boxes, have also been employed to supervise semantic segmentation [103, 105, 108, 154].

Our proposed semantic segmentation algorithm uses only *image tags* for supervision. Image tags indicate which object class(es) are present in the image. They are usually much easier and faster to obtain than the other human annotations described above,

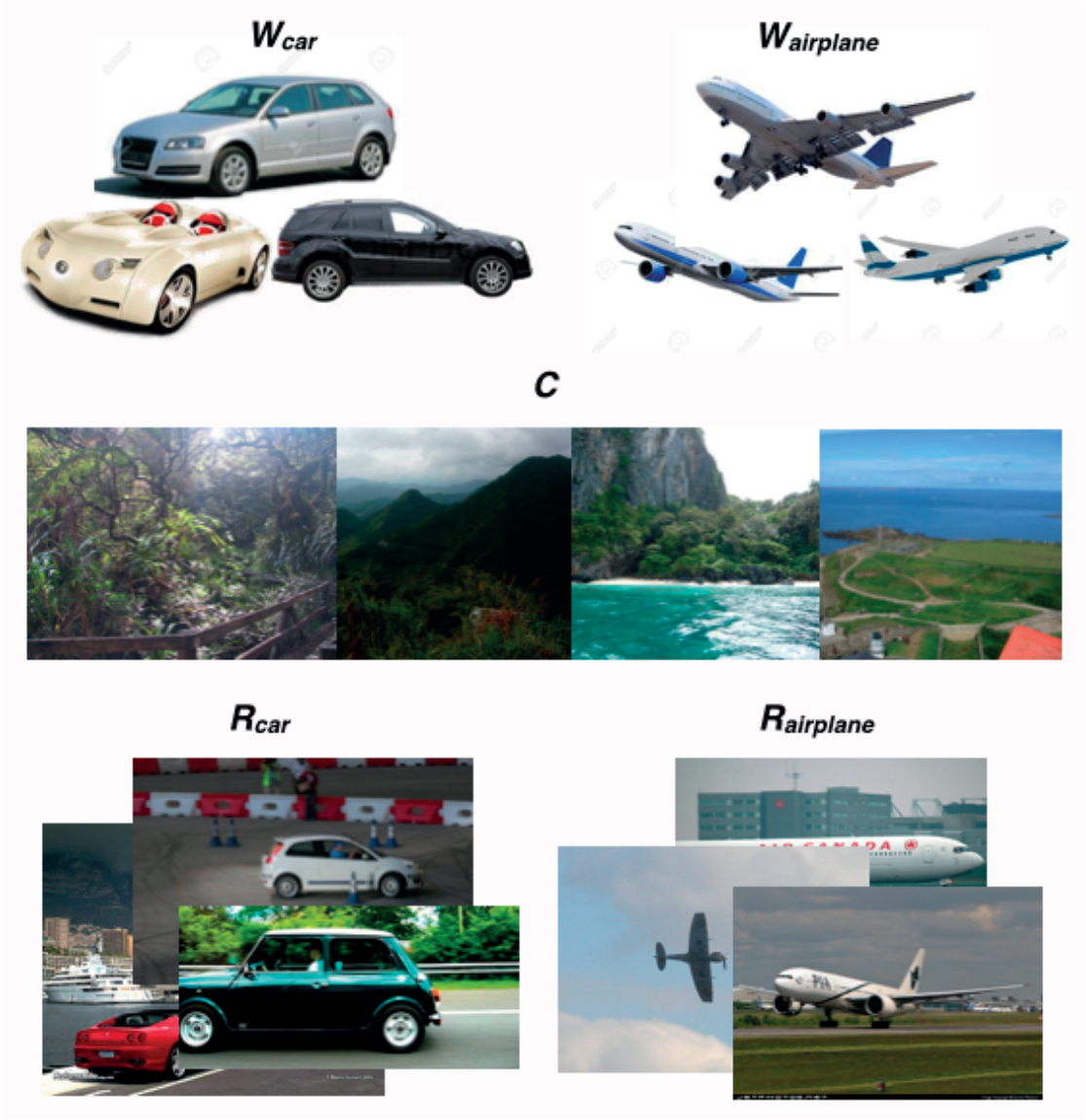


Figure 4.1: To supervise semantic segmentation, we extract three sets of web images: images with white background $\{\mathbf{W_k}\}$, images that contain common background scenes $\mathbf{C}$, and realistic images $\{\mathbf{R_k}\}$. $\{\mathbf{W_k}\}$ and $\mathbf{C}$ sets are used for initial training of the segmentation models. Later, these models are iteratively refined on the realistic image set $\{\mathbf{R_k}\}$. Finally $\{\mathbf{R_k}\}$ is adopted again to train an end-to-end semantic segmentation network.

hence have been explored in many *weakly supervised* semantic segmentation methods [111, 112, 113, 115, 116, 117, 118, 119, 155, 156], as discussed in Chapter 2. However, as opposed to the segmentation masks obtained by the pixel-wise annotations, image tags do not indicate the location of the object(s) in the image, making semantic segmentation much more challenging.

In our framework, we take advantage of the enormous amount of images and their rich context on the Internet. Through cleverly querying and exploiting web images, we build a pipeline to automatically generate segmentation masks for each class. We then train a deep convolutional neural network to segment multiple classes by using the generated masks. We only use image tags to query the web images and to train the network. No additional human annotations or interactions are required.

To supervise semantic segmentation, we extract three sets of images, as shown in Figure 4.1, covering the objects on white background, the common backgrounds, and the realistic scenes of the classes. We propose a novel three-stage procedure to progressively train our semantic segmentation model, as illustrated in Figure 4.3. In the first stage, a Shallow Neural Network (SNN) is initially trained to predict class-specific segmentation masks, using the hypercolumn features [41] from images with white background and images with common backgrounds. We then iteratively refine the SNN of each class on a set of realistic images of that class to generate better segmentation masks. In the last stage, the SNNs of all classes are assembled into one Deep Convolutional Neural Network (DCNN) by training the DCNN with the predicted multi-label segmentation masks from the SNNs. The DCNN, after the last training stage, outperforms current state-of-the-art weakly supervised semantic segmentation algorithms on the PASCAL VOC 2012 segmentation benchmark [22] by a notable margin.

In summary, our **main contributions** in this chapter are:

- We propose to collect three sets of useful web images for supervising segmentation. The first set contains images with a white background, the second set contains images with common background scenes, and the third set contains realistic images of each class.
- We present a novel three-stage pipeline for training semantic segmentation models using the three collected web image sets. The segmentation performance progressively improves following the training pipeline.
- Our DCNN, after the three training stages, achieves state-of-the-art performance on the PASCAL VOC 2012 segmentation benchmark, outperforming the previous weakly supervised semantic segmentation algorithms by more than 3 percent.
- The SNNs from the first two training stages produce state-of-the-art results in an object segmentation application.

4.2 Web Image Sets

Billions of images have been published online with rich context information. By clever querying, analyzing, and extracting images from this giant collection, we propose a novel

pipeline that learns semantic segmentation models with only image tag supervision. In this section, we describe how we query the web image collection and what types of web images we retrieve to supervise semantic segmentation.

For a class k, three sets of web images are collected that cover the visual appearance of the objects of that class and the backgrounds. A white background set (denoted as $\mathbf{W_k}$) is built by querying the text-based image search engine, *e.g.*, Google or Microsoft Bing, with the query "<class> on white background". Images retrieved with this query mostly have salient objects in front of a clean background, hence are easy to segment. We segment these images with dense Conditional Random Fields (CRFs) [157], using saliency maps from [158] as the unary term and the standard color and spatial distance as the pairwise term. Sample images from the $\mathbf{W_k}$ set and the corresponding segmentation masks are shown in Figure 4.2. Note that no human annotations are involved when generating these foreground masks. As images from $\mathbf{W_k}$ are relatively easy to segment, the quality of the segmentation masks, which are generated by saliency and CRF algorithms without using human annotations, is acceptable. Other simple segmentation methods such as grabcut [110] could also be used here to generate the masks. Experiments in Section. 4.4 show that starting from these auto-generated masks, our segmentation networks achieve reliable segmentation performance.

Figure 4.2: Sample images from the white-background set $\mathbf{W_{car}}$ and the corresponding segmentation masks generated from saliency combined with dense CRFs.

Images in the $\mathbf{W_k}$ set encode the foreground information of class k while the backgrounds are missing. We thus collect another set of images $\mathbf{C}$ that is unlikely to contain the classes of interest but contains common background scenes, such as sky, sea, grass, *etc.* $\mathbf{C}$ can be built by retrieving images from image sharing websites, *e.g.*, Flickr or Imgur[1], with common background keywords. Another approach is to use existing online datasets that

[1]www.flickr.com and www.imgur.com

contain mostly common background scenes, such as the Holiday dataset [159].

The third set $\mathbf{R_k}$ contains images of class k, depicting realistic scenes. $\mathbf{R_k}$ can be constructed by crawling image-sharing websites with the given class name or using existing datasets that already cover the class. Example images for these three sets are shown in Figure 4.1. Note that each class has a separate $\mathbf{W_k}$ and $\mathbf{R_k}$ set but shares the same $\mathbf{C}$ set since the $\mathbf{C}$ set contains the common backgrounds for most classes.

4.3 Training the network

Using the three sets of web images, we propose a novel three-stage training pipeline to learn the semantic segmentation models, as illustrated in Figure 4.3. For each class k, a Shallow Neural Network (SNN) is initially trained to output class-specific segmentation masks, using the hypercolumn features from $\mathbf{W_k}$ (images with clean foreground) and $\mathbf{C}$ (images with common background information). The SNN is then iteratively refined on the realistic images in $\mathbf{R_k}$. In the last stage, a DCNN is trained using the multi-label segmentation masks generated by the SNNs of all the classes. The first stage are class-specific where we train and refine a SNN for each keyword, whereas the last stage combines all SNNs into one DCNN to cover all classes.

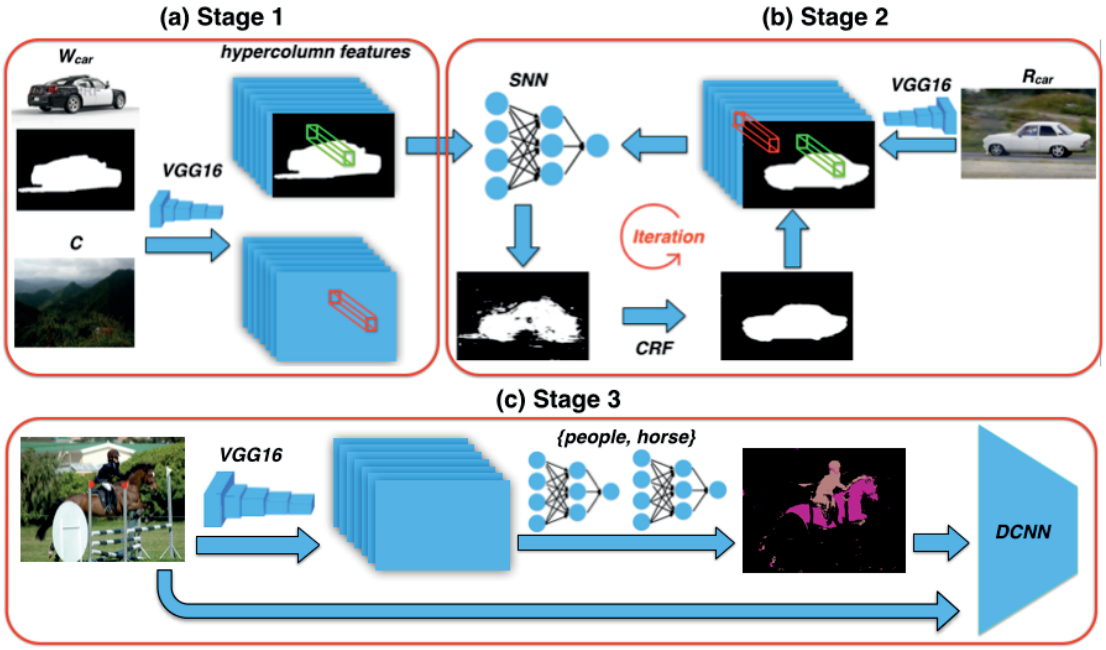


Figure 4.3: The proposed three-stage training pipeline. (a) Stage one: initial training of the SNN using hypercolumn features from $\mathbf{W_k}$ and $\mathbf{C}$. (b) Stage two: iterative refinement of the SNN on realistic images $\mathbf{R_k}$. (c) Stage three: all SNNs are assembled into one DCNN for end-to-end training and testing.

4.3.1 Stage 1: Initial Training

We denote $\Omega = \{1, 2, ..., N\}$ as the set of class names where N is the number of classes. For each class $k \in \Omega$, a SNN, whose parameters are denoted as Θ_k, is trained using the images in $\mathbf{W_k}$ and $\mathbf{C}$. As objects in $\mathbf{W_k}$ are surrounded by a white background, the foreground pixels from $\mathbf{W_k}$, denoted as $\mathbf{W_k^f}$, represent the visual appearance of class k. Correspondingly, the pixels in $\mathbf{C}$ represent the visual appearance of the common backgrounds. We use hypercolumn features [41] that are extracted from the pre-trained VGG16 network [4] to encode the visual appearance. For each pixel x_i^k from $\mathbf{W_k^f}$ or $\mathbf{C}$, we compute its hypercolumn feature:

$$h_i^k = \mathbf{H}(x_i^k) \tag{4.1}$$

Here $\mathbf{H}$ represents the operation of computing hypercolumn features. We then use the hypercolumn features from $\mathbf{C}$ and $\mathbf{W_k^f}$ to train the SNN, as shown in Figure 4.3(a).

Θ_k is trained to minimize the binary crossentropy loss:

$$\min_{\theta_k} \sum_i -\Big(t_i^k \log\Big(f(h_i^k, \theta_k)\Big) + (1 - t_i^k) \log\Big(1 - f(h_i^k, \theta_k)\Big)\Big) \tag{4.2}$$

Here $f(h_i^k, \theta_k)$ represents the SNN output. The SNN takes in the hypercolumn feature h_i^k and outputs the probability of x_i^k belonging to class k.

$$t_i^k = \begin{cases} 0, & \text{if } x_i^k \text{ is from } \mathbf{C} \\ 1, & \text{if } x_i^k \text{ is from } \mathbf{W_k^f} \end{cases} \tag{4.3}$$

An equal number of hypercolumn features are randomly extracted from $\mathbf{C}$ and $\mathbf{W_k^f}$, forming a balanced set for training the SNN. After initial training, the SNN of class k can predict the probability of a pixel belonging to class k, effectively outputting a class-specific mask for an image.

The combination of hypercolumn features and the SNN is similar to the functionality of the Fully Convolutional Network (FCN) [18] introduced in Chapter 2. We separate these two steps to easily balance the class distribution and to parallelize the training for different classes. Moreover, as the network for extracting hypercolumn features is pre-trained and shared, we only need to store Θ_k for a new class. Θ_k is relatively small for a shallow network (around 3.6 MB per class in our experiment), which enables efficient storage and retrieval for a large number of classes.

4.3.2 Stage 2: Refinement

Θ_k is initially trained to separate the objects of class k from the common backgrounds. However, the realistic background of class k may be different from the scenes in the common backgrounds sets $\mathbf{C}$. Therefore, we further iteratively refine Θ_k on realistic images, as shown in Figure 4.3(b). We make use of multiple CRF (Conditonal Random Fields [157]) iterations to improve our SNNs. CRF has been shown to be helpful for semantic segmentation as it recovers missing parts and refines the boundaries in the segmentation masks [115, 117, 150]. Unlike most methods that apply CRF once as post-processing, we apply CRF in each refinement iteration and learn to update the SNNs according to the CRF-refined masks. Consequently, the SNNs are forced to gradually learn to generate more complete segmentation masks with better boundaries.

Assume x_i^k represents the ith pixel in the collection of pixels from the realistic image set $\mathbf{R_k}$. h_i^k and y_i^k are its corresponding hypercolumn feature and label (0 for background and 1 for foreground). We refine Θ_k by minimizing the loss:

$$\min_{\theta_k} \sum_i -\Big(y_i^k \log\Big(f(h_i^k, \theta_k)\Big) + (1 - y_i^k) \log\Big(1 - f(h_i^k, \theta_k)\Big)\Big) \tag{4.4}$$

Given the parameters Θ_k of the SNN, the labels $\{y_i^k\}$ are predicted by minimizing the dense CRF energy function:

$$\min_{\{y_i^k\}} \sum_i \phi_i^k(y_i^k) + \sum_{i,j} \phi_{i,j}^k(y_i^k, y_j^k) \tag{4.5}$$

where the SNN's output is adopted for the unary term:

$$\phi_i^k(y_i^k) = -\log\Big(y_i^k f(h_i^k, \Theta_k) + (1 - y_i^k)\Big(1 - f(h_i^k, \Theta_k)\Big)\Big) \tag{4.6}$$

The pairwise term is set to be the standard color and spatial distance as in [157]. An *Expectation-Maximization* (EM) algorithm is adopted to iteratively optimize Equation 4.4 and Equation 4.5. In each iteration, we first update the labels of all pixels $\{y_i^k\}$ by minimizing Equation 4.5 using the method in [157]. Then we update the parameters Θ_k by minimizing Equation 4.4 using back propagation. The new parameters Θ_k are used in Equation 4.5 to obtain the new $\{y_i^k\}$ as the beginning of a new iteration. Figure 4.4 shows the evolution of the labels $\{y_i^k\}$ through iterations. We can clearly see that the segmentation masks predicted by the SNNs progressively improve over iterations, recovering missing parts and producing better boundaries. We found that 2 refinement iterations already significantly boost the performance while still being efficient in training time. More iterations result in minor performance gains but longer training time.

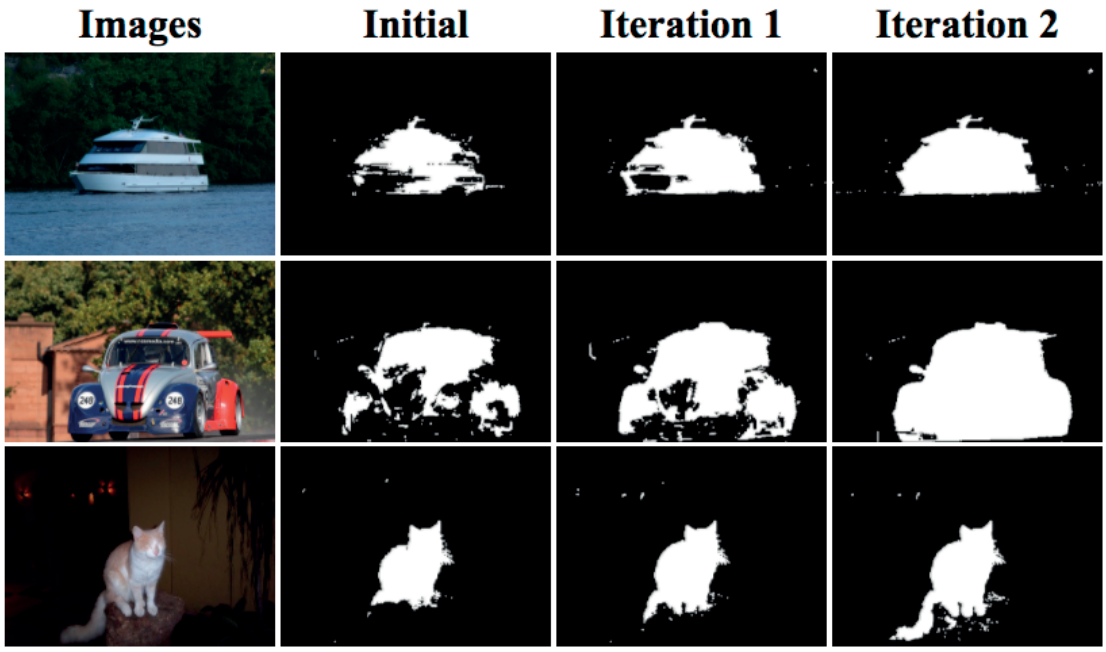


Figure 4.4: Improved segmentation masks by evolving the labels $\{y_i^k\}$ through iterations. Through several iterations of dense CRFs, the generated masks are getting refined over iterations with more missing parts recovered.

4.3.3 Stage 3: Assembling of Classes

The SNNs are trained independently for each class. To perform semantic segmentation for multiple classes in one shot, in this stage we assemble all SNNs into one Deep Convolution Neural Network (DCNN), as shown in Figure 4.3(c).

We use the DCNN architecture proposed in [150] due to its outstanding performance and good documentation. The DCNN is a fully convolutional neural network that takes in an image and directly predicts a multi-label segmentation mask. We train the DCNN on all realistic images in $\{\mathbf{R_k}\}$. Since no pixel-wise human annotations are provided for training, we use the SNNs to automatically generate multi-label segmentation masks as supervision. Specifically, if one image in $\{\mathbf{R_k}\}$ is labeled with tags $C \subseteqq \Omega$, the predicted label y_i for the ith pixel is:

$$y_i = \underset{k \in \{0\} \cup C}{\operatorname{argmax}} f(h_i^k, \Theta_k) \tag{4.7}$$

$$f(h_i^0, \Theta_0) = 1 - \max_{k \in C} f(h_i^k, \Theta_k) \tag{4.8}$$

Effectively, the combined multi-label segmentation mask is produced by taking the pixel-wise maximum of the probability maps across all labels, including background (represented as label 0). The background probability is set as one minus the maximum foreground probability.

After generating the multi-label segmentation masks using the SNNs, we treat them as the groundtruth segmentation masks and perform an end-to-end training of the DCNN. These multi-label segmentation masks are not human annotations, but automatically generated masks from our SNNs, which only require image tags during training. Therefore, the fully supervised DCNN training in [150] is transformed to weakly supervised in our framework.

4.4 Experiments

4.4.1 Setup

Datasets

We validate our algorithm on the standard PASCAL VOC 2012 segmentation benchmark [22]. Following [111, 112, 113, 115, 116, 117, 118, 119], we augment it with the extra annotations from [100], resulting in an augmented training (trainaug) set of 10,582 images, a validation set of 1,449 images, and a test set of 1,456 images, covering 20 classes. We report the standard *Intersection over Union* (IoU) value on both the validation and test sets.

For the three-stage training, we build $\{\mathbf{W_k}\}$ by querying Google with the 20 classes (using the strategy explained in Section 4.2) and obtain on average 340 images per class (6807 images in total). The Holiday dataset [159], which contains 1491 holiday images, serves as the $\mathbf{C}$ set since these images cover some common background scenes, *e.g.*, sky, mountains, grass. We use the trainaug set of PASCAL VOC 2012 as $\{\mathbf{R_k}\}$, as they are all realistic images of the 20 classes. The pixel-wise annotations of the trainaug set are not used. During the first two stages of training, all images are resized such that the larger dimension equals 340. In stage three, images are used in their original size according to [150].

Training and Testing

The hypercolumn features during training are extracted from the *conv1_2* (64 channels), *conv2_2* (128 channels), *conv3_3* (256 channels), *conv4_3* (512 channels) and *conv5_3* (512 channels) layers of the pre-trained VGG16 model, resulting in a 1472 dimensional vector. In the first training stage, each image of $\mathbf{W_k}$ contributes 1000 randomly selected hypercolumn features. An equal number of hypercolumn features are randomly selected from $\mathbf{C}$, forming a balanced set for initial training. In the refinement stage, 1000 hypercolumn features are randomly selected from both the foreground and the background regions of each image in $\mathbf{R_k}$.

The SNN is a fully connected network with 4 layers ($1472 \rightarrow 512 \rightarrow 256 \rightarrow 64 \rightarrow 1$).

We set *Relu* activation in between hidden layers and *Sigmoid* activation after the last layer. The network is trained with the Adam optimizer (lr = 0.0002) for 50 epochs in the initial training stage and 20 epochs in each refinement iteration. The CRF parameters are set through cross-validation on 100 separate validation images [150]. The segmentation performance is robust to the CRF parameters. All the experiments are conducted on a NVIDIA TITAN X GPU with 12GB memory. Training each SNN takes approximately 10 minutes for initial training and 20 minutes per refinement iteration. This part is implemented in the *Theano* framework [160]. The third training stage is performed using Deeplab code [150] that is based on the *Caffe* framework [161]. The parameters of Deeplab training are set according to [150].

4.4.2 Performance of Each Stage

Since we use **Web** images to supervise semantic **S**egmentation, we call our method **WebS**. Table 4.1 illustrates the performance of our method at different training stages on the validation set of the PASCAL VOC 2012 dataset.

To properly evaluate the semantic segmentation performance for all classes, the third stage of assembling all SNNs into a DCNN is always applied. **WebS-i** represents the model that directly combines stage one with stage three, bypassing the refinement stage. **WebS-i1** and **WebS-i2** are the models trained through all three stages, with one refinement iteration and two refinement iterations, respectively. The **WebS-i** model's mIoU of 46.4% is already comparable with several state-of-the-art weakly supervised semantic segmentation methods [111, 112, 113, 116, 154]. This demonstrates the effectiveness of our collected web images ($\{\mathbf{W_k}\}$ and $\mathbf{C}$) for semantic segmentation. The performance progressively improves after each refinement iteration. The first iteration increases the performance by 5.2% to 51.6%, already producing the best performance for weakly supervised semantic segmentation. The **WebS-i2** model further improves the performance by 1.8% to a mIoU value of 53.4%. This fact clearly demonstrates the benefits of adopting CRFs in our refinement stage.

Qualitative results of the three models are shown in Figure 4.5 and Figure 4.6. We can observe that the **WebS-i** model produces rough locations of the objects while missing some parts. Each refinement iteration reveals more details of the objects. The **WebS-i2** model produces accurate segmentation masks with fine-grained boundaries, even for some non-trivial objects such as the partially occluded dog in the 1st row and the hands of the person in the 2nd row of Figure 4.6. This is attributed to the fact that during each iteration, the CRF recovers missing parts of the objects based on the low-level statistics, like color, and produces better boundaries. Consequently, the SNNs learned from the CRF-refined masks also produce more accurate predictions. In the last row of Figure 4.6, our method fails to segment the train from the rail because these two objects often occur together. This is a typical failure case for most weakly supervised methods, as also

Method	bg	aero	bic	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
MIL-FCN [112]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	24.9
EM-Adapt [154]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	38.2
MIL-sppxl* [111]	77.2	37.3	18.4	25.4	28.2	31.9	41.6	48.1	50.7	12.7	45.7	14.6	50.9	44.1	39.2	37.9	28.3	44.0	19.6	37.6	35.0	36.6
CCCN* [113]	68.5	25.5	18.0	25.4	20.2	36.3	46.8	47.1	48.0	15.8	37.9	21.0	44.5	34.5	46.2	40.7	30.4	36.3	22.2	38.8	36.9	35.3
DCSM* [116]	76.7	45.1	24.6	40.8	23.0	34.8	61.0	51.9	52.4	15.5	45.9	32.7	54.9	48.6	57.4	51.8	38.2	55.4	**32.2**	42.6	39.6	44.1
BFBP* [115]	79.2	60.1	20.4	50.7	41.2	**46.3**	62.6	49.2	62.3	13.3	49.7	**38.1**	58.4	49.0	57.0	48.2	27.8	55.1	29.6	54.6	26.6	46.6
STC* [156]	**84.5**	68.0	19.5	60.5	42.5	44.8	68.4	64.0	64.8	14.5	52.0	22.8	58.0	55.3	57.8	**60.5**	**40.6**	56.7	23.0	**57.1**	31.2	49.8
SEC* [117]	82.4	62.9	26.4	61.6	27.6	38.1	66.6	62.7	75.2	**22.1**	53.5	28.3	65.8	57.8	**62.3**	52.5	32.5	62.6	32.1	45.4	45.3	50.7
ours:																						
WebS-i*	82.7	65.5	21.6	56.3	27.3	43.6	69.4	65.3	71.5	5.0	46.2	10.3	65.3	48.0	56.2	33.4	36.4	59.8	17.3	49.4	44.8	46.4
WebS-i1*	84.1	**69.5**	25.9	63.2	50.3	45.3	**70.5**	68.5	77.1	11.6	55.0	13.9	70.4	55.7	56.7	49.5	34.9	62.8	20.7	49.8	**48.5**	51.6
WebS-i2*	84.3	65.3	**27.4**	**65.4**	**53.9**	**46.3**	70.1	**69.8**	**79.4**	13.8	**61.1**	17.4	**73.8**	**58.1**	57.8	56.2	35.7	**66.5**	22.0	50.1	46.2	**53.4**

Table 4.1: Performance on the PASCAL VOC 2012 **validation set** of semantic segmentation methods using only image tag supervision. * represents applying the dense CRF [157] as post-processing.

Method	bg	aero	bic	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
MIL-FCN [112]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	25.7
EM-Adapt [154]	76.3	37.1	21.9	41.6	26.1	38.5	50.8	44.9	48.9	16.7	40.8	29.4	47.1	45.8	54.8	28.2	30.0	44.0	29.2	34.3	46.0	39.6
MIL-sppxl* [111]	74.7	38.8	19.8	27.5	21.7	32.8	40.0	50.1	47.1	7.2	44.8	15.8	49.4	47.3	36.6	36.4	24.3	44.5	21.0	31.5	41.3	35.8
CCCN* [113]	-	24.2	19.9	26.3	18.6	38.1	51.7	42.9	48.2	15.6	37.2	18.3	43.0	38.2	52.2	40.0	33.8	36.0	21.6	33.4	38.3	35.6
DCSM* [116]	78.1	43.8	26.3	49.8	19.5	40.3	61.6	53.9	52.7	13.7	47.3	34.8	50.3	48.9	69.0	49.7	38.4	57.1	34.0	38.0	40.0	45.1
BFBP* [115]	80.3	57.5	24.1	**66.9**	31.7	43.0	67.5	48.6	56.7	12.6	50.9	**42.6**	59.4	52.9	65.0	44.8	41.3	51.1	33.7	44.4	33.2	48.0
STC* [156]	85.2	62.7	21.1	58.0	31.4	55.0	68.8	63.9	63.7	14.2	57.6	28.3	63.0	59.8	67.6	**61.7**	42.9	61.0	23.2	**52.4**	33.1	51.2
SEC* [117]	83.5	56.4	28.5	64.1	23.6	46.5	70.6	58.5	71.3	**23.2**	54.0	28.0	68.1	62.1	70.0	55.0	38.4	58.0	**39.9**	38.4	**48.3**	51.7
ours:																						
WebS-i2*	**85.8**	**66.1**	**30.0**	64.1	**47.9**	**58.6**	**70.7**	**68.5**	**75.2**	11.3	**62.6**	19.0	**75.6**	**67.2**	**72.8**	61.4	**44.7**	**71.5**	23.1	42.3	43.6	**55.3**

Table 4.2: Performance on the PASCAL VOC 2012 **test set** of semantic segmentation methods using only image tag supervision. * represents applying the dense CRF [157] as post-processing.

Figure 4.5: Qualitative results on the PASCAL VOC 2012 validation set of our method before and after refinement iterations. Better viewed on screen. **WebS-i** is the model before refinement. **WebS-i1** and **WebS-i2** are the models with 1 iteration and 2 iterations of refinement, respectively.

Figure 4.6: Qualitative results on the PASCAL VOC 2012 validation set of our method before and after refinement iterations. Better viewed on screen. **WebS-i** is the model before refinement. **WebS-i1** and **WebS-i2** are the models with 1 iteration and 2 iterations of refinement, respectively.

discussed in [117].

4.4.3 Number of Images in $\mathbf{W_k}$ and $\mathbf{R_k}$

We further evaluate the effect of the number of images in $\{\mathbf{W_k}\}$ and $\{\mathbf{R_k}\}$ on the segmentation performance. In Table 4.3, we present the performance of the **WebS-i2** model with different numbers of training images. With only 2,000 images for the $\{\mathbf{W_k}\}$ and 2,000 images for the $\{\mathbf{R_k}\}$ (100 images per class), our method achieves a mIoU of 45.9%, already better than [112, 113, 116, 154], which are supervised with the whole trainaug set of the PASCAL VOC (10,582 images). It is even 9.3% better than [111], refer to Table 4.1, which uses an additional 700K images from ILSVRC [37]. Clearly by adding more images for supervision, the performance of our method improves. We project that our performance can be further improved by collecting more web images. Using only 6,807 Google images[2], 1,491 Holiday images and the 10,582 PASCAL VOC images, our method outperforms all the state-of-the-art methods by a notable margin. Compared to [156], which also uses web images to supervise semantic segmentation, our method requires substantially less images ([156] uses 41K Flickr images as well as the trainaug set of PASCAL VOC), but still produces better results on both the validation set (+3.6% mIoU) and the test set (+4.1% mIoU).

Table 4.3: Performance on the PASCAL VOC 2012 validation set of the **WebS-i2** model using different numbers of training images. **WebS-i2** represents our model with 2 iterations of refinement during the second stage.

#images in $\{\mathbf{W_k}\}$	#images in $\{\mathbf{R_k}\}$	mIoU
2,000	2,000	45.9
2,000	10,582	48.9
4,000	10,582	51.5
6,807	2,000	50.6
6,807	6,000	51.0
6,807	10,582	53.4

4.4.4 Comparison with Weakly Supervised Methods

In this subsection, we compare our method with other CNN based weakly supervised semantic segmentation algorithms. Table 4.1 and Table 4.2 show the performance on the PASCAL VOC 2012 validation and test set, respectively. Here we only compare with methods that require no additional human annotations except image tags [111, 112, 113, 115, 116, 117, 154, 156]. Our method produces excellent results on both the validation and test set of the PASCAL VOC, outperforming all state-of-the-art weakly supervised semantic segmentation methods. On the validation set, we improve over the previous

[2]due to Google search limit and copyright protection.

state-of-the-art SEC [117] method by 2.7% and achieve best scores on 12 out of 21 classes (including background) among all methods. Similar results are observed on the test set, where the evaluation is performed by the PASCAL VOC evaluation server. Our method achieves 3.6% higher mIoU than the state-of-the-art approaches, producing best scores on 14 out of 21 classes.

4.4.5 Object Segmentation Using SNNs

We also investigated the performance of our class-specific SNNs in an object segmentation task, where the user provides the labels of the classes they want to segment in an image. Note that this task is different from typical semantic segmentation where the labels of the images are not given during testing. We evaluate the performance on the Object Discovery (OD) dataset [162], a dataset containing three classes (airplane, car, horse) with 100 images per class. The SNNs are applied to generate segmentation masks by using the strategy explained in Section 4.3.3. The labels of the images are used to retrieve the correct SNN while generating the masks, as shown in Figure 4.3(c).

We compare the mIoU values with state-of-the-art object segmentation methods [162, 163, 164, 165] in Table 4.4. **SNN-i**, **SNN-i1** and **SSN-i2** are our SNNs after the initial training, the first refinement iteration, and the second refinement iteration, respectively. Our **SNN-i2** method improves over the previous state-of-the-art method by a large margin (7.5%). Even the **SNN-i** model already achieves state-of-the-art performance, again demonstrating the effectiveness of our three-stage training pipeline that uses web images.

Table 4.4: Comparison between our SNNs and other object segmentation methods on the *Object Discovery* dataset for the object segmentation task. The performances are measured by the mean intersection-over-union (mIoU) metric.

Method	Car	Horse	Airplane	mIoU
Joulin *et al.* [163]	37.2	30.2	15.4	27.6
Joulin *et al.* [164]	35.2	29.5	11.7	25.5
Rubinstein *et al.* [162]	64.4	51.7	55.8	57.3
Chen *et al.* [165]	64.9	33.4	40.3	46.2
SNN-i	67.7	52.4	53.8	58.0
SNN-i1	74.5	59.6	55.4	63.2
SNN-i2	**76.1**	**61.7**	**56.5**	**64.8**

We show sample results of our **SSN-i2** model compared with previous approaches in Figure 4.7 and Figure 4.8. While previous approaches either miss parts of the objects or segment some background regions as the objects, our SNNs successfully segment out the whole objects with accurate boundaries.

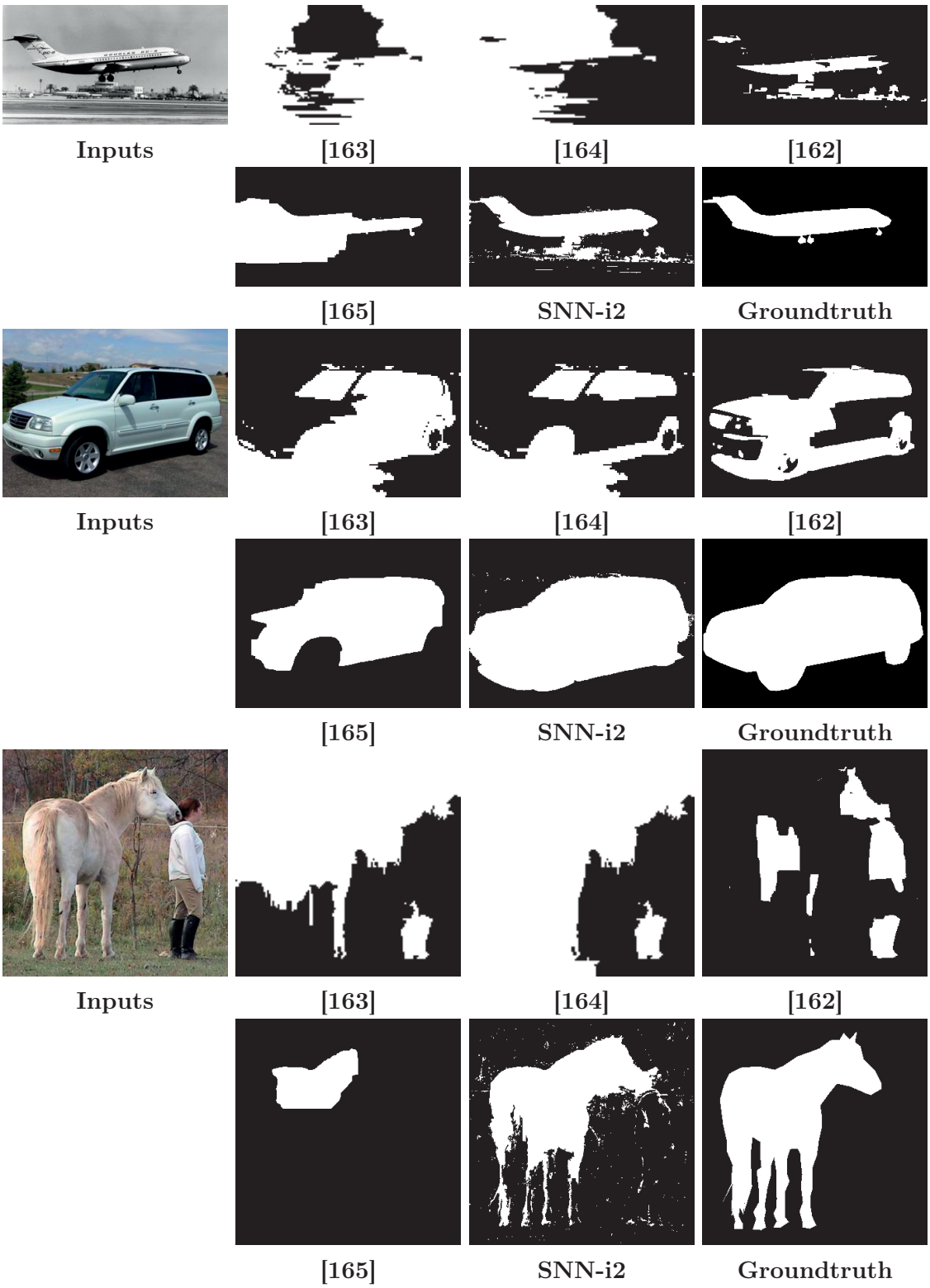


Figure 4.7: Sample results of our method and other state-of-the-art approaches on the OD dataset.

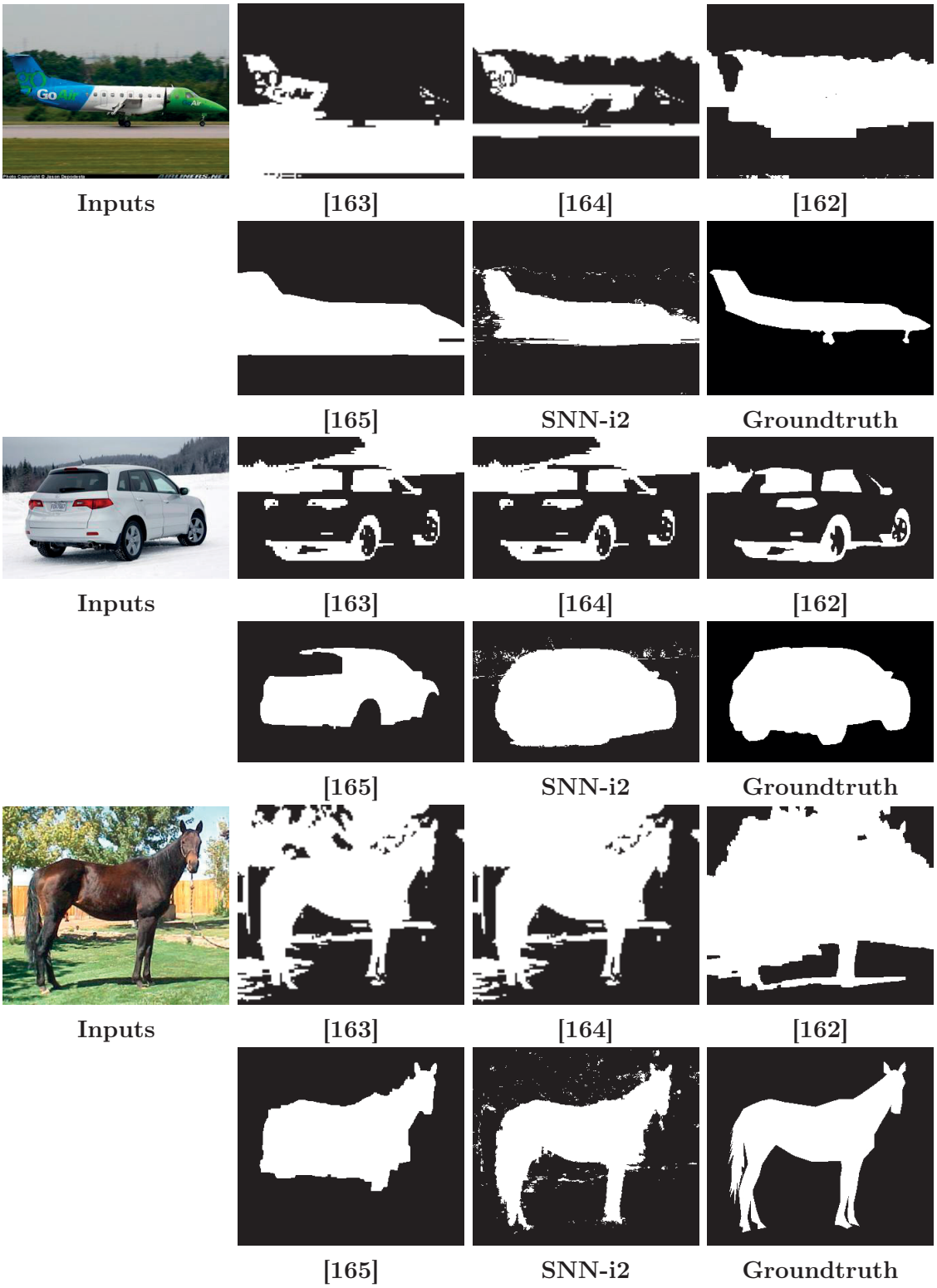


Figure 4.8: Sample results of our method and other state-of-the-art approaches on the *Object Discovery* dataset.

4.5 Conclusion

We propose a novel three-stage training pipeline to progressively learn the semantic segmentation model from three sets of web images. We demonstrate that our method outperforms the previous state-of-the-art weakly supervised semantic segmentation algorithms on the PASCAL VOC 2012 benchmark. The class-specific shallow neural networks (SNNs) learned in the first two training stages also produce excellent results when used in object segmentation. Note that when learning the SNNs, no pixel-wise human annotations are used. Therefore, the segmentation models proposed in this chapter can be used in many application to locate objects, as will be discussed in Chapter 5. Moreover, adopting these SNNs, many fully supervised computer vision methods, such as semantic segmentation [18, 19, 150, 153] and object detection [166], can be easily transformed into a weakly supervised framework.

The initial supervisions for our networks are the crawled images from image search engine with query "<class> on white background". This leads to certain limitations. Our segmentation networks are reliable for concrete objects but produce inferior results for scenery keywords or vague concepts, such as sky, water, blue, happy. The images queried using these keywords cannot be segmented by our saliency and dense CRFs in the first stage of our pipeline, thus leading to errors in the final results. Parsing these types of keywords in a weakly supervised manner is a challenging task that needs further exploration. We also found that the crawled images($\mathbf{W_k}$) for fine-grained classes often contain a considerable amount of mislabels between classes that can lead to errors in SNNs. The corresponding $\mathbf{R_k}$ crawled from image sharing platforms are also less accurate for fine-grained classes. Therefore, our segmentation algorithm is less suitable for fine-grained classes segmentation. Furthermore, some keywords can represent several different objects. For instance, 'mouse' can present an animal as well as a digital device. The crawled images hence encode two types of objects in one category, making it hard for our networks to learn an accurate segmentation mask.

5 Local Color Re-rendering

In this chapter, we present a local color re-rendering system that modifies the colors of certain objects in an input image to be more appealing, according to a keyword input from users. Our system is based on the color re-rendering algorithm in [17]. We propose to incorporate our semantic segmentation algorithm, presented in Chapter 4, in the pipeline, which brings two benefits. First, the segmentation masks help us to better model the correlations between keywords and colors, as will be discussed in Section 5.2.2. Second, in Section 5.2.3, we show that the segmentation masks can be used to locate the keyword-related regions, where the colors are re-rendered. Qualitative comparisons and a psychophysical experiment in Section 5.3 confirm of the performance of this system.

5.1 Introduction

In most consumer cameras, the in-camera processing pipeline contains tone-mapping and color enhancement algorithms to render the captured image visually pleasing. However, the resulting images can still contain unnatural or unsatisfactory colors, due to scene composition, mixed light sources, or other elements that confuse the automatic algorithms. Image color re-rendering, which aims at modifying the image colors for better visual appearance, is therefore a popular image enhancement step, especially for images shared on multimedia platforms.

Keyword-based image color re-rendering is a convenient technique for this, as users only need to specify a keyword to modify the colors. The state-of-the-art keyword-based color re-rendering algorithm [17] proposes to first learn the statistical correlations between keywords and color characteristics, and then to modify the colors of the input image according to the learned correlations. This approach has the advantage of easy extensibility to many keywords, with little human intervention. However, it ignores the semantic meaning of different regions in the image, which can result in unnatural artifacts in the re-rendered image. As shown in Figure 5.1b, with the keyword *strawberry*, [17]

(a) Input image (b) [17] (c) Ours

Figure 5.1: Example color re-rendering result for the keyword *strawberry*. Our method renders the strawberry colors to be more reddish while not affecting the other regions. Noticeable artifacts can be observed in the result of [17] as the whole image is tuned toward red color, which looks unnatural.

re-renders the color of the strawberry to be more vivid, but also makes the other regions to more reddish, such as the yellow cake and the white background, which looks unnatural.

We propose to incorporate semantic information into the color re-rendering pipeline through semantic segmentation. Following [17], we first compute the statistical correlations between keywords and color characteristics. During this process, we use our semantic segmentation algorithm to locate the keyword-related regions and to compute the correlations using only those regions. Color characteristics computed on the located regions, rather than on the whole imagea as in [17], are more accurate representations of the keyword, thus leading to more accurate correlation measures. When applying the color re-rendering, we again use the segmentation masks to indicate where to modify the colors, resulting in visually better results with fewer artifacts. As shown in Figure 5.1c, our method enhances the colors of the strawberries while not affecting the non-strawberry regions, as opposed to that of [17]. Both the qualitative comparisons and the psychophysical experiment validate the superior performance of our system, compared to the state-of-the-art approach [17].

In summary, our **contributions** in this chapter are:

- We incorporate a semantic segmentation algorithm into the color re-rendering pipeline. The segmentation masks help us to achieve better performance than the previous method.

- We achieve more accurate correlation measures between keywords and color characteristics using semantic segmentation masks.

5.2 Method

Figure 5.2 shows the pipeline of our algorithm that mainly consists of two phases. We first analyze the correlations between keywords and color characteristics with significance scores, resulting in a *keyword-feature database* containing such significance scores. Given a keyword and a color characteristic, we can retrieve the corresponding significance values from the database. We then modify the colors of the input image according to the pre-computed significance values. Semantic segmentation is used in both phases to learn accurate correlations and to locate the keyword-related regions for local color re-rendering.

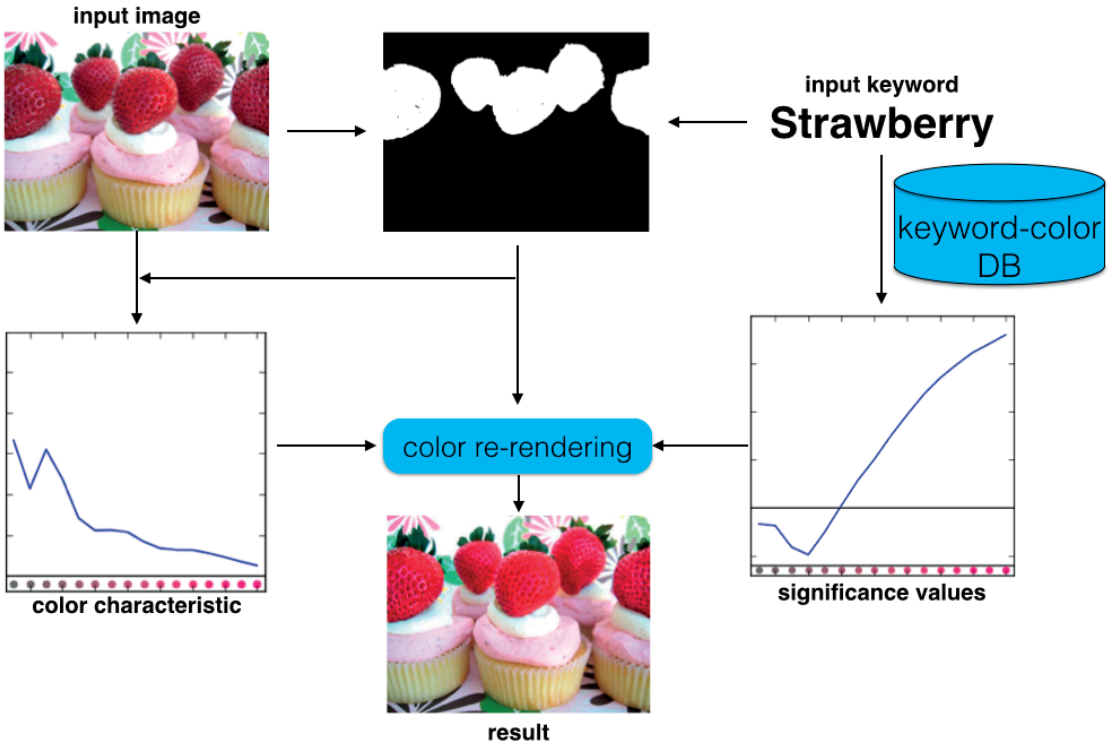


Figure 5.2: Pipeline of our keyword-based color re-rendering algorithm.

5.2.1 Semantic Segmentation

Semantic segmentation algorithms segment an image into regions with labels, indicating the regions that are related to the input keyword. We use the weakly supervised semantic segmentation algorithm in Chapter 4, for its easy extensibility to numerous keywords. As we only search for keyword-related regions, we use the **SNN** models. Figure 5.3 shows an example of the segmentation mask.

(a) Input image (b) Segmentation mask

Figure 5.3: Semantic segmentation result for the keyword *banana*.

5.2.2 Keyword-Color Correlation

Following [17], we statistically analyze the correlations between keywords and color characteristics[1] using a large image-tags database. Such correlations indicate which color characteristic is mostly related to an input keyword, thus indicating how to re-render the image colors according to the keyword. The process can be illustrated in Figure 5.4.

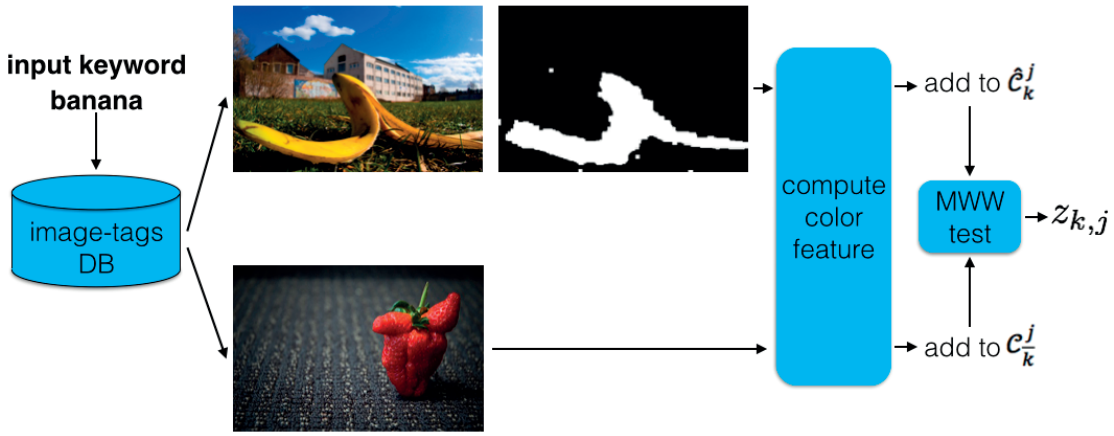


Figure 5.4: Computing the correlations between keywords and color characteristics.

We learn the correlations between keywords and color characteristics on the MIR Flickr dataset [167], a large *image-tags database* containing one million images, each with multiple annotated tags. Given this database of image-tags pairs $\{I_i, A_i\}$, where I_i represents the ith image and A_i is the set of the corresponding tags, the goal is to measure the correlation between a keyword k and a color characteristic j. For each image in the database, Lindner et al. [17] propose to first compute its color characteristic vector:

[1]Such as hue angle, chrome and CIEL histograms. Refer to [17] for all the color characteristics

$$C_i^j = F_j(I_i) \tag{5.1}$$

where F_j represents the function for computing the characteristic j, and C_i^j is the color characteristic vector for image I_i. A keyword k splits the MIR Flickr database into two distinct subsets, $\mathcal{I}_k = \{I_i | k \in A_i\}$ and $\mathcal{I}_{\overline{k}} = \{I_i | k \notin A_i\}$, which leads to two separate color characteristic sets, $\mathcal{C}_k^j = \{C_i^j | I_i \in \mathcal{I}_k\}$ and $\mathcal{C}_{\overline{k}}^j = \{C_i^j | I_i \in \mathcal{I}_{\overline{k}}\}$.

$\mathcal{C}_k^j$ is the set of feature vectors of color characteristic j for images that contain the tag k, and $\mathcal{C}_{\overline{k}}^j$ is the corresponding set for images that do not carry the tag k. Therefore, to assess how a keyword k would influence the color characteristic j, we can compare $\mathcal{C}_k^j$ and $\mathcal{C}_{\overline{k}}^j$ against each other. The difference between them implies the correlation between k and j. However, since C_i^j is computed on the whole image, which also encloses regions that do not correspond to k, the computed correlation is not as accurate due to the noise introduced from those unrelated regions. For example, for the image in Figure 5.3a with keyword *banana*, the C_i^j computed on the whole image not only describes the color of the banana but also the sky and the grass, which introduces noise to the $\mathcal{C}_k^j$ set.

We propose to use semantic segmentation to improve the accuracy of the keyword-color correlations. Assume M_i^k is the semantic segmentation mask of image I_i for keyword k, like in Figure 5.3b. The new color characteristic vector $\hat{C}_i^j$ for image I_i is computed as:

$$\hat{C}_i^j = F_j(M_i^k \cdot I_i) \tag{5.2}$$

Because of the mask M_k^j, the color characteristic vector $\hat{C}_i^j$ now mainly encodes the color information for the regions that correspond to the keyword k with significantly less noise from the unrelated regions. A new characteristic set for $\mathcal{I}_k$ is built as $\hat{\mathcal{C}}_k^j = \{\hat{C}_i^j | I_i \in \mathcal{I}_k\}$, which represents the color of keyword k better than $\mathcal{C}_k^j$ used in [17]. Hence, the dissimilarity between $\hat{\mathcal{C}}_k^j$ and $\mathcal{C}_{\overline{k}}^j$ is a more accurate indication of the correlation between k and j. We do not apply semantic segmentation on $\mathcal{I}_{\overline{k}}$ because images in $\mathcal{I}_{\overline{k}}$ do not carry keyword k.

As in [17], we apply the Mann-Whitney-Wilcoxon (MWW) ranksum test [168] to measure the dissimilarity between $\hat{\mathcal{C}}_k^j$ and $\mathcal{C}_{\overline{k}}^j$. The union set $\hat{\mathcal{C}}_k^j \cup \mathcal{C}_{\overline{k}}^j$ is first sorted to obtain the positional indexes for each element. The ranksum T is defined as the sum of these indexes for all the elements in $\hat{\mathcal{C}}_k^j$. Note that we are not computing the direct sum of elements in $\hat{\mathcal{C}}_k^j$. Instead, the sum is computed on the rank index of each element. Another two statistics are also derived as:

$$\mu_T = \frac{N_k(N_k + N_{\overline{k}} + 1)}{2} \tag{5.3}$$

$$\sigma_T^2 = \frac{N_k N_{\overline{k}}(N_k + N_{\overline{k}} + 1)}{12} \tag{5.4}$$

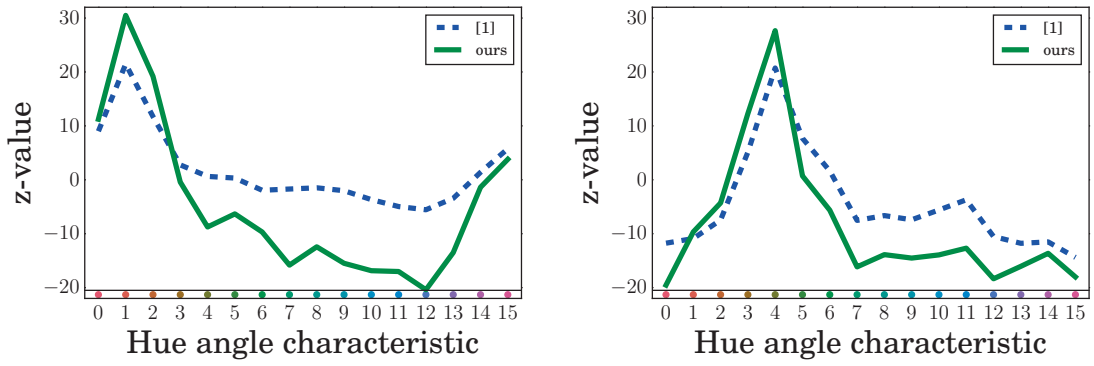
where N_k and $N_{\overline{k}}$ are the number of elements in $\hat{\mathcal{C}}_k^j$ and $\mathcal{C}_{\overline{k}}^j$, respectively. μ_T and σ_T are

basically the expected mean and variance of the ranksum.

According to MWW test, we compute the standardized $z_{k,j}$ value to represent the correlation between k and j:

$$z_{k,j} = \frac{T - \mu_T}{\sigma_T} \tag{5.5}$$

The magnitude of z reflects the strength of the correlation, and its sign shows the direction of the correlation.



(a) Strawberry (b) Sunflower

Figure 5.5: z values between the hue angle characteristic and keywords *strawberry* and *sunflower*.

In Figure 5.5, we show two typical examples of z values computed by our method and [17]. In the examples, we use the hue angle color characteristic that is categorized into 16 bins. Each bin is treated as a color feature, with a z value computed accordingly. We show the corresponding color of each bin at the bottom line of the figure. For keyword *strawberry*, the z values computed using our method show a stronger peak around the red hues with smaller values for the other colors than that from [17], thus demonstrating that our method calculates more accurate keyword-color correlations than [17], as strawberry is only strongly correlated with the red color. We achieve this by using semantic segmentation in the process to filter out the unrelated regions for strawberry, resulting in a stronger signal of red color and less other colors in $\hat{\mathbb{C}}_k^j$. A similar trend is observed for the keyword *sunflower*.

5.2.3 Local Color Re-rendering

Keyword-based image color re-rendering system takes an input image I_i and a keyword k, and modifies the colors of I_i to be visually more appealing according to the keyword k. To determine how to modify the colors, we refer to $\{z_{k,j}|j \in \mathcal{J}\}$, where $\mathcal{J}$ represents the set of all color characteristics, and we choose to enhance the color characteristic j that has the largest $z_{k,j}$ value. For instance, if j is the hue angle histogram, we modify the

hue channel of I_i in LCH color space and convert it back to RGB.

The strength of the color re-rendering operations are conditioned on two factors: (1) the correlation level between the keyword and the color characteristic, namely $z_{k,j}$; and (2) whether the color characteristic in the current image is too strong or too weak. We measure the second term by the difference $\delta^j_{I_i,k}$ between the I_i's color characteristic vector $\hat{C}^j_i$ and those in $\hat{\mathfrak{C}}^j_k$:

$$\delta^j_{I_i,k} = \begin{cases} \max[0, Q_{75\%}(\hat{\mathfrak{C}}^j_k) - \hat{C}^j_i] & \text{if } z_{k,j} \geq 0 \\ \max[0, \hat{C}^j_i - Q_{25\%}(\hat{\mathfrak{C}}^j_k)] & \text{if } z_{k,j} < 0 \end{cases} \tag{5.6}$$

where $Q_p(.)$ is a set's p^{th} quantile value. Conceptually, $z_{k,j}$ indicates whether k and j are correlated, $\delta^j_{I_i,k}$ (rewrite as δ for simplicity) indicates how much the color characteristic of I_i, namely $\hat{C}^j_i$, is different from the majority in $\hat{\mathfrak{C}}^j_k$. Hence $\delta z_{k,j}$ together implies the strength to enhance the colors of I_i, according to the keyword k.

The color re-rendering operation is defined as a nonlinear tone-mapping function that is the same as in [17]. The strength of the re-rendering operations $\delta z_{k,j}$ is used to compute the derivatives m of the tone-mapping function.

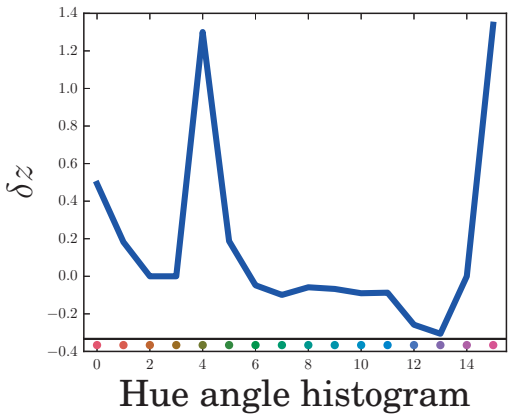
$$m = \begin{cases} \max[1/m_{\max}, 1/(1 + S\delta z_{k,j})] & \text{if } \delta z_{k,j} \geq 0 \\ \min[m_{\max}, 1 + S|\delta z_{k,j}|] & \text{if } \delta z_{k,j} < 0 \end{cases} \tag{5.7}$$

where S is a constant that controls the strength of the nonlinearity. The derivatives are clipped to $[1/m_{max}, m_{max}]$ to reduce extreme enhancement artifacts. The mapping function is obtained by integration over the derivatives. Conceptually, such a mapping function enhances the color characteristic $\hat{C}^j_i$ if $\delta z_{k,j}$ is positive and suppresses $\hat{C}^j_i$ if $\delta z_{k,j}$ is negative.

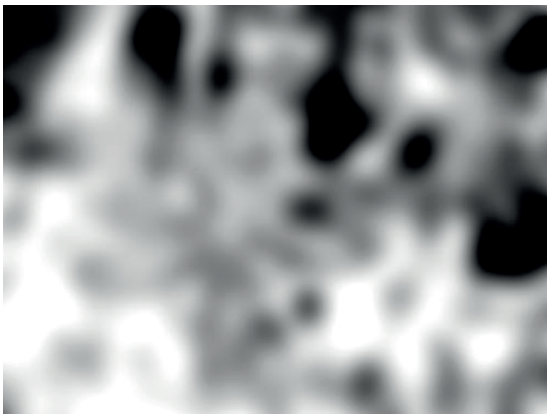
To determine where to apply the mapping function, Lindner et al. [17] build a weight map for the input image, as shown in Figure 5.6c. The re-rendering operation is then weighted by the weight map. Conceptually the re-rendering operations are thus applied to the regions with high weights. The weight values are derived directly as the z value between each pixel's color characteristic and the input keyword. In this sense, the computed weight values only capture the color information, whereas no semantic information is considered, which often mis-identifies the keyword-related regions. For instance, for the keyword 'Ferrari', any objects with red color would be assigned with a high weight according to [17]. Moreover, due to the unrelated regions used to compute $\hat{\mathfrak{C}}^j_k$, the z values obtained by [17] are already less accurate and encode certain amount of noisy colors, thus resulting in more errors in the weight map. As illustrated in Figure 5.6c, the green leaves in the background are also captured in the weight map.

We propose to use semantic segmentation in this step, where we can better locate

(a) Input image



(b) δz values



(c) Weight map in [17]

(d) Result from [17]

(e) Our mask

(f) Our result

Figure 5.6: Local color re-rendering example for *orchid*. We show both our result and the result of [17].

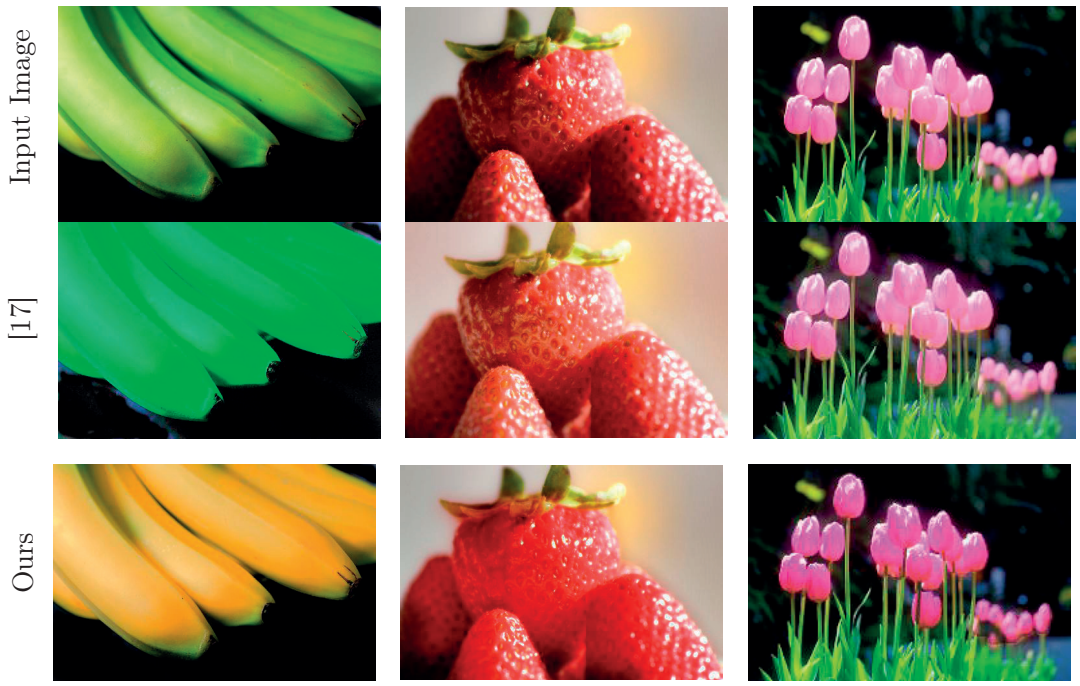
the keyword-related regions than the weight map in [17]. As shown in Figure 5.6e, our segmentation mask is significantly more accurate in locating the orchid. Given a segmentation mask, we first smooth it with a Gaussian filter with $\sigma = 0.02\sqrt{h^2 + w^2}$, where h and w are the image's height and width. This is because the binary mask, as in Figure 5.3b, might introduce artifacts near the edges. The color re-rendering is then weighted according to the smoothed segmentation mask, which results in visually better results than [17]. As illustrated in Figure 5.6d and 5.6f, [17] changes the colors of the orchid as well as the green background due to the errors in the weight map. Our method enhances the color of the orchid while leaving the background unchanged. Figure 5.6b shows the corresponding $\delta z_{k,j}$ values of our method, which clearly indicate that the orchids in the input image need to be enhanced to be more purple and red.

5.3 Experiments

We use the MIR Flickr dataset [167] for computing the keyword-color correlations. This dataset contains one million images, each with multiple annotated tags. The same as in [17], the parameter S in Equation 5.7 is set to 2. m_{max} is set to 5 as a compromise to allow visible changes while reducing the extreme artifacts.

5.3.1 Qualitative Results

We show a qualitative comparison between our method and [17] in Figure 5.7. Clearly our method generates visually more appealing results than [17] with much fewer artifacts. For instance, in Figure 5.7a, Lindner et al. [17] modify the color of the banana to be unnatural cyan, whereas ours correctly re-renders the color of the banana to yellow. This can be attributed to our more accurate keyword-color correlations that use semantic segmentation to filter out non-related regions. Similar observations can be made for Figure 5.7b and Figure5.7c. The colors of strawberries and tulips in our results are clearly more vivid and appealing than those from [17] for the same reason. In addition, in Figure 5.7d and Figure5.7e, our method enhances the colors of the Ferrari and sunflower to be more appealing while not affecting the background, because our segmentation masks accurately locate these objects. Lindner et al. [17] also modify the colors of the background due to the errors of the weight maps, resulting in unnatural artifacts. Figure 5.7f is a failure case of both methods, as our semantic segmentation algorithm segments part of the table as cheese. Such errors can be improved with the development of better segmentation algorithms.



(a) Banana (b) Strawberry (c) Tulip

(d) Ferrari (e) Sunflower (f) Cheese

Figure 5.7: Qualitative comparisons between the results of [17] and our method. In (a) to (e), our methods produce more pleasing colors than [17] with much fewer artifacts. (f) is a failure case for both methods.

5.3.2 Psychophysical Experiment

We further validate our color re-rendering method by a psychophysical experiment on a crowd-sourcing website[2]. The interface for this experiment is shown in Figure 5.8. For this experiment, we choose 50 images that cover different keywords. For each image, the participant is shown the original image, and the two re-rendered results from [17] and our method, and is asked to choose the more appealing one among the two re-rendered results. In total 50 users participate in the experiment with each of them labeling all 50 images.

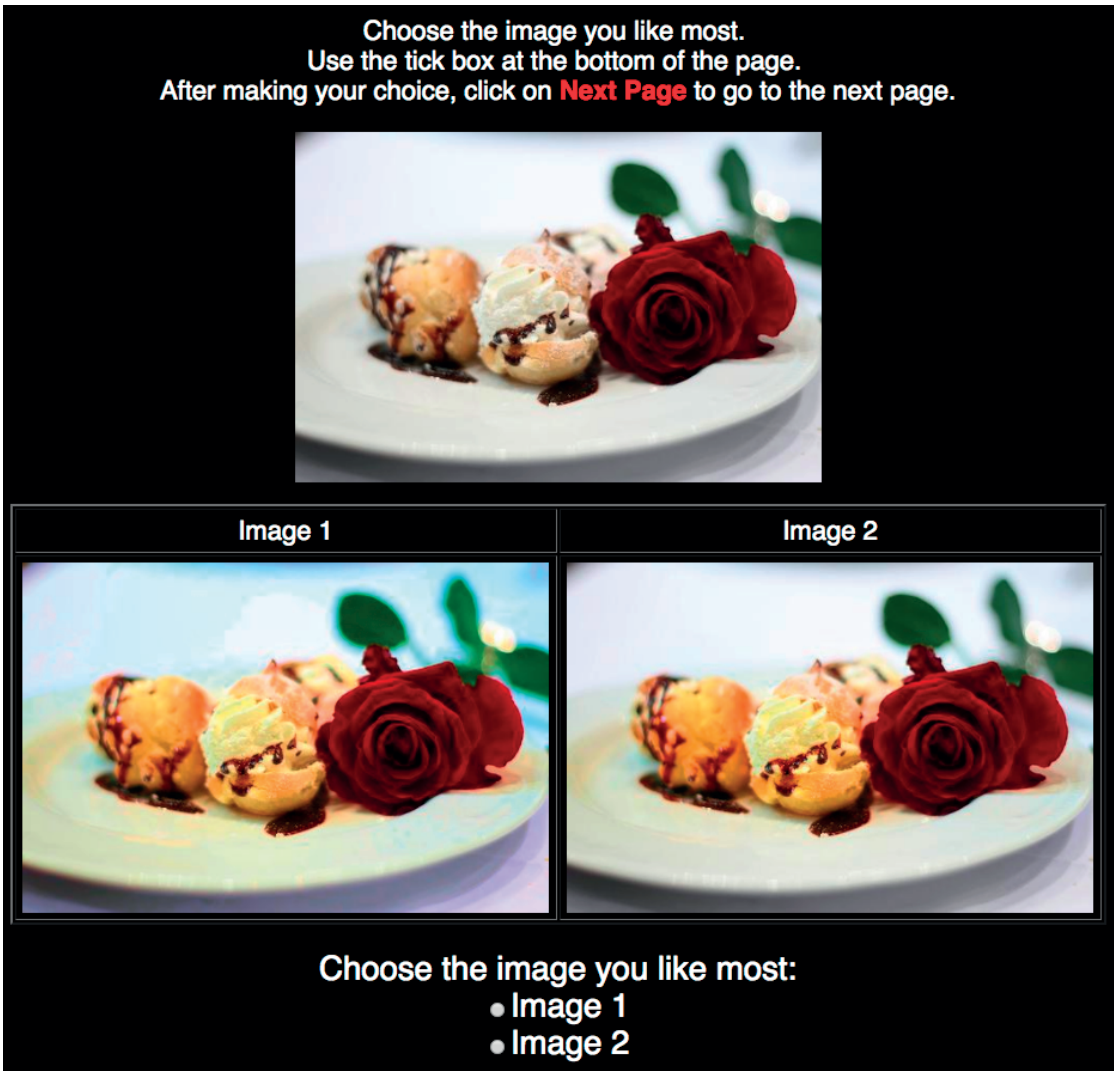


Figure 5.8: Interface of the psychophysical experiment for validating our color re-rendering algorithm.

We show the results of the experiment in Table 5.1. For 35 out of the 50 images, our

[2]www.clickworker.com

color re-rendering results are preferred over the counterpart from [17], which means a 70% preference rate of our results compared to 30% of [17]. Moreover, our method is favored on all keywords, further proving that our method is independent of keywords.

#preference / keywords (#images)	[17]	Ours
strawberry(12)	4	8
banana(11)	3	8
desert(13)	4	9
sunflower(6)	1	5
tulip(5)	2	3
orchid(3)	1	2
all(50)	15	35

Table 5.1: Results of the psychophysical experiment.

5.4 Conclusion

We propose to integrate semantic segmentation with the keyword-based image color re-rendering pipeline. The semantic segmentation is first employed to improve the calculation of the keyword-color correlations, where the segmentation masks help to remove the influence of the non-keyword-related regions and lead to more accurate correlation measures. We also use semantic segmentation to locate the keyword-related regions in the input image, and to re-render their colors according to the computed correlations. By incorporating semantic segmentation, our keyword-based color re-rendering method generates notably better results than the state-of-the-art approach [17]. This is demonstrated by both the qualitative comparisons and the psychophysical experiment.

6 GANs for Image Enhancement

In this chapter, we present an automatic image enhancement algorithm that uses generative adversarial networks. This algorithm is data-driven: no heuristic rules or human interactions are required. The enhancement functions are directly learned from the training data. We design novel Domain Encoding Generative Adversarial Networks (DEGANs) for this purpose.

In Section 6.1, we introduce the automatic image enhancement problem and explain the limitation of current GANs for this task. In Section 6.2, we present our DEGANs architecture and describe the details about each component of DEGANs. We provide qualitative comparisons with the state-of-the-art method in Section 6.3, where we also analyze the effect of each component on the generated results. Finally, we conclude this chapter in Section 6.4.

6.1 Introduction

Many tasks in computational imaging can be viewed as source-to-target image translation problems, such as image colorization [169] and style transfer [170]. The goals of these tasks can be summarized as building translation functions from the source image domain to the target image domain, under certain constraints that are application-specific. Image enhancement is a typical case where the goal is to transform an image from the low quality domain to the high quality domain. Current approaches to achieve such transformations mainly rely on human efforts and expertise. For instance, users can hire professional photographers to take photographs and apply post-processing with tools such as Photoshop or Gimp. In Chapter 5, we introduced a semi-automatic approach for re-rendering image colors, where users are asked to input keywords to direct the enhancement functions. Our focus in this chapter is the development of a fully automatic algorithm for image enhancement.

The recently proposed Generative Adversarial Networks (GANs) [34] and conditional

(a) Input images (b) Our results

Figure 6.1: Example results of our DEGANs for automatic image aesthetics enhancement. Our method produces results that are visually more pleasing than the input images.

GANs [51] enable us to tackle such source-to-target image translation problems [52, 55, 171, 172, 173, 174, 175] in a fully-automatic manner. However, pairs of images are always required to train the conditional GANs. For example, for image super-resolution, each pair of a training sample should contain a low-resolution image and a corresponding high-resolution image (the groundtruth). Without the groundtruth images, conditional GANs cannot learn the proper transformation functions.

For image enhancement, it is very difficult to collect a large set of such pairs of images. Processing one image for better visual quality could take a professional user several minutes, if not more, with an image editing software such as Photoshop. Therefore, it is

very time-consuming and expensive to build thousands of such pairs. Additionally, as "better visual quality" is a subjective concept, there can be a large number of possible operations to enhance a low quality image. Building a complete set of groundtruth images is thus very difficult, if not impossible.

To overcome such limitations on the training data, we propose a novel framework to train GANs without using paired images. Our framework includes two steps. First, we train a domain encoder network on a separate aesthetics prediction task, where the domain encoder learns what type of images are aesthetically of high quality. Training this network is explained in Chapter 3. Second, the pre-trained domain encoder is incorporated into a novel Domain Encoding GANs (DEGANs) architecture; the encoder supervises the DEGANs to learn the proper enhancement operations that improve image aesthetics. The enhancement functions should obey the constraint that the content and the semantics of the generated images need to stay the same as the original images. Therefore, the changes are usually subtle, and the transformation functions include only operations such as color and lightness alterations and contrast adjustments (see Figure 6.1). Note that we do not focus on operations like denoising or super-resolution, nor do we allow hanges in composition or style. The learned enhancement functions essentially resemble the tone-mapping functions used for high dynamic range imaging, like those in [176, 177]. Unlike those functions that are based on heuristically defined rules, our enhancement operations are directly learned from the training data. Our DEGANs achieve similar effects to the color re-rendering operations in Chapter 5, whereas no user interactions or keywords are required.

In summary, our **main contributions** in this chapter are:

- We propose Domain Encoding Generative Adversarial Networks (DEGANs) that learn the image enhancement functions in a fully-automatic manner. Training these DEGANs does not require pairs of images.
- We propose to use a domain encoder network pre-trained on an aesthetics prediction task to encode knowledge about image aesthetics, thus removing the need for groundtruth images in training GANs.
- Our DEGANs automatically enhance the input image to be visually more pleasing, without any user interaction.

6.2 Our Framework

In this section, we introduce our framework for training GANs for image enhancement. We use a pre-trained domain encoder network to encode the knowledge about image aesthetics. The pre-trained domain encoder network is then incorporated into the DEGANs to provide

one of the supervision signals. These DEGANs, as shown in Figure 6.2, are trained to generate natural images that are more aesthetically appealing than the input.

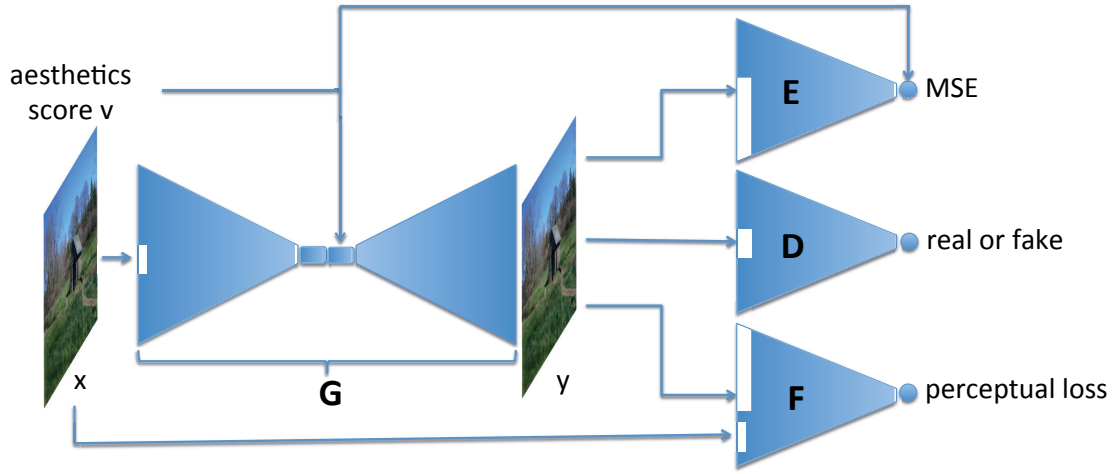


Figure 6.2: The DEGANs consist of a generator G, a domain encoder E, a discriminator D, and a feature extractor F.

6.2.1 Domain Encoder Pre-training

The domain encoder is designed to be a regression network. We train it on an image-aesthetics regression task as described in Chapter 3. We modify the Xception network [178], as this architecture produces state-of-the-art results for image classification tasks and is, in training and testing, much faster than VGG [4]. The top layer is replaced with a fully connected layer that maps to one value (for regression), followed by a *sigmoid* activation before output. The output aesthetics score is in the range of [0,1], where 1 represents the highest visual quality and 0 is the lowest.

Sample results are shown in Figure 6.3. This Xception model achieves the Mean Square Errors of 0.0043 on the *RS-test* and 0.0091 on the *ED-test*. Compared to the network presented in Chapter 3, the domain encoder network in this chapter achieves comparable performance while being faster in the forward pass, thus being more suitable to be incorporated in the training of DEGANs. The domain encoder learns the knowledge about what type of images are aesthetically pleasing. Such knowledge is used to supervise the DEGANs in order to apply image enhancement operations.

6.2.2 DEGANs

The structure of DEGANs is shown in Figure 6.2. It consists of an image-conditioned generator G, a pre-trained domain encoder E, a discriminator D and a feature extractor F. The image-conditioned generator G maps a low quality image to an enhanced result. The domain encoder E measures the aesthetics score for the generated image, while

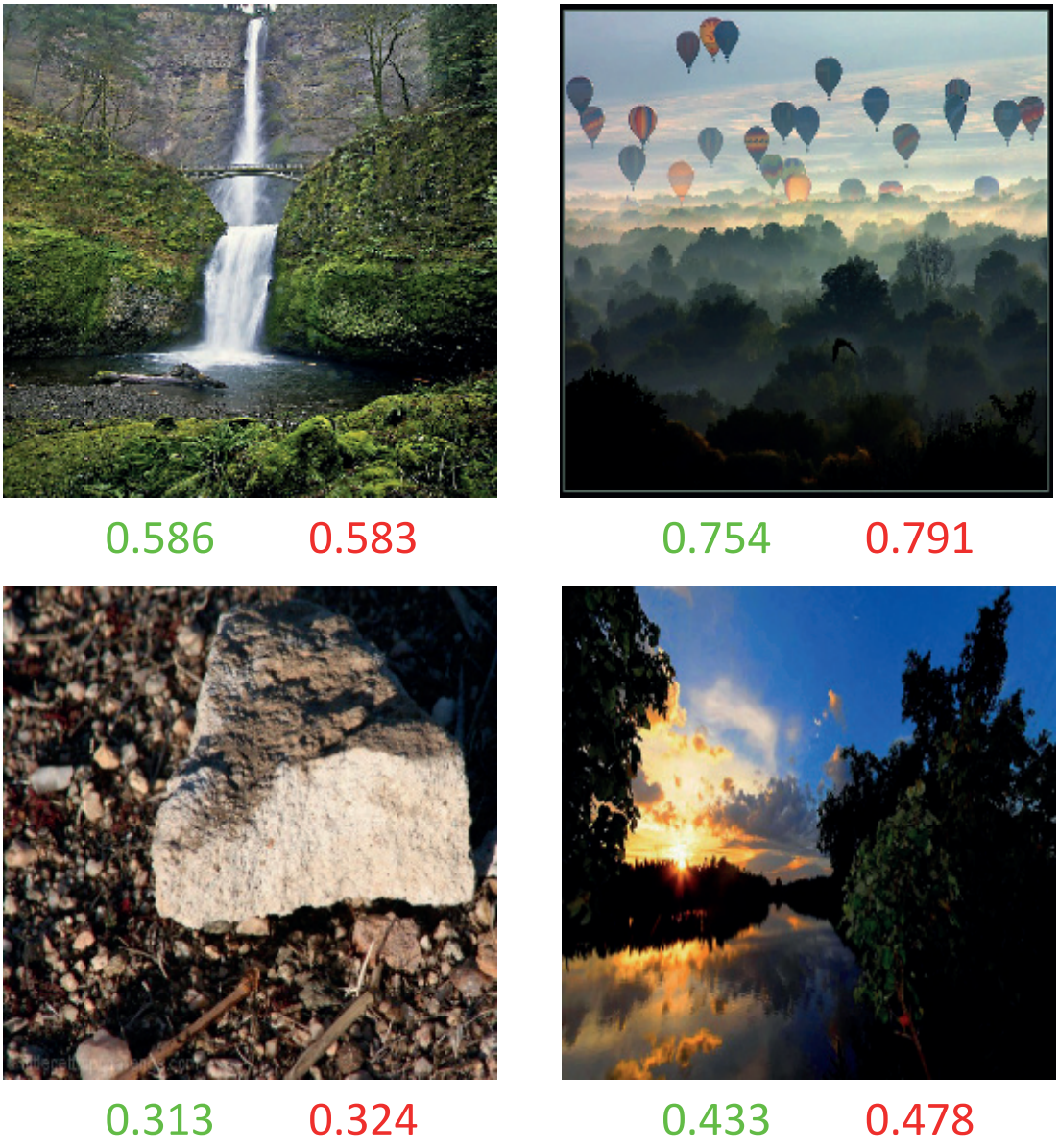


Figure 6.3: Sample results from the pre-trained domain encoder for image aesthetics assessment task. The green value under each image represents the predicted aesthetics score and the red value is the groundtruth score from [12].

the discriminator D judges whether the generated image looks realistic. We specifically add the feature extractor F into the architecture to adapt to the image enhancement application. A perceptual loss [23] is built on top of the feature extractor to fulfill the constraint of content similarity between the input image and the generated result.

6.2.2.1 Generator

The generator maps an input image x and an aesthetics score v to a generated image y: $G : \{x, v\} \rightarrow y$. G is supervised by the weighted sum of three losses:

$$\begin{aligned} L_G(x, v) =& w_E \cdot L_E(y, v) + w_D \cdot L_D(y, 1) \\ &+ w_F \cdot L_F(x, y) \end{aligned} \tag{6.1}$$

L_E, L_D, L_F represent the losses from the domain encoder E, the discriminator D and the feature extractor F, respectively, which will be explained in Section 6.2.2.2, 6.2.2.3 and 6.2.2.4. w_E, w_D, w_F are their corresponding weights.

To increase the stochasticity of the generator, we add an aesthetics score v at the bridging layer, similar to the noise vector used in the traditional GANs [34], as theoretically one input image has an infinite number of aesthetically enhanced target images. v is concatenated with the encoder output before fed into the decoder. Ideally v should serve as an indicator to G, directing G to produce an enhancement result that has the corresponding aesthetics score of v. However, minor stochasticity is observed during experiments with different aesthetics scores, which is consistent with several other techniques that try to improve stochasticity of the generative model [53, 55]. How to increase stochasticity of the generator in GANs remains an open problem for future research. Therefore, we set v to 1 for all images during testing. During training, v is set as the corresponding aesthetics scores of the high-quality images.

6.2.2.2 Domain Encoder

The weights of E carry the knowledge about image aesthetics. To use such knowledge to supervise the generator, the loss of the domain encoder L_E is set as:

$$L_E(y, v) = (E(y) - v)^2 \tag{6.2}$$

where v is the target aesthetics score and $E(y)$ is the predicted aesthetics score for an input image y. When trying to minimize L_E in Equation 6.2, the generator is enforced to produce an output image that meets the aesthetics standard set by v. In this sense, the generator learns about the aesthetics knowledge in E, and is supervised to build the proper mapping function to the high quality image domain. Effectively, the generator learns how to enhance the input image to make it visually more appealing.

6.2.2.3 Discriminator Network

The discriminator D, following the standard GANs protocol [34], distinguishes real images from the generated ones. The loss function L_D is defined as the binary cross-entropy:

$$\begin{aligned} L_D(y,t) &= -t \cdot \log[D(y)] - (1-t) \cdot \log[1 - D(y)] \\ s.t. \quad t &= \begin{cases} 1 & \text{if } y \text{ is a real image} \\ 0 & \text{if } y \text{ is a generated image} \end{cases} \end{aligned} \tag{6.3}$$

Note that when training the discriminator, the label for the generated image is 0. This is different when optimizing Equation 6.1, where the label for the generated image is set to 1. Such adversarial training setting [34] pushes the generator to produce more realistic images, along with the improvement of the discriminator.

6.2.2.4 Feature Extractor

The feature extractor is incorporated to constrain the content similarity between the input image and its enhancement result. A simple choice for incorporating such a constraint is to use a $L2$ difference between the input image and the output image. But, it is known that $L2$ loss is prone to produce blurry images. Therefore, we adopt the recent proposed perceptual loss [23], which is demonstrated to be effective in preserving the image content and not blurring the images.

The perceptual loss L_F is defined as:

$$L_F(y, \hat{y}) = \frac{1}{HWC} ||F(y) - F(\hat{y})||_2^2 \tag{6.4}$$

where $y, \hat{y}$ represents the source image and the target image, respectively. $F(y)$ extracts the feature representations from a certain layer of a pre-trained deep neural network. H, W, C represent the corresponding size of the feature representations. Minimizing L_F in Equation 6.4 encourages the output image to share the same content as the input image, while allowing for image enhancement operations like color, contrast and texture modifications [23].

6.2.2.5 Network Architecture

The generator is an encoder-decoder style generative network. The encoder part consists of eight blocks of Convolution-BatchNormalization-ReLu [179], except that batch normalization is not used in the first block. The output from the encoder is a 512-dimensional vector, which is then concatenated with the aesthetics score, forming the input to the decoder. The decoder architecture is the reverse of the encoder network, in which the

convolution layers are replaced with deconvolution. The final layer of the decoder is a deconvolution that maps to three channels, followed by a *tanh* activation. Inspired by the the "U-Net" architecture [180] and pix2pix network [55], we add additional skip connections between the encoder and decoder, as shown in Figure 6.4. Such skip connections help share information between the input image and the output image. The discriminator architecture is similar to the encoder of G, with the last convolution layer mapping to a single output, followed by a *sigmoid* activation. For the feature extractor, we extract feature representations from the *relu3_3* layer of the VGG-16 network [4] pre-trained on ImageNet [181].

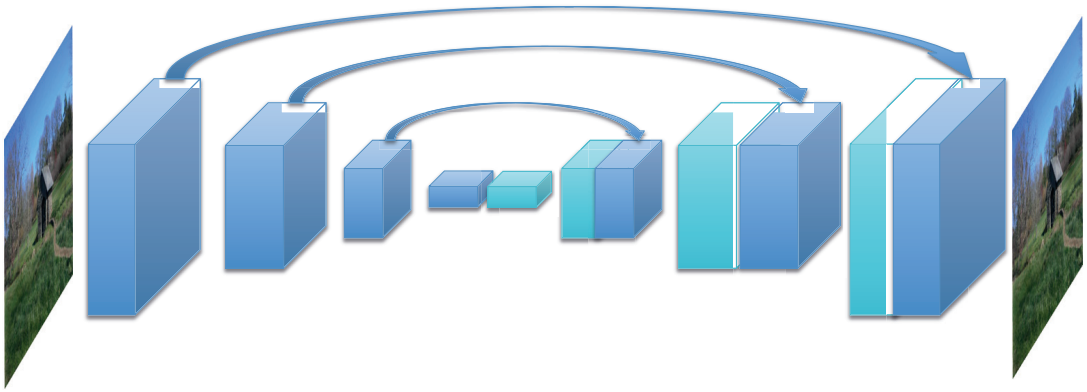


Figure 6.4: The "U-net" architecture with skip connections between the encoder and the decoder. The blue blocks are the encoding layers, and the green blocks are the decoding layers.

6.2.2.6 Training Protocol

In the network architecture, E and F are pre-trained networks. G and D are trained according to the standard adversarial training procedure [34]. Between every iteration of the D optimization, we train two iterations of G, as G needs to minimize the sum of three losses and is hence harder to train. Training D and G only requires a set of low aesthetics quality images, a set of high aesthetics quality images and their corresponding aesthetics scores. v in Equation 6.1 are the aesthetics scores of the high aesthetics quality images. This directs the generator to produce similar aesthetics levels as the real-world high aesthetics quality images. The whole training procedure is described in Algorithm 2.

When testing, an input image is fed into the generator, together with a random aesthetics score (we use 1 in all experiments as there is little difference in results as explained in Sec. 6.2.2.1). One forward pass of the generator produces the enhanced images.

Algorithm 2: Training for DEGANs

Data: a set of low aesthetics quality images $\{x_l\}$, a set of high aesthetics quality images $\{x_h\}$ and their aesthetics scores $\{v_h\}$.
Init: initialize G and D with Gaussian weights, E and F with pre-trained weights.
for *number of epochs* **do**
  for *number of batches per eopch* **do**
    Get: get a low aesthetics quality batch $\{x_l\}^B$, a high aesthetics quality batch $\{x_h\}^B$ together with the aesthetics scores batch $\{v_h\}^B$
    Enhance: produce a generated batch $\{\hat{x_g}\}^B$ from $\{x_l\}^B$ with current G
    Update: label $\{\hat{x_g}\}^B$ as 0, $\{x_h\}^B$ as 1, update D according to L_D in Equation 6.3
    Update: label $\{\hat{x_g}\}^B$ as 1, update G according to L_G in Equation 6.1
    Get: get a new low aesthetics quality batch $\{\hat{x}_l\}^B$, a new high aesthetics quality batch $\{\hat{x}_h\}^B$ together with the aesthetics scores batch $\{\hat{v}_h\}^B$
    Enhance: produce a generated batch $\{\hat{x_g}\}^B$ from $\{\hat{x}_l\}^B$ with current G
    Update: label $\{\hat{x_g}\}^B$ as 1, update G according to L_G in Equation 6.1
  end
end

6.3 Experiments

6.3.1 Setup

Datasets

The training data for DEGANs are collected from the AVA dataset [12], as introduced in Chapter 2. As the image enhancement operations might differ for different types of images [82], we train a separate generator for each of the following five categories: animal, cityscape, landscape, nature, and portrait. We choose these five categories as they are the most frequent five classes in the AVA dataset.. We rank the images according to their aesthetics scores in descending order, where the first 4000 images are taken as the high aesthetics image set and the last 4000 images as the low aesthetics set. We do not experiment on categories that have less than 8000 images. A random $80\% - 20\%$ split is performed to form the training data and the test data. We show sample high aesthetics quality and low aesthetics quality images in Figure 6.5.

Implementation Details

Training the domain encoder for an aesthetics regression task is explained in Chapter 3. For training DEGANs, all the images are resized to 256×256 and trained with a batch size of 1. With batch size 1, the batch normalization inside our network becomes "instance normalization" [182], which is demonstrated to be effective for image generation tasks. The DEGANs are trained with the Adam optimizer with a learning rate of 0.001 for 50

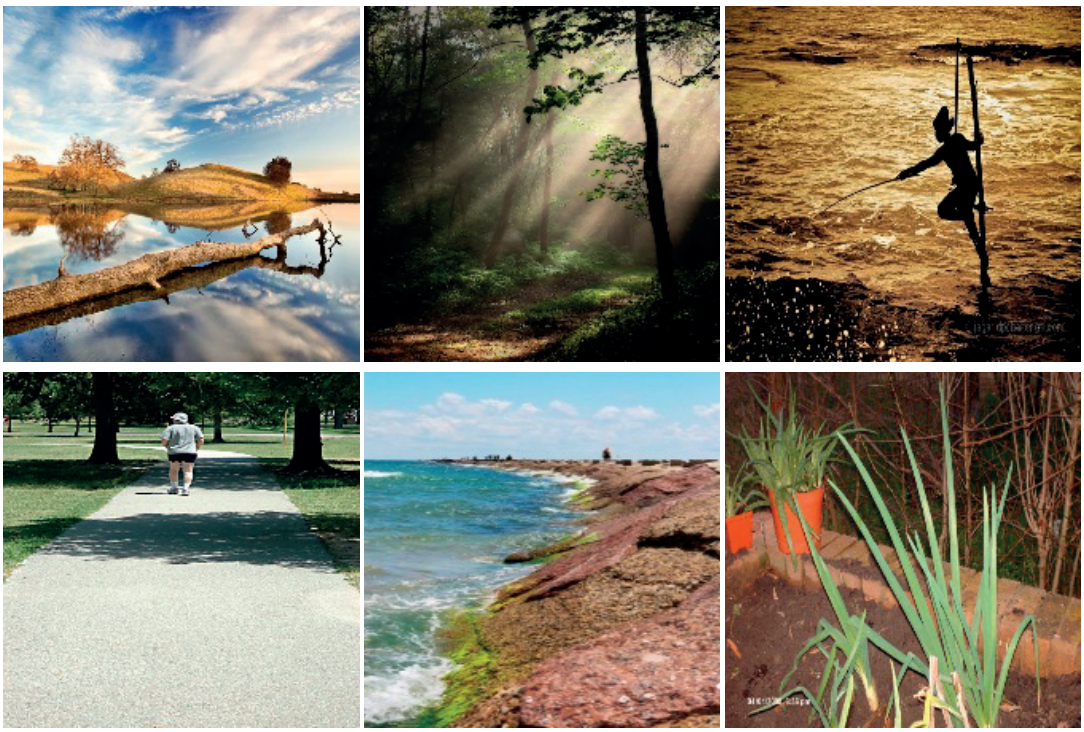


Figure 6.5: Sample images from AVA: the first row are the high aesthetics quality images, the second row are the low aesthetics quality ones. Note that we resize these images to be square due to the input constraint of neural networks.

epochs. The weights for L_D, L_E, L_F are set to 1, 10, 0.0003, considering the different scales of the losses. The algorithm is implemented in Keras with *Tensorflow* backend [183]. Training DEGANs takes 16 hours per category on a single GTX Titan X GPU.

6.3.2 Qualitative Comparison

For the image aesthetics enhancement task, the goal is to produce an image that has a better aesthetics quality, *i.e.*, is visually more appealing. Full reference image quality metrics, such as PSNR or SSIM, are not suitable for evaluating the enhancement results as no groundtruth results are available. Furthermore, as discussed in Chapter 2.2.1, our model is trained to improve image aesthetics, while metrics like PSNR and SSIM focus on modeling image quality. Therefore, we judge the aesthetics enhancement results through qualitative comparisons.

Image enhancement operations can be treated as translation functions from the source domain (low quality) to the target image domain (high quality). In this sense, we compare with the state-of-the-art image translation method, pix2pix [55], as discussed in Chapter 2. Input-groundtruth image pairs are needed for training the original pix2pix network, but

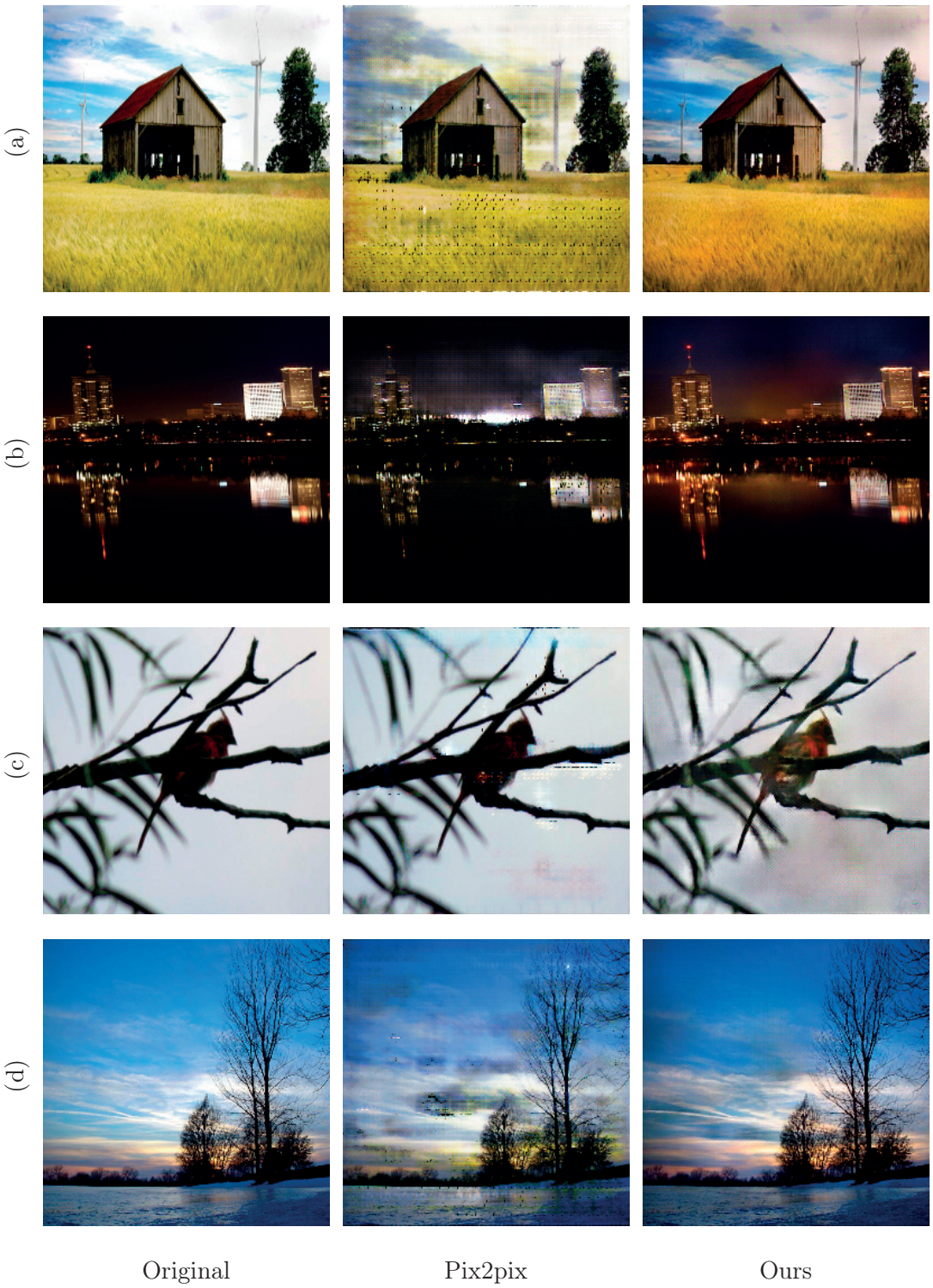


Figure 6.6: Qualitative comparison between the pix2pix [55] results and our aesthetics enhancement results, best viewed on screen.

Figure 6.7: Qualitative comparison between the pix2pix [55] results and our aesthetics enhancement results, best viewed on screen.

Figure 6.8: Qualitative comparison between the pix2pix [55] results and our aesthetics enhancement results, best viewed on screen.

they are not available. Hence, we modify the content loss in pix2pix to be computed with respect to the input image, which potentially models the content similarity between the input image and the result image. This is similar to the perceptual loss in our DEGANs. The generator and discriminator architectures in pix2pix are set the same as those in our model. Figure 6.6, Figure 6.7 and Figure 6.8 show the qualitative comparisons between the pix2pix results and our results (on the test set).

Our network has the domain encoder to learn the image aesthetics knowledge, whereas the pix2pix model does not. Therefore, it fails to perform image enhancement operations. Moreover, the pix2pix uses L1 loss to model content similarity, which leads to noticeable artifacts in the results, like the black dots on almost every result and the white bar artifacts in Figure 6.6b. Our DEGANs, using the domain encoder to transfer the aesthetics knowledge to the generator, produces results that have an aesthetics quality better than the input images, as well as significantly less artifacts than pix2pix.

In Figure 6.6a and 6.7b, our DEGANs modify the sky to be more blue, as well as make the color of the ground to be more saturated, resulting in visually more pleasing images. The lighting effect in Figure 6.6b is enhanced to increase the contrast of the buildings with respect to their background. In Figure 6.6c, our method significantly brightens the

bird, rendering the details of the main object more visible. The color of the sky and the sunset glow are modified to be more vivid in Figure 6.6d and Figure 6.7c, resulting in better visual quality. In Figure 6.7a, our DEGANs reduce the lightness of the image since the original image is over-exposed. In Figure 6.7d and 6.8a, the images are improved to have more pleasing colors. For Figure 6.8b, our method darkens the surroundings of the image and brightens the center, thus enhancing the contrast for the main object. In all examples, our method edits the input image to achieve a better aesthetics quality without using groundtruth images for training. The proper image enhancement operations are automatically inferred and applied, according to the input image.

(a) Input images (b) Pix2pix (c) Ours

Figure 6.9: Sample failure cases of our DEGANs.

We also show some failure cases in Figure 6.9. In the first row, our DEGANs make the center window colorful, which appears to be unnatural. For the second row, both our result and the pix2pix result are almost the same as the input image, which implies that the DEGANs do not know what operations to apply to enhance this image. Note that certain level of ringing artifacts can be observed in both pix2pix and our results. For example, in the first row of Figure 6.6, obvious filter artifacts can be observed in the cloud regions in both pix2pix and our results. This is a common problem for many generative models, as the generative models need to produce high-resolution images using deconvolution layers. Since the training set contain a wide variety of images with different textures and shapes, the deconvolution layers need to model and store all these information, rendering it difficult for the deconvolution kernels to converge. Including

more data or improve the network architecture may be required to reduce the artifacts.

6.3.3 Effect of Different Losses

Our DEGANs use the sum of three losses to supervise the generator. In this section, we investigate the effect of each individual loss on the output results. By setting the weight of a loss in Equation 6.1 to 0, we compute the performance difference against the full-model results in order to analyze its influence.

6.3.3.1 The Effect of D

By setting w_D to 0 in Equation 6.1, the generator G is trained to produce results that have the target aesthetics quality, as well as the content of the input images. Ringing artifacts can be observed from the results without D, as seen in Figure 6.10c and 6.10h. These artifacts (Refer to the inceptionism effect [184]) manage to fool the domain encoder to produce high aesthetics scores [185]. Incorporating the discriminator network would significantly reduce the amount of such artifacts, as these artifacts make it easy for D to distinguish between real and generated images. Through adversarial training, the generator learns to avoid generating these artifacts, hence is pushed to learn the actual aesthetics knowledge from E.

6.3.3.2 The Effect of E

The domain encoder E carries the knowledge about aesthetics. Without E, the GANs cannot learn how to enhance the input image for better aesthetics quality. As shown in Figure 6.10d and 6.10i, the output image is almost identical to the input image. Note that this result is similar to the result that uses only the perceptual loss. However, as pointed out in Sec. 6.3.3.1, the discriminator D is needed to suppress the artifacts in the results and to ensure that G learns the actual aesthetics knowledge.

6.3.3.3 The Effect of F

We build the perceptual loss on top of the feature extractor F to ensure the content similarity between the input image and the output image. Without F, the input image and the output image are not linked in content, rendering the output image unrecognizable, as shown in Figure 6.10e and 6.10j. Note that Figure 6.10e and 6.10j are almost identical. This is known as the mode collapse [186] problem of GANs, where all input images map to the same output image and the optimization fails to make progress. The feature extractor F is thus needed to prevent such problem.

(a) Input Image (b) Ours_fullmodel

(c) Ours_noD (d) Ours_noE (e) Ours_noF

(f) Input image (g) Ours_fullmodel

(h) Ours_noD (i) Ours_noE (j) Ours_noF

Figure 6.10: Effects of different loss functions on the enhancement results.

6.3.4 Comparison with Traditional Method

As can be seen in Figure 6.6, 6.7 and 6.8, our network is trained to perform relatively subtle modifications to the input image for better aesthetics, such as color and contrast adjustment. In this sense, we further compare with a traditional color enhancement method [124] (*Histeq*), which defines a histogram equalization process that balances the color histogram of an image. The corresponding results can be seen in Figure 6.11. Since Histeq applies histogram equalization operations to all images with no understandings of the image content, the corresponding results thus often contain certain artifacts. For

(a) Original (b) Histeq [124] (c) Ours

Figure 6.11: Comparison with traditional color enhancement methods.

instance, in the first row of Figure 6.11, our method enhances the lighting effect of the buildings without affecting the dark background. Histeq instead increases the brightness of the whole image, resulting in over-exposure of the buildings. In the second row, our network makes the sky more blue and the flower more yellow, while not affecting the green mountains. The result of Histeq modifies the colors of the whole image but does not improves its aesthetics.

6.3.5 Quantitative Comparison

Our DEGANs focus on improving the aesthetics level of the input image. Quantitatively evaluation of aesthetics is a difficult problem as explained in Chapter 3. There are no existing metrics suitable for this task. Therefore, we use the pre-trained aesthetics predictor from Chapter 3 to evaluate the aesthetics. In fact, we use the domain encoder network as described in Section 6.2.1 for its superior performance and high speed. We build a random test set with 800 low aesthetics images and generate the corresponding enhanced results using our generator (v is set to 1 for all test images). We also generate the results for Histeq and pix2pix We then measure the average aesthetics score of each set, as shown in Table 6.1. The average aesthetics score for the original images is 0.45. The traditional color enhancement method (Histeq) only slightly improves this score by 0.2, because Histeq has no understanding of the content of the image, thus often resulting in noticeable artifacts as shown in Fig. 6.11. Pix2pix achieves better enhancement results than Histeq but worse results than our method, because they do not have domain encoder network that implicitly model the aesthetics knowledge. Our method has the highest gain in the average aesthetics score. Note that the average aesthetics score for the enhanced results is still not very high. We think the main reason is the large variety of aesthetically appealing images, rendering it difficult for the GANs to learn the complete set of enhancement operations. Our network learns to apply subtle changes to the images that can result in limited improvement of aesthetics. It requires further research efforts to incorporate more sophisticated operations into the system.

Table 6.1: Average aesthetics score of the enhanced results from different methods. The aesthetics score is predicted by the domain encoder network, which is in the range of 0 (low aesthetics) to 1 (high aesthetics).

Original images	Histeq	Pix2pix	Ours
0.45	0.47	0.50	0.58

6.4 Conclusion

In this chapter, we use generative adversarial networks for automatic image enhancement. Due to the lack of paired training samples, we propose a novel DEGANs framework in order to learn image enhancement operations. Our framework first trains a domain encoder

for an image aesthetics prediction task, where only aesthetics scores are required. This task enriches the domain encoder of knowledge about image aesthetics. Such knowledge is then used in the DEGANs to supervise the generator. The generator thus learns the proper image enhancement functions that edit the input images to make them visually more appealing.

Our DEGANs are the first attempt to learn image enhancement operations in a fully-automatic manner from the training data. Although producing promising results, we observe several problems with the current system. Due to the lack of training data, our high quality images cannot cover the whole range of the aesthetics domain. Many rarely seen or unseen objects and scenes can lead to noticeable artifacts in the generated results. Building a larger set of training images for each category could benefit this task. Furthermore, our DEGANs can only perform tone adjustment to the images, such as color and contrast modification. It is desirable to have an automatic system that can perform more sophisticated operations, such as content removal or composition adjustment.

7 Conclusion

7.1 Thesis Summary

In this thesis, we developed several useful techniques to ease users' efforts in obtaining aesthetically pleasing images, including an aesthetics predictor that helps rank images, and three image enhancement algorithms that edit images to be visually more appealing. All the proposed algorithms involve neural networks to some extent. Therefore, in Chapter 2, we first reviewed several well-known neural network architectures, including CNNs, FCN and GANs, and their corresponding training techniques. Afterwards, we introduced the field of computational image aesthetics. Early algorithms in this field use handcrafted features or heuristic rules to estimate image aesthetics, whereas recent approaches all adopt neural networks. Furthermore, we inspected the related algorithms for semantic segmentation and image enhancement, respectively, in the last two sections.

In Chapter 3, we presented the details about our aesthetics predictor. By analyzing the most widely used aesthetics dataset, we revealed the unbalanced distribution of aesthetics scores in the training set, which would introduce bias to the networks. Therefore, we introduced sample weights into our loss function to reduce the influence of such bias. Furthermore, we proposed to predict the histograms of user labels, in addition to the average labels. The predicted histogram not only reveals the average aesthetics quality of an image, but also indicates the difficulty of such an estimation. The quantitative comparison on the AVA dataset proved the state-of-the-art performance of our algorithm. Using our aesthetics predictor, we built an automatic image cropping algorithm as described in Chapter 3.4. Our cropping algorithm automatically extracts an aesthetically pleasing patch from the input image, according to a size template. We validated the performance by a pyschophysical experiment.

We described another image enhancement algorithm for color re-rendering, in Chapter 4 and Chapter 5. Users specify a keyword for an input image and our algorithm edits the corresponding colors to be visually more pleasing. To locate the keyword-related

regions, we first developed a weakly supervised semantic segmentation algorithm. Our segmentation algorithm does not require pixel-wise human annotations for training. Instead, three sets of web images are collected to supervise our segmentation networks. Through a three-stage training pipeline, our segmentation networks are progressively refined, achieving state-of-the-art performance on the semantic segmentation benchmark. After obtaining the keyword-related regions, we then applied color re-rendering operations on these regions. The color re-rendering operations are defined as tone-mapping functions, which are derived from the pre-learned correlations between keywords and colors. Qualitative comparisons and a pyschophysical experiment demonstrate the performance of our color re-rendering system.

In Chapter 6, we developed a fully-automatic image enhancement algorithm using GANs. We proposed a novel DEGANs architecture to learn the proper enhancement functions, without using the groundtruth images. The image aesthetics knowledge is pre-encoded into a domain encoder network through the aesthetics prediction task. Such knowledge is then used to supervise the generator of the DEGANs, together with another two networks that force the generated images to be natural and content-wise similar to the input images. Given an input image, this algorithm automatically enhances it to be visually more appealing.

7.2 Future Work

In this thesis, we proposed one aesthetics predictor and three image enhancement techniques. More explorations can be conducted both in computational aesthetics and in image enhancement.

In Chapter 3, to obtain the fixed size input to the neural networks, we used a simple preprocessing method (global resizing). More sophisticated methods could be incorporated to improve the performance. For example, it is shown in [93,141] that the combination of multiple views of an image (global resizing result and multiple local patches) could benefit the aesthetics prediction task. Additionally, image aesthetics are also highly correlated with several image attributes [78], such as depth-of-field and rule-of-thirds. It would be advantageous to integrate these attributes with the aesthetics predictor. Attempts have been made in this direction [78,97,149], where the attributes are combined with aesthetics in a multi-task learning framework. However, multi-task learning is very label-hungry as each image needs to have not only aesthetics labels but also image attributes labels. Developing a more efficient method to use image attributes for aesthetics estimation could be beneficial.

Our aesthetics network focuses on modeling the average aesthetics scores of the crowd. However, as image aesthetics is a relatively subjective concept, modeling personal preferences of aesthetics can also be very useful in real-world applications. One example

would be recommendation systems that can adjust their content based on users' aesthetics preferences. Relatively few research efforts have been devoted in this area.

For image enhancement, our cropping algorithm mainly considers the aesthetics point-of-view. Typical failure cases were observed during our experiments, as shown in Figure 7.1. The main objects of the images are either excluded or partially cut in the generated crops. This is due to the fact that semantics, such as completeness of objects and saliency, are not considered during the cropping procedure. Designing systems that integrate aesthetics with semantics could be a promising research direction that might produce better cropping results.

(c) Original images (d) Cropping results

Figure 7.1: Failure cases of the cropping technique. The main object is either cut out for cut through in the result crop.

From our keyword-based color re-rendering algorithm, several interesting research directions can be derived. One possibility is to extend keyword supervision to phrases, or even sentences. Further exploration in this line could be to incorporate speech in the pipeline, thus being more interactive and user-friendly for real-world applications. Although the current speech recognition system [187] produces reliable results, how to

interpret the recognized phrases/sentences, instead of keywords, for image editing and enhancement still remains unsolved. This is a challenging task because the system needs to semantically understand the phrases or sentences, then build computational models that map the semantic meanings to image features. Advances in natural language processing, such as word2vec [188], might help achieve this goal.

Another possible direction is to extend the image enhancement operations from color re-rendering to more sophisticated ones, such as image composition adjustment and image inpainting. Using keywords/phrases/sentences to supervise other image-related operations, such as organizing or archiving images, are also worth investigation. In general, many interesting projects can be conducted to bridge the gap between natural language processing and image processing/computer vision.

For fully-automatic image enhancement, our DEGANs can perform subtle changes to the images, such as color and contrast adjustment. It requires further research efforts to incorporate more sophisticated operations into the system. To achieve this, building a larger set of images with varying aesthetics quality might be useful as the first step.

Bibliography

[1] *https://www.flickr.com/photos/franckmichel/6855169886.*

[2] *https://www.statista.com/.*

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

[7] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, "Deep face recognition," in *Proceedings of the British Machine Vision Conference*, vol. 1, p. 6, 2015.

[8] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Advances in Neural Information Processing Systems*, pp. 1790–1798, 2014.

[9] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, "Learning to deblur," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1439–1451, 2016.

[10] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proceedings of European Conference on Computer Vision*, pp. 184–199, 2014.

[11] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1646–1654, 2016.

[12] N. Murray, L. Marchesotti, and F. Perronnin, "AVA: A large-scale database for aesthetic visual analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2408–2415, 2012.

[13] A. Maslennikova and V. Vezhnevets, "Interactive local color transfer between images," *Graphicon Conference*, 2007.

[14] G. P. Laput, M. Dontcheva, G. Wilensky, W. Chang, A. Agarwala, J. Linder, and E. Adar, "Pixeltone: a multimodal interface for image editing," in *SIGCHI Conference on Human Factors in Computing Systems*, pp. 2185–2194, 2013.

[15] L. Xu, Q. Yan, and J. Jia, "A sparse control model for image and video editing," *ACM Transactions on Graphics*, vol. 32, no. 6, p. 197, 2013.

[16] X. Chen, D. Zou, Q. Zhao, and P. Tan, "Manifold preserving edit propagation," *ACM Transactions on Graphics*, vol. 31, no. 6, p. 132, 2012.

[17] A. Lindner, A. Shaji, N. Bonnier, and S. Süsstrunk, "Joint statistical analysis of images and keywords with applications in semantic image enhancement," in *ACM International Conference on Multimedia*, pp. 489–498, 2012.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[19] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.

[20] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

[21] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *arXiv preprint arXiv:1612.01105*, 2016.

[22] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[23] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proceedings of European Conference on Computer Vision*, pp. 694–711, 2016.

[24] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[25] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[27] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[28] J. Botha and P. Blunsom, "Compositional morphology for word representations and language modelling," in *Proceedings of the International Conference on Machine Learning*, pp. 1899–1907, 2014.

[29] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of the International Conference on Computational Linguistics*, pp. 69–78, 2014.

[30] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3223–3230, 2015.

[31] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, 2016.

[32] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in Neural Information Processing Systems*, pp. 127–135, 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[35] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model.," *Interspeech*, vol. 2, p. 3, 2010.

[36] Y. LeCun and M. Ranzato, "Deep learning tutorial," in *Tutorials in International Conference on Machine Learning*, 2013.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[38] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning*, pp. 807–814, 2010.

[39] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1665–1672, 2011.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proceedings of European Conference on Computer Vision*, pp. 346–361, 2014.

[41] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015.

[42] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433, 2015.

[43] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, *et al.*, "From captions to visual concepts and back," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1473–1482, 2015.

[44] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI Conference on Artificial Intelligence*, 2017.

[45] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Proceedings of European Conference on Computer Vision*, pp. 297–312, 2014.

[46] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[47] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[48] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[49] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint arXiv:1609.04802*, 2016.

[50] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proceedings of the International Conference on Machine Learning*, vol. 3, 2016.

[51] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[52] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv preprint arXiv:1511.05440*, 2015.

[53] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems*, pp. 1486–1494, 2015.

[54] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu, "Unsupervised diverse colorization via generative adversarial networks," *arXiv preprint arXiv:1702.06674*, 2017.

[55] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[56] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the International Conference on Machine learning*, pp. 1096–1103, 2008.

[57] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015.

[58] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

[60] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[61] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[62] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[63] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[64] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[65] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[66] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[67] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[68] L. Prechelt, "Early stopping-but when?," *Neural Networks: Tricks of the Trade*, pp. 553–553, 1998.

[69] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, 2006.

[70] Y. Yuan, Q. Guo, and X. Lu, "Image quality assessment: a sparse learning way," *Neurocomputing*, vol. 159, pp. 227–241, 2015.

[71] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference image quality assessment in the spatial domain," *IEEE Transactions on Image Processing*, vol. 21, no. 12, pp. 4695–4708, 2012.

[72] L. Kang, P. Ye, Y. Li, and D. Doermann, "Convolutional neural networks for no-reference image quality assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1733–1740, 2014.

[73] S. Bosse, D. Maniry, T. Wiegand, and W. Samek, "A deep neural network for image quality assessment," in *Proceedings of IEEE International Conference on Image Processing*, pp. 3773–3777, 2016.

[74] Y. Liang, J. Wang, X. Wan, Y. Gong, and N. Zheng, "Image quality assessment using similar scene as reference," in *Proceedings of European Conference on Computer Vision*, pp. 3–18, 2016.

[75] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Studying aesthetics in photographic images using a computational approach," in *Proceedings of European Conference on Computer Vision*, pp. 288–301, 2006.

[76] X. Sun, H. Yao, R. Ji, and S. Liu, "Photo assessment based on computational visual attention model," in *Proceedings of the ACM International Conference on Multimedia*, pp. 541–544, 2009.

[77] X. Tang, W. Luo, and X. Wang, "Content-based photo quality assessment," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 1930–1943, 2013.

[78] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes, "Photo aesthetics ranking network with attributes and content adaptation," in *Proceedings of European Conference on Computer Vision*, pp. 662–679, Springer, 2016.

[79] Y. Luo and X. Tang, "Photo and video quality evaluation: Focusing on the subject," in *Proceedings of European Conference on Computer Vision*, pp. 386–399, 2008.

[80] L.-K. Wong and K.-L. Low, "Saliency-enhanced image aesthetics class prediction," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 997–1000, 2009.

[81] S. Bhattacharya, R. Sukthankar, and M. Shah, "A framework for photo-quality assessment and enhancement based on visual aesthetics," in *Proceedings of the 18th ACM International Conference on Multimedia*, pp. 271–280, 2010.

[82] F. Simond, N. Arvanitopoulos Darginis, and S. Süsstrunk, "Image aesthetics depends on context," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 3788–3792, 2015.

[83] L. Marchesotti and F. Perronnin, "Learning beautiful (and ugly) attributes," in *Proceedings of the British Machine Vision Conference*, 2013.

[84] L. Marchesotti, N. Murray, and F. Perronnin, "Discovering beautiful attributes for aesthetic image analysis," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 246–266, 2014.

[85] L. Marchesotti, F. Perronnin, D. Larlus, and G. Csurka, "Assessing the aesthetic quality of photographs using generic image descriptors," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1784–1791, 2011.

[86] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[87] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22, 2004.

[88] G. Csurka and F. Perronnin, "Fisher vectors: Beyond bag-of-visual-words image representations," in *International Conference on Computer Vision, Imaging and Computer Graphics*, pp. 28–42, 2011.

[89] C. Cortes and V. Vapnik, "Support vector machine," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[90] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[91] Z. Dong, X. Shen, H. Li, and X. Tian, "Photo quality assessment with dcnn that understands image well," in *Proceedings of the International Conference on Multimedia Modeling*, pp. 524–535, 2015.

[92] W. Wang, M. Zhao, L. Wang, J. Huang, C. Cai, and X. Xu, "A multi-scene deep learning model for image aesthetic evaluation," *Signal Processing: Image Communication*, vol. 47, pp. 511–518, 2016.

[93] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Z. Wang, "Deep multi-patch aggregation network for image style, aesthetics, and quality estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 990–998, 2015.

[94] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, "Rating image aesthetics using deep learning," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2021–2034, 2015.

[95] L. Mai, H. Jin, and F. Liu, "Composition-preserving deep photo aesthetics assessment," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 497–506, 2016.

[96] Y. Kao, K. Huang, and S. Maybank, "Hierarchical aesthetic quality assessment using deep convolutional neural networks," *Signal Processing: Image Communication*, vol. 47, pp. 500–510, 2016.

[97] Y. Kao, R. He, and K. Huang, "Visual aesthetic quality assessment with multi-task deep learning," *arXiv preprint arXiv:1604.04970*, 2016.

[98] Y. Kao, C. Wang, and K. Huang, "Visual aesthetic quality assessment with a regression model," in *Proceedings of the IEEE International Conference on Image Processing*, pp. 1583–1587, 2015.

[99] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of European Conference on Computer Vision*, pp. 740–755, 2014.

[100] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 991–998, 2011.

[101] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation," *arXiv preprint arXiv:1611.06612*, 2016.

[102] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[103] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *Proceedings of European Conference on Computer Vision*, 2016.

[104] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[105] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[106] P. Vernaza and M. Chandraker, "Learning random-walk label propagation for weakly-supervised semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7158–7166, 2017.

[107] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[108] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1635–1643, 2015.

[109] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele, "Simple does it: weakly supervised instance and semantic segmentation," *arXiv preprint arXiv:1603.07485*, 2016.

[110] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM Transactions on Graphics*, vol. 23, pp. 309–314, 2004.

[111] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1713–1721, 2015.

[112] D. Pathak, E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional multi-class multiple instance learning," in *Proceedings of the International Conference on Learning Representations*, 2015.

[113] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1796–1804, 2015.

[114] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Advances in Neural Information Processing Systems*, pp. 561–568, 2002.

[115] F. Saleh, M. S. A. Akbarian, M. Salzmann, L. Petersson, S. Gould, and J. M. Alvarez, "Built-in foreground/background prior for weakly-supervised semantic segmentation," in *Proceedings of European Conference on Computer Vision*, pp. 413–432, 2016.

[116] W. Shimoda and K. Yanai, "Distinct class-specific saliency maps for weakly supervised semantic segmentation," in *Proceedings of European Conference on Computer Vision*, pp. 218–234, 2016.

[117] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *Proceedings of European Conference on Computer Vision*, 2016.

[118] X. Qi, Z. Liu, J. Shi, H. Zhao, and J. Jia, "Augmented feedback in semantic segmentation under image level supervision," in *Proceedings of European Conference on Computer Vision*, pp. 90–105, 2016.

[119] Y. Wei, X. Liang, Y. Chen, Z. Jie, Y. Xiao, Y. Zhao, and S. Yan, "Learning to segment with image-level annotations," *Pattern Recognition*, 2016.

[120] S. J. Oh, R. Benenson, A. Khoreva, Z. Akata, M. Fritz, and B. Schiele, "Exploiting saliency for object segmentation from image level labels," *arXiv preprint arXiv:1701.08261*, 2017.

[121] S. Hong, D. Yeo, S. Kwak, H. Lee, and B. Han, "Weakly supervised semantic segmentation using web-crawled videos," *arXiv preprint arXiv:1701.00352*, 2017.

[122] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 328–335, 2014.

[123] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3286–3293, 2014.

[124] R. Hummel, "Image enhancement by histogram transformation," *Computer Graphics and Image Processing*, vol. 6, no. 2, pp. 184–195, 1977.

[125] A. Polesel, G. Ramponi, and V. J. Mathews, "Image enhancement via adaptive unsharp masking," *IEEE transactions on Image Processing*, vol. 9, no. 3, pp. 505–510, 2000.

[126] D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu, "Color harmonization," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 624–630, 2006.

[127] C. Sauvaget and V. Boyer, "Harmonic colorization using proportion contrast," in *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pp. 63–69, 2010.

[128] B. Wang, Y. Yu, T.-T. Wong, C. Chen, and Y.-Q. Xu, "Data-driven image color theme enhancement," *ACM Transactions on Graphics*, vol. 29, no. 6, p. 146, 2010.

[129] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein, "Palette-based photo recoloring," *ACM Transactions on Graphics*, vol. 34, no. 4, p. 139, 2015.

[130] G. Ciocca, C. Cusano, F. Gasparini, and R. Schettini, "Content aware image enhancement," *Congress of the Italian Association for Artificial Intelligence*, pp. 686–697, 2007.

[131] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001.

[132] X. Xiao and L. Ma, "Color transfer in correlated color space," in *Proceedings of the ACM International Conference on Virtual Reality Continuum and its Applications*, pp. 305–309, 2006.

[133] Y.-W. Tai, J. Jia, and C.-K. Tang, "Local color transfer via probabilistic segmentation by expectation-maximization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 747–754, 2005.

[134] F. Wu, W. Dong, Y. Kong, X. Mei, J.-C. Paul, and X. Zhang, "Content-based colour transfer," *Computer Graphics Forum*, vol. 32, no. 1, pp. 190–203, 2013.

[135] B. Wang, Y. Yu, and Y.-Q. Xu, "Example-based image color and tone style enhancement," *ACM Transactions on Graphics*, vol. 30, no. 4, p. 64, 2011.

[136] S. B. Kang, A. Kapoor, and D. Lischinski, "Personalization of image enhancement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1799–1806, 2010.

[137] X. Wang, J. Jia, and L. Cai, "Affective image adjustment with a single word," *The Visual Computer*, vol. 29, no. 11, pp. 1121–1133, 2013.

[138] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, pp. 341–349, 2012.

[139] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[140] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, "Automatic photo adjustment using deep neural networks," *ACM Transactions on Graphics*, vol. 35, no. 2, p. 11, 2016.

[141] X. Lu, Z. Lin, H. Jin, X. Yang, Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, "Rapid: Rating pictorial aesthetics using deep learning," in *Proceedings of the ACM International Conference on Multimedia*, pp. 457–466, 2014.

[142] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (mos) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 2016.

[143] O. Wu, W. Hu, and J. Gao, "Learning to predict the perceived visual quality of photos," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 225–232, 2011.

[144] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[145] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[146] J. M. Joyce, "Kullback-leibler divergence," in *International Encyclopedia of Statistical Science*, pp. 720–722, 2011.

[147] B. Fuglede and F. Topsoe, "Jensen-shannon divergence and hilbert space embedding," in *Proceedings of International Symposium on Information Theory*, p. 31, 2004.

[148] X. Jin, L. Wu, C. Song, X. Li, G. Zhao, S. Chen, J. Chi, S. Peng, and S. Ge, "Predicting aesthetic score distribution through cumulative jensen-shannon divergence," *arXiv preprint arXiv:1708.07089*, 2017.

[149] Y. Deng, C. C. Loy, and X. Tang, "Image aesthetic assessment: An experimental survey," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 80–106, 2017.

[150] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *Proceedings of International Conference on Learning Representations*, 2015.

[151] G. Lin, C. Shen, I. Reid, *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[152] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015.

[153] G. Bertasius, J. Shi, and L. Torresani, "Semantic segmentation with boundary neural fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[154] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1742–1750, 2015.

[155] N. Pourian, S. Karthikeyan, and B. Manjunath, "Weakly supervised graph based semantic segmentation by learning communities of image-parts," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1359–1367, 2015.

[156] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, Y. Zhao, and S. Yan, "STC: a simple to complex framework for weakly-supervised semantic segmentation," *arXiv preprint arXiv:1509.03150*, 2015.

[157] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems*, pp. 109–117, 2011.

[158] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3166–3173, 2013.

[159] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometry consistency for large scale image search-extended version," in *Proceedings of European Conference on Computer Vision*, 2008.

[160] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, 2016.

[161] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014.

[162] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1939–1946, 2013.

[163] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1943–1950, 2010.

[164] A. Joulin, F. Bach, and J. Ponce, "Multi-class cosegmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 542–549, 2012.

[165] X. Chen, A. Shrivastava, and A. Gupta, "Enriching visual knowledge bases via object discovery and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2027–2034, 2014.

[166] M. Najibi, M. Rastegari, and L. S. Davis, "G-cnn: an iterative grid based object detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[167] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *ACM International Conference on Multimedia Information Retrieval*, pp. 39–43, 2008.

[168] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[169] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Proceedings of European Conference on Computer Vision*, pp. 649–666, 2016.

[170] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[171] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

[172] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *Proceedings of European Conference on Computer Vision*, pp. 702–716, 2016.

[173] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Proceedings of European Conference on Computer Vision*, pp. 597–613, 2016.

[174] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *Proceedings of European Conference on Computer Vision*, pp. 318–335, 2016.

[175] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, "Pixel-level domain transfer," in *Proceedings of European Conference on Computer Vision*, pp. 517–532, 2016.

[176] L. Meylan and S. Susstrunk, "High dynamic range image rendering with a retinex-based adaptive filter," *IEEE Transactions on Image Processing*, vol. 15, no. 9, pp. 2820–2830, 2006.

[177] G. Eilertsen, R. Mantiuk, and J. Unger, "A comparative review of tone-mapping algorithms for high dynamic range video," in *Computer Graphics Forum*, vol. 36, pp. 565–592, 2017.

[178] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint arXiv:1610.02357*, 2016.

[179] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[180] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, 2015.

[181] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[182] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.

[183] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[184] "https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html."

[185] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

[186] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[187] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the International Conference on Machine Learning*, pp. 173–182, 2016.

[188] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

Bin Jin

BC 318, Station 14
Lausanne, 1015, Switzerland
+41 78 935 72 28
jinbin1991@gmail.com
06/22/1991, Chinese
Permit B of Switzerland

Strengths:
- Machine learning, deep learning and computer vision specialist
- Extensive experience in data analysis and pattern recognition
- Strong analytical thinking and problem solving skills
- Quick learner and good self-management

Education

2012–2018 **PhD** in Computer Science, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne (**EPFL**), Switzerland (Expected graduation date: **January 2018**).

2008–2012 Bsc in Communication Systems, School of Communication Systems and Electrical Engineering, Shanghai JiaoTong University, China.

Experience

2012–Present Research Assistant, *Image and Visual Representation Lab*, EPFL, Switzerland.
- 5 years of experience in machine learning, deep learning, computer vision and data analysis fields.
- Solved several data mining and pattern discovery projects, always quickly grasped the gist of new problems and proposed solutions with state-of-the-art results.
- Developed machine learning and deep learning algorithms that handle *vast dataset (millions of images)*.
- Extensive experience of programming in Python, Matlab and C/C++.

May–Sep 2017 Intern, ***Facebook***, Seattle, USA.
- Worked in the **Applied Machine Learning** group.
- Collected data and built machine learning, deep learning algorithms to discover the patterns in the dataset.
- Quickly learned the new platforms and tools. Produced useful results since the first week.
- Worked independently with self-motivation.
- Well communicated with several co-workers in geographically distributed settings.

Sep–Dec 2014 Intern, *Océ*, Paris, France.
- Worked in the *R&D* department of the company.
- Efficiently collaborated with several co-workers, implemented tools that significantly enhanced the visualization and manipulation of images.

2010–2011 Research Assistant, *Video & Image Processing Lab*, Shanghai, China.
- Enthusiastic about new technology, published several papers on different problems.

Skills

Programming Languages **Python**(very good), Matlab(very good), C(good), C++(good), Bash Scripting(good), Html(basic), PHP(basic), Javascript(basic), VHDL(basic)

Technical Tools scikit-learn, Keras, jupyter notebook, Matlab, Theano, Tensorflow, Caffe, Caffe2, OpenCV, vl_feat, Xcode, Visual Studio, LaTeX, Microsoft Office, Git
-In-depth understanding of machine learning and deep learning algorithms and packages.
-Extensive experience in applying machine learning and deep learning techniques for data analysis and pattern recognition.
-In-depth understanding of computer vision and image processing algorithms and techniques.

Operating Systems Linux, MacOS, Windows.
-Years of experience in working on Linux servers with GPU computation.

Selected Projects

2012–Present Research Assistant, *Image and Visual Representation Lab*, EPFL, Switzerland.

- *Image semantic segmentation*: I developed an deep learning algorithm for image segmentation. My algorithm requires no user annotations for training while still achieving state-of-the-art performance. I successfully built a large image dataset by taking images from different sources. A novel neural network architecture is proposed to discover the object patterns in the unlabeled dataset. **This work is published at CVPR (the best conference in this field)**.
- *Image aesthetics prediction*: I applied machine learning and deep learning techniques for the problem of image aesthetics prediction. After statistically analyzing the training data, I proposed a novel technique to overcome the bias in the dataset. I implemented a convolutional neural network that improves the aesthetics prediction accuracy by 32 percent. *Theano* and *Keras* packages are used for this project. **This work is bought by CliqOrange**.
- **Face saliency**: I developed a novel algorithm to model the saliency of different faces in the images. I implemented a crowd-sourcing website using Html, Javascript and PHP, and statistically analyzed the user data with a multi-dimensional Gaussian model. This method produces more accurate saliency estimation for face images and won the **best paper runner-up award at HVEI 2015**.

2010–2011 Research Assistant, *Video & Image Processing Lab*, Shanghai, China.

- *Images retrieval*: I built statistical models for the users' expectation on the layout of the keywords. These models are used to re-rank the image retrieval outputs, leading to visually more consistent results.

Honours and Awards

2016 EPFL IC Teaching Assistant Award

2015 HVEI Best Paper Runner-up Award

2012 EPFL EDIC fellowship

2009, 2010 2nd Prize for Academic Excellence by Shanghai Jiao Tong University, twice

2009, 2010 The Jidian Electron Scholarship, twice

Languages

English Working proficiency, $100/120$ in *TOEFL* test, 2011

Chinese Mother tongue

French A1, willing to learn

German willing to learn

Selected Publications

2017 **B. Jin**, S. Süsstrunk, "Preserving Perceptual Contrast in Decolorization with Optimized Color Orders", *IEEE International Conference on Image Processing, 2017*.

2017 F. Lahoud, **B. Jin**, M. Ortiz Segovia, S. Süsstrunk, "Keyword-based Image Color Re-rendering with Semantic Segmentation", *IEEE International Conference on Image Processing, 2017*.

2017 **B. Jin**, M. Ortiz Segovia, S. Süsstrunk, "Webly Supervised Semantic Segmentation", *IEEE International Conference on Computer Vision and Pattern Recognition, 2017*.

2016 **B. Jin**, M. Ortiz Segovia, S. Süsstrunk, "Image aesthetic predictors based on weighted CNNs", *IEEE International Conference on Image Processing, 2016*.

2015 **B. Jin**, G. Yildirim, C. Lau, A. Shaji, M. Ortiz Segovia, S. Süsstrunk, "Are Faces Equally Important?", *IS&T/SID Human Vision and Electronic Imaging Conference, 2015*.

2012 **B. Jin**, W. Lin, J. Wu, T. Wu, "Layout-expectation-based Model For Image Search Re-ranking", *IEEE International Conference on Multimedia and Expo Workshops, 2012*.

2011 J. Hu, **B. Jin**, W. Lin, J. Huang, "Saliency-based Visualization for Image Search", *IEEE International Workshop on Multimedia Signal Processing, 2011*.