An Investigation of Deep Learning for Image Processing Applications

Xianxu Hou

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy

May 2018

Abstract

Significant strides have been made in computer vision over the past few years due to the recent development in deep learning, especially deep convolutional neural networks (CNNs). Based on the advances in GPU computing, innovative model architectures and large-scale dataset, CNNs have become the workhorse behind the state of the art performance for most computer vision tasks. For instance, the most advanced deep CNNs are able to achieve and even surpass human-level performance in image classification tasks. Deep CNNs have demonstrated the ability to learn very powerful image features or representations in a supervised manner. However, in spite of the impressive performance, it is still very difficult to interpret and understand the learned deep features when compared to traditional human-crafted ones. It is not very clear what has been learned in the deep features and how to apply them to other tasks like traditional image processing problems.

In this thesis, we focus on exploring deep features extracted from pretrained deep convolutional neural networks, based on which we develop new techniques to tackle different traditional image processing problems.

First we consider the task to quickly filter out irrelevant information in an image. In particular, we develop a method for exploiting object specific channel (OSC) from pretrained deep CNNs in which neurons are activated by the presence of specific objects in the input image. Building on the basic OSC features and use face detection as a specific example, we introduce a multi-scale approach to constructing robust face heatmaps for rapidly filtering out non-face regions thus significantly improving search efficiency for potential face candidates. Finally we develop a simple and compact face detectors in unconstrained settings with state of the art performance.

Second we turn to the task to produce visually pleasing images. We investigate two generative models, variational autoencoder (VAE) and generative adversarial network (GAN), and propose to construct objective functions to train generative models by incorporating pretrained deep CNNs. As a result, high quality face images can be gener-

ated with realistic facial parts like clear nose, mouth as well as the tiny texture of hair. Moreover, the learned latent vectors demonstrate the capability of capturing conceptual and semantic information of facial images, which can be used to achieve state of the art performance in facial attribute prediction.

Third we consider image information augmentation and reduction tasks. We propose a deep feature consistent principle to measure the similarity between two images in feature space. Based on this principle, we investigate several traditional image processing problems for both image information augmentation (companding and inverse halftoning) and reduction (downscaling, decolorization and HDR tone mapping). The experiments demonstrate the effectiveness of deep learning based solutions to solve these traditional low-level image processing problems. These approaches enjoy many advantages of neural network models such as easy to use and deploy, end-to-end training as a single learning problem without hand-crafted features.

Last we investigate objective methods for measuring perceptual image quality and propose a new deep feature based image quality assessment (DFB-IQA) index by measuring the inconsistency between the distorted image and the reference image in feature space. The proposed DFB-IQA index performs very well and behave consistently with subjective mean opinion scores when applied to distorted images created from a variety of different types of distortions.

Our works contribute to a growing literature that demonstrates the power of deep learning in solving traditional signal processing problems and advance the state of the art on different tasks.

Acknowledgments

I would like to thank many people who contributed to the four wonderful years of my experience as a PhD student.

First and foremost, I would especially like to thank my PhD supervisor, Guoping Qiu, for taking me under his wing. Through around 500 emails, 1,600 WeChat messages and many meetings, Guoping introduces me into this wonderful field of academic research and encourages me to explore new fields and research directions, which help me from a confused student to a competent researcher. Guoping's passion, foresight and ambition for perfection are infectious.

I would like to thank my close collaborators and lab-mates who I had a wonderful pleasure of working with and learning from. I would like to thank Linlin Shen who gives insightful comments, feedback and advice. I would like to thank Qian Zhang, Chao Zhang, Zhan Xu, Bolei Xu, Jingxin Liu, Zhuo Lei, Ke Sun, Fei Yang, Qing Li and Rui Cao for thoughtful discussions and ideas sharing. I would also like to thank Bozhi Liu for his help during my visiting at Shenzhen University.

I am thankful to funding support institutes in the University of Nottingham that supported my research, including The International Doctoral Innovation Centre (IDIC) and The Horizon Centre for Doctoral Training (CDT). I would also like to thank Emma Juggins and Jessica Wang for administrative support in UK and China.

I would like to thank many people who are around during my daily life to make my experience in Nottingham and Ningbo so pleasant. I would like to thank Xiaoping Jiang, Dongsheng Zhao, Long Wen, Chuang Gao, Jorge Mendez Astudillo, Kai Xing, Yitong Huang, Jiayang Wang, Kate Arnold, Sam Doehren, Gregor Engelmann, Iona Fitzpatrick, Shadab Mashuk, Obrien Sim, Pepita Stringer, Tatiana Styliari, Matt Voigts, Alex Young, Dan Wang, Xiangzhi Zhang and Xi Gao.

I would like to thank my girlfriend Lu Liu, for all her love and support.

I would like to extend thanks to my friends who I had the pleasure of interacting with and learning from during my Master's period. Especially Xukun Lu, Qiang Wei, Xi-

aoxin Li, Youfei Li and Yang Li.

I would like to thank my close high school friends Zheng Zhang, Wei Li and Xun Gao, who especially support and encourage me to pursue a PhD in computer science from geology.

Last but not least, I would like to give a big thank you to my parents, Mingshu Hou and Guozhen Huang, and my sister Dan Hou, for supporting me to pursue my dreams. I am deeply indebted to the love from them. This thesis is dedicated to them with my sincere gratitude.

Publications

1. KeSun, JiasongZhu, ZhuoLei, XianxuHou, QianZhang, JiangDuan, and Guoping Qiu. Learning deep semantic attributes for user video summarization. In 2017 IEEE International Conference on Multimedia and Expo, ICME 2017, Hong Kong, China, July 10-14, 2017, pages 643–648, 2017. doi: 10.1109/ICME.2017.8019411. URL https://doi.org/10.1109/ICME.2017.8019411

2. XianxuHou, JiasongZhu, KeSun, LinLinShen, and GuopingQiu.Objectspecific deep feature and its application to face detection. In Fifteenth IAPR International Conference on Machine Vision Applications, MVA 2017, Nagoya, Japan, May 8-12, 2017, pages 173–176, 2017. doi: 10.23919/MVA.2017.7986829. URL https://doi.org/10.23919/MVA.2017.7986829

3. Xianxu Hou, LinLin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In 2017 IEEE Winter Conference on Applications of Com- puter Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017, pages 1133– 1141, 2017. doi: 10.1109/WACV.2017.131. URL https://doi.org/10.1109/WACV.2017.131

4. Xianxu Hou and Guoping Qiu. Image companding and inverse halftoning using deep convolutional neural networks. CoRR, abs/1707.00116, 2017. URL http://arxiv.org/abs/1707.00116

5. Xianxu Hou, Jiang Duan, and Guoping Qiu. Deep feature consistent deep image transformations: Downscaling, decolorization and HDR tone mapping. CoRR, abs/1707.09482, 2017. URL http://arxiv.org/abs/1707.09482

Contents

CHAPTER 1

Introduction

1.1 Overview

One fundamental task in computer vision is visual recognition that gives the computer the ability to identify visual content information from an image composed of a grid of raw pixel values. Humans can easily achieve to classify a given image or identify different objects presented in that image. A quick glimpse at an image is enough for us to catch all the important visual details. Humans are able to learn from only a few examples and naturally adapt to a variety of conditions such as brightness, scale, rotation, deformation, angle and so on. Although these tasks seem very natural and straightforward for us, it should be noted how difficult these challenges are for a computer. Visual recognition and other machine vision perception cannot be easily solved by manually designing rules which are used for processing visual inputs. For a computer, an image is represented as an array of numbers and an intelligent system is designed to be able to transform a bunch of structured numbers to high level concepts like "faces". Moreover, not only pixel values for faces from different people are not the same, but also the pattern of pixel values for the same face could be completely different due to different visual conditions.

In spite of the difficulty of those tasks, we have witnessed the advanced progress in the area of visual recognition. With structured visual input, traditional approaches are trying to figure out effective ways to extract visual features that can preserve useful and robust semantic information as well as important details against different variations. Computer vision community has enjoyed the benefits of powerful hand-crafted features like SIFT [1], HOG [2] and Haar [3] and achieved consistent high performance in different visual recognition tasks. More importantly, significant breakthroughs have been achieved in recent years with the advanced development of deep learning. State

of the art deep convolutional neural network (CNN) models have the ability to correctly distinguish image categories on large-scale datasets with millions of images and thousands of categories with human-level performance [4]. Other related visual tasks like object detection [5–7], segmentation [8, 9] and image captioning [10, 11] are also dramatically improved in recent years. In a word, deep CNNs have become the workhorse behind the state of the art performance for most computer vision tasks. Moreover, the deep convolutional neural network features (usually called deep features) have become powerful self-learned visual representations different from classical human crafted ones and have enabled effective transfer learning from large dataset like ImageNet [12].

Following the impressive performance of deep convolutional neural network models for visual recognition, it is natural to ask why they perform so well, or how they might be improved, or what they have learned, or what on earth deep features are, or how they can be transferred to other tasks. Without a clear understanding of how and why they work, developing better models or applying them to new applications would be reduced to trial-and-error experiments, which is unsatisfactory from a scientific point of view. In computer vision community, there is a trend to develop visualization techniques to reveal the internal representations of different convolutional layers in deep models. These efforts could help get a better understanding of the existing systems, and enable more powerful vision systems. At the same time, introducing new methods to solve well known problems can always provide new perspectives and directions to the research field. In this thesis, beyond the visual recognition problems we adopt deep learning based solutions and design deep neural network models for traditional image processing tasks.

1.2 Contributions

In this thesis we develop different techniques to tackle traditional image processing problems based on self-learned features from deep convolutional neural networks.

In particular we first consider the task to quickly filter out irrelevant information in an image. We take face images as a specific example and try to filter out non-face regions to help build face detector. Specifically we explore and exploit the internal representations in pretrained deep convolutional neural networks and propose object specific channel (OSC) features, which are successfully used to rapidly filter out background regions and build compact face detector with state of the art performance in the wild settings. In addition, we consider generating visually pleasing images from the per-

spective of image generation. Particularly we design new architectures to train two generative models, i.e., variational autoencoder (VAE) and generative adversarial network (GAN) by incorporating deep features in the objective functions. As a result, realistic and sharp face images can be randomly generated when tested on face image database, moreover the learned latent face image representations can be used to achieve state of the art performance in facial attributes prediction. Third, we revisit several image information augmentation and reduction tasks. We develop different deep models by adopting an end-to-end learning paradigm. We formulate each low-level image processing task as a single optimization problem in which we use deep convolutional neural networks as non-linear mapping functions to achieve different image transformation. More importantly, a deep feature consistent (DFC) framework is proposed to help construct object function for measuring the perceptual loss for each image transformation task. Finally a general deep feature based image qualitative assessment (DFB-IQA) index is designed to measure perceptual image quality, which is robust to different types of distortions and can work in a variety of applications.

1.3 Outline

In order to make each part of the thesis self-contained, we will leave the background and related work to each individual chapter. The rest of thesis is organized as follows.

Chapter 2 provides the essential background on machine learning in general and reviews the basic concept and knowledge of neural network (especially convolutional neural network), gradient decent optimization algorithm as well as other novel model architectures and techniques, which are the essential components for deep learning.

In **Chapter 3** we start to analyze the internal representations in deep convolutional neural networks and present a method for exploiting object specific channel (OSC) features and use face detection as a case study. In particular, we seek to discover and exploit the convolutional channels of a pretrained CNN in which neurons are activated by the presence of specific objects in the input images based on both qualitative visualization and quantitative indicators. Specifically a method for explicitly fine-tuning a pretrained CNN to induce an OSC and systematically identifying it for the human face object has been developed. Based on the basic OSC features, we introduce a multi-scale approach to constructing robust face detector with state of the art performance in unconstrained settings.

Chapter 4 focuses on two generative models used to produce high quality images. Specifically we consider improving the performance of two popular generative models,

variational autoencoder (VAE) and generative adversarial network (GAN), by incorporating pretrained deep features in the objective functions. Both models can process a random vector and produce an image with a natural appearance. Moreover, we also show that our VAE model can capture perceptual and semantic information in the latent representation which can be used to achieve state of the art performance for facial attributes prediction.

Having discussed the effectiveness of deep features for generative models, **Chapter 5** and **Chapter 6** propose a deep feature consistent principle and move to tackle several low-level image processing tasks. In **Chapter 5** image companding and inverse halftoning are considered. They are essentially information augmentation tasks to expand an image from a lower bit depth to a higher bit version. We also advance the model architecture design by adding skip connections between different layers to strengthen feature propagation and encourage feature reuse. As a result, our models can achieve superior results for both tasks.

In **Chapter 6** we investigate another 3 challenging image processing problems: image downscaling, decolorization (colour to grayscale conversion) and high dynamic range (HDR) image tone mapping. Unlike companding and inverse halftoning that requires inferring new information, these three tasks essentially try to use reduced images with less information (low resolution, less color channel and low dynamic range) to express the original ones. We propose a deep feature consistent deep image transformation (DFC-DIT) framework and build deep models to unify these 3 image processing tasks in a similar way. To the best of our knowledge, this is the first work that successfully uses deep learning to solve these three low level image processing problems in a unified framework. The experimental results demonstrate the effectiveness of the DFC-DIT technique and its state of the art performances.

In **Chapter 7** we further explore the deep feature consistent concept from the perspective of image quality assessment and propose an objective method for assessing perceptual image quality. The key insight is the capability of pretrained deep convolutional neural network to contain structural and perceptual image information, which can be used as an alternative motivating principle for the design of image quality metric. Compared with other image quality assessment methods (PSNR, SSIM and CW-SSIM), the proposed method yields superior results and behaves consistently with subjective mean opinion scores when applied to distorted images created from different types of distortions.

Finally, in **Chapter 8** we come to a conclusion based on previous chapters, and identify the remaining challenges as well as discuss the path forward.

CHAPTER 2

Background

Computer vision seeks to generate intelligent and human-understood descriptions of visual objects and scenes from raw pixels. However it is too difficult to be solved by manually designed rules for computers to follow. Instead, a special category of algorithms are designed for machine to *learn* the internal patterns and useful visual representations from a given dataset, and the learned features can be then applied to different computer vision tasks. This alternative approach is usually referred to *machine learning* that gives computers the ability to learn without being explicitly programmed. This chapter provides the essential background on machine learning and deep learning in general, especially deep convolutional neural networks.

2.1 Machine Learning

Machine learning focuses on designing software or learning algorithms that give computers the ability to learn from and make predictions on data. This approach is quite useful for a variety of complicated tasks like computer vision related problems, where designing strictly static program instructions with good performance is infeasible and impractical based on hard-code knowledge. Instead, machine learning systems are able to empower computers to solve these problems by discovering patterns and requiring their own knowledge from data.

A more formal and widely quoted definition of machine learning is *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"*. In order to design a machine learning system for computer vision and image processing tasks, the experience E usually includes different image samples and corresponding labels if available. Image samples are usually represented as a 4D array $X \in \mathbb{R}^{n \times c \times w \times h}$,

where n is the batch size of the images; c, w and h are the color channel, width and height of a given image. The image samples can be also encoded as a matrix $X \in \mathbb{R}^{n \times m}$ and each of n rows describes one sample with m features. In addition, the labels indicating the categories for each image are usually encoded as a vector $y \in \{0, 1, ..., k\}^n$, each y_i represents which of k categories sample i belongs to. For example, in our work we use 1 to indicate face images while 0 for non-face images.

In this thesis, we explore different tasks T under the machine learning framework.

- *Face detection:* In this task, the machine learning algorithm is designed to learn a non-linear function $f : \mathbb{R}^{c \times w \times h} \rightarrow \mathbb{R}^m$. Here $f(x)$ is used to estimate the coordinates y of faces for an input image x. Additionally a face image classification system is also designed as the essential part of the face detector system. The classification algorithm is asked to output a function $f : \mathbb{R}^{c \times w \times h} \rightarrow \mathbb{R}$, to decide whether the input is a face image or not.

- *Density estimation:* In this task, the machine learning algorithm is used to tackle density estimation in high dimensional spaces to learn generative models, i.e., $f : \mathbb{R}^m \rightarrow \mathbb{R}^{c \times w \times h}$. Specifically this task is asked to output new sample images following the same probabilistic distribution of a given image dataset from random vectors. In order to train models to generate data like the ones in a given dataset, it is crucial to design performance measure P to force the model to efficiently discover and internalize the essence of the dataset. In this work, we attack two popular generative models, variational autoencoder (VAE) and generative adversarial network (GAN).

- *Image transformation:* In this task, the machine learning algorithm is designed to learn a transformation function $f : \mathbb{R}^{c1 \times w1 \times h1} \rightarrow \mathbb{R}^{c2 \times w2 \times h2}$, to process an input image to an output image which contains certain properties for different transformation purposes. In this work, we explore several image transformation tasks which can be categorized as two opposite directions: one is information augmentation to expand lower bit images to higher bit ones (like inverse halftoning) by adding more tiny details and spatial information, the other direction is to present images with less information (like resolution, color channel and dynamic range), which can still preserve visually important details and perceptual quality.

Each task should be evaluated by corresponding performance measure P, which is the objective function or loss function to guide the training. The objective function is designed to estimate the similarity between the model outputs and the ground truth,

mathematically quantifying the amount by which the prediction deviates from the actual values from experience E. After the objective function is decided for a given task, an optimization step is needed to find the *"best available"* values for a given model. We will introduce different optimization methods used in this work in the following sections.

2.2 Supervised Learning

Supervised learning is one type of machine learning algorithms to infer a function from labeled training data. Each training sample contains the desired output of the model in advance, which is served as the supervisory signal during training. Mathematically supervised tasks can be formulated as requiring a computer to estimate a mapping $f : X \to Y$, where X is the input space and Y is the desired output space. One example of this is aforementioned face image classification task, where X is the space of input images and Y is a value between 0 and 1, indicating the probability of a face appearing somewhere in the image.

Assumptions. The final goal of a supervised learning algorithm is to apply the optimized function f to map unseen samples in which the desired outputs cannot be observed. Therefore, the learning algorithm is required to learn the internal representations or patterns of the training data and generalize properly to unseen situations. In fact, we need to assume that there indeed exists some common pattern in the data and the samples ($(x, y) \in X \times Y$) are independently and identically distributed, i.e., i.i.d. That's to say, each sample (x_i, y_i) is drawn from the same distribution and independently generated from each other.

Objective. The learning objective is to select the *"best"* mapping $f : X \to Y$ from a given set of candidates $\mathbb{F}$. f is desired to fit the training data well and generalize well for unseen data too. The selection is usually based on scalar-valued loss function $L(y, \hat{y})$ to measure the inconsistency between the prediction $\hat{y}_i = f(x_i)$ and the ground truth y_i. Mathematically, the learning problem can be formulated as an optimization problem as follows:

$$f^* = \arg\min_{f \in \mathbb{F}} \frac{1}{n} \sum_{i}^{n} L(y_i, \hat{y}_i) \tag{2.2.1}$$

$$= \arg\min_{f \in \mathbb{F}} \frac{1}{n} \sum_{i}^{n} L(y_i, f(x_i)) \tag{2.2.2}$$

Loss functions. The definition of the loss functions could vary from task to task. In

principle, the scalar-valued loss should reflect the disagreement between the prediction and the ground truth. For regression tasks there are two commonly used loss functions. One is the squared distance $L(y, \hat{y}) = \sqrt{||y - \hat{y}||^2}$, which is also called L_2 loss, the other is the absolute distance $L(y, \hat{y}) = |y - \hat{y}|$, which is also called L_1 loss.

For classification tasks, Softmax function and Logistic function are used to define classification losses. In fact, Softmax classifier is the generalization of Logistic regression from binary classification to multiple categories. Specifically the corresponding loss (often called cross-entropy loss) for one sample is defined as follows:

$$L_i = -log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{2.2.3}$$

where f_j is the j^{th} element for the output vector and y_j indicates the ground-truth category. $\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ is the Softmax function to normalize the an arbitrary output vector to a vector of values between 0 to 1. Intuitively cross-entropy loss tries to penalize the inconsistency between the prediction and corresponding ground truth. What's more, Softmax classifier could interpret the outputs f_j as the unnormalized log probabilities, therefore minimizing the negative log likelihood of the correct class is, in fact, to perform maximum likelihood estimation.

Regularization. In practice, the mapping f_W is usually defined as a complicated function parameterized by weights W, and neural network models are often used for deep learning. Now that the architecture or shape of f is predefined, the optimization is transferred to choose the best set of parameters W. However, one issue needed to be considered is that this set of parameters is not necessarily unique, and many similar W can achieve the same goal like correctly classifying all the training samples. What we desired for is that W can generalize well to new and unseen samples, and the *generalization* is what makes machine learning different from pure optimization. One technique commonly used to improve generalization is to extend the loss function with a regularization penalty $R(W)$. More precisely, L_1 or L_2 norm is commonly used to discourage large weights through an element-wise quadratic penalty over all parameters $R(f) = R(W) = \sum W^2$. Thus, the final loss function is the combination of data loss and the regularization loss weighted by a hyper-parameter λ:

$$f^* = \arg\min_{f \in \mathbb{F}} \frac{1}{n} \sum_i^n L(y_i, f(x_i)) + \lambda R(f) \tag{2.2.4}$$

$$= \arg\min_{f \in \mathbb{F}} \frac{1}{n} \sum_i^n L(y_i, f(x_i)) + \lambda R(W) \tag{2.2.5}$$

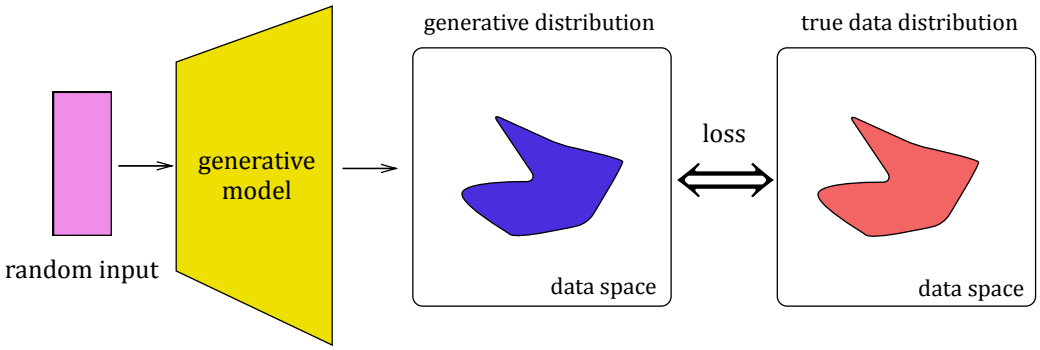


Figure 2.1: An overview of a generative system.

This penalty gives preferences to select some W over others and follows the principle of Occam's razor, which can be stated as: *"Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better"*. Mathematically L_1 norm has the property to penalize large weights and prefers smaller and more diffuse weights that can reduce the complexity of function f_W, therefore the final predictions will take all the input dimensions into small amount rather than depending on only a few dimensions. For deep learning there exist other effective regularization methods and we will leave them in the neural network section.

2.3 Unsupervised Learning

Unsupervised learning is another type of machine learning method to infer a function to describe hidden structure from unlabeled data. The goal is to explore and discover something about the training data, for example, the trained generative models can generate high quality data that are similar to the training data from a random input. Figure 2.1 shows an overview of a generative system.

In this work, we attempt to tackle unsupervised learning problems based on deep generative models. Specifically two popular generative models are considered in this work. One is variational autoencoder encoder (VAE) [13, 14], the other is generative adversarial network (GAN) [15]. VAE allows us to formalize image generation task in the framework of probabilistic graphical models with latent variables. VAE models can be trained by optimizing a lower bound loss which is the sum of the reconstruction loss and KL divergence loss. GANs try to formalize the training as a game between two separate networks: a generator network used to generate new samples and a second discriminator network used to distinguish between these samples from the model distribution and these from true distribution. During the training, each time the

discriminator identifies a difference between the two distributions, the generator will adjust itself to make the difference go away. Finally, the discriminator in theory can be "fooled" by the samples produced by the generator, which means an equilibrium has reached from the perspective of game theory. Therefore the generator is expected to produced samples similar to those coming from the true distribution.

In addition, not only can generative models produce new samples based on a training dataset, but also provide an effective way to learn reusable feature representations for supervised learning.

2.4 Deep Neural Networks

In previous sections, we have shown that we need to estimate a function $f : X \to Y$ for different mapping problems. However, we have not yet given any details about the function f. We now turn to specify the f we considered in this thesis. In particular, we focus on formulating the complicated mappings with deep convolutional neural networks. In the following sections, we will introduce the basic architecture of neural networks, and one specific biologically inspired variant – convolutional neural network (CNN, or ConvNet), which has been widely used to process visual data. Finally we show how to optimize deep CNNs to choose the best model.

2.4.1 Regular Neural Network

Neural network is originally inspired by the goal of modeling biological neural systems and is among the most successful algorithms for machine learning tasks. In order to build a neural network system, we first need to model *artificial neurons*. The earliest type of artificial neuron is called *perceptron* [16], which takes several binary inputs $x_1, x_2, ..., x_i$ and produces a single binary output y. The value of a perceptron output is determined by comparing the weighted sum $\sum_i w_i x_i$ to a given threshold as follows:

$$perceptron\ output = \begin{cases} 0 & \text{if } \sum_i w_i x_i \leq \text{threshold} \\ 1 & \text{if } \sum_i w_i x_i > \text{threshold} \end{cases} \tag{2.4.1}$$

One problem of perceptron neuron is the discontinuity on itself that a small change of weights can cause the output of that perceptron to flip completely. More advanced type of artificial neurons are then introduced to solve this problem. New type of neurons are similar to perceptrons but use a continuous function σ to achieve nonlinearity, which

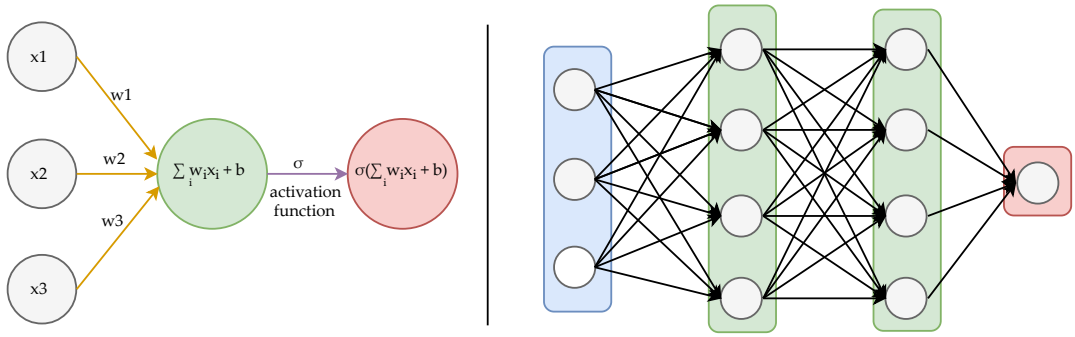


Figure 2.2: Diagram for a single artificial neuron and a two layer neural network.

can be formulated as $y = \sigma(\sum_i w_i x_i + b)$. Each input x_i has a corresponding weight w_i and an overall bias b. A neuron is diagrammed in the left part in Figure 2.2.

There are several other activation functions σ commonly used in practice: Tanh function $(e^x - e^{-x})/(e^x + e^{-x})$, logistic function $1/(1 + e^{-x})$, Rectified Linear Unit (ReLU) $max(x, 0)$ [17] and other variants of ReLU such as PReLU [18], RReLU [19], CReLU [20] and LeakyReLU [19].

Neural networks are often modeled hierarchically as a collection of neurons which are organized into different layers. Different layers are connected as an acyclic graph in someway and the data information can be transmitted from one layer to another. The simplest and most commonly used architecture is the fully connected layer: two adjacent layers are pairwise connected and the neurons in the same layer share no connections. The right diagram in Figure 2.2 shows two layer fully-connected neural network. The last layer is an output layer in which the size of neurons is determined by different tasks. For instance, only one neuron is needed for a binary classification task.

To put it in more precise algebraic terms, neural networks can be constructed by repeating matrix multiplications and element-wise non-linearities. For example, a three-layer neural network can be formulated as $f(x) = W_3\sigma(W_2\sigma(W_1 x))$, where W_1, W_2 and W_3 are matrices.

2.4.2 Deep Convolutional Neural Network

Motivation. Traditionally neural networks are fully connected between all the hidden and input units, however it is computationally very expensive, even infeasible to tackle large dataset. Intuitively one solution is to restrict the connections between hidden units and input units, i.e., locally connected, allowing each hidden unit only connects part of input units. This idea draws the inspiration from biological studies on

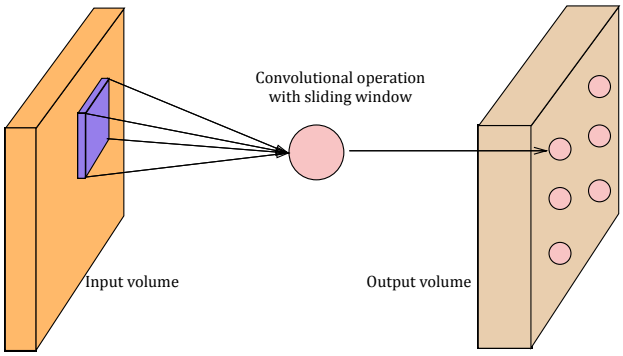


Figure 2.3: Diagram for a convolutional layer. Each neuron in the output layer is the result of a dot product between a small filter (like 3×3) and a local input. For a given convolutional layer, there are several different filters and each slides over all spatial locations and computes dot products independently.

visual system, especially, neurons in virtual cortex have localized receptive field [21]. Specifically the neurons in convolutional neural network are arranged in 3 dimensions: height, width and depth. Each layer of CNNs transforms one volume of activations to another in a more sensible way.

Convolutional layer. It is one type of neural network architectures [22] specifically designed to handle data with spatial topology like image and video. It is served as the key block to construct convolutional neural networks to achieve local connections and parameters sharing (Figure 2.3). A convolutional layer takes an input tensor to a similar output tensor by spatially convolving the input volume with a set of different filters. Each filter works independently and computes a dot product with a small chunk of input by sliding across all spatial positions of the input. What's more, the weights of one filter remain unchanged when sliding different locations. As a result, the number of parameters can be reduced dramatically by this kind of sharing scheme.

CNN architecture. In the past few years, innovated convolutional neural network architectures are proposed like AlexNet [23], VGGNet [24], GoogLeNet [25], ResNet [26] and DenseNet [27]. Typically a convolutional neural network is built by stacking convolutional and other layers hierarchically, like $[Input - [Conv - BN - ReLU - Pool] \times N - FC]$. Take image as an example, the $Input$ represents a 4-dimensional tensor of a batch of images and $Conv$ represents a convolutional layer described above. BN is a Batch Normalization layer [28] that performs the normalization for each training mini-batch, allowing to use much higher learning rates and stabilize the training for very deep models. ReLU (Rectified Linear Unit) is commonly used as the non-linearity to threshold at 0. $Pool$ stands for a typical filter (2×2) pooling layer with stride (2×2) for downsampling operation along the spatial dimensions. Max and average pooling are commonly used. FC is a fully-connected layer stacked in the last few layers to squash

the output tensors to vectors, which can be further processed for different tasks. In theory, the architecture could be arbitrary "deep" by stacking different layers.

2.4.3 Optimization

In previous sections, we saw that machine learning tasks can be formulated as an optimization problem of the form $W^* = \arg\min_W L(W)$, where $L(W)$ is the loss function parameterized by the weights W coming from a convolutional neural network. Now we move to the techniques on how to optimize deep CNN models.

Gradient descent algorithm. The loss function $L(W)$ defined above is a complicated and often non-convex function, which can not be easily optimized by any tools in optimization literature. For neural network optimization, we usually try to find a direction in the parameter space that would improve our weights to give us a lower loss step by step, which is the key insight of *gradient descent algorithm*. In particular, we could restrict us only using continuous functions to build deep models and calculate the gradients $\nabla_W L(W)$ of the loss function $L(W)$ with respect to the weights W. Backpropagation [29] is the algorithm we use to efficiently compute the gradients of a deep network based on the chain rule from calculus. Mathematically the gradients are the partial derivatives and give the direction to increase of the loss $L(W)$ locally, therefore, we can improve the weights W by subtracting a small amount (weighted by learning rate lr) along the gradient direction, i.e., $W := W - lr\, \nabla_W L(W)$. In practice, a mini-batch of training examples are only used to estimate the gradients rather than the whole dataset, which can be very large (e.g. ImageNet [12] contains more than 1 million image samples). The resulting algorithm is often called *Stochastic Gradient descent (SGD)*, which repeats the same process to reduce the loss until convergence and it works well in most practical applications.

One critical hyperparameter to implement SGD is the learning rate, which decides the step of the descent each time. In principle, the learning rate should not be set too big or too small. The optimization may not converge or even diverge if the learning rate is too big, on the other hand, the training will be too slow when using a too small learning rate. In practice, a good rule of thumb is to initialize relatively a big learning rate and decay it during the training.

Advanced SGD. Standard SGD does not guarantee good convergence and suffers from several challenges like how to choose a proper learning rate, how to balance the oscillations, and how to avoid suboptimal local minima, etc. There are a few more advanced techniques which are commonly used by deep learning to deal with aforementioned

challenges and improve better convergence for deep networks. The simplest method is *Momentum* [30] that helps speed up the training in a relatively small but consistent direction along the gradient. *Momentum* does it by decaying the sum of previous gradient directions for each step. Another technique is called *Adagrad* [31] that does not restrict the learning rate to be the same for all the parameter updates. Instead, it adapts the learning rate according to the parameters: smaller updates for frequent parameters. In addition, Adaptive Moment Estimation (Adam) [32] is a more complicated method to compute adaptive learning rate for each parameter by exponentially decaying both the average of past gradients and squared gradients. These improved methods can equalize the updating for different parameters and achieve faster and better convergence.

2.5 Applications

In this section, we will briefly describe the applications of deep learning. Though deep learning can be widely used to solve applications in computer vision, natural language processing and speech recognition, we would like to focus on image related tasks.

Computer vision is a broad research field that encompassing a variety of ways to process image and video data and has a diversity of different applications. The final goal is to simulate or reproduce the ability of the human visual system. Most deep learning techniques for computer vision are used to object recognition tasks like image classification, which is one of the core problems in computer vision that a variety of other seemingly distinct problems (like object detection) can be reduced to image classification tasks. In fact, new innovated deep network architectures [23–27] are first designed for image classification problem for large-scale dataset like ImageNet [12]. Other applications include object detection, which means annotating an image with bounding boxes around each object [6, 7, 33, 34]; image segmentation, which means labeling each pixel in an image with the identity of the object it belongs to [8, 9, 33]; image captioning, which means transcribing the content of an image with a sentence [10, 11]; style transfer, which means recomposing images in the style of other images [35–37]; image synthesis, which means generating realistic new images [13–15].

2.6 Summary

Deep learning can perform *automatic feature extraction* without human intervention through an end-to-end training manner. A typical workflow for deep learning can be summarized as follows:

Data preparation. For a given machine learning task, the first step is to obtain a dataset $\{X, y\}$, where X is the images used in the task, and y is the labels for the corresponding images. Usually the dataset will be then divided into 3 folds: training, validation and testing split. Training split is used for parameters optimization, validation split is for tuning different hyperparameters and testing split is for evaluating the final performance of the trained model. What's more, it is often a good idea to preprocess the data before being fed into the neural networks. In the case of images, it is a very common practice to compute a mean image across the whole dataset and subtract it from every image. As a result, all the images are *zero-centered* around 0 and range from approximately [-1, 1] or [-127, 127] for different scales.

Model architecture and objective function design. When the data is ready, what we need is to design the architecture of the deep model and the objective function for a given task. It is commonly built by stacking convolutional and other layers hierarchically, like $[Input - [Conv - BN - ReLU - Pool] \times N - FC]$. But how deep should the network be, anyway? There are no strict rules on the scales of deep models, and they are often tuned to find the best hyperparameters. A rough rule of thumb is to use deeper models when large scale data are available. Additionally, the objective function is often designed as a loss layer that can be directly stacked at the end of the model to achieve end-to-end training. The last step is to choose the deep learning tools to implement the designed models. There are a few widely used deep learning frameworks we can choose such as *Caffe*[38], *TensorFlow*[39], *Torch* [40], *Theano*[41], MXNet [42], Chainer [43], PyTorch and so on.

Parameters optimization. After the model is decided, we can move to the training stage. The optimization with stochastic gradient descent algorithm is a loop process across the whole dataset until convergence. In addition we can train several independent models by trying different hyperparameters (like learning rate). The final published model is the one that achieves the best performance on the validation dataset.

Evaluation. After the best model is selected, we can abandon the training data and use it on testing split to report the final performance. We can quantify the performance via different metrics like accuracy, area under ROC curve, confusion matrix, etc.

Other tricks. In order to implement excellent deep convolutional neural networks, we adopt several tricks or tips in this thesis.

- Data augmentation. It is a must-do thing for training a deep convolutional neural network. Horizontal flipping, random crops and scaling are often used to augment image data. It is a common way to try the combinations of different

methods.

- Initialization. It is a good idea to initialize all the parameters with random small numbers to train deep models from scratch. We can also achieve better initialization by using the weights from a pretrained model, and fine-tune the model for different tasks.
- Regularization. For deep learning, it is common to apply weight decay to regularize deep models and add Dropout layer [44] to prevent overfitting for better generalization.
- Learning rate decay. It is often benefit to anneal the learning rate during the training. This can be achieved either by multiplying a constant (e.g. 0.5) for predefined steps, or using continuous annealing schedules.

CHAPTER 3

Object Specific Deep Feature for Face Detection

In this chapter, we consider the task to quickly filter out irrelevant information in an image and use face detection as a case study. In particular we develop approaches for exploring the internal representations of deep convolutional neural networks and propose object specific features, which are successfully used to tackle high-level image processing problem, i.e., face detection, in the wild settings. In particular, a method for explicitly fine-tuning a pre-trained CNN to induce object specific channel (OSC) and systematically identifying it for the human faces has been developed. We also introduce a multi-scale approach for constructing robust face heatmaps based on OSC features for rapidly filtering out non-face regions thus significantly improving search efficiency for detection. We show that multi-scale OSC can be used to develop simple and compact face detectors in unconstrained settings with state of the art performance.

3.1 Introduction

A key motivation of this work is based on the observation that certain convolutional channels of CNNs exhibit object specific responses [45, 46]. An *object specific channel (OSC)* is a convolutional feature map at a hidden layer of a CNN, in which neurons are strongly activated by the presence of a certain class of objects at the neurons' corresponding regions in the input image. An example is shown in Figure 3.1 where the last image at the top row is a face specific OSC, in which spatial locations corresponding to the face regions have strong responses (white pixels) while areas corresponding to non-face regions have weak responses (black pixels). If such an OSC can be reliably identified, then it can be exploited for various tasks including object detection. Do such

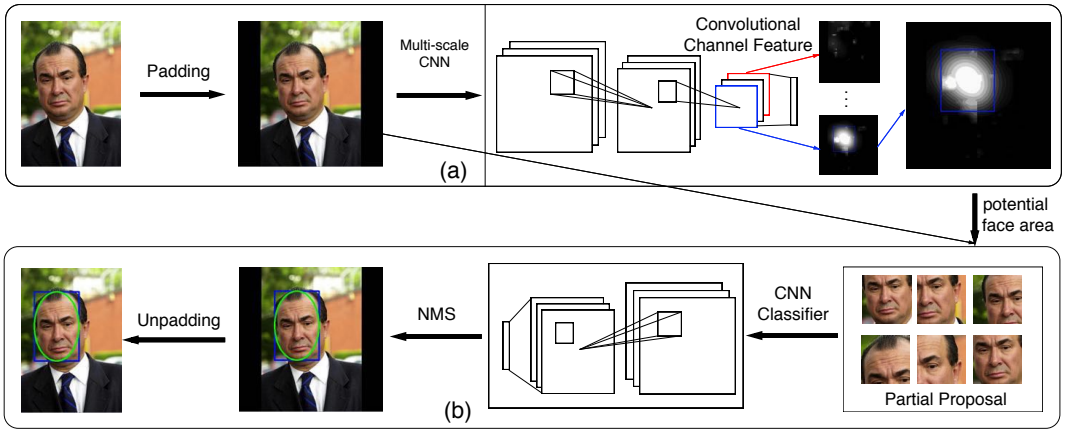


Figure 3.1: (a) An input image is first processed by a CNN, its face specific convolutional channels are identified and a face response heatmap is generated from multi-scale face specific convolutional channel features. (b) Face proposals are generated based on the heatmap and processed by a binary CNN classifier. Finally detected faces are combined through a Non-Maximum Suppression (NMS) algorithm.

channels exist for a given class of objects? If so, how can we systematically identify such channels? If not, can we tune a pre-trained CNN to have such channels? In this chapter, we propose the object specific channel (OSC) based on pretrained deep convolutional neural network and systematically identify it for the human faces. We show that OSC features can be used for rapidly filtering out non-face regions and building compact face detector with state of the art performance.

3.2 Related Work

Face detection models in the literature can be divided into four categories: Cascade-based model, Deformable Part Models (DPM)-based model, Exemplar-based model and Neural-Network-based model. The most famous cascade-base model is the VJ detector [3] based on Haar-like features, which have demonstrated excellent performance for frontal face detection. However Harr-like features have limited representation ability to deal with variational settings. Some works try to improve VJ detector via using more complicated features such as SURF [47], HoG [48] and polygonal Haar-like features [49]. Aggregate channel features [50] are also introduced for solving multi-view face detection problems.

Another category is DPM-based model [51], which treats face as a collection of small parts. DPM-based model can benefit from the fact that different facial parts independently have lower visual variations, therefore it is reasonable to build robust detectors

by combining different models trained for individual parts. For example, Part-based structural models [52] have achieved success in face detection and a vanilla DPM can achieve top performance over more sophisticated DPM variants [53].

Exemplar-based detectors [54] try to bring image retrieval techniques into face detection to avoid explicitly modeling different face variations in unconstrained settings. Specifically, each exemplar casts a vote following the Bag-of-Words (BOW) [55] retrieval framework to get a voting map and uses generalized Hough Voting [56] to locate the faces in the input image. As a result, faces can be effectively detected in many challenging settings. However, a considerable amount of exemplars is required to cover all kinds of variations.

Neural-Networks-based detectors are usually based on deep convolutional neural networks. Faceness [57] tries to find faces through scoring facial parts' responses by their spatial structure and arrangement, and different facial parts correspond to different CNNs. A two-stage approach is also proposed by combining multi-patch deep CNNs and deep metric learning [58]. The CCF detector [59] uses an integrated method called Convolutional Channel Features, transferring low-level features extracted from pre-trained CNN models to a boosting forest model. Cascade architectures based on CNNs [60] have been also designed to help reject background regions at low resolution, and select face area carefully at high resolution. The DDFD detector [61] uses a single model based on deep convolutional neural networks for multi-view face detection, and points out that CNNs can benefit from better sampling and more sophisticated data augmentation techniques.

We try to directly use a single convolutional channel to produce face response heatmap, which can be used to quickly locate potential face area. The heatmap is similar to the voting map in exemplar-based approach [54], but the difference is that their voting map is produced by the Bag-of-Words (BOW) [55] retrieval framework, while our heatmap is directly extracted from a convolutional channel. Another similar approach is faster R-CNN [62], which also uses a feature map to extract potential region proposals for object detection. However, faster R-CNN uses a region proposal network to achieve different region proposals by regressing the coordinates of bounding boxes while we directly threshold the intensive values of the feature map to enable face region proposals.

3.3 Method Overview

Our goal is to discover and exploit face specific convolutional channel (features) to help locate the face areas quickly for further processing. Our system contains two

stages as shown in Figure 3.1: In the first stage, the face heatmap of a face image can be generated by face specific channel in a trained CNN with a multi-scale approach. Thresholding the heatmap quickly filters out non-face regions, dramatically reduces face search space without throwing away genuine face regions. In the second stage, a set of face candidate windows can be quickly identified based on the heatmap, and all candidates are then processed by a CNN based binary classifier. Finally all face windows are merged using Non-Maximum Suppression (NMS) [63] to obtain the final detection results.

3.4 Face Specific CNN Channel Extraction

3.4.1 Training Data Preparation and CNN Fine-tuning

We start with the pre-trained "AlexNet" [23] provided by the open source Caffe Library [38]. In order to adapt the CNN model to our face detection problem, we change the last classification layer from ImageNet-specific 1000 classes to 2 classes, which represent images with faces and images with masked faces (Figure 3.2) respectively. Specifically, let Conv be a convolutional layer, LRN a local response normalization layer, P a max pooling layer and F a fully connected layer, the architecture can be described as Conv1($55 \times 55 \times 96$) - LRN - P - Conv2($27 \times 27 \times 256$) - LRN - P - Conv3($13 \times 13 \times 384$) - Conv4($13 \times 13 \times 384$) - Conv5($13 \times 13 \times 256$) - P - F(4096) - F(4096) - F(2), where the numbers inside the brackets indicate the topology of each layer.

The authors of [45, 46] have tried to gain insight into the operation of the CNN models via visualizing the features and filters of their hidden units. The authors of [46] have shown that the CNN model is highly sensitive to local structure of the input image and that particular regions of an image are responsible for firing specific neural units. [45] discovers that there exist many invariant detectors for faces, shoulders, texts, etc. in convolutional layers. It shows that CNNs can learn partial information even though no explicitly labeled faces or texts exist in the training datasets.

If we can be certain that specific objects will fire specific hidden neurons in a CNN, then it will be very useful because we can infer the objects in the input image from the response of the hidden neurons. Based on above observations, we believe it is possible to fine-tune a CNN such that particular neurons will respond to specific objects if suitably prepared object specific training examples are used. To verify this idea, we use Annotated Facial Landmarks in the Wild (AFLW) database [64] as training dataset to fine-tune the AlexNet. AFLW database contains 25,993 faces in 21,997 real world

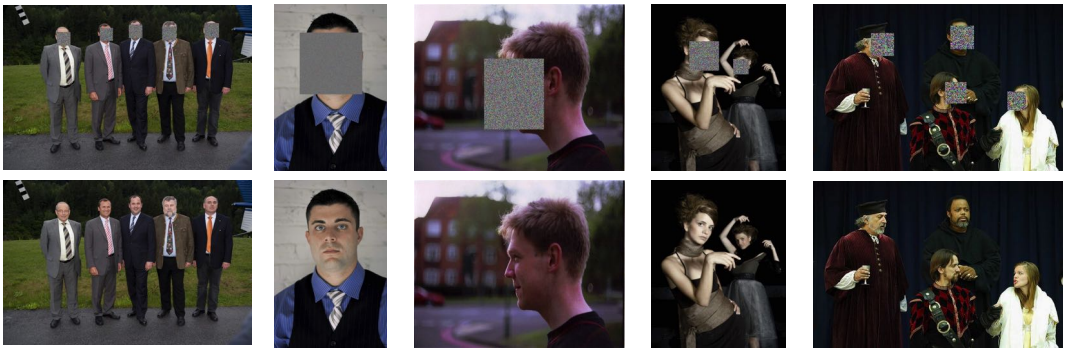


Figure 3.2: Examples of masked face images (first row) and original images with faces (second row) from AFLW [64] for fine-tuning.

images collected from Flickr with a range of diversity and variation in poses, ages and illuminations. Unlike most common methods that crop the ground-truth face area as positive samples and non-face area as negative samples, we use the original images as positive samples. For negative samples, we also use the same images, but mask the facial parts with random noises (R, G, B value are randomly generated from 0-255). Some examples of these positive and negative samples are shown in Figure 3.2. The idea is that if the CNN model can discriminate the masked image from the original image, it will be forced to use the partial information from face area for classification. What we desired is that through re-enforcing the CNN to discriminate images with faces and images without faces, then the network could organise itself such that certain units will be responsible for representing faces, thus enabling the extraction of face specific channels for various post-processing.

3.4.2 Face Specific Convolutional Channel Identification

In order to quantify how well each convolutional channel responds to faces after fine-tuning, we've studied the 5^{th} convolutional layer which has a $13 \times 13 \times 256$ topology and can be visualized as 256 different 2-D heatmaps (13×13). We resize every heatmap from shape (13×13) to (227×227), the same size as the input image, using a bicubic interpolation. We calculate the average intensity value of the heatmap both inside and outside the face areas respectively, which are denoted as "face-score", i.e., $\frac{1}{wh}\sum_{i=x}^{x+w}\sum_{j=y}^{y+h} I_{i,j}$, for a given bounding box, where (x, y) is the top left coordinate, (w, h) are the width and height, and $I_{i,j}$ is the intensity value at point (i, j). The resized heatmap has the same shape as the input image, and the face area in the heatmap can be located by directly using the ground-truth face annotations. To calculate the face-score in the face areas, all the intensity value of pixels inside the face area are added up and divided by the number of pixels in that face area. Similarly, the face-score outside

the face area can be calculated easily. We then use 1,000 images randomly chosen from AFLW [64] to calculate the face-scores inside and outside face areas of all the channels in the 5^{th} convolutional layer of the fine-tuned model. The final value of face-score is the average value across all the 1,000 images. The results are shown in Figure 3.3. We can see that the 196^{th} channel has the highest face-score inside the face areas, followed by the 139^{th} channel. The value of face-score outside face area is small in all channels. This shows that there do exist face specific channels where specific neurons are fired at the spatial positions corresponding to the face regions in the input image. Though it seems empirical to choose the channel index, however consistent responding channels can be produced by randomly choosing different face images. We use the 196^{th} channel in all experiments, which can get better results than using 139^{th} channel.

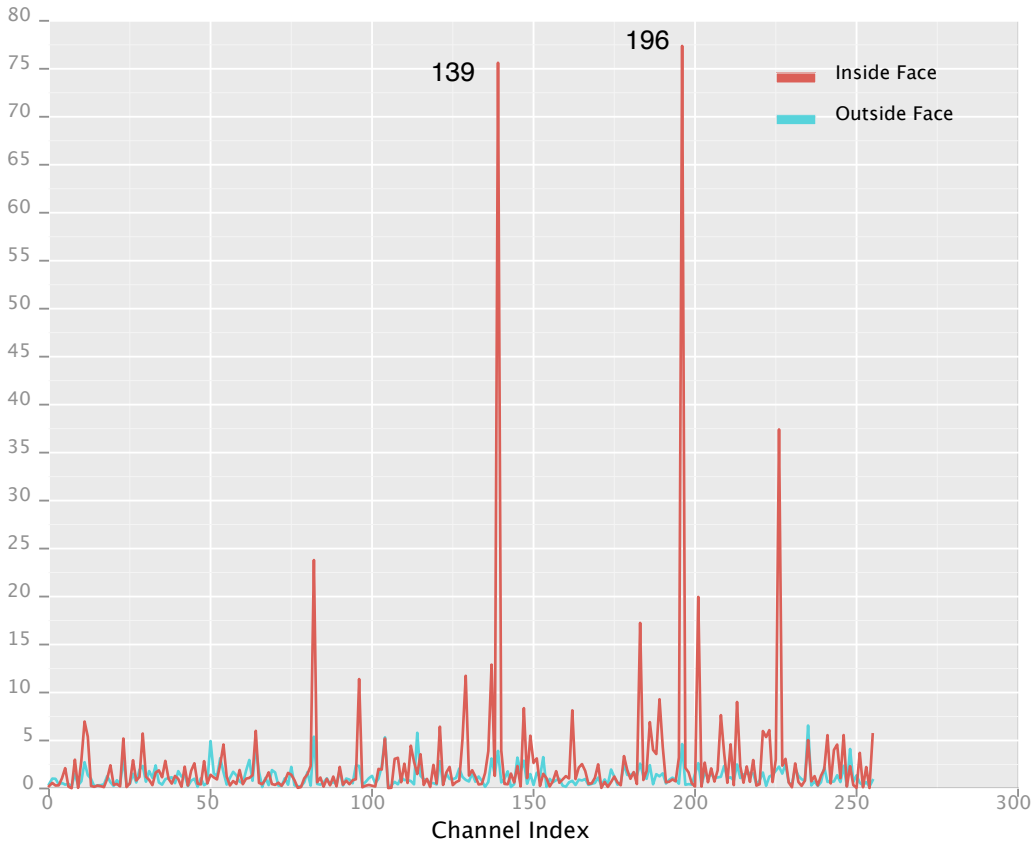


Figure 3.3: Face-scores of inside and outside face area for each channel in conv5 layer, averaged over 1,000 images randomly chosen from ALFW.

3.4.3 Multi-scale Features

From above, we could compute the face heatmap ($conv5_{196}$) and generate the bounding box of potential face regions by a given threshold. As a result, the bounding box could

be used to represent the face position. However, as shown in Figure 3.4, there are two problems if we want to use the heatmap directly for face detection. One is that the heatmap cannot capture small faces, the other is that the strong firing neurons in the heatmap tend to shift to the edge of the image if the face is not in the center of an image. We have developed a multi-scale approach to solving these problems: the input image is divided into small sub-images by a sliding window at multiple scales. We specify the stride, with which the sliding window is moved each time, to be half of the sub-image's size. Thus the adjacent sub-images are able to overlap with each other. In this work, the sizes of the sub-images are 1/4, 1/6 and 1/8 of the size of the original image to achieve multi-scale feature extraction. All sub-images and the original image are then resized and fed to the fine-tuned CNN to extract the channel heatmaps ($conv5_{196}$), which are then merged to be a single heatmap according to the corresponding locations in the original image scale. Specifically, each heatmap of sub-images is extended to the original image scale, and the intensity value outside the sub-image areas are set to be zero. The merging is achieved by selecting the maximum intensity value of each heatmap at each pixel position in the original image scale, i.e., $I_{i,j} = \max_{1 \leq l \leq n} I^l_{i,j}$, where $I_{i,j}$ is the intensity value of the merged heatmap at point (i, j) in the original image scale, and $I^l_{i,j}$ is the intensity value of the l^{th} sub-image's heatmap at point (i, j). As a result (see Figure 3.4), the merged heatmap is able to capture small faces and locate faces in the input image more precisely. This is because small faces in the original scale become "larger" in the sub-image scale, which can be captured by our model. In particular, different scales of sub-images can be used to extract different scales of face response heatmaps, thus faces with different sizes can be detected. However it will increase the amount of computational cost when using different scales of sub-images.

3.4.4 Fast Non-Face Region Filtering

The face heatmap contains information about the likelihood of a pixel belonging to a face region. The higher the intensity the more likely it is from a face region. Therefore, this allows us to quickly filter out non-face regions by simply setting a threshold to the heatmap. Only pixels above a certain threshold will likely contain faces. Therefore, in the face detection stage, we only need to search regions covered by these pixels thus significantly reducing the search space and speeding up face detection. To get an idea about the kind of saving we can achieve, we randomly pick 1,000 images from ALFW and threshold the heatmaps. We count the average number of pixels above the threshold which contains areas of potential faces. At the same time, we also count the average number of faces that are covered by these remaining pixels. Obviously, we

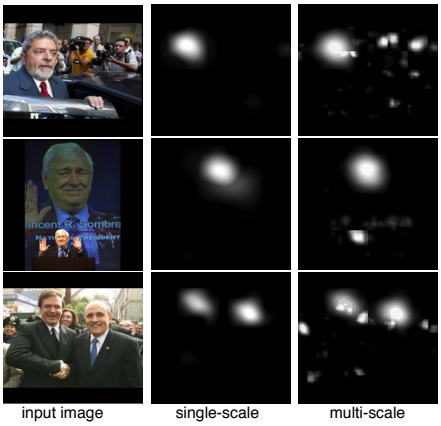


Figure 3.4: Examples for single-scale and multi-scale feature extraction.

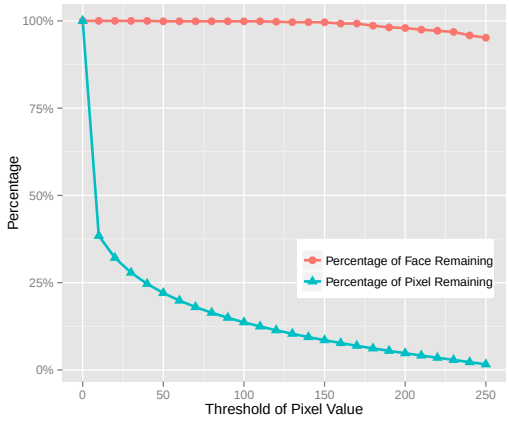


Figure 3.5: Diagram for the average percentage of remaining pixels (blue line) and remaining faces (red line) above different thresholds.

want the number of remaining pixels to be small and at the same time these remaining pixels should cover all genuine faces. Figure 3.5 shows the percentage of pixels above different thresholds, and for each threshold, the percentage of genuine faces covered by the remaining pixels. It is seen that by simply thresholding the heatmap, we can dramatically reduce the search regions without missing genuine faces. For example, by setting a threshold value to about 110, just a little over 10% of the pixels remain and at the same time all faces are covered by these pixels. This means that compared with full search, this procedure will reduce face detection search region by 90%.

3.5 Face Proposals and Detection

The second stage of our method (Figure 3.1(b)) locates faces more precisely and separates overlapped faces which cannot be distinguished in the heatmap. We first use face-score defined above (average intensity of a specific area in a heatmap) to quickly select face proposals. Then another binary CNN classifier is trained to discriminate face images from non-face images. In the end, Non-Maximum Suppression is used to merge multiple face windows for final detection.

3.5.1 Face Proposals by Face-Score

Instead of using category-independent region proposals method like selective search [65] used in R-CNN [33] for general object detection, we simply use a multi-scale sliding window to select potential face windows based on face-score defined above.

Specifically, while scanning the input image, the face-score of the sliding window is calculated at the same position in the corresponding heatmap. The sliding windows are selected as face region proposals if the face-scores exceed a given threshold (80 in our case). This approach can reject non-face regions effectively based on the above observation that the pixel intensity values around the face regions are higher than non-face regions. All potential face proposals of one image are collected as a batch to feed to a binary CNN classifier described below.

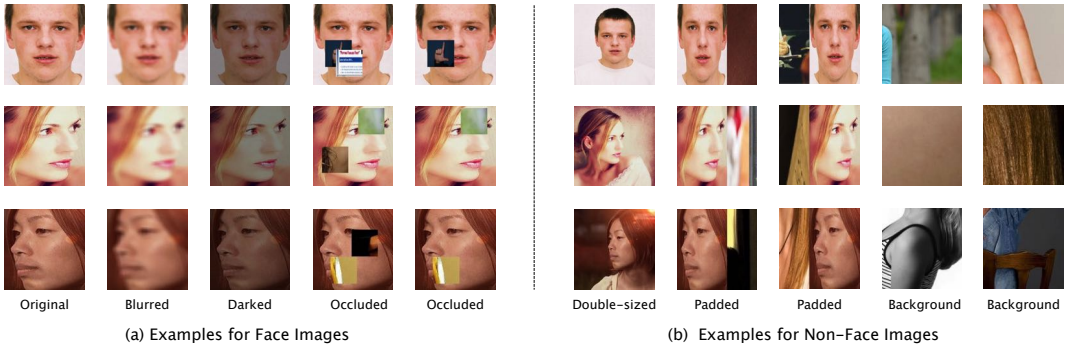


Figure 3.6: Example face images(a) and non-face images (b) from AFLW for face classifier training.

3.5.2 Candidate Face Window Selection by CNN

Due to a range of visual variations such as poses, expressions, lightings and occlusions, a robust face classifier is trained by using augmented data. The fine-tuned CNN model used to extract face response heatmap is used as pre-trained model, and then fine-tuned again with face images and non-face images. All the face images are cropped from AFLW dataset [64] by the ground-truth bounding box, and augmented by making them darkened, blurred and occluded (Figure 3.6(a)). Non-face images are collected in the following ways (Figure 3.6(b)): (1) background images randomly cropped from AFLW images with a given Intersection-over-Union (IoU) ratio (0, 0.1 and 0.2) to a ground-truth face; (2) cropped by double-sized ground-truth faces; (3) face images padded with non-face images. Each candidate face window has a classification score after being processed by the trained binary classifier. Finally all detected face windows are sorted from highest to lowest based on classification score, and then non-maximum suppression (NMS) is applied to the detected windows to reject the window if it has an Intersection-over-Union (IoU) ratio bigger than a given threshold.

3.5.3 Face Detection Experiments

As described above, the AFLW dataset is used to train our model, and then we use PASCAL Face [52] and FDDB dataset [66] to evaluate our face detector. PASCAL Face dataset is a widely used face detection benchmark, containing 851 images and 1,341 annotated faces. FDDB dataset is a larger face detection benchmark, consisting of 5,171 annotated faces in 2,845 images. It contains a wide range of difficulties including occlusions, various poses and out-of-focus faces.

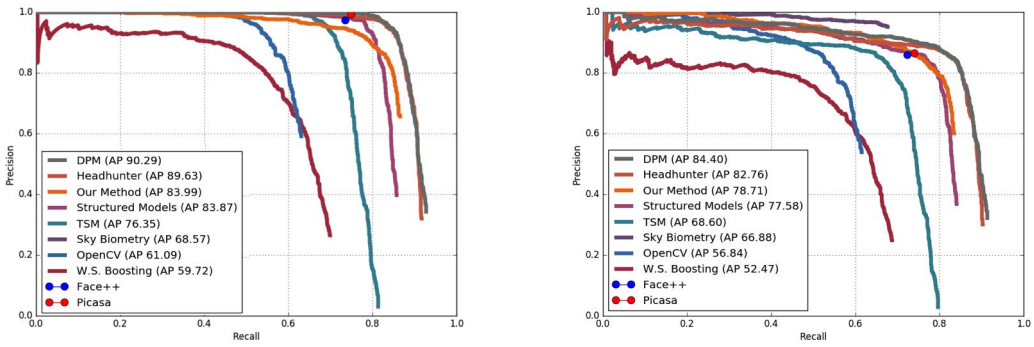
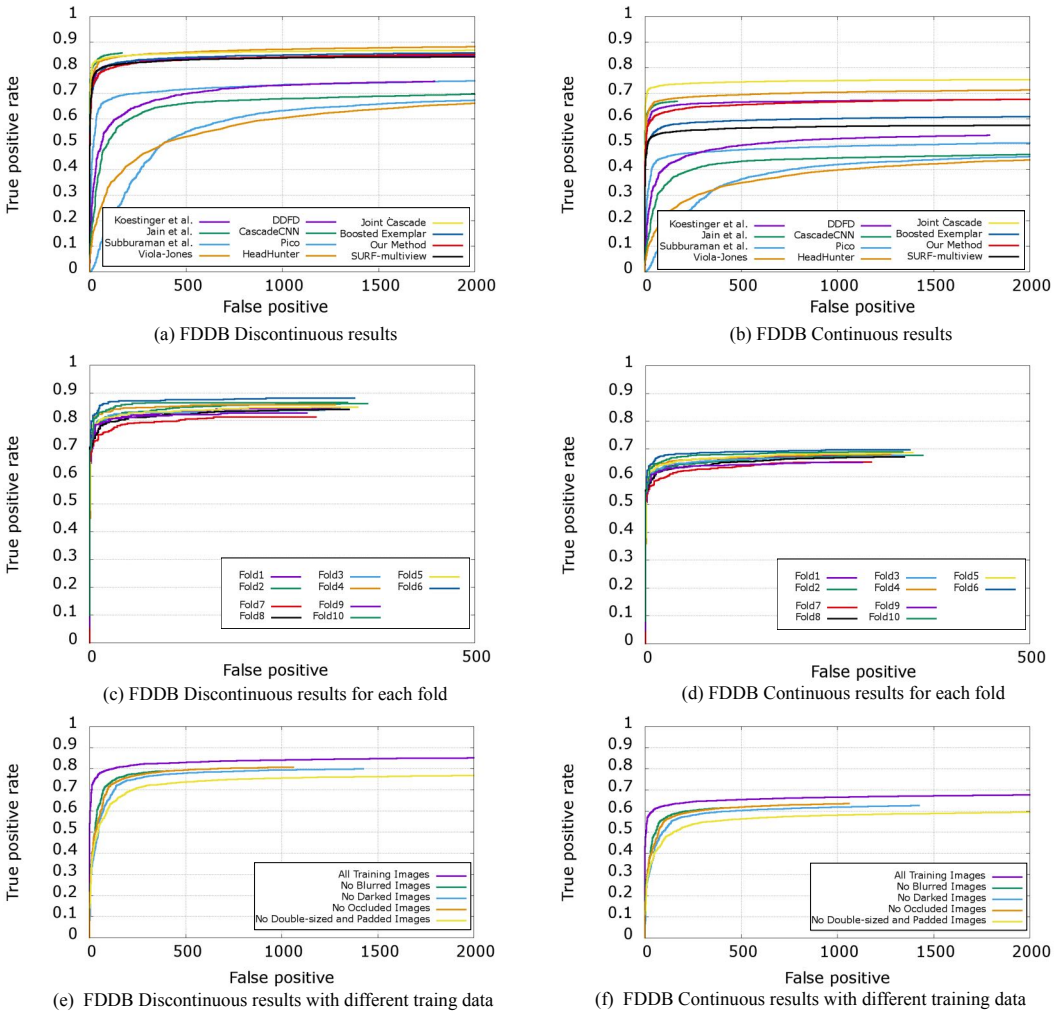


Figure 3.7: Performance comparison on PASCAL Face dataset for the original (left) and adjusted annotations (right).

One problem in the evaluation of face detection is the different annotations between training datasets and testing datasets such as policies for what constitutes a face, size of annotation boxes and minimum/maximum of face size. In order to solve this problem, some works [50, 53, 60] try to manually adjust the annotations to get better results. In our work, we use both the original and adjusted annotations [53] of PASCAL Face dataset with the toolbox provided by [53]. FDDB dataset is evaluated with the original elliptical annotations by two evaluation protocols provided in [66]: the continuous score and discontinuous score. In order to better fit the elliptical annotations that cover the whole faces, we extend our detected square boxes vertically by 40% to upright rectangles for FDDB dataset. We also fit the largest upright ellipses for the extended rectangles as elliptical outputs for evaluation (Figure 3.9).

We report the average precision on PASCAL Face dataset (Figure 3.7), discontinuous and continuous ROC for all the 10 folds (Figure 3.8 (a) and (b)) and individual ROC (Figure 3.8 (c) and (d)) for each fold on FDDB dataset. By comparing with other state-of-the-art face detectors [48, 54, 60, 61, 67–70], our method can achieve similar results both on PASCAL faces [52] and FDDB [66] datasets while having lower complexity.

In addition, we design several comparative experiments to investigate the impact of data augmentation. The CNN binary classifier in the second stage is retrained with a



(a) FDDB Discontinuous results

(b) FDDB Continuous results

(c) FDDB Discontinuous results for each fold

(d) FDDB Continuous results for each fold

(e) FDDB Discontinuous results with different traing data

(f) FDDB Continuous results with different training data

Figure 3.8: Performance comparison on FDDB dataset with discrete and continuous protocols.

slightly different training dataset, while all the other components are left unchanged. For face images, we remove the darkened, blurred and occluded face images (Figure 3.6(a)) respectively, and then the padded and double-sized images (Figure 3.6(b)) are removed from non-face images. As before, all the newly trained models are tested on FDDB dataset as shown in Figure 3.8(e) and (f). We can see that the augmented training dataset has a significant influence on the performance of the detector. By removing one type of augmented training data, there is an obvious drop in the performance of detection. For example, the padded and double-sized non-face images could help the model locate face area more precisely, i.e., neither too big nor too small, thus bounding box regression is not needed for post-processing. Therefore, better data augmentation makes full use of the high-capacity of CNN to achieve better performance. In our case, all kinds of facial variations can be represented in a single CNN model. Some qualitative results on PASCAL Face and FDDB dataset together with the face response

Figure 3.9: Qualitative face detection results and face response heatmap on PASCAL face (top three rows) and FDDB dataset (bottom three rows).

heatmaps are shown in Figure 3.9. It is seen that our detectors can successfully detect faces in challenging settings.

3.6 Discussion

The contribution of this work is not only on the face detector itself, but rather the introduction of a face specific channel in a deep neural network that can be easily exploited for face detection. This is the first work that introduces the concept of object specific channel in a deep convolutional neural network. Although we use face as a specific case study, our method does not depend on any human knowledge about faces or any special processing for faces. Our method could be extended to other classes of objects in a similar way to turn the convolutional feature maps of a CNN into object specific feature channels for object detection and other computer vision tasks.

Compared to many methods in the literature, our method is much simpler: ground-truth bounding boxes are the only information needed to train our model, and one single model can capture all the facial variations based on the carefully designed data augmentation. In contrast, Faceness [57] needs additional hair, eye, nose, mouth and beard annotations to train several attribute-aware face models and uses bounding box regression to refine detected windows. DPM and HeadHunter [53] use extra annotation to train view-specific components to tackle facial variations. JointCascade [68] uses face alignment to help face detection with manually labeled 27 facial points. Our method only needs a single feed-forward calculation to achieve fast multi-scale face detection for a batch size of images in an end-to-end manner without any post-processing like bounding box regression. The runtime of our model is around 70 ms on a single GPU (Nvidia Tesla k40) for a 640×480 VGA image.

3.7 Summary

In this chapter, we have developed a method to exploit the internal representation power of hidden units of a deep convolutional neural network. We seek to discover and exploit the convolutional channels of a CNN in which neurons are activated by the presence of specific objects in the input image. Through a purposefully designed face specific training, object specific channels are extracted and automatically identified for human faces. Building on the object specific channels, a multi-scale approach is used to build state of the art face detectors with the advantage of being simple and compact.

CHAPTER 4

Image Generation

In this chapter, we develop *generative models* to produce visually pleasing images from a latent space. A generative model trained with a given dataset can be used to generate data having similar properties as the samples in the dataset, learning the internal essence of the dataset and "storing" all the information in the limited parameters that are significantly smaller than the training dataset. In particular, variational autoencoder (VAE) [13] and generative adversarial networks (GAN) [15] are the two generative models considered in this work. Although the two approaches have achieved some success in generating natural images of the real world in the original work, the samples often suffer from being blurry for VAE and being noisy and incomprehensible for GANs. To overcome these limitations, we try to construct objective functions to train generative models by incorporating deep features extracted from pretrained deep convolutional neural networks. Our experiments demonstrate that our models can produce high quality images and the learned latent representations that can capture the semantic information. The rest of this chapter is organized around two projects that leverage the capability of these models to generate high quality images. We will introduce two improved models for variational autoencoder and generative adversarial network in turn.

4.1 Variational Autoencoder (VAE)

Variational Autoencoder [13, 14] has become a popular generative model, allowing us to formalize image generation task in the framework of probabilistic graphical models with latent variables. A VAE [13] helps us to do two things. Firstly it allows us to encode an image x to a latent vector $z = Encoder(x) \sim q(z|x)$ with an encoder network, and then a decoder network is used to decode the latent vector z back to an image that

will be as similar as the original image $\bar{x} = Decoder(z) \sim p(x|z)$. That's to say, we need to maximize the marginal log-likelihood of each observation (pixel) in x, and the VAE reconstruction loss $\mathcal{L}_{rec}$ is the negative expected log-likelihood of the observations in x. Another important property of VAE is the ability to control the distribution of the latent vector z, which has characteristic of being independent unit Gaussian random variables, i.e., $z \sim \mathcal{N}(0, I)$. Moreover, the difference between the distribution of $q(z|x)$ and the distribution of a Gaussian distribution (called KL Divergence) can be quantified and minimized by gradient descent algorithm [13]. Therefore, VAE models can be trained by minimizing the sum of the reconstruction loss ($\mathcal{L}_{rec}$) and KL divergence loss ($\mathcal{L}_{kl}$) by gradient descent.

$$\mathcal{L}_{rec} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] \tag{4.1.1}$$

$$\mathcal{L}_{kl} = D_{kl}(q(z|x)||p(z)) \tag{4.1.2}$$

$$\mathcal{L}_{vae} = \mathcal{L}_{rec} + \mathcal{L}_{kl} \tag{4.1.3}$$

4.1.1 Related Work

Several methods have been proposed to improve the performance of VAE. [71] extends the variational auto-encoders to semi-supervised learning with class labels, [72] proposes a variety of attribute-conditioned deep variational auto-encoders, and demonstrates that they are capable of generating realistic faces with diverse appearance, Deep Recurrent Attentive Writer (DRAW) [73] combines spatial attention mechanism with a sequential variational auto-encoding framework that allows iterative generation of images. Considering the shortcoming of pixel-by-pixel loss, [74] replaces pixel-by-pixel loss with multi-scale structural-similarity score (MS-SSIM) and demonstrates that it can better measure human perceptual judgments of image quality. [75] proposes to enhance the objective function with discriminative regularization. Another approach [76] tries to combine VAE and generative adversarial network (GAN) [15, 77], and use the learned feature representation in the GAN discriminator as the basis for the VAE reconstruction objective.

By default, pixel-by-pixel measurement like L_2 loss, or logistic regression loss is used to measure the difference between the reconstructed and the original images. Such measurements are easily implemented and efficient for deep neural network training. However, the generated images tend to be very blurry when compared to natural images. This is because the pixel-by-pixel loss does not capture the perceptual difference and spatial correlation between two images. For example, the same image offsetted

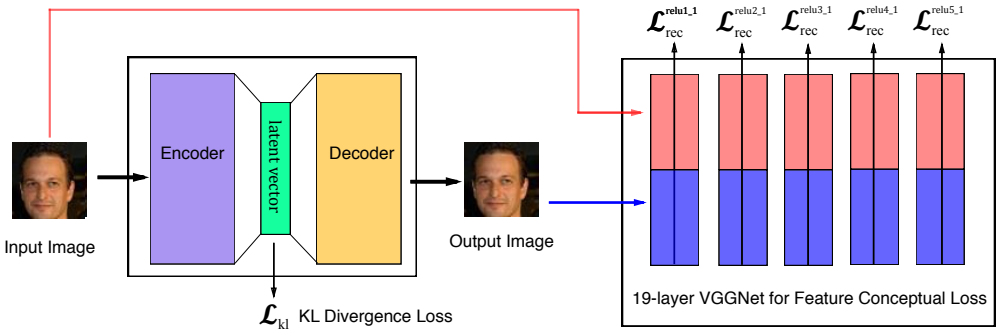


Figure 4.1: Model overview. The left is a deep CNN-based Variational Autoencoder and the right is a pretrained deep CNN used to compute feature perceptual loss.

by a few pixels will have little visually perceptual difference for humans, but it could have a very high pixel-by-pixel loss. This is a well known problem in the image quality measurement community [78].

In this chapter, we propose replacing the pixel-by-pixel loss with feature perceptual loss, which is defined as the difference between two images' hidden representations extracted from a pretrained deep CNN such as AlexNet [23] and VGGNet [24] trained on ImageNet [12]. The main idea is trying to improve the quality of generated images of a VAE by ensuring the consistency of the hidden representations of the input and output images, which in turn imposes spatial correlation consistency of two images.

4.1.2 Model

Overview. Our system consists of two main components as shown in Figure 4.1: an autoencoder network including an encoder network $E(x)$ and a decoder network $D(z)$, and a loss network Φ that is a pretrained deep CNN to define feature perceptual loss. An input image x is encoded as a latent vector $z = E(x)$, which will be decoded back to image space $\bar{x} = D(z)$. After training, a new image can be generated by the decoder network with a given vector z. In order to train a VAE, we need two losses, one is KL divergence loss $\mathcal{L}_{kl} = D_{kl}(q(z|x)||p(z))$ [13], which is used to make sure that the latent vector z is an independent unit Gaussian random variable. The other is feature perceptual loss. Instead of directly comparing the input image and the generated image in the pixel space, we feed both of them to a pre-trained deep CNN Φ respectively and then measure the difference between hidden layer representations, i.e., $\mathcal{L}_{rec} = \mathcal{L}^1 + \mathcal{L}^2 + ... + \mathcal{L}^l$, where $\mathcal{L}^l$ represents the feature loss at the l^{th} hidden

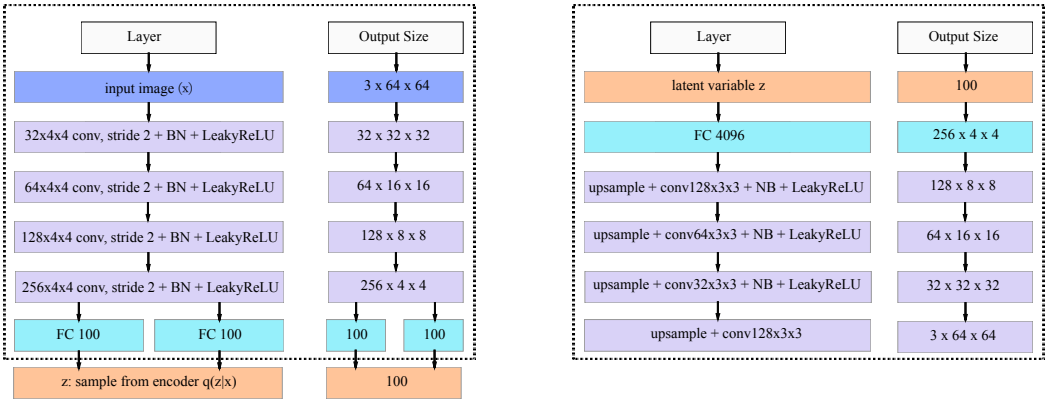


Figure 4.2: Autoencoder network architecture. The left is encoder network and the right is decoder network.

layer. Thus, we use the high-level feature loss to better measure the perceptual and semantic differences between the two images, this is because the pretrained network on image classification has already incorporated perceptual and semantic information we desired for. During the training, the pretrained loss network is fixed and just for high-level feature extraction, and the KL divergence loss $\mathcal{L}_{kl}$ is used to update the encoder network while the feature perceptual loss $\mathcal{L}_{rec}$ is responsible for updating parameters of both the encoder and decoder.

Variational autoencoder network architecture. Both encoder and decoder network are based on deep CNN like AlexNet [23] and VGGNet [24]. We construct 4 convolutional layers in the encoder network with 4 x 4 kernels, and the stride is fixed to be 2 to achieve spatial downsampling instead of using deterministic spatial functions such as maxpooling. Each convolutional layer is followed by a batch normalization layer and a LeakyReLU activation layer. Then two fully-connected output layers (for mean and variance) are added to the encoder, and will be used to compute the KL divergence loss and sample latent variable z (see [13] for details). For decoder, we use 4 convolutional layers with 3 x 3 kernels and set stride to be 1, and replace standard zero-padding with replication padding, i.e., feature map of an input is padded with the replication of the input boundary. For upsampling we use nearest neighbor method by a scale of 2 instead of fractional-strided convolutions used by other works [8, 77]. We also use batch normalization to help stabilize training and use LeakyReLU as the activation function. The details of autoencoder architecture are shown in Figure 4.2.

4.1.3 Feature Perceptual Loss

Feature perceptual loss of two images is defined as the difference between the hidden features in a pretrained deep convolutional neural network Φ. Similar to [35], we use VGGNet [24] as the loss network in our experiment, which is trained for classification problem on ImageNet dataset. The core idea of feature perceptual loss is to seek the consistency between the hidden representations of two images. As the hidden representations can capture important perceptual quality features such as spatial correlation, a smaller difference of hidden representations indicates the consistency of spatial correlations between the input and the output, as a result, we can get a better visual quality of the output image.

Specifically, let $\Phi(x)^l$ represent the representation of the l^{th} hidden layer when input image x is fed to network Φ. Mathematically $\Phi(x)^l$ is a 3D volume block array of the shape [C^l x W^l x H^l], where C^l is the number of filters, W^l and H^l represent the width and height of each feature map for the l^{th} layer. The feature perceptual loss for one layer ($\mathcal{L}^l_{rec}$) between two images x and $\bar{x}$ can be simply defined by squared Euclidean distance. Actually it is quite like pixel-by-pixel loss for images except that the color channel is not 3 anymore.

$$\mathcal{L}^l_{rec} = \frac{1}{2C^l W^l H^l} \sum_{c=1}^{C^l} \sum_{w=1}^{W^l} \sum_{h=1}^{H^l} (\Phi(x)^l_{c,w,h} - \Phi(\bar{x})^l_{c,w,h})^2 \tag{4.1.4}$$

The final reconstruction loss is defined as the total loss by combining different layers of VGG Network, i.e., $\mathcal{L}_{rec} = \sum_l \mathcal{L}^l_{rec}$. Additionally we adopt the KL divergence loss $\mathcal{L}_{kl}$ [13] to regularize the encoder network to control the distribution of the latent variable z. To train VAE, we jointly minimize the KL divergence loss $\mathcal{L}_{kl}$ and the feature perceptual loss $\mathcal{L}^l_{rec}$ for different layers, i.e.,

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{kl} + \beta \sum_{i}^{l} (\mathcal{L}^l_{rec}) \tag{4.1.5}$$

where α and β are weighting parameters for KL Divergence and image reconstruction. It is quite similar to style transfer [35] if we treat KL Divergence as style reconstruction.

4.1.4 Experiment

We perform experiments on face images to test our method. Specifically we evaluate the image generation performance and compare with other generative models. Fur-

Figure 4.3: Generated fake face images from 100-dimension latent vector $z \sim \mathcal{N}(0, 1)$ from different models. The first part is generated from the decoder network of plain variational autoencoder (PVAE) trained with pixel-based loss [13], the second part is generated from generator network of DCGAN [77], and the last two parts are the results of VAE-123 and VAE-345 trained with feature perceptual loss based on layers relu1_1, relu2_1, relu3_1, and relu3_1, relu4_1, relu5_1 respectively.

thermore, we also investigate the latent space and study the semantic relationship between different latent representations and apply them to facial attribute prediction.

Training details. Our model is trained on CelebFaces Attributes (CelebA) Dataset [79]. CelebA is a large-scale face attribute dataset with 202,599 face images, 5 landmark locations, and 40 binary attributes annotations per image. We build the training dataset by cropping and scaling the aligned images to 64 x 64 pixels like [76, 77]. We train our model with a batch size of 64 for 5 epochs over the training dataset and use Adam method for optimization [32] with an initial learning rate of 0.0005, which is decreased by a factor of 0.5 for the following epochs. The 19-layer VGGNet [24] is chosen as loss network Φ to construct feature perceptual loss for image reconstruction.

We experiment with different layer combinations to construct feature perceptual loss

Figure 4.4: Image reconstruction from different models. The first row is input image, the second row is generated from decoder network of plain variational autoencoder (PVAE) trained with pixel-based loss [13], and the last two rows are the results of VAE-123 and VAE-345 trained with feature perceptual loss based on layers relu1_1, relu2_1, relu3_1, and relu3_1, relu4_1, relu5_1 respectively.

and train two models, i.e., VAE-123 and VAE-345, by using layers relu1_1, relu2_1, relu3_1 and relu3_1, relu4_1, relu5_1 respectively. In addition, the dimension of latent vector z is set to be 100 like DCGAN [77], and the loss weighting parameters α and β are 1 and 0.5 respectively. Our implementation is built on deep learning framework Torch [40].

In this work, we also train additional two generative models for comparison. One is the plain Variational Autoencoder (PVAE), which has the same architecture as our proposed model, but trained with pixel-by-pixel loss in the pixel space. The other is Deep Convolutional Generative Adversarial Networks (DCGAN) consisting of a generator and a discriminator network [77], which has shown the ability to generate high quality images from noise vectors. DCGAN is trained with open source code [77] in Torch.

Qualitative results for image generation. The comparison is divided into two parts: arbitrary face images generated by the decoder based on latent vectors z drawn from $\mathcal{N}(0, 1)$, and face image reconstruction.

In the first part, random face images (shown in Figure 4.3) are generated by feeding latent vector z drawn from $\mathcal{N}(0,1)$ to the decoder network in our models and the generator network in DCGAN respectively. We can see that the generated face images by plain VAE tend to be very blurry, even though the overall spatial face structure can be preserved. It is very hard for plain VAE to generate clear facial parts such as eyes and noses, this is because it tries to minimize the pixel-by-pixel loss between two images. The pixel-based loss does not contain the perceptual and spatial correlation information. DCGAN can generate clean and sharp face images containing clearer facial features, however it has the facial distortion problem and sometimes generates weird faces. Our method based on feature perceptual loss can achieve better results. VAE-123 can generate faces of different genders, ages with clear noses, eyes and teeth, which are better than VAE-345. However, VAE-345 is better at generating hair with different textures.

We also compare the reconstruction results (shown in Figure 4.4) between plain VAE and our two models, and DCGAN is not compared because of no input image in their model. We can get a similar conclusion as above. In addition, VAE-123 is better at keeping the original color of input images and generating clearer eyes and teeth. The VAE-345 can generate face images with more realistic hair, but the color could be different from the original in the input images. This is because VAE-345 is trained with higher hidden layers of VGGNet and captures spatial correlation on a coarser scale than VAE-123, hence the images generated by VAE-345 are more blurry than those of VAE-123. Additionally as textures such as hair reflects larger area correlations, this may explain why VAE-345 generates better textures than VAE-123. The other way around, local patterns like eyes and noses reflect smaller area correlations, thus VAE-123 can generate clearer eyes and noses than VAE-345.

Linear interpolation of latent space. As shown in Figure 4.5, we investigate the linear interpolation between the generated images from two latent vectors denoted as z_{left} and z_{right}. The interpolation is defined by linear transformation $z = (1-\alpha)z_{left} + \alpha z_{right}$, where $\alpha = 0, 0.1, \ldots, 1$, and then z is fed to the decoder network to generate new face images. Here we show three examples for latent vector z encoded from input images and one example for z randomly drawn from $\mathcal{N}(0,1)$. From the first row in Figure 4.5, we can see the smooth transitions between *vector*("Woman without smiling and short hair") and *vector*("Woman with smiling and long hair"). Little by little the hair becomes longer, the distance between lips becomes larger and teeth are shown in the end as smiling, and pose turns from looking slightly right to looking front. Additionally we provide examples of transitions between *vector*("Man without eyeglass")

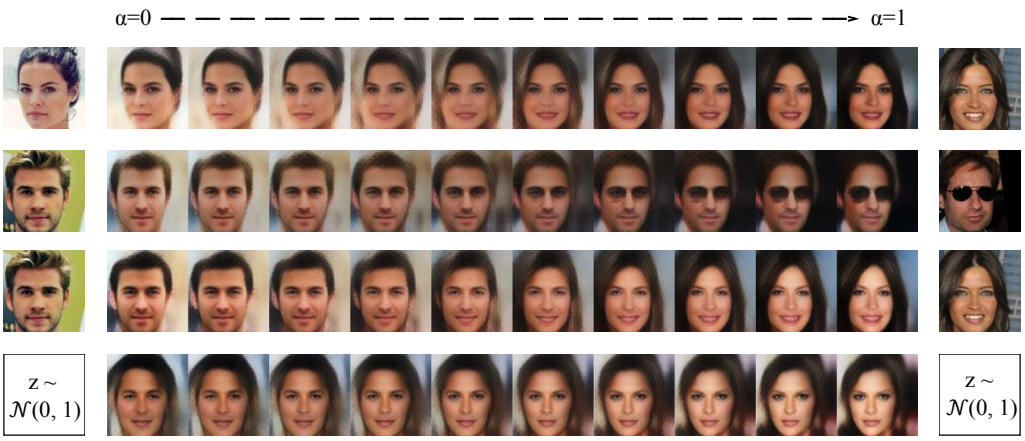


Figure 4.5: Linear interpolation for latent vector. Each row is the interpolation from left latent vector z_{left} to right latent vector z_{right}. e.g. $(1-\alpha)z_{left} + \alpha z_{right}$. The first row is the transition from a non-smiling woman to a smiling woman, the second row is the transition from a man without eyeglass to a man with eyeglass, the third row is the transition from a man to a woman, and the last row is the transition between two fake faces decoded from $z \sim \mathcal{N}(0, 1)$.

and *vector*("Man with eyeglass"), and *vector*("Man") and *vector*("Woman").

Facial attribute manipulation. The experiments above demonstrate interesting smooth transitional property between two learned latent vectors. In this part, instead of manipulating the overall face images, we seek to find a way to control a specific attribute of face images. In previous works, [80] shows that *vector*("King") - *vector*("Man") + *vector*("Woman") generates a vector whose nearest neighbor is the *vector*("Queen") when evaluating learned representation of words. [77] demonstrates that visual concepts such as face pose and gender could be manipulated by simple vector arithmetic. In this work, we investigate two facial attributes wearing eyeglass and smiling. We randomly choose 1,000 face images with eyeglass and 1,000 without eyeglass respectively from the CelebA dataset [79]. The two types of images are fed to our encoder network to compute the latent vectors, and the mean latent vectors are calculated for each type respectively, denoted as $z_{pos\_eyeglass}$ and $z_{neg\_eyeglass}$. We then define the difference $z_{pos\_eyeglass} - z_{neg\_eyeglass}$ as eyeglass-specific latent vector $z_{eyeglass}$. In the same way, we calculate the smiling-specific latent vector $z_{smiling}$. Then we apply the two attribute-specific vectors to different latent vectors z by simple vector arithmetic like $z + \alpha\, z_{smiling}$. As shown in Figure 4.6, by adding a smiling vector to the latent vector of a non-smiling man, we can observe the smooth transitions from non-smiling face to smiling face (the first row). Furthermore, the smiling appearance becomes more obvious

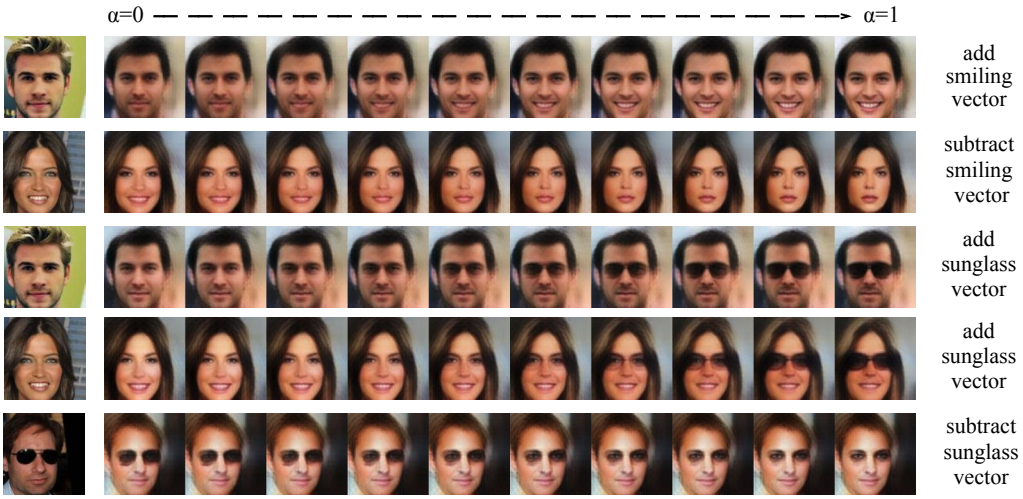


Figure 4.6: Vector arithmetic for visual attributes. Each row is the generated faces from latent vector z_{left} by adding or subtracting an attribute-specific vector, i.e., $z_{left} + \alpha\ z_{smiling}$, where $\alpha = 0, 0.1, \ldots, 1$. The first row is the transition by adding a smiling vector with a linear factor α from left to right, the second row is the transition by subtracting a smiling vector, the third and fourth row are the results by adding a eyeglass vector to the latent representation for a man and women, and the last row shows results by subtracting an eyeglass vector.

when the factor α is bigger, while other facial attributes are able to remain unchanged. The other way around, when the latent vector of smiling woman is subtracted by the smiling vector, the smiling face can be translated to not smiling by only changing the shape of mouth (the second row in Figure 4.6). Moreover, we could add or wipe out an eyeglass by playing with the calculated eyeglass vector.

Correlation between attribute-specific vectors. Considering the conceptual relationship between different facial attributes in natural images, for instance, bald and gray hair are often related to old people. We selected 13 of 40 attributes from CelebA dataset and calculate their attribute-specific latent vectors respectively (the calculation is the same as calculating eyeglass-specific vector above). We then calculate the correlation matrix (Pearson's correlation) of the 13 attribute-specific vectors, and visualize it as shown in Figure 4.7. The results are consistent with human interpretation. We can see that *Attractive* has a strong positive correlation with *Makeup*, and a negative correlation with *Male* and *Gray Hair*. It makes sense that female is generally considered more attractive than male and uses more makeup. Similarly, *Bald* has a positive correlation with *Gray Hair* and *Eyeglasses*, and a negative correlation with *Young*. Additionally, *Smiling* seems to have no correlation with most of other attributes and only has a weak negative correlation with *Pale Skin*. It could be explained that *Smiling* is

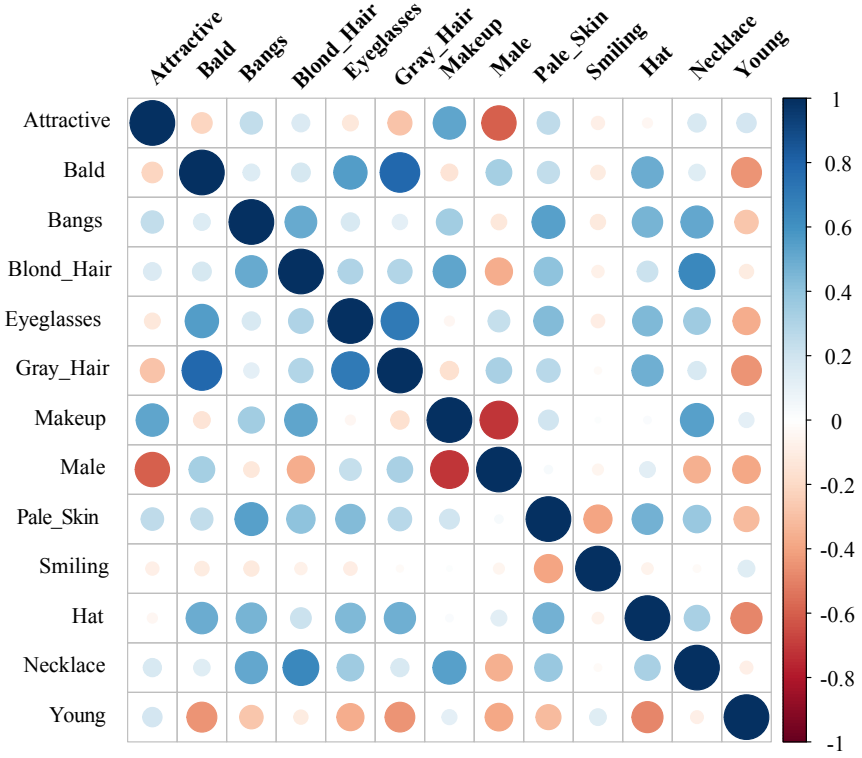


Figure 4.7: Diagram for the correlation between selected facial attribute-specific vectors. The blue indicates positive correlation, while red represents negative correlation, and the color shade and size of the circle represent the strength of the correlation.

a very common human facial expression and it could have a good match with many other attributes.

Visualization of latent vectors. Considering that the latent vectors are nothing but the encoding representation of the natural face images, we think that it may be interesting to visualize the natural face images based on the similarity of their latent representations. Specifically we randomly choose 1600 face images from CelebA dataset and extract the corresponding 100-dimensional latent vectors, which are then reduced to 2-dimensional embedding by t-SNE algorithm [81]. t-SNE can arrange images that have similar high-dimensional vectors (L_2 distance) to be nearby each other in the embedding space. The visualization of 40 x 40 images is shown in Figure 4.8. We can see that images with a similar background (black or white) tend to be clustered as a group, and females with smiling are clustered together (green rectangle in Figure 4.8). Furthermore, the face pose information can be also captured even no pose annotations in the dataset. The face images in the upper left (blue rectangle) are those looking to the right and samples in the bottom left (red rectangle) are those looking to the left, while in other area the faces look to the front.

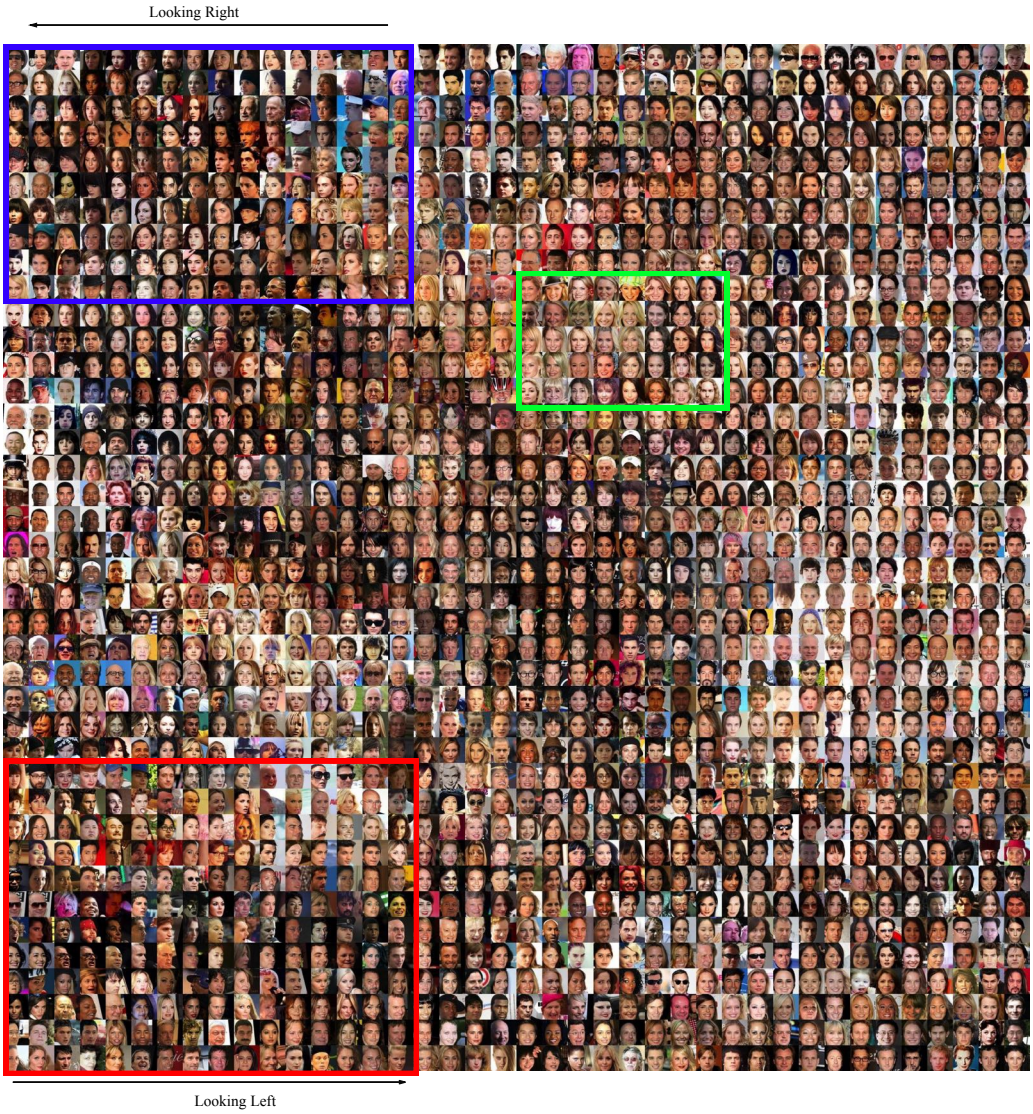


Figure 4.8: Visualization of 40 x 40 face images based on latent vectors by t-SNE algorithm [81].

Facial attribute prediction. We further evaluate our model by applying the latent vector to facial attribute prediction, which is a very challenging problem. Similar to [79], 20,000 images from CelebA dataset are selected for testing and the rest for training. Instead of using a face detector, we use ground truth landmark points to crop out the face parts of the original images like PANDA-l [83], and the cropped face images are fed to our encoder network to extract the latent vectors for both VAE-123 and VAE-345, which are then used to train standard Linear SVM [84] classifiers with the corresponding 40 binary attributes annotations per image provide by CelebA. As a result, we train 40 binary classifiers for each attribute in CelebA dataset respectively. As a baseline, we also train different Linear SVM classifiers for each attribute with 4096-dimensional

Method	5 Shadow	Arch. Eyebrows	Attractive	Bags Un. Eyes	Bald	Bangs	Big Lips	Big Nose	Black Hair	Blond Hair	Blurry	Brown Hair	Bushy Eyebrows	Chubby	Double Chin	Eyeglasses	Goatee	Gray Hair	Heavy Makeup	H. Cheekbones	Male
FaceTracer	85	76	78	76	89	88	64	74	70	80	81	60	80	86	88	98	93	90	85	84	91
PANDA-w	82	73	77	71	92	89	61	70	74	81	77	69	76	82	85	94	86	88	84	80	93
PANDA-l	88	78	**81**	79	96	92	67	75	85	93	86	77	86	86	88	98	93	94	90	86	97
LNets+ANet	**91**	79	**81**	79	**98**	**95**	68	78	**88**	**95**	84	**80**	**90**	91	92	**99**	**95**	**97**	**90**	**87**	**98**
VAE-123	89	77	75	81	**98**	91	76	79	83	92	**95**	**80**	87	**94**	95	96	94	96	85	81	90
VAE-345	89	**80**	78	**82**	**98**	**95**	**77**	**81**	85	93	**95**	**80**	88	**94**	**96**	**99**	**95**	**97**	89	85	95
VGG-FC	83	71	68	73	97	81	51	77	78	88	94	67	81	93	93	95	93	94	79	64	84

Method	Mouth S. O.	Mustache	Narrow Eyes	No Beard	Oval Face	Pale Skin	Pointy Nose	Reced. Hairline	Rosy Cheeks	Sideburns	Smiling	Straight Hair	Wavy Hair	Wear. Earrings	Wear. Hat	Wear. Lipstick	Wear. Necklace	Wear. Necktie	Young	Average
FaceTracer	87	91	82	90	64	83	68	76	84	94	89	63	73	73	89	89	68	86	80	81.13
PANDA-w	82	83	79	87	62	84	65	82	81	90	89	67	76	72	91	88	67	88	77	79.85
PANDA-l	**93**	93	84	93	65	91	71	85	87	93	**92**	69	77	78	96	**93**	67	91	84	85.43
LNets+ANet	92	95	81	**95**	66	91	72	89	90	**96**	**92**	73	**80**	82	**99**	**93**	71	**93**	**87**	87.30
VAE-123	80	**96**	**89**	88	73	**96**	73	**92**	**94**	95	87	79	74	82	96	88	**88**	**93**	81	86.95
VAE-345	88	**96**	**89**	91	**74**	**96**	**74**	**92**	**94**	**96**	91	**80**	79	**84**	98	91	**88**	**93**	84	**88.73**
VGG-FC	60	93	87	84	66	**96**	58	86	93	85	65	68	70	49	98	82	87	89	74	79.85

Table 4.1: Performance comparison of 40 facial attributes prediction. The accuracies of FaceTracer [82], PANDA-w [83], PANDA-l [83], and LNets+ANet [79] are collected from [79]. PANDA-l, VAE-123, VAE-345 and VGG-FC use the truth landmarks to get the face part.

deep features extracted from the last fully connected layer of pretrained VGGNet [24].

We then compare our method with other state-of-the-art methods. The average of prediction accuracies of FaceTracer [82], PANDA-w [83], PANDA-l [83], and LNets+ANet [79] are 81.13, 79.85, 85.43 and 87.30 percent respectively. The results of our method with latent vectors of VAE-123 and VAE-345 are 86.95 and 88.73 respectively, whilst that of VGG last layer features (VGG-FC) is 79.85. From Table 4.1, we can see that our method VAE-345 outperforms other methods. In addition, we notice that all the methods can achieve a good performance to predict *Bald*, *Eyeglasses* and *Wearing_Hat* while it is difficult for them to correctly predict attributes like *Big_Lips* and *Oval_Face*. It might be explained that attributes like *wearing_hat* and *Eyeglasses* are much more obvious in face images, than attributes like *big_lips* and *Oval_face*, and the extracted features are not able to capture such subtle differences. Future work is needed to find a way to extract better features which can also capture these tiny variations of facial attributes. In addition, we discover that though some attributes like necklaces and hats are not visible due to the original images has been cropped tightly to the face, these attributes can be still predicted correctly to some degree. This is because the cropped training images are labelled with different attributes even though they are not visible. Moreover some attributes like Necklace and Blond Hair can coexist in the same image,

thus training with one attribute (Blond Hair) can be useful for other attributes prediction (Necklace).

4.2 Generative Adversarial Network (GAN)

Now we move to generative adversarial network, which is another popular generative model first introduced by [15] based on minmax game. Two networks are simultaneously trained: a generator network G tries to map a noise variable $p_z(z)$ to data space $G(z)$ to generate new samples, and a discriminator network D aims to distinguish between the samples from the true data and generated samples by maximizing the probability of assigning the correct label for each category. The generator network G is trained simultaneously to minimize $log(1 - D(G(z)))$ by playing against the adversarial discriminator network D. Thus the minmax game between G and D can be formulated as follows:

$$\min_G \max_D L(D, G) = \mathbb{E}_x[log(D(x))] + \mathbb{E}_z[log(1 - D(G(z)))] \tag{4.2.1}$$

where D and G are the discriminator and generator network respectively, z is the random vector sampled from a Gaussian distribution, and x is the generated image.

4.2.1 Related Work

There are some following works [77, 85–88] trying to improve GAN either focusing on generating clearer and better perceptual quality images, or learning better representation for semi-supervised learning. LAPGAN [85] adopts a cascade of convolutional networks with a Laplacian pyramid framework to generate high quality images. [77] proposes deep convolutional generative adversarial networks known as DCGAN that contains certain architectural constrains, and can learn a hierarchy of representations both in the generator and discriminator. Several improved techniques are also introduced by [87] to encourage convergence of the GANs game, and improve the performance of semi-supervised classification problems. Additionally InfoGAN [88] can learn interpretable representations by maximizing the mutual information between a small subset of the latent variables and the observations.

In this work, we adopt deep feature matching strategy to improve the visual quality of the generated images and the stability of GANs training. We propose to distinguish deep features of the real and generated images to train the discriminator and generator network. That's to say, our discriminator is able to distinguish the deep features of

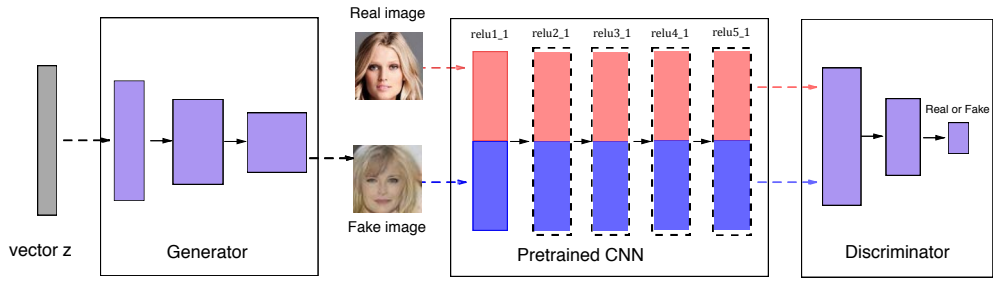


Figure 4.9: Model overview. The left is a generator network to produce fake images from a noise vector z, the middle is a pretrained CNN to extract features for both real and fake images, and the right is a discriminator network to distinguish the deep features of real images from fake images.

generated samples from the ones of real samples, meanwhile the generator is capable of producing fake samples that have similar deep features to the real samples.

4.2.2 Model

Our GAN system consists of three components as shown in Figure 4.9: A generator network $G(z)$ used to generate fake samples from a given noise vector z, a feature extraction network Φ used to extract deep features, and a discriminator network $D(x)$ designed to be a binary classifier. During the training, the generator (G) transforms a noise vector to a fake image through in-network sampling. Unlike traditional GAN that directly feeds the raw real images and fake images to a discriminator, our method first preprocesses the images with a pretrained deep CNN to extract the corresponding features, which are then fed to the discriminator network. Because the convolutional features are of a 3-dimensional array, they could be interpreted as "images" except that the color channel is not 3 anymore.

Generator network. Our generator and discriminator network are based on the architectural innovations of DCGAN [77]. For generator, fractionally-strided convolution or full convolution is used for upsampling the input by a scale of 2 and each fractionally-strided convolution is followed by a spatial batch normalization layer [28] and a ReLU nonlinear layer [17] with the exception of the output layer. The output layer uses a scaled tanh layer to restrict the pixel value of the generated image to the range $[0, 255]$. Additionally we also use total variation regularizer [89] to help improve the spatial smoothness of the generated images.

Discriminator network. As for discriminator, stride convolution is used for down-

sampling by a scale of 2 instead of deterministic functions such as maxpooling. In our model the filter size of the first stride convolution is not fixed to 3 (RGB channels) any longer and should be compatible with the shape of extracted deep features. Each stride convolution is followed by a batch normalization and LeakyReLU layer [90] except for the last one, which is flattened and then fed into a sigmoid layer to achieve binary classification.

Feature extraction network. The 19-layer VGGNet [24] trained on ImageNet [12] is chosen as pretrained CNN Φ for feature extraction. Specifically let $\Phi(x)^l$ stand for the l^{th} hidden representations of an input image x, and mathematically the extracted deep feature $\Phi(x)^l$ is a 3-dimensional array of shape [C^l x W^l x H^l], where C^l is the number of filters, W^l and H^l represent the width and height of the feature map for the l^{th} layer. And then the extracted features $\Phi(x)^l$ of the real and fake images instead of raw images x are used to train the discriminator and generator. Because of the same 3-dimensional array structure of $\Phi(x)$ and image x, $\Phi(x)$ could be considered as a "feature image" that consists of more than 3 color channels. Thus, the first convolutional layer of discriminator should be consistent with the number of color channels of the input "feature image".

4.2.3 Experiments

We perform two experiments in this work: face images generation and a Turing test. We use public face and flower image dataset to train our GAN model and compare the visual quality of generated samples with other generative models. Moreover we conduct a visual Turing test to figure out how well humans can distinguish the real and generated face images.

Training details. Our GAN models are also trained on CelebFaces Attributes Dataset (CelebA) [91] similar to experiments for VAE in the previous section. We use the aligned images in CelebA and crop out each face image to the shape [64 x 64] as the final training dataset. In addition, all the training images are scaled to the range [-1, 1]. Our models are trained with a batch size of 64 for 10 epochs across the training images. Adam [32] optimizer is used for stochastic optimization. The learning rate is initially set to 0.0001 and decreased by a factor of 0.5 every 3 epochs. Additionally the constant scale factor of the tanh layer in the generator is fixed to 150 that is compatible with the image preprocessing of VGGNet [24]. The generated images are regularized by total variation regularization with a strength of 1 x 10^{-8}. The 100-dimensional noise vector z is randomly drawn from normal distribution $\mathcal{N}(0, 1)$.

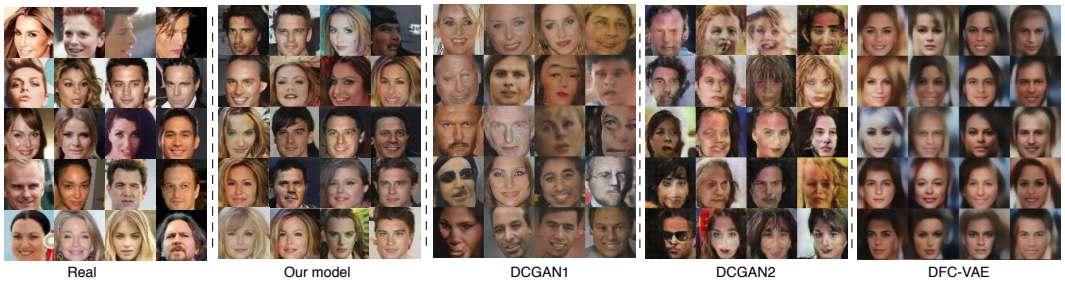


Figure 4.10: Generated face images from 100-dimension latent vector $z \sim \mathcal{N}(0, 1)$ for different models. The most left part is the real faces, the second part is the results generated by our model trained with relu1_1 representations of VGGNet. The two parts in the middle are the samples by DCGAN [77] (DCGAN1 is the samples directly from the paper [77] and DCGAN2 is retrained with raw images), and the most right part is results of DFC-VAE in previous section.

We also adjust the training process in order to improve the stability of training. GAN [15] and DCGAN [77] take turns to train the generator and discriminator for each iteration and try to achieve a balance between them in the end. However we discover that the discriminator is much easier for training and produce lower training loss than the generator. Ideally if the generated images are very similar to the real ones then the output of discriminator should be 0.5, because the discriminator is fooled and cannot tell the difference. Therefore, both the training loss for discriminator and generator (D_{loss} and G_{loss}) should be around $-log(0.5) = 0.69$ due to the loss function for binary classification. In order to prevent the discriminator from overtraining, we perform up to N (0 to 10) more gradient decent learning on the generator if the G_{loss} is bigger than 0.8 or the D_{loss} is less than 0.7. Our implementation is built on machine learning framework Torch [40].

Qualitative results for image generation. We separately experiment with different layer representations of VGGNet as deep features to train our models. In additional, we compare our generated images with the samples of DCGAN [77] and our DFC-VAE from previous section.

In Figure 4.10 we show the qualitative examples comparing our results with those of other generative models. The resolution of all the generated samples is 64 by 64, and all the models use the same training dataset CelebA [91]. Our model is trained with relu1_1 features. In these results, DCGAN can generate sharp images, however the facial structure could be distorted and generate weird faces which can be easily identified by humans. The faces produced by DFC-VAE have clear mouths, noses and eyes but blurred hairs. In general our model can produce sharp and more realistic face images

Figure 4.11: Generated samples of our models trained with different layer features. The GAN_11, GAN_31 and GAN_51 are trained with extracted features of layer relu1_1, relu3_1 and relu5_1 of VGGNet respectively.

with clear facial parts like the tiny texture of hair and reasonable background, showing that relu1_1 features are effective to train better GAN models than directly raw pixels.

One explanation could be that the first convolution layer in VGGNet transfers the raw image of shape [3 x 64 x 64] to "feature image" of shape [64 x 64 x 64], which can contain more diverse information of the raw image. This is because the first layer of the pretrained networks mainly responds to corners and other edge/color conjunctions [45], which can be used for transfer learning. On the other hand, one could argue that the first layer in the discriminator is able to achieve the same effect. However, it is more difficult to obtain the filters trained from scratch as good as those of networks pretrained on a large dataset like ImageNet.

As demonstrated in Figure 4.11, we then compare the performance of our models trained with relu1_1, relu3_1 and relu5_1 layer representations of VGGNet, denoted as GAN_11, GAN_31 and GAN_51. And the discriminators contains 5, 3 and 1 convolution layers respectively due to the maxpooling in VGGNet. It is clear that the generated samples of GAN_11 are much better than those of the other two models. The samples of GAN_31 can keep the facial structure in general with recognized mouths and eyes, which are better than those produced by GAN_51. Moreover, both the samples of GAN_31 and GAN_51 lack of diversity and the generated faces could be visually the same with different input vectors z. It shows that the lower layer features can be effective for GAN training. This could be explained that more detailed pixel information

Figure 4.12: Randomly generated flowers by our method.

is preserved in lower layers while higher layers contain more abstract information for classification, and the high-level features could be very similar if the inputs are just slightly different face images.

In addition, we also test our method on a flower dataset [92], containing 8,189 images for 102 different categories, and all the images are scaled to 64 x 64 as training samples. Figure 4.12 shows the randomly generated samples and real flowers. Though our model can produce reasonable flowers, it is still very difficult to generate tiny textures.

Linear interpolation of input vectors. Like [77, 93] we also investigate the linear property between two given noise vectors for image generation. Specifically we choose pairs of noise vectors and compute the intermediate representations by linear interpolation between them, and then generate images using the intermediate vectors. As shown in Figure 4.13, the generated images shows smooth transitions between each other in pixel space.

Human evaluation. In order to evaluate to how much the generated fake faces are similar to the real faces, we design a visual Turing test via a web interface [1] [2] [3] to invite 35 undergraduate volunteers from faculty of science and engineering in our university to distinguish the fake face images from real ones. We choose 50 fake images generated by our GAN_11, DCGAN1, DFC-VAE and real samples respectively (Figure 4.10). All the images are randomly shuffled for different volunteers and they need to choose a "Real" or "Fake" label for each image, which is a binary classification task. The average accuracies on the whole dataset are 0.71, 0.84 and 0.87 for GAN_11, DCGAN1 and DFC-VAE. It shows that our method can generate more realistic face images and the

[1] https://goo.gl/forms/cuNsJBAVJTUlizt23
[2] https://goo.gl/forms/KyEg5Kee13GFGCSJ2
[3] https://goo.gl/forms/eRcCQkBr0EFN5twp1

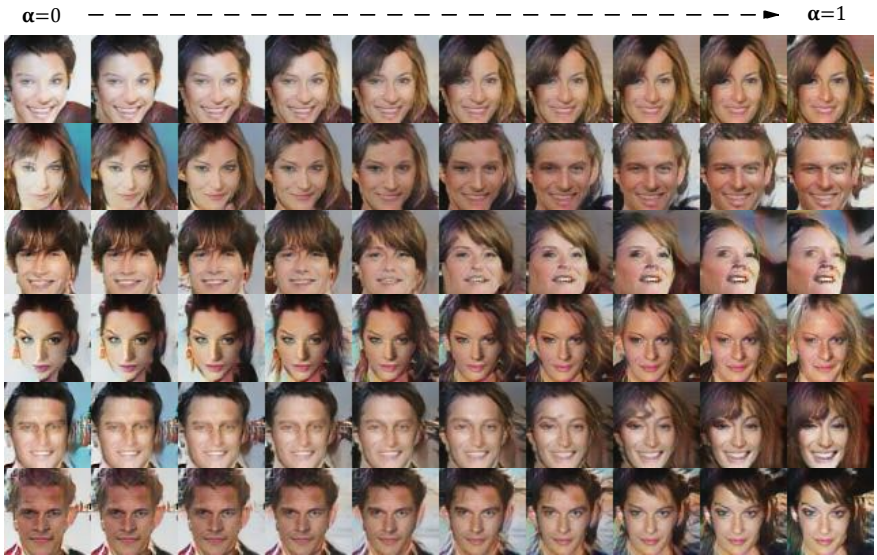


Figure 4.13: Linear interpolation of input vector z. Each row is the generated images for intermediate vectors from left vector z_{left} to right vector z_{right}. e.g. $(1-\alpha)z_{left} + \alpha z_{right}$.

volunteers could be "fooled" and cannot distinguish fake faces from real ones easily.

4.3 Summary

In this chapter we have developed two generative models that can produce new realistic images similar to the ones in a given dataset. We propose to incorporate pretrained deep convolutional neural network that helps construct objective functions for variational autoencoder and generative adversarial network respectively. We show that the pretrained CNN can be seamlessly stacked on VAE and GAN, therefore, the entire models can be optimized end-to-end on raw image dataset. The experiments demonstrate that our models can generate realistic images better than previous methods. In addition, We also show that our method can produce latent vectors that can capture the conceptual and semantic information of natural images. In particular, we achieve new state of the art performance in facial attribute prediction by using the learned latent representations in VAE.

As shown in this chapter, deep features extracted from pretrained CNN can be used to train state of the art generative models. In Chapter 5 and Chapter 6, we will further explore the effectiveness of deep features based loss as image quality measurement for two categories of image processing tasks: image information augmentation and information reduction.

CHAPTER 5

Image Information Augmentation

In this chapter, we develop deep learning techniques to tackle two traditional low-level image processing tasks, *image companding* and *inverse halftoning*, which are essentially information augmentation problems that require inferring new information. In particular, we propose to train a deep convolutional neural network as a nonlinear transformation function to map a lower bit depth image to a higher bit depth or from a halftone image to a continuous tone image, and at the same time employs another pretrained deep CNN as a feature extractor to derive visually important features to construct the objective function to guide the training. Extensive experimental results are presented to show that the new deep learning based solution significantly outperforms previous methods and achieves new state of the art results.

5.1 Introduction

Companding is a process of compression and then expanding, allowing signals with a higher dynamic range to be transmitted with a lower dynamic range by reducing the number of bits. This technique is widely used in telecommunication and signal processing such as audio processing. For image processing, companding could be regarded as an encoding and decoding framework. The encoding or quantized compression process, while fairly simple and efficient, could also produce a lot of undesirable artifacts, such as blocking artifacts, contouring and ringing effects (see Figure 5.1). These degraded artifacts become more obvious with lower bit quantization.

Inverse halftoning, another similar image processing problem considered in this work, is the inversed process for halftoning. Halftone images are binary images served as analog representation and widely used in digital image printing, trying to convey the illusion of having a higher number of bit levels (continuous-tone) to maintain the over-

all structure of the original images. As a result, distortions will be introduced to the halftone images due to a considerable amount of information being discarded. Inverse halftoning, on the other hand, addresses the problem of recovering a continuous-tone image from the corresponding halftoned version. This inversed process is needed since typical image processing techniques such as compression and scaling can be successfully applied to continuous-tone images but very difficult to halftone images.

However, these two problems are ill-posed considering that there could be an infinite number of possible solutions due to different information augmentation. They are essentially one-to-many mappings and the input image could be transformed into an arbitrary number of plausible outputs even if the compression and halftone methods are known in advance. Solving both problems requires finding a way to estimate and add more information into the images that do not exist. There are no well-defined mathematic functions or guidelines to describe the mappings to produce high-quality images.

In this work, we take advantage of the recent development in machine learning, in particular deep convolutional neural networks (CNNs), which have become the state of the art workforce for most computer vision tasks [23, 24]. Unlike previous human-engineered methods [94–97], we formulate the two image processing problems, i.e., companding and inverse halftoning, from the perspective of machine learning. We train deep convolutional neural networks as non-linear mapping functions in a supervised manner to expand images from a lower bit depth to a higher bit depth in order to reduce artifacts in image companding and produce continuous-tone images in inverse halftoning. Moreover, we also investigate the effect to construct loss functions based on different level convolutional layers, which have shown different properties when applying an inverting processing to reconstruct the encoded images [98].

Our core contributions in this work are two folds. Firstly, to the best knowledge of the authors, this is the first work that has successfully developed deep learning based solutions to these two traditional image processing problems. This not only introduces new methods to tackle well-known image processing problems but also contributes to the literature that demonstrates the power of deep learning in solving traditional signal processing problems. Secondly, building on the insights into the properties of visual quality of images and the hidden representation properties of deep CNNs, and also inspired by recent works that use deep CNNs in other image processing applications [35, 36, 99], we take full advantage of the convolutional neural networks both in the nonlinear mapping functions and in the neural networks loss functions for low-level image processing problems. We not only use a deep CNN as a nonlinear transfor-

mation function to map a low bit depth image to a higher bit depth image or from a halftone image to a continuous tone image, but also employ another pre-trained deep CNN as a feature extractor or convolutional spatial filter to derive visually important features to construct the objective function for the training of the transformation neural network. Through these two low-level image processing case studies, we demonstrate that a properly trained deep CNN can capture the spatial correlations of pixels in a local region and other visually important details, which can be used to infer the "correct" values of pixels and their neighbors as well. Our work further demonstrates that halftone images and heavily compressed low bit depth images, even though showing visually annoying artifacts, they have still preserved the overall structure information of the images which are sufficient to enable deep neural networks to recover the original signals to a high degree of fidelity.

5.2 Related Work

5.2.1 Image Companding

Companding, a combination of the words **com**pressing and ex**panding**, is a signal processing technique to allow signals with a large dynamic range transmitted in a smaller dynamic range format. This technique is widely used in digital telephony systems. Image companding [94, 100, 101] is designed to squeeze higher-bit images to lower bit ones, based on which to reproduce outputs with higher bits. Multi-scale subband architecture [94] successfully compressed high dynamic range (HDR) images to displayable low dynamic range (LDR) ones. They also demonstrated that the compression process can be inverted by following the similar scheme as the previous compression. As a result, low dynamic range images can be expanded to approximate the original higher-bit ones with minimal degradation.

5.2.2 Halftoning and Inverse Halftoning

The typical digital halftoning process is considered as a technique of converting a continuous-tone grayscale image with 255 color levels (8 bits) into a binary black-and-white image with only 0 and 1 two color levels (1 bit). These binary images could be reproduced to "continuous-tone" images for humans based on an optical illusion that tiny dots are blended into smooth tones by human eyes at a macroscopic level. In this work, we focus on the most popular halftoning technique known as error diffusion, in which the residual quantization error of a pixel is distributed to neighbor-

ing pixels. Floyd-Steinberg dithering is commonly used by image manipulation software to achieve error diffused halftoning based on a simple kernel. The reversed processing known as inverse halftoning is to reconstruct the continuous-tone images from halftones. Many approaches to addressing this problem have been proposed in the literature, including non-linear filtering [96], vector quantization [102], projection onto convex sets [103], MAP projection [104], wavelets-based [105], anisotropic diffusion [95], Bayesian-based [106], a method by combining low-pass filtering and super-resolution [107], Look-up table [108], sparse representation [109], local learned dictionaries [110] and coupled dictionary training [111].

5.2.3 Deep Learning for Image Transformation

In this work, we seek to formulate the image companding and inverse halftoning as image transformation problems and employ deep convolutional neural networks as non-linear functions to map input images to output images for different purposes. Recent deep CNNs have become a common workhorse behind a wide variety of image transformation problems. These problems can be formulated as per-pixel classification or regression by defining low level loss. Semantic segmentation methods [8, 112, 113] use fully convolutional neural networks trained by per-pixel classification loss to predict dense scene labels. End-to-end automatic image colorization techniques [114, 115] try to colorize grayscale image based on low level losses. Other works for depth [79, 116] and edge detection [117] are also similar to transform input images to meaningful output images through deep convolutional neural networks, which are trained with per-pixel classification or regression loss. However the per-pixel measurement essentially treats the output images as "unstructured" in a sense that each pixel is independent with all other pixels for a given image.

Considering the shortcoming of per-pixel loss, other "structured" measurements have been proposed such as structural similarity index measure (SSIM) [118] and conditional random fields (CRF) [119], which take context into account. These kinds of "structured" loss have been successfully applied to different image transformation problems. However these measurements are human-crafted, the community has successfully developed structure loss directly learned from images. Generative adversarial networks (GANs) [15] are able to generate high-quality images based on adversarial training. Many works have tried to apply GANs in conditional settings such as discrete labels [120], texts [121] and of course images. Image-conditioned GANs involve style transfer [122], inpainting [123], frame prediction [124]. In addition, image-to-image translation framework [125] based on adversarial loss can effectively synthesize photos under dif-

ferent circumstances.

Another way to improve per-pixel loss is to generate images by optimizing a perceptual loss which is based on high level features extracted from pretrained deep convolutional neural networks. By optimizing individual deep features [45] and maximizing classification score [126], images can be generated for a better understanding of hidden representations of trained CNNs. By inverting convolutional features [127], the colors and the rough contours of an image can be reconstructed from activations in pretrained CNNs. In addition, artistic style transfer [35] can be achieved by jointly optimizing the content and style reconstruction loss based on deep features extracted from pretrained CNNs. A similar method is also used for texture synthesis [128]. Similar strategies are also explored to achieve real-time style transfer and super-resolution [36].

5.2.4 Image Quality Metric

Peak Signal to Noise Ratio (PSNR). PSNR is the simplest and most widely used full-reference image quality metric and represents the ratio between the maximum possible power of a signal and the power of distortion noise. In the context of image processing, distortion noise is measured by the per-pixel mean squared error (MSE) between the original image and the distorted image while the maximal power of a signal is the maximum possible pixel value of the image (e.g. 255 for 8 bits images). Mathematically PSNR can be formulated as follows:

$$PSNR = 10\ log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{5.2.1}$$

We can see that PSNR is quite simple and easy to calculate with meaningful physical interpretation. In general, a higher PSNR value usually indicates better image quality, however in some cases it may not and could not generalize well to perceived visual quality [129–132]. In particular, for image reconstruction or generation the output images tend to be very blurry when compared to natural images. This is because the per-pixel loss (MSE) only considers point-wise signal difference rather than the perceptual difference and spatial correlation between two images.

Structural Similarity (SSIM) Index. SSIM [118] is another widely used and cited full-reference image quality metric and designed to improve on traditional methods like PSNR. The essential difference with respect to PSNR and MSE is that these methods only estimate the absolute point-wise error while SSIM is a more perceptual-based approach by comparing the structures of the reference and the distorted signals. SSIM is based on the assumption that the human visual system is highly adapted to extract

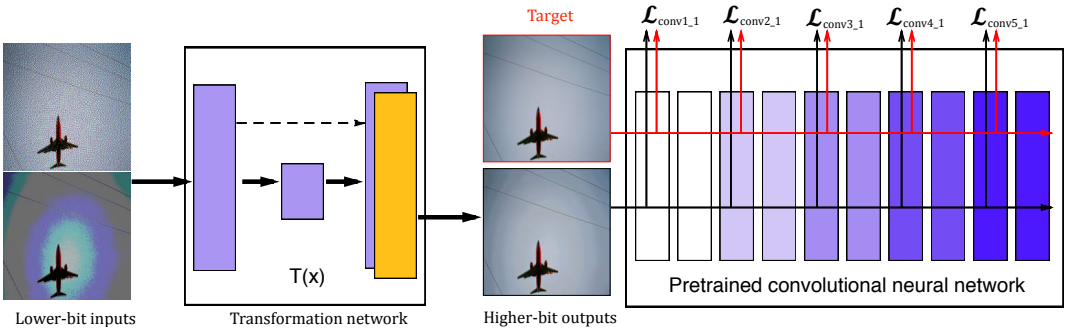


Figure 5.1: Method overview. A transformation convolutional neural network (CNN) to expand lower-bit images to higher-bit ones. A pretrained deep CNN for constructing perceptual loss to train the transformation network.

structural information from visual information and considers image degradations as perceived changes in structural information variation. The final SSIM index between two patches (x, y) is the combination of *Luminance*, *Contrast* and *Structural* comparison as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.2.2}$$

where μ_x and μ_y are the mean intensity of x and y; σ_x^2 and σ_y^2 are the variance of x and y; σ_{xy} is the covariance of x and y; $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are used to stabilize the division with weak denominator; L is the dynamic range of the pixel values; $k_1 = 0.01$ and $k_2 = 0.03$ by default. The "universal quality index" (UQI) proposed in [132] is the special case when k_1 and k_2 are equal to 0.

5.3 Method

In this work, we propose to use deep convolutional neural networks with skip connections as non-linear mapping functions to expand images from a lower bit depth to a higher bit depth. The objective of generating the higher bit depth version of the image is to ensure that this image is visually pleasing and can capture visually important properties of the original version of the image. Instead of using per-pixel losses, i.e. measuring pixel-wise difference between the output image and its target (the original) image, we measure the difference between the output image and target image based on the high level features extracted from pretrained deep convolutional neural networks. The key insight is that the pretrained networks have already encoded perceptually useful information we desired, such as the spatial relationship between pixels nearby. Our system is diagrammatically illustrated in Figure 5.1, which consists of two parts:

an autoencoder transformation neural network with skip connections $T(x)$ to achieve end-to-end mapping from an input image to an output image, and a pretrained neural network $\Phi(x)$ to define the loss function.

5.3.1 Network Architecture

Our non-linear mappings are deep convolutional neural networks, which have been demonstrated to have state-of-the-art performances in many computer vision tasks. Successful network architecture like AlexNet [23], VGGNet [24] and ResNet [26] are designed for high level tasks like image classification to output a single label, and they cannot be directly applied to image processing problems. Instead, previous works have employed an encoder-decoder architecture [36, 99] to firstly encode the input images through several convolutional layers until a bottleneck layer, followed by a reversed decoding process to produce the output images. Such encoder-decoder architecture forces all the information to pass through the networks layer by layer. Thus the final generated images are produced by higher layers' features. However for image processing, the output images can retain a great deal of lower layers' information of the input images, and it would be better to incorporate lower layers' features in the decoding process. Based on the architecture guidelines of previous work on image segmentation [133], image-to-image translation [125] and DCGAN [77], we add skip connections to construct a "U-Net" network to fuse lower layers and higher layers features and employ fully convolutions for image transformation.

The details of our model are shown in Figure 5.2, we first encode the input image to lower dimension vector by a series of stride convolutions, which consists of 4 x 4 convolution kernels and 2 x 2 stride in order to achieve its own downsampling. We also use a similar approach for decoding to allow the network to learn its own upsampling by using deconvolutions [8]. Spatial batch normalization [28] is added to stabilize the deep network training after each convolutional layer except the input layer of the encoder and the last output layer of the decoder as suggested in [77]. Additionally leaky rectified activation (LeakyReLU) and ReLU are served as non-linear activation functions for encoder and decoder respectively. Finally we directly concatenate all the encoding activations to the corresponding decoding layers to construct a symmetric "U-Net" structure [133] to fuse the features from both low layers and high layers.

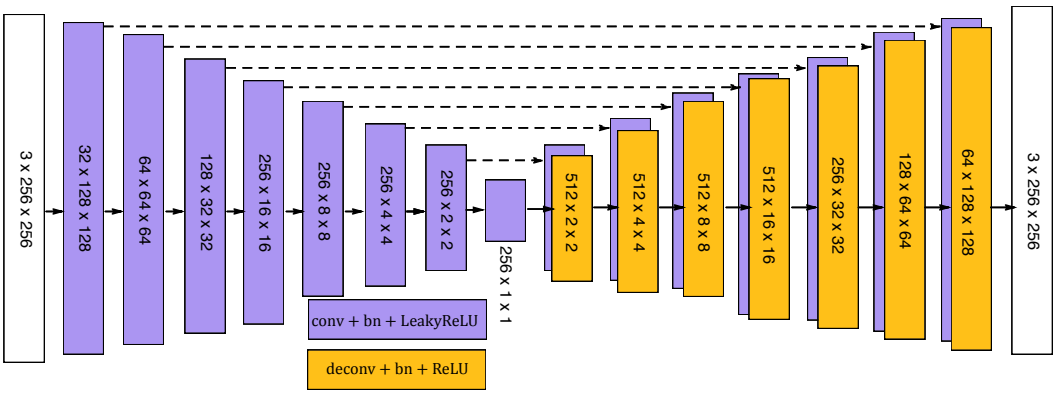


Figure 5.2: The architecture of the transformation networks. The "U-Net" network is an encoder-decoder with skip connections between the encoder and decoder. The dash-line arrows indicate the features from the encoding layers are directly copied to the decoding layers and form half of the corresponding layers' features.

5.3.2 Perceptual Loss

It is well known that per-pixel loss for regression and classification is problematic and could produce blurry outputs or other visual artifacts. This is because each pixel is regarded as an individual object for optimization, resulting in average outputs to some degree. A better strategy is to construct the loss by incorporating the spatial correlation information. Rather than encouraging matching each individual pixels of input and output images, we follow previous works [35, 36, 99] to measure the difference between two images at various deep feature levels based on pretrained deep convolutional neural networks. We seek to capture the input images' spatial correlations by means of convolution operations in the deep CNNs.

We denote the loss function as $\mathcal{L}(\hat{y}, y)$ to measure the perceptual difference between two images. As illustrated in Figure 5.1, both the output image $\hat{y} = T(x)$ generated by the transformation network and the corresponding target image y are fed into a pretrained deep CNN Φ for feature extraction. We use $\Phi_i(y)$ to represent the hidden representations of image y at i^{th} convolutional layer. $\Phi_i(x)$ is a 3D array of shape [C_i, W_i, H_i], where C_i is the number of filters, W_i and H_i are the width and height of the given feature map of the i^{th} convolutional layer. The final perceptual loss of two images at i^{th} layer is the Euclidean distance of the corresponding 3D arrays as following:

$$\mathcal{L}_i(\hat{y}, y) = \frac{1}{C_i W_i H_i} \sum_{c=1}^{C_i} \sum_{w=1}^{W_i} \sum_{h=1}^{H_i} (\Phi_i(\hat{y})_{c,w,h} - \Phi_i(y)_{c,w,h})^2 \tag{5.3.1}$$

In fact, above loss still follows the per-pixel manner if we treat the hidden features

which are 3D arrays as "images" with more than 3 color channels. However this kind of loss has already incorporated the spatial correlation information because the "pixels" in these images are the combinations of the original pixels through convolution operations.

5.3.3 Training Details

Our implementation uses open source machine learning framework Torch [40] and a Nvidia Tesla K40 GPU to speed up training. The pretrained 19-layer VGGNet [24] is chosen as the loss network for deep feature extraction which is fixed during the training. In addition, due to the similar convolutional architecture, the loss network can be seamlessly stacked to our "U-Net" neural network to achieve end-to-end training. The training images are of the shape 256×256 and we train our model with a batch size of 16 for 30,000 iterations. Adam optimizer [32] is used for stochastic optimization with a learning rate of 0.0002. For the LeakyReLU in the encoder, the slope of the leak is set to 0.2 in all layers. Additionally we experiment with conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 layers in VGGNet to construct perceptual loss for comparison.

5.4 Experimental Results

In our experiments, we use Microsoft COCO dataset [134] which is a large-scale database containing more than 300,000 images as our training images. We resize the training images to 256×256 as our inputs to train our models. We perform experiments on two image processing problems: image companding and inverse halftoning.

5.4.1 Image Companding

One essential part of image companding is to expand lower bit images to higher bit outputs. This technique has been investigated in the context of high dynamic range (HDR) imaging [94], firstly compressing the range of an HDR image into an LDR image, at which point the process is then reversed to retrieve the original HDR image.

Since it is impossible to display a true HDR image with more than 8 bits, we use 8 bit images as our highest bit depth images in the experiments. The 8 bit images are reduced by different depths as the lower bit depth images, and then expanded back to 8 bits. Take 4 bit images for example, they can only have 16 different levels for each color channel while there are 256 different levels for 8 bit images. The default approach [94] for converting 8 bit images to 4 bit images is to divide by 16 to quantize

the color level from 256 to 16, which will be then scaled up to fill the full range of the display. Mathematically we can use the formula below to easily convert 8 bit images to different lower bit outputs. This operation can be applied to both grayscale images and color images by processing each channel separately.

$$I_{low} = \lfloor \frac{I_{high}}{2^{(h-l)}} \rfloor 2^{(h-l)} \tag{5.4.1}$$

where the I_{low} and I_{high} are the pixel intensity of converted lower and higher bit depth images respectively, l and h are the bit depth for lower and higher bit depth images.

We first preprocess the training images to different lower-bit ones as input data, and use the original images as higher-bit targets we want to retrieve. After training, the validation split of Microsoft COCO is used for testing. We first compare the results of different lower-bit input images, and then evaluate how the perceptual loss constructed from different convolutional layers affects the expanding quality.

Different bit depths. We have separately trained models for different lower-bit input images for comparison and use the conv1_1 layer of VGGNet to construct the perceptual loss for all the models.

Figure 5.3 and Figure 5.4 show the qualitative results for a variety of color and grayscale images taken from Microsoft COCO 2014 validation split. We can see that the linearly quantized lower-bit images display severe blocking and contouring artifacts. The compression process amplifies low amplitudes and high frequencies which dominate the quantization artifacts because we try to show a lower dynamic range image on a higher dynamic range displayable device. For instance, our device is appropriate for the original 8 bit targets with 256 color levels. We could drop the bit depths of the original images by 5 bits and linearly quantize them to 3 bit images with only 8 color levels. Since the compressed images contain 5 fewer bits, they should be theoretically displayed on 1/32 dynamic range device. It is obvious that this kind of lossy compression introduces visible artifacts in pixel blocks and at block boundaries.

We also show the corresponding expanded images retrieved from our models in Figure 5.3 and Figure 5.4. The blocking and contouring artifacts are effectively reduced to show smooth appearance in the expanded outputs. For example in the airplane image in Figure 5.4, the compressed images show obvious contouring artifacts in the sky while the expanded images have homogeneous gradually changing colors. This can be further validated from the distribution of intensity histograms. Figure 5.5 shows the intensity histograms for the compressed 2 and 4 bit airplanes and the expanded ones in Figure 5.4. It is clear that our methods are able to infer the "correct" values for a

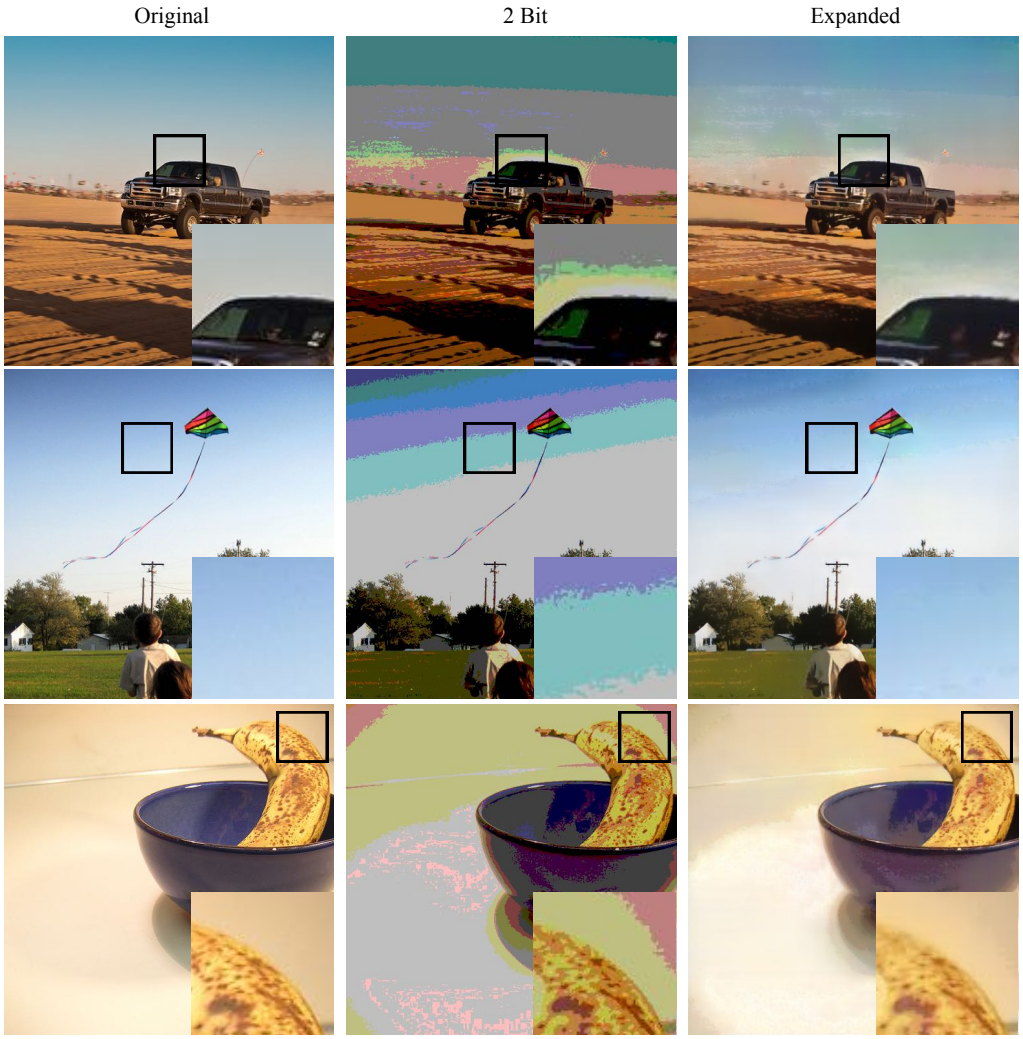


Figure 5.3: Results on color images from Microsoft COCO validation split for blocking and contour artifacts reduction. A pair of compressed 2 bit images and the corresponding expanded ones are shown together. Additionally an enlarged sub-image of each image is given at the bottom for better comparison.

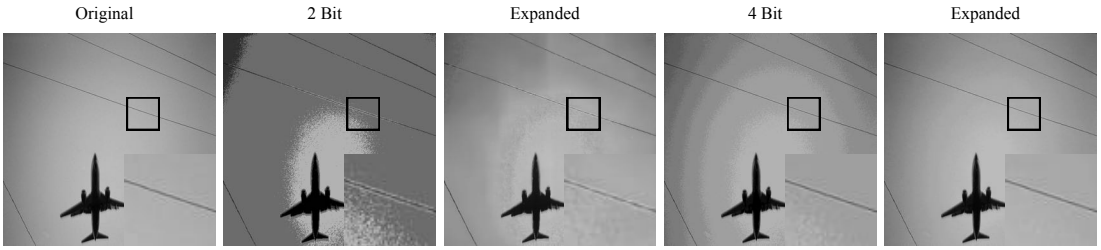


Figure 5.4: Results on grayscale images from Microsoft COCO validation split for blocking and contour artifacts reduction. A pair of compressed 2 bit and 4 bit images and the corresponding expanded ones are shown together. Additionally an enlarged sub-image of each image is given at the bottom for better comparison.

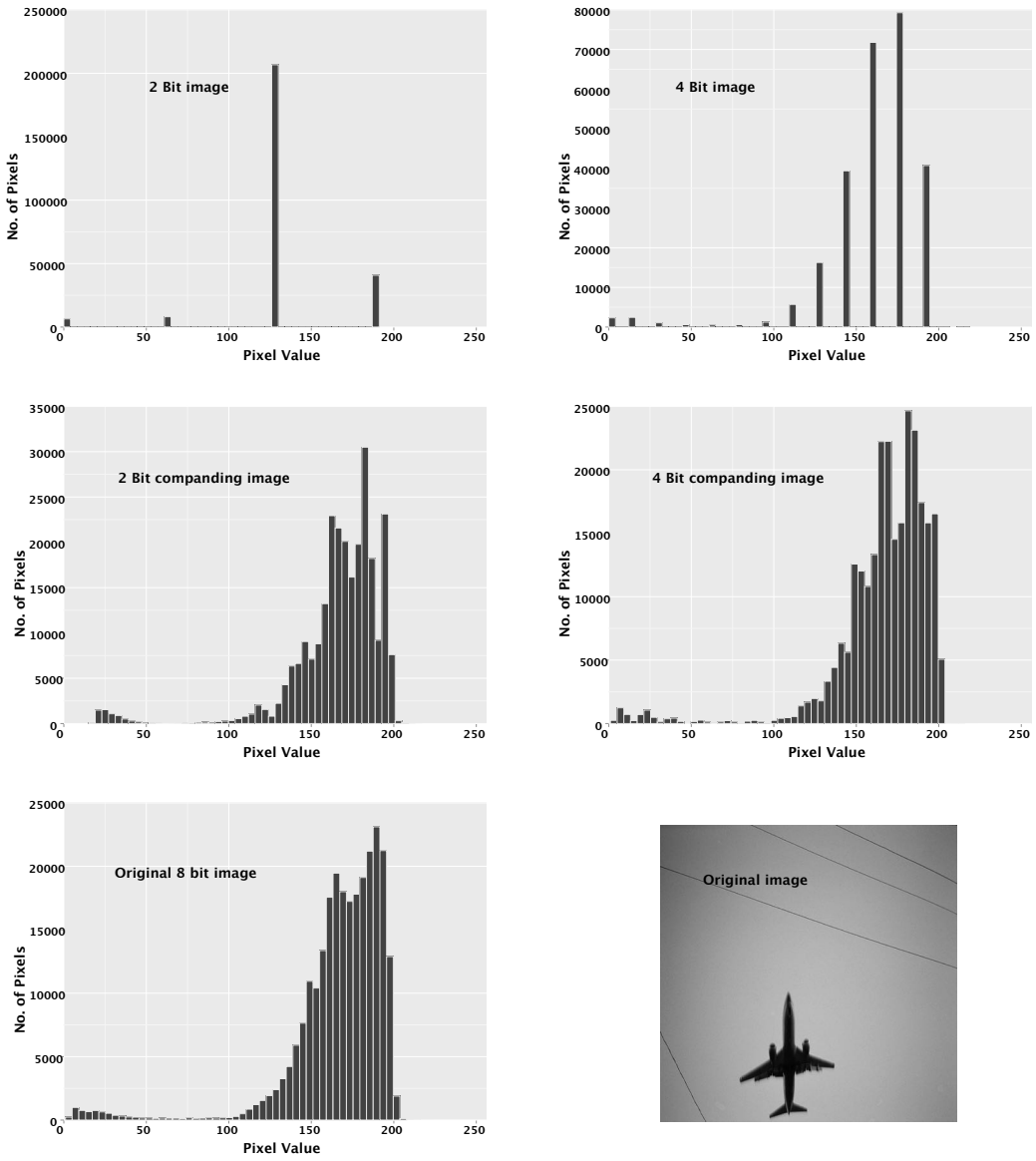


Figure 5.5: Intensity histogram of different compressed and expanded images.

single pixel based on its neighbors, and convey a more attractive impression with rich and saturated colors.

In order to have a comprehensive quantitative evaluation for our models, we report peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [118] for quality assessment. PSNR is per-pixel based measurement defined via the mean squared error (MSE) while SSIM index is known as perceptual-aware method for measuring the structural similarity between two images. For both measurements, a higher value indicates better quality. Table 5.1 summarizes the average PSNR (dB) and SSIM values of 100 images selected from COCO validation split. Similar to qualitative results, the higher bit images have higher PSNR and SSIM values, indicating better image

Table 5.1: The average companding results of PSNR(dB) and SSIM for 100 color and grayscale images randomly selected from Microsoft COCO validation split. The expanded results were based on a perceptual loss constructed using conv1_1 layer.

Input Bit-depth		PSNR		SSIM	
		Compressed	Expanded	Compressed	Expanded
Bit 1	Color	11.73	18.67	0.40	0.55
	Grayscale	11.58	18.81	0.35	0.49
Bit 2	Color	17.37	25.65	0.67	0.81
	Grayscale	17.29	25.84	0.61	0.74
Bit 3	Color	23.13	30.79	0.85	0.90
	Grayscale	23.16	31.33	0.78	0.87
Bit 4	Color	29.03	34.52	0.94	0.95
	Grayscale	29.19	36.69	0.90	0.94
Bit 5	Color	34.85	37.59	0.98	0.97
	Grayscale	35.08	40.24	0.96	0.97

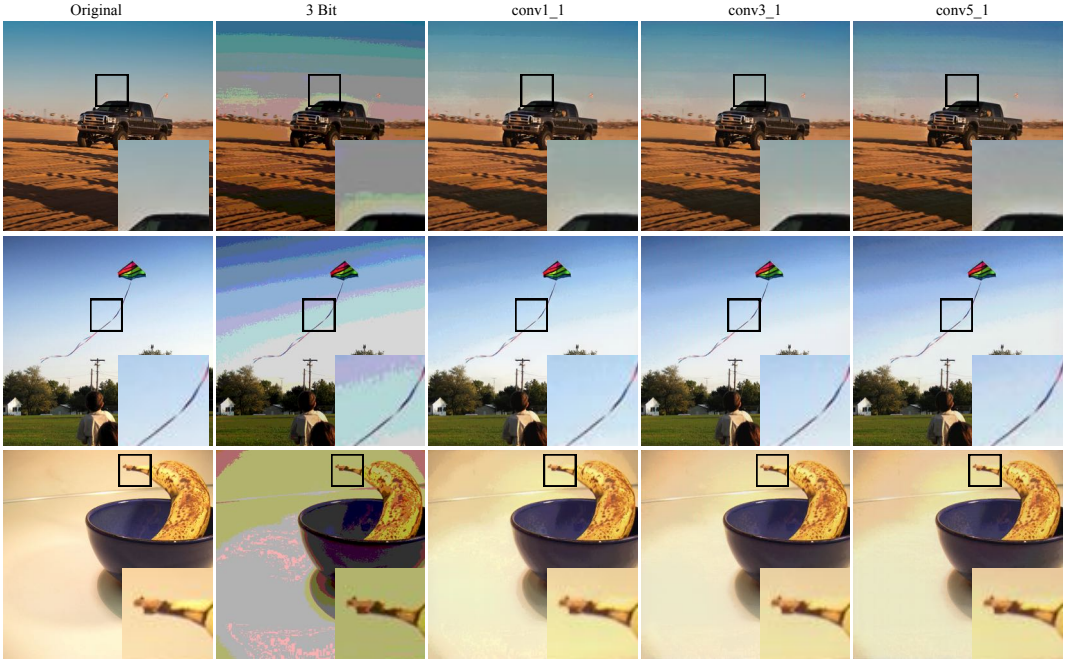


Figure 5.6: Results on color images for blocking and contour artifacts reduction. The compressed images are fixed to 3 bits with 8 color levels for each channel. The conv1_1, conv3_1, conv5_1 are the expanded results produced by the models trained with perceptual loss constructed by corresponding convolutional layers. Additionally an enlarged sub-image of each image is given at the bottom for better comparison.

quality. Additionally, the expanded images produced by our method have significantly higher PSNR and SSIM values compared to the corresponding compressed ones. It is clear that our method can effectively improve the image quality especially for lower bit depth images.

Perceptual loss at different convolutional layers. Due to the multi-layer architecture of deep convolutional neural networks, the perceptual loss can be defined by different convolutional layers. Therefore, we conduct experiments to investigate the performance for different perceptual losses. In all the experiment we use 3 bit depths (8 color levels) as input images to train our deep networks for both color and grayscale images.

As shown in Figure 5.6, all the expanded outputs can effectively reduce the blocking and contouring artifacts and reveal continuous-tone results in general. However, the reconstruction based on perceptual loss of higher-level layers could introduce new artifacts such as grid patterns as shown in images for conv3_1 and conv5_1 layers. We observed similar phenomenons for input images of different bit-depths. One explanation is that the higher layers will cover a larger area in the input image. The areas covered by conv3_1 and conv5_1 layers are too large to construct a natural looking image and the spatial correlations across a large area of the image do not capture the natural appearances of an image. Expanding based on perceptual loss of deep features of conv1_1 layer or lower layers does not have this kind of artifacts. This could be also validated by previous work [98] that tries to compute an approximate inverse image from its deep features. It shows that the first few layers in a pretrained CNN are essentially an invertible code of the image and maintain a photographically faithful representations, and the higher level features are corresponding to a more coarse space area of the encoded image.

Table 5.2 shows the average PSNR and SSIM values for 100 COCO testing images based on perceptual losses constructed with different convolutional layers. On the one hand, both the PSNR and SSIM of our expanded images are much higher than those of the compressed lower bit images, and the compressed images can be significantly improved by our method. On the other hand, the expanded images based on perceptual losses of lower level layers have higher PSNR and SSIM values. This is because new artifacts like grid pattern will be introduced (Figure 5.6) although the blocking artifacts can be reduced.

5.4.2 Inverse halftoning

Another similar image processing problem we are interested in is inverse halftoning. This task is to generate a continuous-tone image from halftoned binary images. This problem is also inherently ill-posed since there could exist multiple continuous-tone images corresponding to the halftoned ones. In our experiments we try to use deep feature based perceptual loss to allow the inversed halftones perceptually similar to the given targets. We experiment with both color and grayscale images by using the same

Table 5.2: The average companding results of PSNR(dB) and SSIM for 100 color and grayscale testing images. The compressed input images are 3 bits, and the expanded results based on perceptual loss constructed with different convolutional layers are shown.

Perceptual Loss Layer		PSNR		SSIM	
		Compressed	Expanded	Compressed	Expanded
Conv1	Color	23.13	32.64	0.85	0.93
	Grayscale	23.16	32.57	0.78	0.91
Conv2	Color	23.13	30.00	0.85	0.88
	Grayscale	23.16	30.74	0.78	0.85
Conv3	Color	23.13	28.17	0.85	0.87
	Grayscale	23.16	29.60	0.78	0.81
Conv4	Color	23.13	25.74	0.85	0.84
	Grayscale	23.16	29.54	0.78	0.82
Conv5	Color	23.13	25.43	0.85	0.87
	Grayscale	23.16	27.50	0.78	0.77

approach and employ error diffusion based Floyd-Steinberg dithering for halftoning.

Qualitative results. We test our models on random samples of images from Microsoft COCO validation split. The inverse halftoning results are shown in Figure 5.7. We can see that the inversed outputs produced by our method are visually similar to the original images. All the outputs can show much smoother textures and produce sharper edges. For instance, sharp kite line and smooth sky can be reconstructed in the kite image. When comparing with the inversed outputs produced by using perceptual loss of different level layers, the outputs from lower-level layer is visually better than those from higher-level layer. Like image companding, grid pattern artifacts can be introduced when using higher-level layer to construct perceptual loss.

In addition, we also compare our method on two widely used grayscale images *Lenna* and *Peppers* with other algorithms. Figure 5.8 shows comparative grayscale results against previous Fastiht2 [95] and Wavelet-based WInHD [105] algorithms. We also report the PSNR / SSIM measurement for each image. It is clear that our learning-based method can achieve state-of-the-art results and produce sharp edges and fine details, such as the hat in the Lenna image. Our deep models can effectively and correctly learn the relevant spatial correlation and semantic information between different pixels and infer the "best" values for a single pixel based on its neighbors. Moreover, our method can be naturally adapted to color images and produce high-quality continuous-tone color images from corresponding halftones. Figure 5.9 shows the resulting images for the *Koala* and *Cactus* image, which include fine textures and structures. We compare our results (CNN Inverse) with those of two recent methods GLDP [109] and LLDO [110]. We can see that our method can provide better resulting images with well expressed

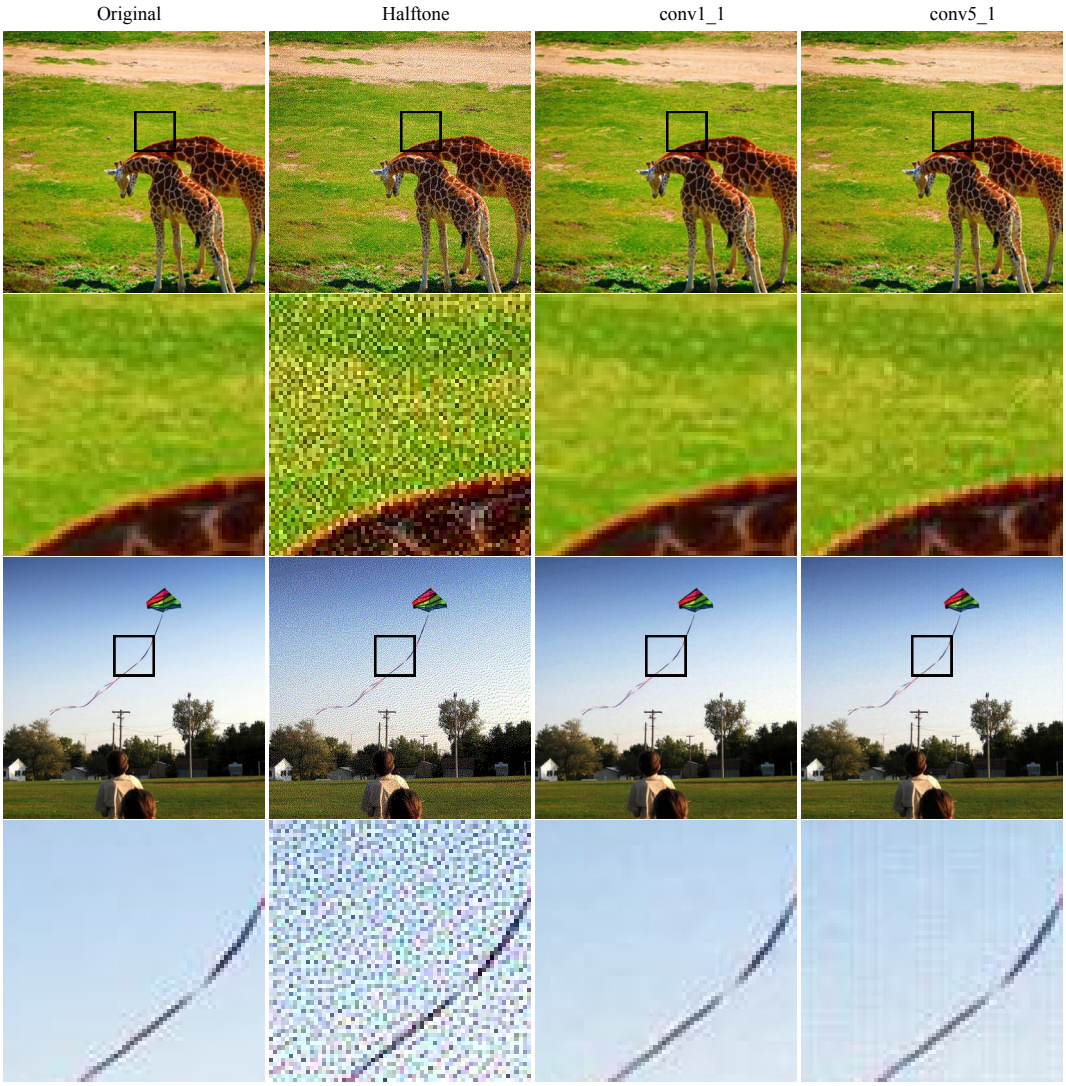


Figure 5.7: Inverse halftoning results on images from Microsoft COCO validation split. The conv1_1, conv5_1 are the results produced by the models trained by the perceptual losses of corresponding convolutional layers. Additionally an enlarged sub-image of each image is given at the bottom for better comparison.

fur and bark in the *Koala* image, and distinct boundaries of the fine sand and sharpened edges of splines in the *Cactus* image.

Quantitative results. We use PSNR and SSIM as quality metrics to quantitatively evaluate our inverse halftoning results. Table 5.3 shows the average PSNR and SSIM values for 100 COCO testing images constructed from different convolutional layers. It is clear that based on these image evaluation metrics, our method can improve the images by a large margin for both color and grayscale images. In our experiment, the best results are produced by the model trained with conv1_1 layer. When using perceptual loss based on higher layers gives rise to a slight grid pattern artifacts visible under magni-

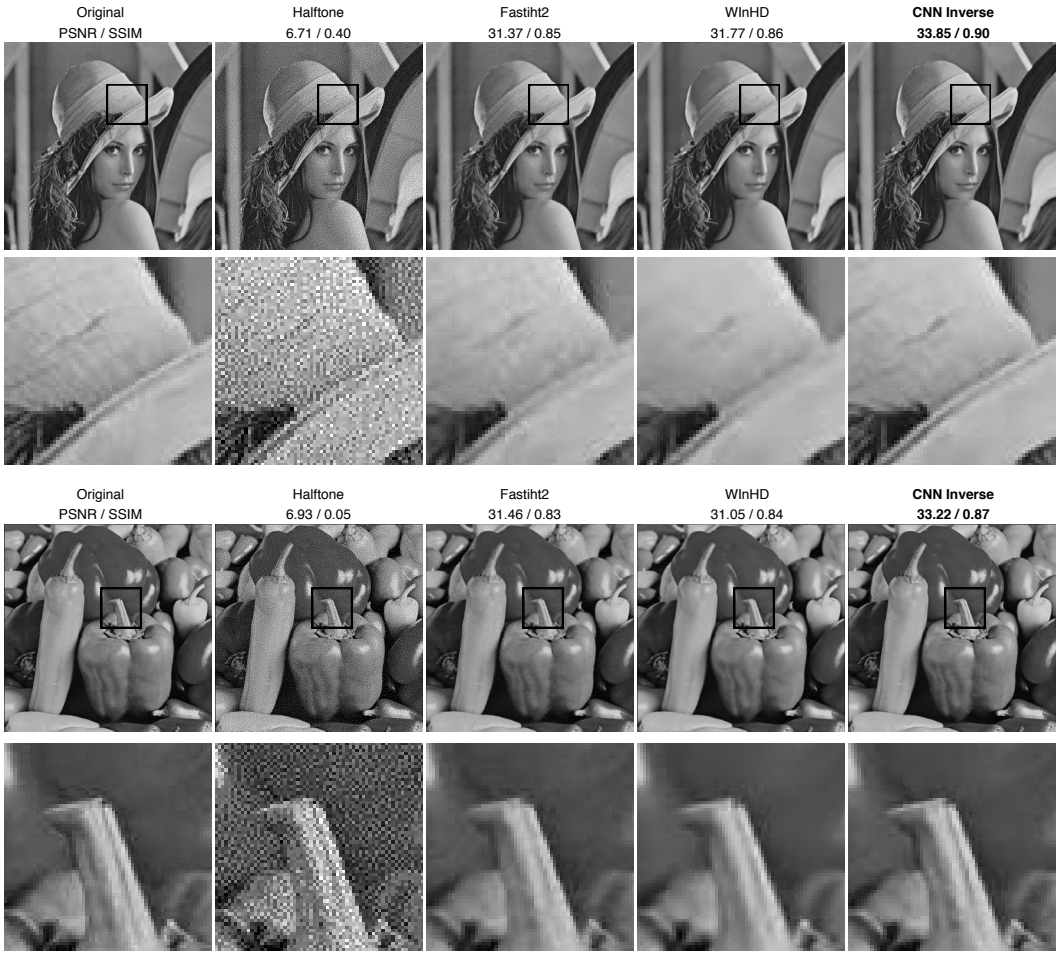


Figure 5.8: A comparison of inverse halftoning results on grayscale *Lena* and *Peppers* images by different methods. We compare our CNN Inverse method with those of Fastiht2 [95] and Wavelet-based WInHD [105]. We report PSNR / SSIM for each example.

Table 5.3: The average inversed halftoning results of PSNR(dB) and SSIM for 100 color and grayscale images selected from Microsoft COCO validation split.

Perceptual Loss Layer		PSNR		SSIM	
		Halftone	CNN Inverse	Halftone	CNN Inverse
Conv1	Color	8.08	31.43	0.20	0.91
	Grayscale	7.92	31.36	0.14	0.90
Conv2	Color	8.08	20.98	0.20	0.59
	Grayscale	7.92	23.98	0.14	0.67
Conv3	Color	8.08	24.05	0.20	0.73
	Grayscale	7.92	27.44	0.14	0.74
Conv4	Color	8.08	26.48	0.20	0.85
	Grayscale	7.92	27.82	0.14	0.76
Conv5	Color	8.08	25.47	0.20	0.84
	Grayscale	7.92	26.48	0.14	0.69

fication, which harms the PSNR and SSIM.

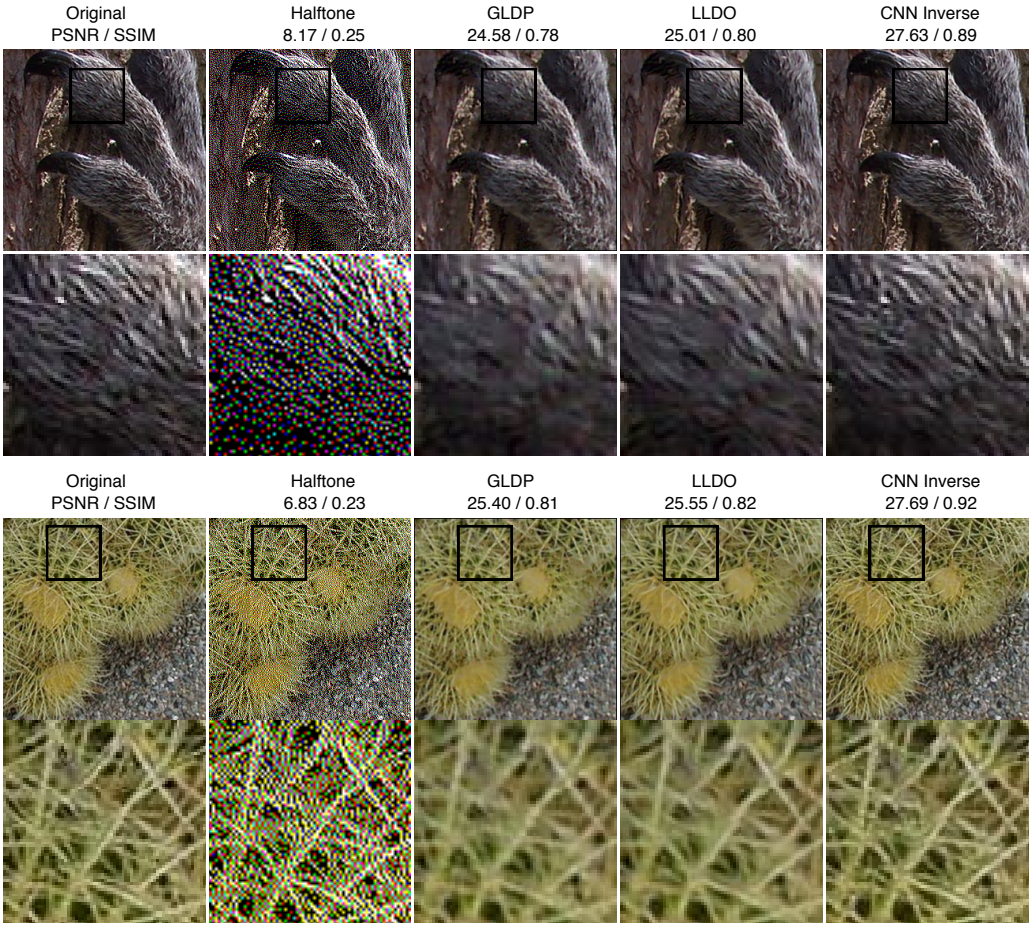


Figure 5.9: A comparison of inverse halftoning results on color *Koala* and *Cactus* images by different methods. We compare our CNN Inverse method with those of GLDP [109] and LLDO [110]. We report PSNR / SSIM for each example.

Moreover, we conduct experiments to compare with several previous methods. We use 6 images *Koala*, *Cactus*, *Bear*, *Barbara*, *Shop* and *Peppers*, the same as [110] for testing. Table 5.4 shows the PSNR and SSIM results for conventional methods based on MAP estimation [104], ALF [95], LPA-ICI [135] and recent GLDP [109] and LLDO [110]. We can see that our algorithm (CNN Inverse) can achieve new state-of-the-art results and significantly outperform previous methods for inverse halftoning.

5.5 Summary

Image companding and inverse halftoning are two similar image processing problems in the sense that they attempt to use a lower bit depth image to represent a higher bit depth version of the same image to achieve information augmentation. In order to expand the compressed images and inverse the halftones, traditional methods usually

Table 5.4: PSNR (dB) and SSIM comparison of different inverse halftoning methods for color images: MAP [104], ALF [95], LPA-ICI [135], GLDP [109], LLDO [110] and our CNN Inverse.

Image	ALF		MAP		LPA-ICI		GLDP		LLDO		CNN Inverse	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Koala	22.36	0.66	23.33	0.74	24.17	0.76	24.58	0.78	25.01	0.80	27.63	0.89
Cactus	22.99	0.64	23.95	0.77	25.04	0.79	25.40	0.81	25.55	0.82	27.69	0.92
Bear	21.82	0.62	22.63	0.72	23.14	0.72	23.66	0.77	24.17	0.78	26.35	0.89
Barbara	25.41	0.71	26.24	0.78	27.88	0.83	27.12	0.80	28.48	0.85	31.79	0.92
Shop	22.14	0.64	22.46	0.69	24.12	0.77	23.86	0.75	24.61	0.80	27.27	0.89
Peppers	30.92	0.87	28.25	0.77	30.70	0.87	30.92	0.87	31.07	0.87	31.44	0.89

need to design expanding and inverse operators manually. For example, the halftone technique such as the specific dithering algorithms should be given in advance in order to design an inverse operator. In this chapter, we show that a learning based method can formulate the two problems in the same framework and a perceptual loss based on pretrained deep networks can be used to guide the training. Our method is very effective in dealing with compressed blocking and contouring artifacts for companding and reproduces state of the art continuous-tone outputs from binary halftone images.

CHAPTER 6

Image Information Reduction

In the last chapter, we saw that we can effectively train deep convolutional neural network for two image information augmentation problems: image companding and inverse halftoning. The key point is to measure the similarity between the outputs and ground-truth images based their deep features extracted from pretrained deep models. In this chapter, we turn to other image processing problems that require reducing image information such as image downscaling, decolorization (colour to grayscale conversion) and high dynamic range (HDR) image tone mapping (Figure 6.1). We cannot directly apply the strategy used in the last chapter because no ground-truth are available for these problems. In this chapter, we propose the Deep Feature Consistent Deep Image Transformation (DFC-DIT) framework and build deep models to tackle these 3 image processing tasks in a similar way. DFC-DIT uses a convolutional neural network as a non-linear mapper to transform an input image to an output image following the deep feature consistent principle which is enforced through another pretrained and fixed deep convolutional neural network. For each problem, we reason about a suitable learning objective function based on image features extracted from deep models and develop an effective solution under the DFC-DIT framework. This is the first work that uses deep learning to solve and unify these three common image processing tasks. We present experimental results to demonstrate the effectiveness of the DFC-DIT technique and its state of the art performances.

6.1 Related Work

Image downscaling. Classical image downscaling techniques usually involve processing the input images by applying a spatially constant low-pass filter, subsampling, and reconstructing the result to prevent aliasing in the reconstructed signal. Approxima-

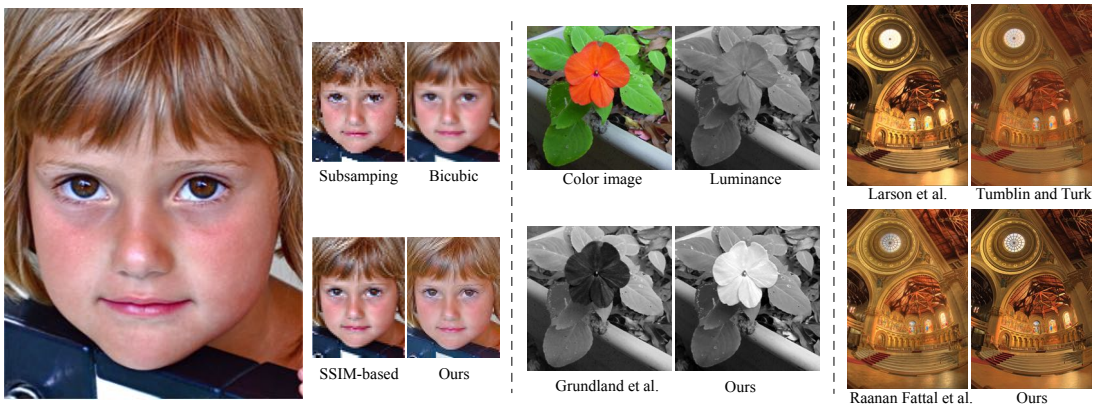


Figure 6.1: Examples of classic image transformation tasks. Image downscaling (left) where we show results of our method, two traditional methods (subsampling and bicubic) and a state-of-the-art SSIM-based method [136]. Decolorization (middle) where we show results of our method, the Luminance channel and a state-of-the-art method [137]. HDR image tone mapping (right) where we show results of our method and 3 methods from the literature [138–140].

tions to the theoretically optimum sinc filter such as the Lanczos filter, and other filters (e.g., bilinear and bicubic) have been developed and used in practice. However the filtering kernels of these methods do not adapt to the image content. A recent content-adaptive technique [141] is proposed to overcome the above shortcoming by adapting the shape and location of every kernel to the local image content and demonstrates better downscaling results. A better depiction of the input image was proposed [136] by formulating image downscaling as an optimization problem with the structural similarity (SSIM) [118] as the perceptual image quality metric. In addition convolutional filters [142] are used to preserve visually important details in downscaled images.

Decolorization. Decolorization aims to convert color images into grayscale images while preserving structures and contrasts as much as possible. The baseline method is to extract the luminance channel of a given color image from the RGB channels. However it could fail to express salient structures of the color image because of the fixed weights for combining RGB channels. Other more advanced techniques are proposed to obtain better results by either focusing on local contrasts or global contrasts. Local contrasts [143, 144] use different mapping functions in different local regions of the image, while global contrasts [145–148] are designed to produce one mapping function for the whole image. [149] takes into account multi-scale contrast preservation in both spatial and range domain and uses bilateral filtering to mimic human contrast perception. [150] uses a bimodal objective function to alleviate the restrictive order constraint for color mapping. In addition image fusion based strategy [151] is proposed for image

and video decolorization.

HDR image tone mapping. HDR image tone mapping aims to reproduce high dynamic range radiance maps in low dynamic range reproduction devices. Tone mapping operators can be classified as global operators and local operators. Global operators [152–154] usually employ the same mapping function for all pixels and can preserve the intensity orders of the original scenes to avoid "halo" artifacts, however the global operators will generally cause loss of details in the mapped image. In contrast, local operators [139, 155, 156] use mapping functions which vary spatially across the image. Most local operators employ a pipeline to decompose an image into different layers or scales and then recompose the mapped results from various scales after contrast reduction. However, the major shortcoming of local operators is the presence of haloing artifacts. In addition, global operator is used in the local regions to reproduce local contrast and ensure better quality [157]. What's more, an up-to-date, detailed guide on the theory and practice of high dynamic range imaging is included in the book [158], which also provide MATLAB code for common tone mapping operators. In this chapter, we use their code to reproduce previous methods.

Image quality metrics. The choice of image quality metric is essential for image transformation tasks. Standard pixel-by-pixel measurement like mean square error is problematic and the resultant images are often of low quality. This is because the measurement is poorly correlated with human perception and can not capture the perceptual difference and spatial correlation between two images. Better metrics have been proposed for image quality assessment in recent years. Structural similarity (SSIM) index [118] is one of the most popular metrics, which computes a matching score between two images by local luminance, contrast, and structure comparisons. It has been successfully used for image downscaling [136], image denoising [118] and super-resolution [159].

Relevant deep learning/CNN literature. Recently, there has been an explosion of publications on deep learning/CNN, similar to previous sections we here briefly review the most related publications to our current work. A number of papers have successfully generated high-quality images based on the high-level features extracted from pretrained deep convolutional neural networks. By optimizing individual deep features [45, 93, 160, 161], better visual quality images can be generated, which in turn can help understand the learned representations of deep networks. Additionally [35] have achieved style transfer by minimizing content and style reconstruction loss which are also based on features extracted from deep networks. Other works try to train a feed-forward network for real-time style transfer and super-resolution [36]. Different loss

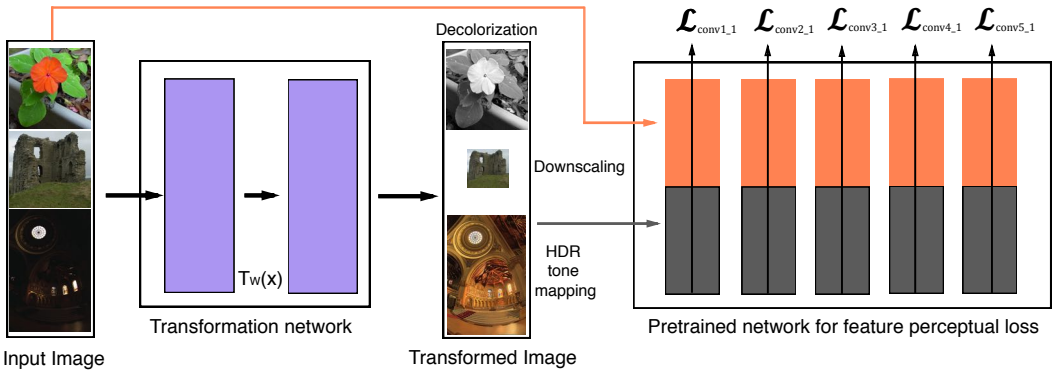


Figure 6.2: The Deep Feature Consistent Deep Image Transformation (DFC-DIT) framework. A convolutional neural network transforms an input to an output. A pretrained deep CNN is used to compute feature perceptual loss for the training of the transformation network.

functions are compared for image restoration with neural networks [162]. In addition image-to-image translation framework [125] are proposed to generate high quality images based on adversarial training.

It is worth noting that the downscaling problem studied in this work has the opposite goal to super-resolution. Deep CNN based super-resolution training data has a unique corresponding target for a given input image and is a many-to-one mapping. The downscaling operation, however, is a one-to-many mapping; for a given input, there is no known target in the training data. Therefore, existing end to end super-resolution learning [36, 163, 164] and other similar CNN based image processing techniques such as colorization [115, 165] cannot be directly applied to the problems studied in this work.

6.2 Method

We seek to train a convolutional neural network as a non-linear mapper to transform an input image to an output image following what we call the deep feature consistent principle. The schematic is illustrated in Figure 6.2. Our system consists of two components: a transformation network $T_W(x)$ and a loss network $\Phi(x)$. The transformation network is a convolutional neural network parameterized by weights W, which transforms an input image x to an output image $\hat{x}$, i.e. $\hat{x} = T_W(x)$. The other component is the loss network Φ which is a pretrained deep convolutional neural network to help define the feature perceptual loss function for training $T_W(x)$. We feed both the original image x and the transformed image $\hat{x}$ to Φ and compute the feature perceptual loss $\mathcal{L}(x, \hat{x})$. Training $T_W(x)$ is to find the weights W that minimize $\mathcal{L}(x, \hat{x})$, i.e.

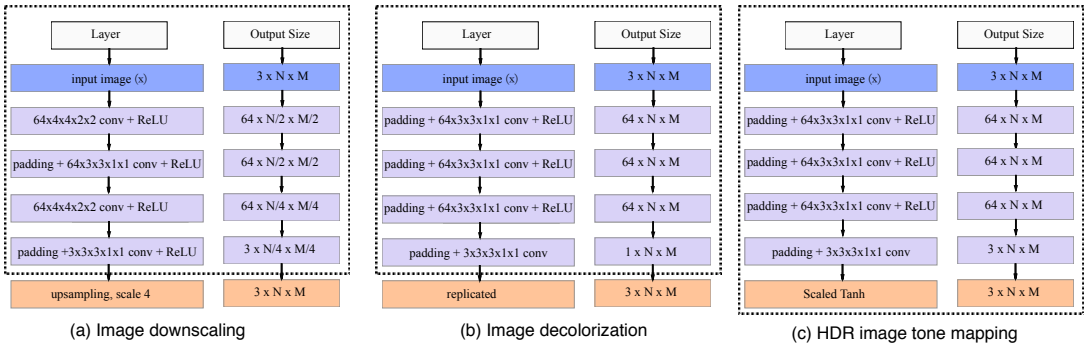


Figure 6.3: Transformation neural network architecture for image downscaling, decolorization and HDR image tone mapping.

$$W^* = \arg\min_W E_x[\mathcal{L}(x, T_W(x))] \tag{6.2.1}$$

Our approach in this chapter can be regarded as the extensions of the previous chapter. But these extensions are non-trivial and non-obvious; each requires in-depth understanding of the problem and ingenuity that cannot be readily derived from existing works. Unlike previous applications, none of our problems has a known ground truth or target for a supervised learning network. Instead, we have to reason about the suitable target and develop solutions to construct the perceptual loss for each application accordingly. In downscaling, we created a perceptual loss to match two images with different shapes (sizes). In colour2gray, we constructed a perceptual loss to match two images with different number of colour channels. In HDR tone mapping, we introduced a perceptual loss to match two images with different dynamic ranges.

6.2.1 Feature Perceptual Loss

As alluded to earlier, the spatial correlation of an image is a major determining factor of the visual integrity of an image. The goal of image transformation in Figure 6.2 and the tasks in Figure 6.1 is to ensure $\hat{x}$ preserves the visual integrity of x. This can be alternatively stated as making the spatial correlations in $\hat{x}$ consistent with those in x. Instead of using handcrafted functions to describe an image's spatial correlations, we make use of a pretrained deep CNN. The hidden layers outputs, which we call deep features, capture the spatial correlations of the input image.

The definition of feature perceptual loss is following the previous section. Specifically, let $\Phi_i(x)$ represent the i^{th} hidden activations when feeding the image x to Φ. If the i^{th} is a convolutional or ReLU layer, $\Phi_i(x)$ is a feature map of shape $[C_i, W_i, H_i]$, where C_i is the number of filter for the i^{th} convolutional layer, H_i and W_i are the height and

width of the given feature map respectively. The feature perceptual loss $\mathcal{L}_i(x, \hat{x})$ for a given layer of two images x and $\hat{x} = T_W(x)$ is defined as the normalized Euclidean distance between the corresponding 3D feature maps. The final loss $\mathcal{L}_i(x, T_W(x))$ is the total loss of different layers.

$$\mathcal{L}_i(x, T_W(x)) = \frac{1}{C_i W_i H_i} \sum_{c=1}^{C_i} \sum_{w=1}^{W_i} \sum_{h=1}^{H_i} (\Phi_i(x)_{c,w,h} - \Phi_i(T_W(x))_{c,w,h})^2 \tag{6.2.2}$$

$$\mathcal{L}(x, T_W(x)) = \sum_i \mathcal{L}_i(x, T_W(x)) \tag{6.2.3}$$

It is worth noting that Φ is pre-trained and fixed during the training of $T_W(x)$, it is used as convolutional filters to capture the spatial correlations of the images.

6.2.2 Transformation Networks Architecture

The transformation networks are convolutional neural networks based on the architecture guidelines from VGGNet [24] and DCGAN [77], and the details of the architecture vary with different image transformation tasks (Figure 6.3).

Image downscaling. For image downscaling we use strided convolutions to construct the networks with 4 x 4 kernels. The stride is fixed to be 2 x 2 to achieve in-network downsampling instead of deterministic spatial functions such as max pooling and average pooling. The ReLU layer is used after the first convolutional layer as non-linear activation function. Thus after two strided convolutions, the size of the input image can be downscaled to $1/4$. In order to compute the feature perceptual loss we need to make sure that the transformed image and the original image have the same shape. In our experiments we apply a 2D upsampling of a factor of 4 over every channel of the transformed output (see Figure 6.4), thus upscaling the downscaled image back to the same size as the original input. The nearest neighbor upsampler is chosen to ensure the upsampled image has the same information as the downscaled image. Thus we can feed the upscaled version and the original image into the loss network to compute the feature perceptual loss.

Image decolorization. The image decolorization transformation only affects the color of the input images, and there is no need to incorporate downsampling architecture in the network. We use 3 x 3 kernels with 1 x 1 stride for all the convolutions. In addition, each feature map of a given input is padded by 1 pixel with the replication of the input boundary before the convolution operation. Thus the convolutional layers do not change the size of the input image. Like the image downscaling network we use ReLU

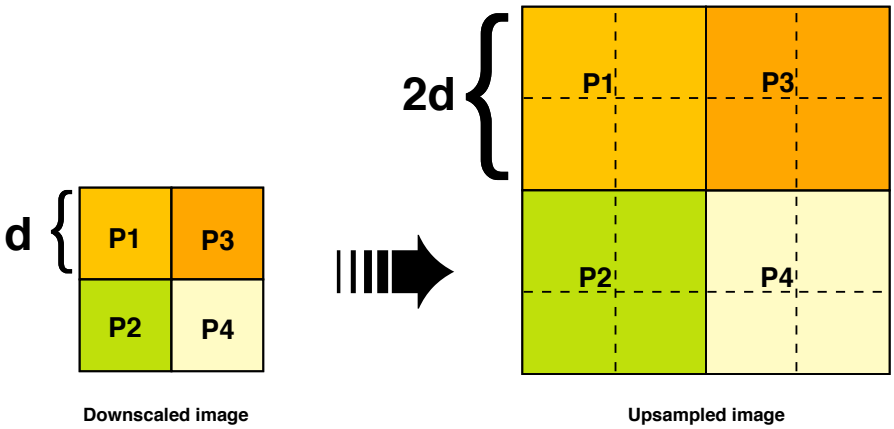


Figure 6.4: Nearest neighbor upsampling for the transformed image. The upsampled image contains the same amount of information as the downscaled image and is the same size as the original input image.

layer after the first convolutional layer, but only a single filter for the last convolution to represent the transformed grayscale image. What we desired is the deep feature consistency of the decolorized output and the original image. We replicate the single channel of the decolorized output to a 3 channel color image (3 channels are identical), which is then fed to the loss network $\Phi(x)$ to calculate the feature perceptual loss with the original input. This is designed to ensure the replicated 3 channel color image have the same amount of information as the decolorized output.

HDR image tone mapping. The network architecture for HDR image tone mapping is similar to the one used in image decolorization above. We use replication to pad the input boundary, and all the convolutions are 3 x 3 kernels with 1 x 1 stride. The difference is that 3 filters are needed for the last convolutional layer for reproducing a color image. The output layer is a scaled Tanh layer, restricting the pixel value of the transformed image to the displayable range [0, 255] from a high dynamic range. During the training we seek the deep feature consistency of the tone mapped and the original high dynamic range image. Specific implementation details of each of the applications are described in the experimental section.

6.3 Experiments

We present experimental results on three image transformation tasks: image downscaling, image decolorization and HDR image tone mapping to demonstrate the effectiveness of our method. We also investigate how the feature perceptual loss constructed with different hidden layers of the loss network affects the performances.

6.3.1 Training Details

Our image downscaling and decolorization transformation CNNs are trained offline using Microsoft COCO dataset released in 2014 [134], which is a large-scale dataset containing 82,783 training images. We resize all the image to 256×256 as the final training data, and train our models with a batch size of 16 for 10 epochs over all the training images. Once the transformation CNN is trained, it can be used to perform downscaling or decolorization.

For HDR image tone mapping, the transformation CNN is trained online, i.e., an HDR image is compressed using the transformation CNN trained with its own data. The practical consideration is that it is difficult to collect large enough training dataset. With large enough collection of training data, the model can also be trained offline.

For training, Adam [32] method is used for stochastic optimization with a learning rate of 0.0002. A pretrained 19-layer VGGNet [24] is used as loss CNN Φ to compute feature perceptual loss which is fixed during the training of the transformation CNN. When constructing the feature perceptual loss for a pretrained network, the first step is to decide which layer (layers) should be used. Unlike image generation works [36, 93] using ReLU layers, we use convolution layers for feature extraction. This is because the ReLU activation is just the corresponding convolutions output thresholded at 0, the convolutions could contain more subtle information when compared with ReLU output. Specifically we experiment feature perceptual loss by using convolutional layer conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 for comparison. Our implementation is built on open source machine learning framework Torch [40].

6.3.2 Image Downscaling

Image downscaling is trying to transform a high-resolution input image to a low-resolution output image. In our experiments we focus on the $\times 1/4$ image downscaling similar to previous works [136, 141]. This seemingly simple routine image operation is actually a technically challenging task because it is very difficult to define the correct low-resolution image. As already discussed, this is a one-to-many mapping and there are many plausible solutions. Based on our DFC-DIT framework, we ensure that the downsampled image and the original image will have similar deep features which means that the output will maintain the spatial correlations of the original image thus keeping the visual integrity of the original image.

Qualitative results. Although our network is trained on images of shape 256×256, it can be adapted to any image sizes because of its fully convolutional architecture. After

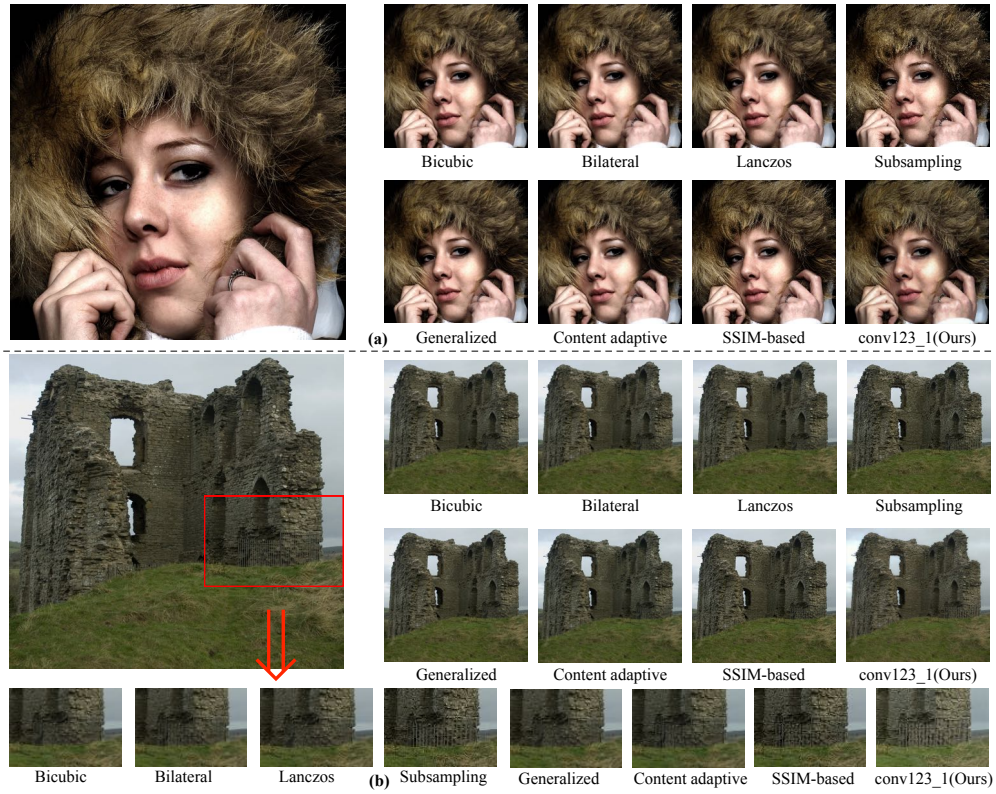


Figure 6.5: A comparison of natural images downscaled by different methods. The results are downscaled by a factor of $\times$ 1/4 while the original inputs are resized for better display. For each image, results of common filters such as Bicubic, Bilateral, Lanczos and Subsampling are shown in the first row. Results of recent methods, generalized sampling [166], content-adaptive[141] and SSIM-based downsampling [136] and ours are shown in the second row. Our conv123_1 results are produced by a model trained with a combined loss of convolutional layers conv1_1, conv2_1, and conv3_1. The bottom row of the second image shows a local region of the downscaled image by different methods. All the images are courtesy of [136]. The results are best viewed in native resolution electronically.

training, we evaluate our method on the testing images from [136]. We first show the qualitative examples and compare our results with other state of the art methods. We then evaluate how perceptual losses constructed at different convolutional layers affect the performances.

Figure 6.5 shows qualitative examples of our results, other common techniques and state of the art methods. We only show results of downscaling by a factor of $\times$ 1/4, the original images are resized for better display. We can see that bicubic filter is known to lead to oversmoothing results and cannot preserve fine structures such as the fence area highlighted by the red rectangle (Figure 6.5(b)). Other filters such as bilateral filter and Lanczos filter achieve sharper downscaled results, however these filters are also prob-

Figure 6.6: A comparison of natural images downscaled with the DFC-DIT framework with different levels of feature perceptual loss. The examples, from left to right, are $\times$ 1/4 downscaling results with perceptual losses computed with individual hidden layers of VGGNet (from layer 1 to layer 5). The last column is the results based on a perceptual loss combining the first 3 layers.

lematic. Bilateral filter can lead to ringing artifacts (the hair in Figure 6.5(a)), and Lanczos filter could not preserve small-scale features such as the fence area in Figure 6.5(b). More advanced methods such as generalized sampling [166], and content-adaptive downscaling [141] and SSIM-based downscaling [136] could produce better results, but still cannot preserve all perceptually important details. In contrast our method trained by a feature perceptual loss constructed using layer conv1_1, conv2_1 and conv3_1 deep features can capture important fine textures and produce better transformed results, visually closer to the original high-resolution inputs. From Figure 6.5(b), the fine textures of the fence area can be seen clearly in the downscaled image. Although simple (nearest neighbor) subsampling can also achieve sharper images, the results are sometimes noisy and suffer from aliasing (see the hair in Figure 6.5(a)). Our algorithm avoids both oversmoothing and aliasing problems and produces a crisp and noise-free image. These results demonstrate that by keeping the deep features of the downscaled image consistent with those of the original can indeed preserve the visual integrity of the input.

Deep feature consistency at different layers. Figure 6.6 shows results of DFC-DIT downscaled images using perceptual losses computed using conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1 layer of the VGGNet individually. We find that keeping the deep feature consistent at these individual layers can in general preserve the original texture or content well. However for the high level layers, the downscaled images could lose detailed pixel information such as pixel color. For example, results of conv4_1 and conv5_1 in Figure 6.6 have higher color contrasts. We also found that by combining the first three layer deep features in general works very well.

6.3.3 Image Decolorization

Like in image downscaling we also train a four-layer convolutional network to transform color images into grayscale images using the DFC-DIT framework. One of the major problems in traditional approach to this task is that in iso-luminant areas the color contrasts will disappear in the grayscale image because even though the pixels have different colors their luminance levels are the same. In our neural network based nonlinear mapping framework, we enforce deep feature consistency which means that the spatial correlations of the color images are preserved in the grayscale image. Thus even in iso-luminant regions, the color contrasts will be preserved as grayscale contrasts.

Qualitative results. Again, our fully convolutional neural network architecture can be applied to process images of any sizes even though the training images have a fixed size. Figure 6.7 shows several comparative results against standard luminance and recent color to grayscale methods [143, 144, 150, 167]. Our training-based approach can preserve the color contrasts of the original images, the grayscale images appear sharp and fine details are well protected. It is interesting to note that unlike previous methods, we did not explicitly compute color contrasts and grayscale contrasts, instead we only enforce deep feature consistency of the color and the decolorized images. From these examples, we have shown convincingly that our DFC-DIT framework is an effective decolorization method.

Deep feature consistency at different layers. We also conduct experiments to evaluate how deep feature consistency at different hidden layer of the loss network affects the decolorization results. Results produced by models trained with perceptual loss of different hidden layers are shown in Figure 6.8. Again we can see that all the transformed images are able to reconstruct the content of the original color image and preserve the contrasts. Compared to lower layers, the decolorized images from higher layers do a better job at reconstructing fine details, especially the contrast preservation that is desired. Specifically the results from lowest layers, i.e., conv1_1 are similar to the luminance channel (Figure 6.7), isoluminant regions are mapped onto the same output intensity and global appearance is not well preserved. Constructing feature perceptual loss from higher layer is better for contrast preserving. However when using the highest conv5_1 layer (Figure 6.8), the contrast of the outputs is too high that makes the decolorized images look unnatural. Our best model is trained by using conv4_1 layer.

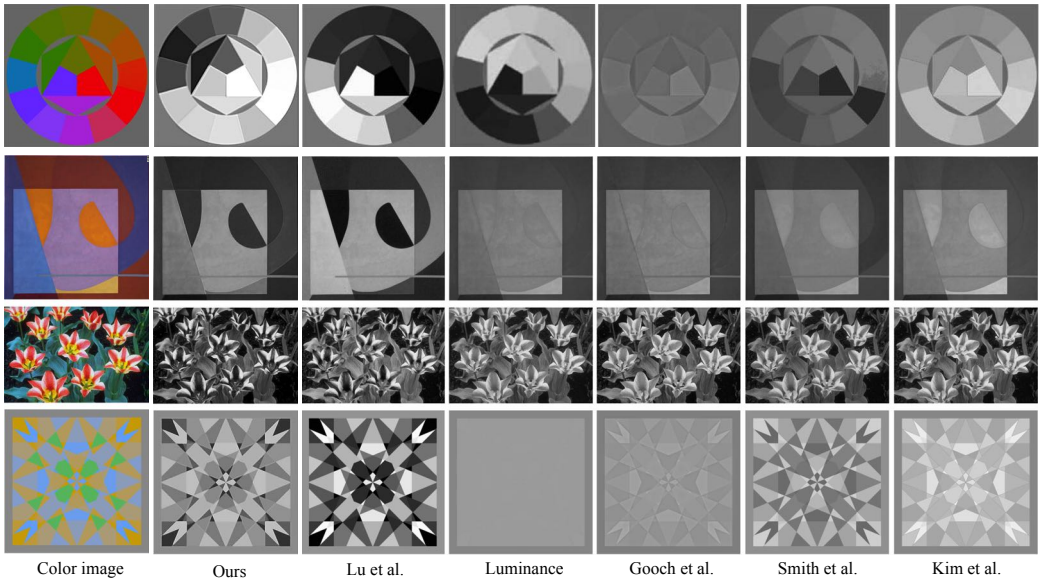


Figure 6.7: A comparison of decolorized images by different methods. We compare our method trained with conv4_1 layer with standard luminance and other recent methods [143, 144, 150, 167]. The results are best viewed electronically.

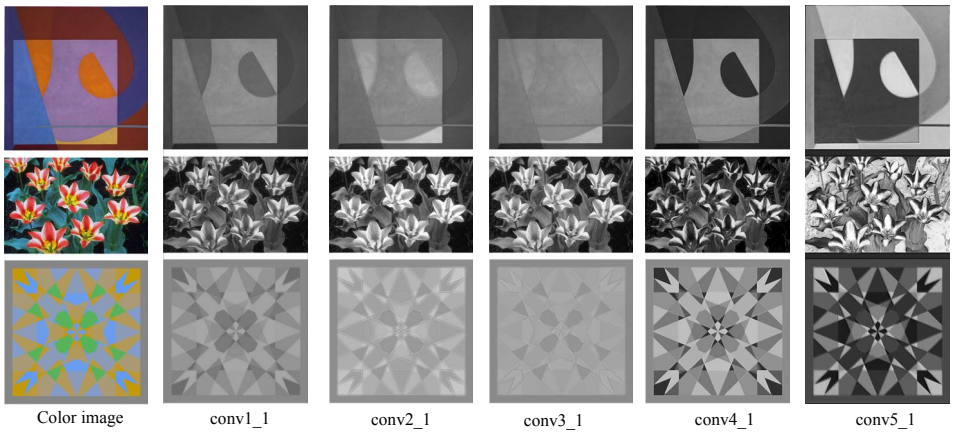


Figure 6.8: A comparison of decolorization by our methods trained with different level feature perceptual loss. The examples are trained from low level to high level layers in VGGNet. The results are best viewed electronically.

6.3.4 HDR Image Tone Mapping

Unlike image downscaling and decolorization where a single model is trained offline using a large collection of training images and used to process all testing images, we adapt one network to a single HDR image due to the lack of large HDR dataset available for training. This can be seen as an online process where we use an HDR image's own data to optimize its own transformation function. It is important to note that this approach is realistic in practice as the process only needs the HDR input to produce

its tone mapped output and there is no need to use any other extra information. The only slight disadvantage is that it requires online training the neural network using an HDR image's own data before outputting the final tone mapped image. Comparing with training the model offline using a large collection of training images, this online approach will be slower because it needs to adapt the neural network to the current testing image before producing the output tone mapped image. In our implementation on a machine with an Intel Core i7-4790K CPU and a Nvidia Tesla K40 GPU, it takes around 20 seconds to tone map a 768 x 512 HDR image.

It is a common practice to process the HDR radiance map in the logarithmic domain, we feed the logarithm of the radiance signal to the transformation CNN. Dynamic range compression is achieved by a *Tanh* function in the last layer of the transformation network (Figure 6.3(c)). In practice, the dynamic range of the input HDR radiance signal is compressed to the displayable range $[0, 255]$. Following the principle of DFC-DIT, the HDR tone mapping transformation network is optimized by enforcing deep feature consistency between the transformation output image and the original HDR radiance map.

Rendering display image. The output of the transformation network will have the correct dynamic range suitable for display, however, its colour may not be correct due to the nonlinear mapping operations of the transformation CNN. We therefore need to render the output of the transformation network to have the correct colour for display. Therefore, we only use the luminance channel of tonemapped image and combine it with the color channels of the original HDR image. As in other tone mapping method [154], the final tone mapped image is rendered as:

$$R_{out} = (\frac{R_{in}}{L_{in}})^{\gamma} L_{out} \tag{6.3.1}$$

$$G_{out} = (\frac{G_{in}}{L_{in}})^{\gamma} L_{out} \tag{6.3.2}$$

$$B_{out} = (\frac{B_{in}}{L_{in}})^{\gamma} L_{out} \tag{6.3.3}$$

where R_{out}, G_{out} and B_{out} are the final tone-mapped RGB channels, R_{in}, G_{in} and B_{in} are the original radiance values in the corresponding HDR channels, and γ can be used to render the correct display colour. L_{in} and L_{out} are respectively the luminance value of the HDR radiance map and the luminance value of the transformation image by the transformation CNN. According to the literature, γ should be set between 0.4 and 0.6 and we set it to 0.5 in all our results.

Qualitative results. Figure 6.9 and Figure 6.10 show examples of tone mapping results

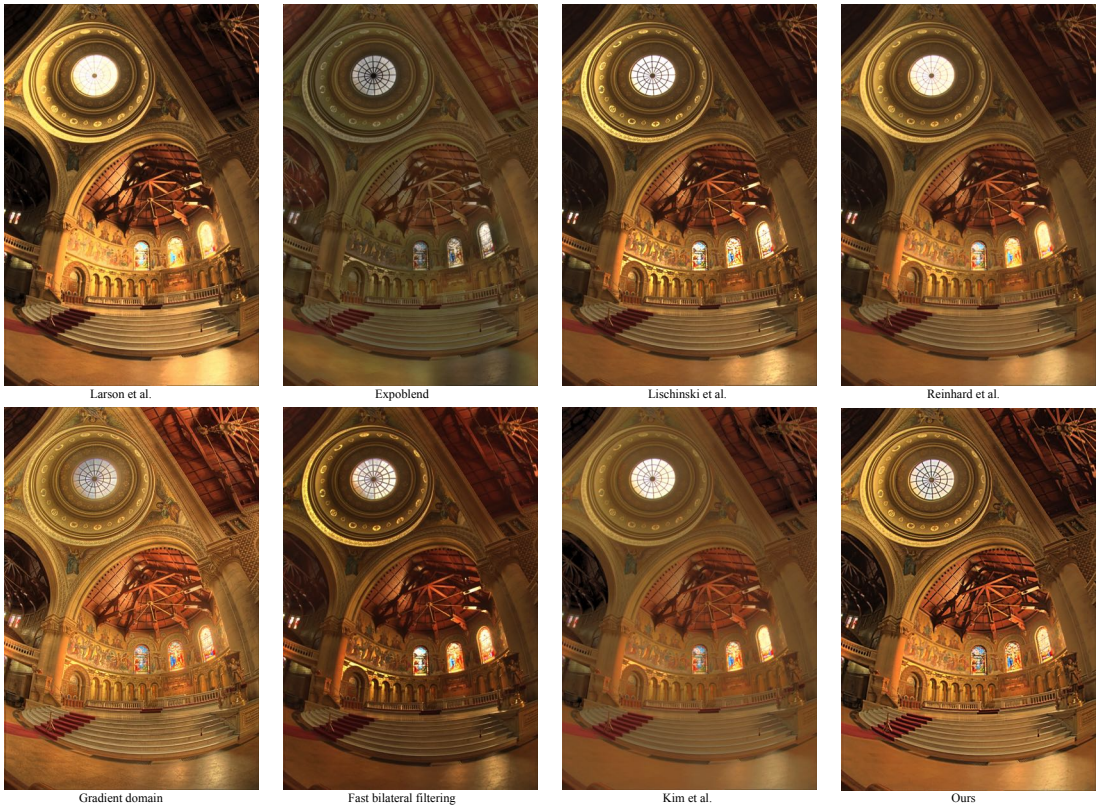


Figure 6.9: Stanford Memorial Church displayed using different methods. We show those of Larson et al. [138], Expoblend [168], Lischinski et al. [169], Reinhard et al. [156], gradient domain [140], fast bilateral filtering [155] and Kim et al. [170]. Our results are based on feature perceptual loss of 3 layers conv1_1, conv2_1 and conv3_1.

Figure 6.10: Sunset image displayed using different methods. We show those of [138], Expoblend [168], Lischinski et al. [169], Reinhard et al. [156], gradient domain [140], fast bilateral filtering [155] and Kim et al. [170]. Our results are based on feature perceptual loss of 3 layers conv1_1, conv2_1 and conv3_1. Our results are based on feature perceptual loss of 3 layers conv1_1, conv2_1 and conv3_1.

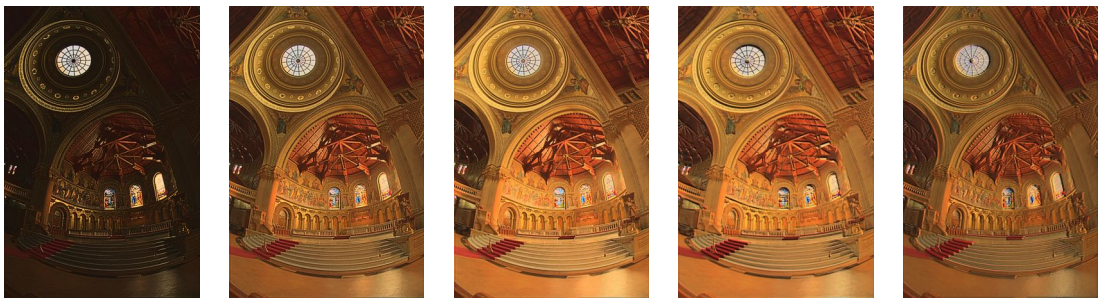


Figure 6.11: A comparison of HDR image tone mapping by our methods trained with different level feature perceptual loss. The results are best viewed electronically.

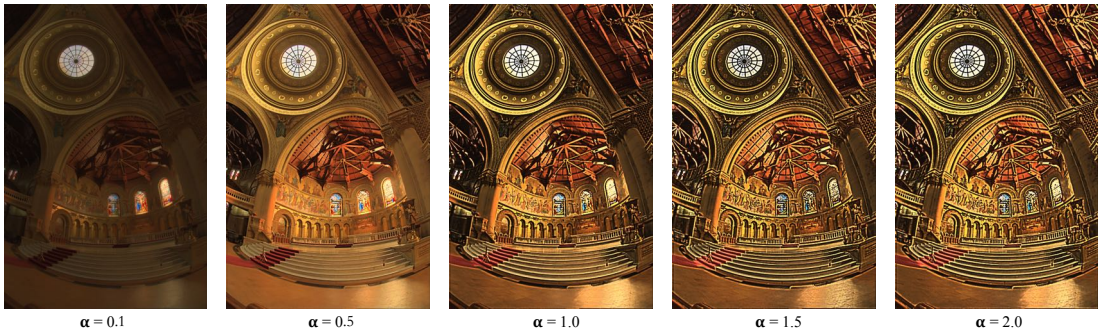


Figure 6.12: A demonstration of the effects of logarithmic compression based on feature perceptual loss of 3 layers conv1_1, conv2_1 and conv3_1.

of some HDR radiance maps of real scenes that are widely used in the literature, i.e., "Stanford Memorial Church" and "Vine Sunset". We compare our results with some of the best known and latest methods in the literature including Larson et al. [138], Expoblend [168], Lischinski et al. [169], Reinhard et al. [156], gradient domain [140], fast bilateral filtering [155] and Kim et al. [170]. From Figure 6.9 and Figure 6.10, we can see that our method is able to render the images with excellent visual appearances to keep tiny details and contrast of the radiance map, which are at least as good as those produced by the best methods.

Deep feature consistency at different layers. In Figure 6.11 we show how feature perceptual loss from different hidden layers affect the tone mapped images of the DFC-DIT framework for HDR tone mapping. Overall the tone mapped images based on perceptual losses from the middle level (conv2_1 and conv3_1) have a good balance between local and global contrasts. Combining the perceptual losses of first several layers together tend to produce somewhat better results than using a single layer. The tone mapped outputs based on higher layers (conv4_1 and conv5_1) appear slightly bumpy effect on different regions.

The effects of logarithmic compression. As mentioned above, we first compress the

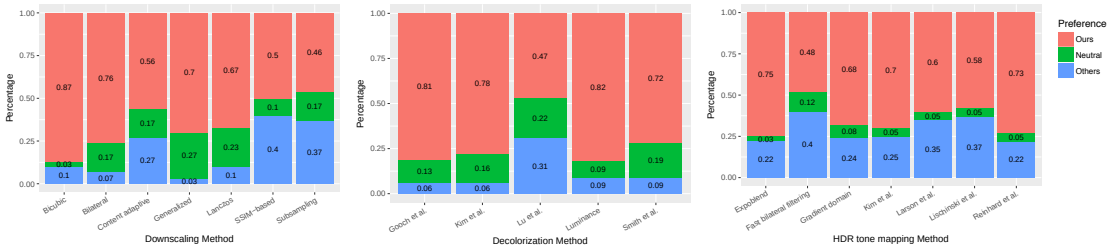


Figure 6.13: Subjective evaluation results. The red areas represent the percentage that our algorithm is selected, green areas for no preference and the blue ones for the other methods.

HDR radiance map with the logarithmic functions and try to seek the deep feature consistency in the logarithmic domain. We can multiply the compressed radiance map with a factor α to control the logarithmic transformation. The tone mapping results with different α are shown in Figure 6.12. It can be seen that a higher α can lead to a more noticeable local contrast and crisp appearance of tone mapped results. This is because the compressed HDR radiance map with a higher α retains a higher dynamic range in logarithmic domain and retain more important details. It is clear that our method can extract exquisite details from high-contrast images. It works well when α is around 0.5 in our experiments.

6.3.5 Subjective Evaluation of DFC-DIT Framework

We have conducted a subjective evaluation of results of downscaling, decolorization and HDR tone mapping of the new DFC-DIT framework. For each transformation, we evaluate our technique against several best techniques in the literature. For downscaling, we use bicubic, bilateral, lanczos, subsampling, generalized sampling [166], content-adaptive [141] and SSIM based method [136] as the benchmarks. For decolorization, we use luminance the methods of Smith et al. [144], Kim et al. [167], Gooch et al [143] and Lu et al. [150] as benchmarks. For HDR tone mapping we use Larson et al. [138], fast bilateral filtering [155], gradient domain [140], Expoblend [168], Kim et al. [170], Lischinski et al. [169] and Reinhard et al. [156] as benchmarks. For each image, we show the original input image (in the case of HDR tone mapping, the original radiance map cannot be shown), a version produced by our method and a version of one benchmark technique to subjects and ask which version they prefer or indicate no preference. 50 undergraduate science and engineering students from our university evaluated 10 pairs of images for image downscaling and 8 pairs of images for image decolorization and HDR tone mapping. Figure 6.13 shows the voting results. We can see that there is an obvious preference for our method against all other methods for all

the transformation tasks. These results demonstrate DFC-DIT framework is comparable to or better than state of the art techniques. In image downscaling, subsampling and SSIM-based are two competing methods to produce sharp and crisp downscaled images, however subsampling sometimes suffers strong aliasing artifacts like the hair in Figure 6.5. In image decolorization, the method of Lu et al. [150] is the best competing candidate that maximally preserves color contrast. However some participants prefer ours than theirs because the decolorized versions of Lu et al. [150] may show too strong contrast while the corresponding color images in fact have low contrasts. For HDR image tone mapping, fast bilateral filtering [155] is the best comparable tone mapping operator in our study.

6.4 Summary

Traditional image processing tasks like image downscaling, image decolorization and HDR image tone mapping require reducing image information while preserving visually and perceptually important details. These problems are inherently ill-posed and there is no well-defined ground truth and they are one-to-many mapping problems. For image downscaling fine details should be preserved from visually ambiguous high-resolution inputs; for image decolorization the gray image should be semantically similar to the original color version and preserve the contrast as much as possible in spite of drastic loss of color information; for HDR image tone mapping we want to compress the scene radiance to displayable range while preserving details and color appearance to appreciate the original scene content. In this chapter, we propose to compare perceptual similarities between two images in feature space and have developed the deep feature consistent deep image transformation (DFC-DIT) framework to tackle these image information reduction problems. Our experiments demonstrated the effectiveness of the method and its state of the art performances.

CHAPTER 7

Deep Feature Based Image Quality Assessment

With the previous chapters showing the effectiveness of deep models to tackle different image processing problems like image generation and image transformation (information augmentation and reduction) by constructing objective functions that ensure the consistency of deep features between two images, in this chapter we further explore the deep feature consistent concept from the perspective of image quality assessment. To this end, we propose a new deep feature based metric for assessing perceptual image quality by measuring the inconsistency between the distorted image and the reference image in feature space built on deep convolutional neural network. We further demonstrate the effectiveness and promise of the proposed method through a set of intuitive examples, as well as quantitative comparison to other commonly used image quality assessment methods based on a large subjective image quality database with different distortion types.

7.1 Related Work

Objective image quality assessment (IQA) aims to measure the degradation and predict the quality of images in accordance with human perception. Depending on the accessibility of the original images, with which distorted images are compared, IQA algorithms can be classified into different categories [171], i.e., *full-reference*, *no-reference* and *blind* image quality assessment. In this section, we will review several commonly used full-reference image quality metrics.

The simplest one is Peak Signal to Noise Ratio (PSNR), which represents the ratio between the maximum possible power of a signal and the power of distortion noise,

which is based on per-pixel mean squared error (MSE) between the original image and the distorted image. In general, a higher PSNR value usually indicates better image quality, however in some cases it could not generalize well to perceived visual quality [129–132]. In particular, for image reconstruction or generation tasks the output images tend to be very blurry when compared to natural images. This is because the per-pixel loss (MSE) only considers point-wise signal difference rather than the perceptual difference.

Another widely used and cited one is SSIM index [118], which is designed to improve on traditional methods like PSNR. The essential difference with respect to PSNR is that SSIM is a more perceptual-based approach by comparing the structures of the reference and the distorted signals instead of absolute point-wise error. SSIM is based on the assumption that the human visual system is highly adapted to extract structural information from visual information and considers image degradations as perceived changes in structural information variation.

Additionally complex wavelet structural similarity (CW-SSIM) index [172] is proposed as an extension of the SSIM to the complex wavelet domain as a general purpose image similarity metric. The key idea is that certain image distortions can lead to consistent phase change in the local wavelet coefficients, and the structural content of the image doesn't change with a consistent phase shift of the coefficients.

The image quality assessment indexes mentioned above are purely human-crafted, and the key insight is to design better method by incorporating the structural information to measure the inconsistency (or similarity) of a given pair of images. In this work, we start from a learning-based approach to constructing IQA index in feature space.

7.2 Deep Feature Based Image Quality Assessment (DFB-IQA)

7.2.1 Motivation

The previously mentioned IQA indexes, i.e., SSIM and CW-SSIM demonstrate that it is crucial to manually extract structural information of an image either in pixel or complex wavelet domain. In parallel, the fact that natural image signals are highly structured is also considered to build state of the art object recognition systems like deep convolutional neural networks (CNNs). In particular, CNNs use a set of filters or kernels like $11 \times 11 \times 3$ (i.e. 11 pixel width and height, and 3 for channel depth) to spatially convolve across the width and height of the input volume to compute dot products between the entries of each filter and the input at any position. Thus, each output ele-

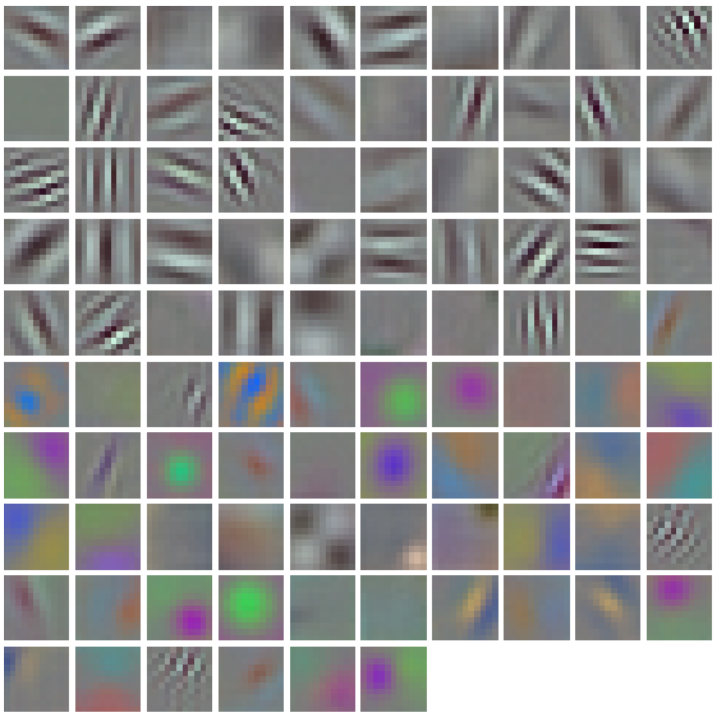


Figure 7.1: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer of pretrained AlexNet [23].

ment is the result of a local region of the input volume with a given filter. Conceptually these kind of convolution operations are able to capture the structural information due to the local connectivity. Previous works on image recognition tasks have shown that deep convolutional neural networks can learn interpretable filters based on a large-scale dataset. Figure 7.1 shows the 96 convolutional kernels of size $11 \times 11 \times 3$ learned by first convolutional layer of AlexNet [23] trained on ImageNet [12]. We can see that there are a variety of meaningful frequency and orientation-selective kernels, as well as various colored blobs.

At each hidden layer, a deep feature covers a different size of an input receptive field as illustrated in Figure 7.2. Equivalently, a deep feature in convolutional layer 1 computes the local structure of a 3x3 spatial block. Inspecting the convolutional weights in Figure 7.1, it is not difficult to see that some filters will compute the local luminance or weighted average (when all weights have the same signs), some will compute local contrasts along different directions (when the weights have both positive and negative values). For higher layers, each hidden unit covers a (larger and larger) local block in the input image and it can be easily shown that each hidden unit in these higher layers also has an equivalent convolutional kernel the same size as its corresponding receptive field covering the input spatial block. Again, depending on the signs of the filter

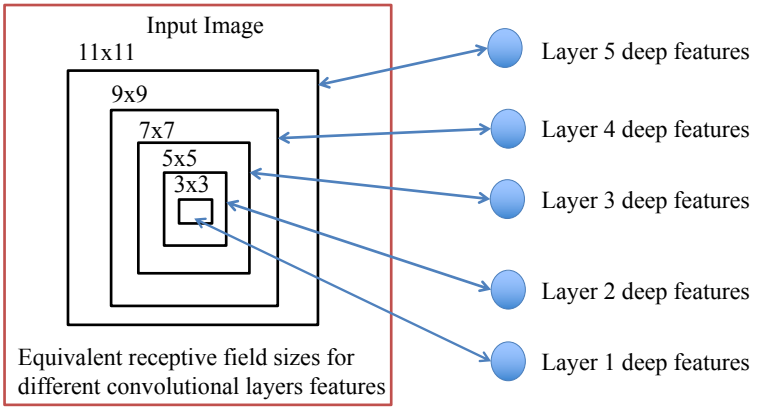


Figure 7.2: Deep features at different hidden layers and their corresponding equivalent receptive field size. In essence, each hidden layer unit (deep feature) computes the local pixel structure of its corresponding receptive field region.

coefficients, they either compute the local luminance or local contrasts of various sorts.

7.2.2 Deep Feature Based Image Quality Assessment Index

Our DFB-IQA index tries to provide a good approximation to perceived image distortion by incorporating image pixel spatial correlation and object structure information, which are implicitly captured by the the learned filters through a large-scale image dataset training instead of human engineering. In other words, the proposed DFB-IQA index measures the structural similarity of two images in the learned feature space.

In order to measure the similarity between a distorted image and a reference image, we first need to process the input images from pixel space to feature space. Similar to the definition of feature loss in the previous chapters, we choose the pretrained VGGNet (denoted as $\Phi(x)$) [24] on ImageNet [12] as our deep feature extraction filter. Specifically, let $\Phi_i(x)$ represent the i^{th} hidden activations when feeding the image x to Φ. $\Phi_i(x)$ is a feature map of shape [C_i, W_i, H_i], where C_i is the number of filter for the i^{th} convolutional layer, H_i and W_i are the height and width of the given feature map respectively. Thus, we can easily define the deep feature based mean squared error (DFB-MSE) for a reference image x and a distorted image $\tilde{x}$ as follows:

$$DFB\text{-}MSE(x,\tilde{x}) = \frac{1}{C_i W_i H_i} \sum_{c=1}^{C_i} \sum_{w=1}^{W_i} \sum_{h=1}^{H_i} (\Phi_i(x)_{c,w,h} - \Phi_i(\tilde{x})_{c,w,h})^2 \tag{7.2.1}$$

The final DFB-IQA index is defined via the DFB-MSE in logarithmic scale similar to PSNR. DFB-IQA index can be formulated as:

$$
\begin{aligned}
DFB\text{-}IQA(x, \tilde{x}) &= 10\, log_{10} \frac{MAX^2(\Phi(x), \Phi(\tilde{x}))}{DFB\text{-}MSE} \\
&= 20\, log_{10}(MAX(\Phi(x), \Phi(\tilde{x}))) - 10\, log_{10}(DFB\text{-}MSE)
\end{aligned}
$$

where $MAX(\Phi(x), \Phi(\tilde{x}))$ is the maximum possible signal value of deep feature extracted from a reference x and the corresponding distorted image $\tilde{x}$.

7.3 Experiments

Image quality assessment is a very subjective task, which requires a plenty of human labeled dataset and carefully designed experiments. Although many existed image quality assessment methods can achieve consistent results for a given set of distorted images with the same type of distortions, the effectiveness of these algorithms for image quality prediction could be diluted significantly when applied to more complicated conditions in which there exist a variety of different types of distortions. Thus, cross-distortion testing environment is needed to evaluate the generalizability of image quality assessment algorithms.

7.3.1 Testing dataset

In this work, we evaluate the proposed DFB-IQA index on Release 2 version of LIVE Image Quality Assessment Database [118, 173, 174], which is a large dataset including 30 reference images and 779 distorted versions. It contains 5 distortion types: JPEG Compression (169 images), JPEG2000 Compression (175 images), Gaussian Blur (145 images), White Noise (145 images) and Fast Fading Rayleigh (145 images). In addition, the difference mean opinion score (DMOS) (1 to 100) value for each distorted image is computed according to human rating. Specifically about 20-29 human observers rated each image and all the observers were asked to provide the perception of quality on a continuous linear scale marked as adjectives "Bad", "Poor", "Fair", "Good" and "Excellent". The raw scores for each subject were converted to difference scores (between the test and the reference) by the mean and variance of scores for that subject, and then the entire dataset was scaled and shifted to the range 1 to 100. In our experiments, we convert difference mean opinion score (subtracted by 100) to *mean opinion score* (MOS) for comparison.

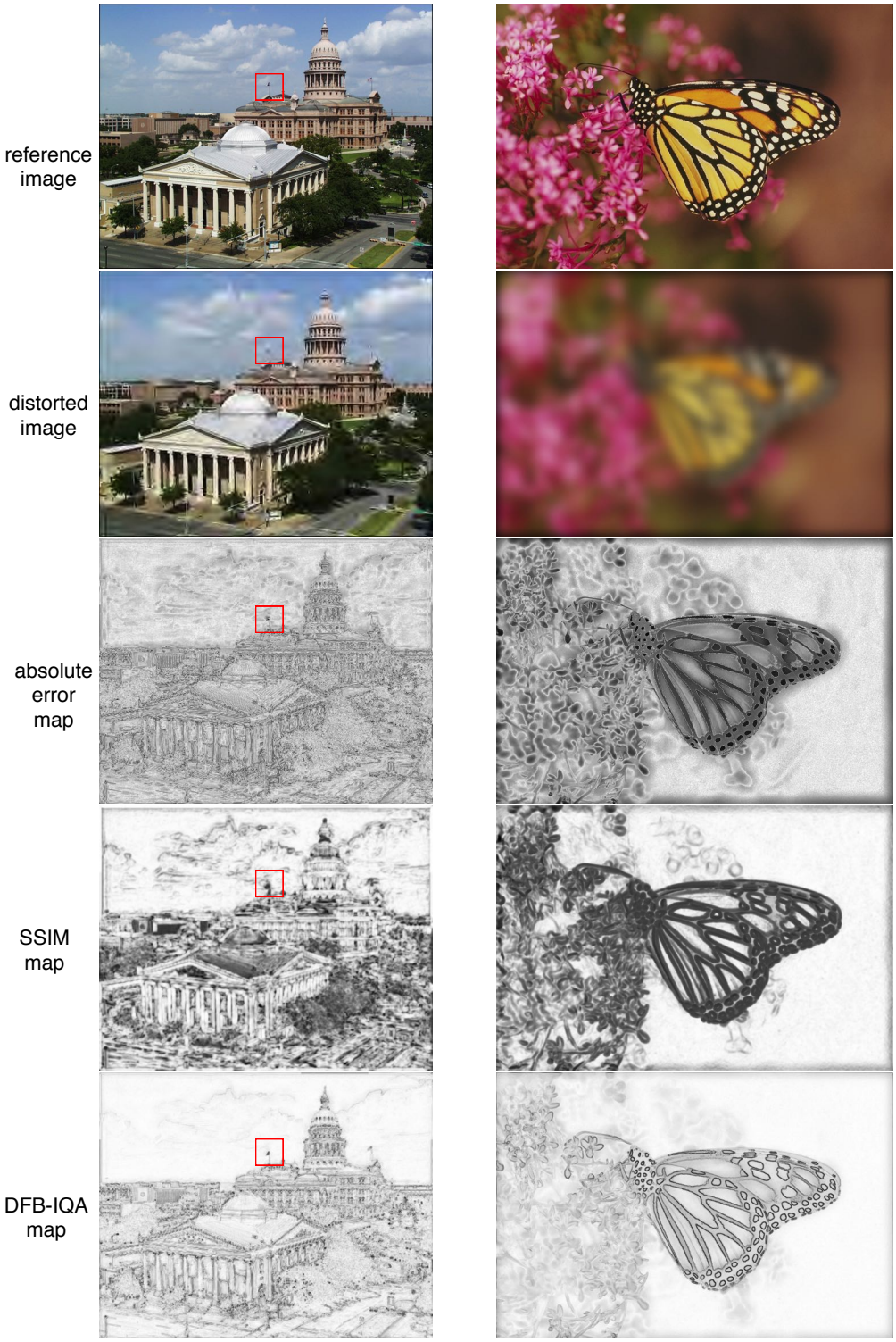


Figure 7.3: Samples with different distortions, the left is compressed by JPEG2000 and the right is processed by Gaussian bur. The last 3 rows show absolute error map, SSIM index map and DFB-IQA index map.

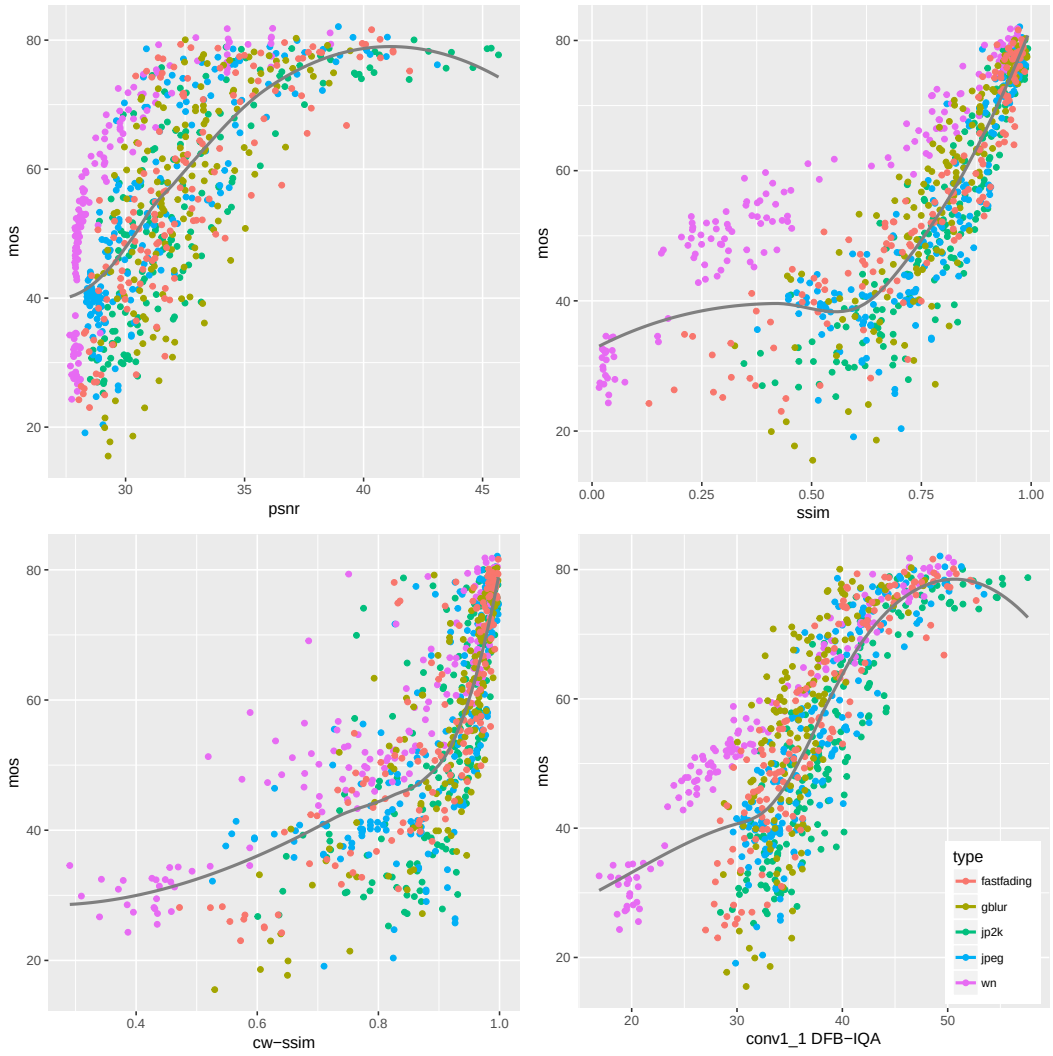


Figure 7.4: Scatter plots of mean opinion scores (MOS) versus PSNR, SSIM, CW-SSIM and DFB-IQA. DFB-IQA index is calculated based the features from *conv*1_1 layer.

7.3.2 Results

Based on LIVE Image Quality Assessment Database, we conduct experiments to compare the performance of the proposed DFB-IQA index with PSNR and SSIM. One the other hand, due to the hierarchical architecture of deep convolutional neural network we also investigate the effect of different level features on the performance of DFB-IQA.

Qualitative analysis. In Figure 7.3, we show two samples in the LIVE database with different types of distortions: the left image is called"churchandcapitol" and compressed by JPEG2000 with bit rate of 0.13, the right one is "monarch" and distorted by Gaussian bur with σ of 11.33. Additionally the absolute error map, SSIM index map and DFB-IQA index map are calculated for each distorted image. The absolute error

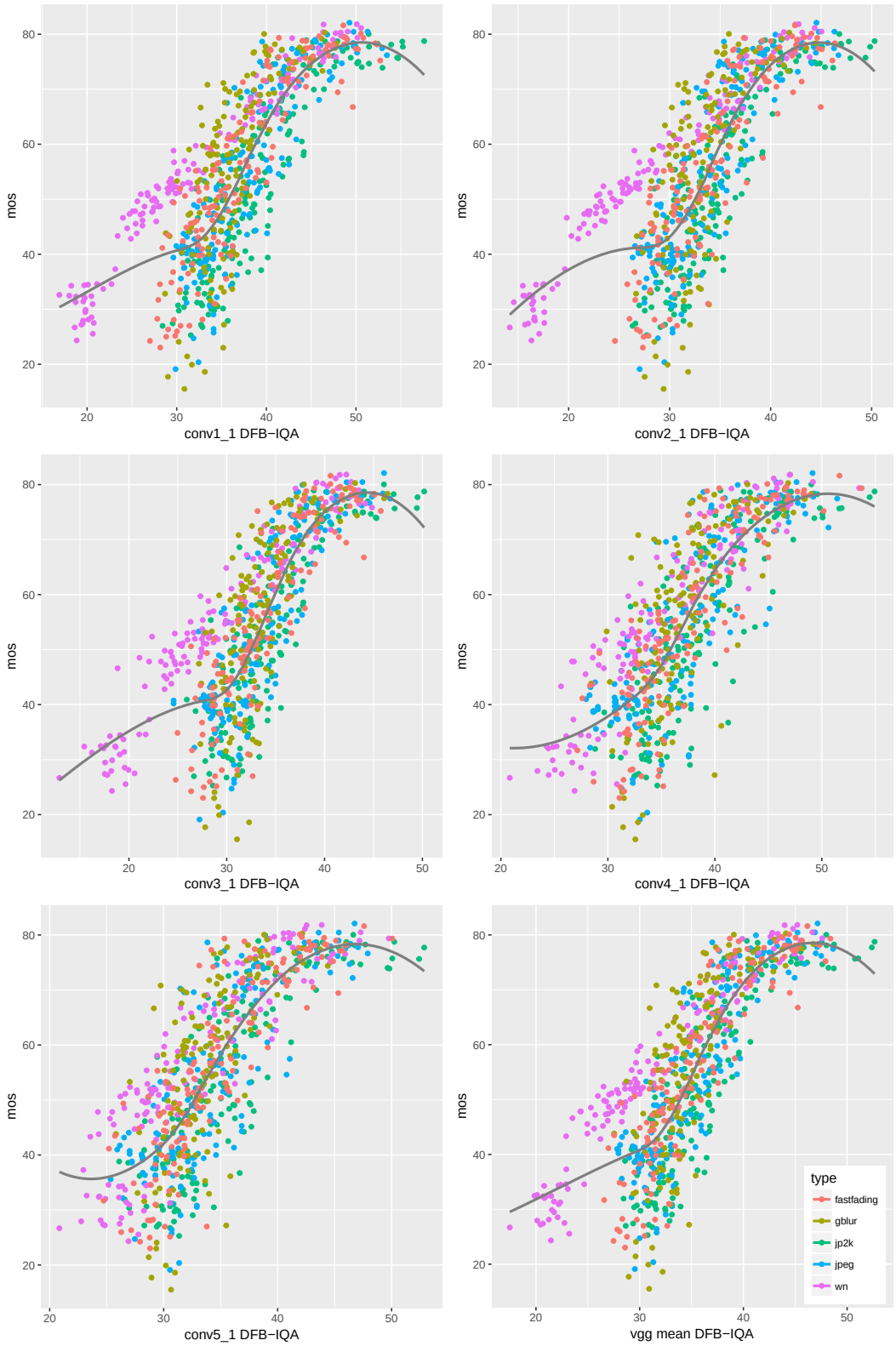


Figure 7.5: Scatter plots of mean opinion scores (MOS) versus DFB-IQA indexes calculated by different convolutional layers. The vgg mean is the average results across the 5 convolutional layers in VGGNet.

map and DFB-IQA index map are contrast-inverted for easier comparison with SSIM index map.

From Figure 7.3, we can see that the absolute error map is quite sensitive to the point-wise difference between each pixel and contains some "visual noise" (visually not important for humans), which gives the error map a mottled or grainy appearance (e.g. the sky and buildings in the church image). As a result, the absolute error map focuses more on the point-wise details rather than overall structural information, which is more important to perceived visual quality. In contrast, the error maps of SSIM and DFB-IQA index are more sensitive to structural difference and can capture the perceived changes in structural information variation. Moreover, the DFB-IQA index map shows a much clearer structural appearance and local structures of the original image are better preserved, for example, we can observe the well-structured flag in the "church" image and sharp skeleton of the butterfly in Figure 7.3.

In order to get a more comprehensive comparison of different image quality assessment algorithms, we show the correlation map of MOS versus different model predictions with different types of distortions. The DFB-IQA index is based on features from the first convolutional layer *conv*1_1. From Figure 7.4, we can see that the proposed DFB-IQA performs quite well in this test and behaves consistently with MOS. Though SSIM performs well for a single type of distortion, it cannot generalize well for cross-distortion testing. On the contrary, the scatter plot of DFB-IQA index is more compact, demonstrating that it supplies remarkably good prediction of the mean opinion scores.

In addition, we investigate the effect of different level features on the performance of DFB-IQA for subjective MOS prediction. In Figure 7.5, we show the scatter plots of MOS versus DFB-IQA by using layer conv1_1, conv2_1, conv3_1, conv4_1, conv5_1 and the average results of the 5 layers. We can observe the trend that DFB-IQA indexes constructed from higher layers behave more consistently with mean opinion scores when applied to distorted images created from different types of distortions. Specifically, there is an obvious divergence in the scatter plot for *white noise* distortion in lower layers (conv1_1 and conv2_1), while it shows a more compact distribution for all types of distortions by using higher layers (conv4_1 and conv5_1). This could be explained that higher layers of the convolutional neural network mainly capture the high-level content in terms of objects and overall structures with bigger receptive field size instead of limiting to detailed pixel information. From Figure 7.2, it can be easily understood because these higher layers cover a larger receptive field in the input image. Thus, the difference between different types of distortions will not be quite significant, what matters is the degradation degree for each type of distortion method.

Table 7.1: Performance comparison of different image quality assessment indexes. CC: correlation coefficient after regression analysis; MAE: mean absolute error; RMS: root mean square error; SROCC: Spearman rank-order correlation coefficient. 6 DFB-IQA results are calculated: the first 5 are calculated based on DFB-IQA of single convolutional layer in VGGNet, DFB-IQA-Mean is based on the average features from the 5 convolutional layers.

	Linear Regression			Loess Regresssion			Rank-order
Model	CC	MAE	RMS	CC	MAE	RMS	SROCC
PSNR	0.70	9.56	11.54	0.72	10.10	11.96	0.72
SSIM	0.74	9.17	10.84	0.83	8.38	9.84	0.85
CW-SSIM	0.69	9.71	11.65	0.74	9.68	11.53	0.79
DFB-IQA1	0.79	8.15	9.82	0.83	8.64	10.15	0.82
DFB-IQA2	0.77	8.60	10.37	0.82	8.86	10.42	0.80
DFB-IQA3	0.78	8.16	10.02	0.83	8.57	10.20	0.82
DFB-IQA4	0.80	7.58	9.65	0.82	8.57	10.36	0.82
DFB-IQA5	0.79	7.94	9.91	0.81	8.71	10.48	0.81
DFB-IQA-Mean	0.81	7.68	9.46	0.85	8.23	9.80	0.84

Quantitative analysis. We further conduct quantitative experiments to measure the performances of different image quality assessment algorithms. We follow the evaluation procedures in [118] to fit different regression models between subjective mean opinion scores and other objective image quality assessment indexes, and use the predictions of the regression models to build quantitative measures. Specifically we consider both linear regression and non-linear regression model in our experiments, and we choose Local Polynomial Regression (known as LOESS) as our non-linear fitting. LOESS is an attractive non-parametric regression method that combines much of the simplicity of linear least squares regression with the flexibility of nonlinear regression. LOESS fitting curves are shown in Figure 7.4.

We then choose three metrics to evaluate the performances of different IQA indexes. We first calculate the correlation coefficient (CC) between the subjective MOS scores and objective IQA scores after regression analysis, served as an evaluation of prediction correlation and dependence; we also calculate the mean absolute error (MAE) and root mean square error (RMS) between the subjective and objective scores after regression, providing the evaluation of accuracy; Lastly Spearman rank-order correlation coefficient (SROCC) between the subjective MOS scores and objective IQA scores is used as a measure of prediction monotonicity.

The evaluation results are shown in Table 7.1. All the results are calculated based on all the types of distortions in the LIVE Image Quality Assessment Database [174]. Specifi-

cally we compare DFB-IQA with PSNR, SSIM and CW-SSIM. DFB-IQA1 to DFB-IQA5 are calculated based on the features extracted from the conv1_1 to conv5_1 layer and DFB-IQA-Mean is based on the average features from the 5 convolutional layers. We can see that our proposed DFB-IQA methods work well in general and DFB-IQA-Mean performs better than all the other methods. It is demonstrated that DFB-IQA index has a better consistent and stable correlation with subjective mean opinion scores in cross-distortion evaluation.

7.4 Discussion

In this and previous chapters, we show that the learned features extracted from pre-trained deep convolutional neural network for image classification task can be applied to different image processing problems. This is because modern deep neural networks can learn from low-level to high-level visual features when trained on image dataset, and the learned features can be used for different image related tasks under the transfer learning framework [175]. The first-layer features extracted from pretrained deep CNN usually resemble either edge detectors or color blobs (Figure 7.1), which are considered as general features. On the other hand, the features computed by the last layer (i.e., fully connected layer for image classification tasks) of a trained network are considered as specific features, because they must depend greatly on the chosen dataset and specific task. Therefore the features extracted from first few layers of a pretrained model contain low-level and general image information, which are suitable for the image processing tasks considered in this work.

7.5 Summary

In this chapter, we propose the use of learned features to design image quality assessment metrics under deep learning framework. The key insight is the capability of pretrained deep convolutional neural network to incorporate structural and perceptual image information, which can be used as an alternative motivating principle for the design of image quality measures. Our experiments demonstrate the effectiveness of the proposed DFB-IQA index with respect to subjective mean opinion scores prediction in cross-distortion settings. DFB-IQA can be seen as an alternative or complementary to traditional approaches like SSIM, which tries to incorporate structural information by human engineering while DFB-IQA seeks to extract image pixel spatial correlation and object structure information from learned features based on image recognition tasks.

CHAPTER 8

Concluding Remarks

8.1 Summary of the Thesis and Contributions

It is both surprising and thrilling to witness the rapid advances in the fields of deep learning in the past few years. Deep convolutional neural networks have become the workhorse behind the state of the art performance for most computer vision and object recognition tasks. Deep CNN methods have been shown to learn robust feature representations directly from image pixels through end-to-end training. In this thesis we developed models and techniques that push the frontier of traditional image processing problems by exploring the deep features extracted from pre-trained deep CNNs.

Concretely, in Chapter 3 we first consider quickly filtering out irrelevant information in an image for a given task. Specifically we dived into certain hidden neurons of the deep CNN and proposed Object Specific Channel (OSC) that responds more strongly to certain object categories. We then used face as a case study and introduced a multi-scale approach for extracting OSC features, based on which we built a compact and fast face detector with state of the art performance in unconstrained settings. In Chapter 4 we turned to the task to produce visually pleasing images. We proposed to incorporate pretrained deep CNN into the architecture of two popular generative models: variational autoencoder and generative adversarial networks. This allowed us to produce high quality images and learn powerful image representations in the latent space. In Chapter 5 and Chapter 6, we revisited both image information augmentation and reduction tasks. We proposed a deep feature consistent framework and developed deep models to tackle traditional image processing problems for both image information augmentation (companding and inverse halftoning) and reduction (downscaling, decolorization and HDR tone mapping). Finally, in Chapter 7 we further explored the deep feature consistent concept and proposed a new deep feature based metric for

assessing image quality from the perspective of learned structural image information, which demonstrated it's effectiveness and promise for image quality assessing in cross-distortion settings.

From a modeling perspective, the methods and models developed in this work belong to the category of deep learning. In particular, our deep models are designed seamlessly integrated with a pretrained convolutional neural network, which is used to extract learned features in order to define objective functions. As a result, the whole architecture is just a single differentiable function used to process raw images for different image processing tasks and it can be optimized through an end-to-end training manner on GPUs. We hope the design of our model architecture can be reused or referred in future work.

8.2 Future Work

In spite of the rapid advance of deep learning and progress in visual recognition, the research opportunities in computer vision and image processing are far from diminishing, on the contrary, deep learning has energized the related research fields and stimulated a variety of new problems and research directions. Moreover, introducing new methods to solve well known problems can always provide new perspectives and directions to the research field. It is clear that many challenges still remain and new ideas need further explore. For example, in this thesis we consider the traditional image processing problems in a learned feature space which is based on high-level visual recognition tasks. In comparison, the traditional methods mainly focus on manually engineering in low-level pixel space. In other words, the traditional image processing paradigm is a bottom-up approach, piecing together of base elements which are then linked together to give rise to more complex systems; Our new paradigm is a top-down approach, starting with an overview of the whole system and mimicking the overall hypothesized functionality of the input. With this new top-down paradigm, visual recognition systems can be used as "black boxes" to assist traditional image processing tasks. To make further progress, this kind of visual recognition assisted image processing paradigm is a worthy research direction we can turn to next.

References

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[4] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[8] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[9] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to

supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1635–1643, 2015.

[10] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[11] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663, 2017.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286, 2014.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[16] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[19] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[20] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*, pages 2217–2225, 2016.

[21] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[29] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[30] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[31] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[32] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[35] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[36] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXiv:1603.08155*, 2016.

[37] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*, 2016.

[38] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[39] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[40] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[41] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra,

Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[42] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[43] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. URL `http://learningsys.org/papers/LearningSys_2015_paper_33.pdf`.

[44] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[45] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[46] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[47] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.

[48] Jianguo Li and Yimin Zhang. Learning surf cascade for fast and accurate object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3468–3475, 2013.

[49] Minh-Tri Pham, Yang Gao, Viet-Dung D Hoang, and Tat-Jen Cham. Fast polygonal integration and its application in extending haar-like features to improve object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 942–949. IEEE, 2010.

[50] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014.

[51] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[52] Junjie Yan, Xuzong Zhang, Zhen Lei, and Stan Z Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790–799, 2014.

[53] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *Computer Vision–ECCV 2014*, pages 720–735. Springer, 2014.

[54] Haoxiang Li, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Gang Hua. Efficient boosted exemplar-based face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1843–1850, 2014.

[55] Ravi Shekhar and CV Jawahar. Word image retrieval using bag of visual words. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 297–301. IEEE, 2012.

[56] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004.

[57] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3676–3684, 2015.

[58] Jinguo Liu, Yafeng Deng, and Chang Huang. Targeting ultimate accuracy: Face recognition via deep embedding. *arXiv preprint arXiv:1506.07310*, 2015.

[59] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Convolutional channel features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 82–90, 2015.

[60] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[61] Sachin Sudhakar Farfade, Mohammad J Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 643–650. ACM, 2015.

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networ` `pdf`.

[63] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855. IEEE, 2006.

[64] Martin Köstinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2144–2151. IEEE, 2011.

[65] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[66] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[67] Nenad Markuš, Miroslav Frljak, Igor S Pandžić, Jörgen Ahlberg, and Robert Forchheimer. A method for object detection based on pixel intensity comparisons organized in decision trees. *arXiv preprint arXiv:1305.4537*, 2013.

[68] Dong Chen, Shaoqing Ren, Yichen Wei, Xudong Cao, and Jian Sun. Joint cascade face detection and alignment. In *Computer Vision–ECCV 2014*, pages 109–122. Springer, 2014.

[69] Martin Köstinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Robust face detection by simple means. In *DAGM 2012 CVAW workshop*, 2012.

[70] Venkatesh Bala Subburaman and Sébastien Marcel. Fast bounding box estimation based face detection. In *ECCV, Workshop on Face Detection: Where we are, and what next?*, number EPFL-CONF-155015, 2010.

[71] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[72] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. *arXiv preprint arXiv:1512.00570*, 2015.

[73] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[74] Karl Ridgeway, Jake Snell, Brett Roads, Richard Zemel, and Michael Mozer. Learning to generate images with perceptual similarity metrics. *arXiv preprint arXiv:1511.06409*, 2015.

[75] Alex Lamb, Vincent Dumoulin, and Aaron Courville. Discriminative regularization for generative models. *arXiv preprint arXiv:1602.03220*, 2016.

[76] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[77] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[78] Jon C Leachtenauer, William Malila, John Irvine, Linda Colburn, and Nanette Salvaggio. General image-quality equation: Giqe. *Applied Optics*, 36(32):8322–8328, 1997.

[79] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[80] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[81] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[82] Neeraj Kumar, Peter Belhumeur, and Shree Nayar. Facetracer: A search engine for large collections of images with faces. In *European conference on computer vision*, pages 340–353. Springer, 2008.

[83] Ning Zhang, Manohar Paluri, Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1644, 2014.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[85] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[86] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*, 2016.

[87] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[88] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

[89] Hussein A Aly and Eric Dubois. Image up-sampling using total-variation regularization with a new observation model. *IEEE Transactions on Image Processing*, 14(10):1647–1659, 2005.

[90] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[91] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[92] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[93] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. *arXiv preprint arXiv:1610.00291*, 2016.

[94] Yuanzhen Li, Lavanya Sharan, and Edward H Adelson. Compressing and companding high dynamic range images with subband architectures. In *ACM transactions on graphics (TOG)*, volume 24, pages 836–844. ACM, 2005.

[95] Thomas D Kite, Niranjan Damera-Venkata, Brian L Evans, and Alan C Bovik. A fast, high-quality inverse halftoning algorithm for error diffused halftones. *IEEE Transactions on Image Processing*, 9(9):1583–1592, 2000.

[96] Mei-Yin Shen and C-C Jay Kuo. A robust nonlinear filtering approach to inverse halftoning. *Journal of Visual Communication and Image Representation*, 12(1):84–95, 2001.

[97] Glenn R Easley, Vishal M Patel, and Dennis M Healy Jr. Inverse halftoning using a shearlet representation. In *SPIE Optical Engineering+ Applications*, pages 74460C–74460C. International Society for Optics and Photonics, 2009.

[98] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[99] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141, March 2017.

[100] Bian Yang, Martin Schmucker, Wolfgang Funk, Christoph Busch, and Shenghe Sun. Integer dct-based reversible watermarking for images using companding technique. In *Electronic Imaging 2004*, pages 405–415. International Society for Optics and Photonics, 2004.

[101] Sunil Bhooshan, Vinay Kumar, and HP Solan. 2d t-law: a novel approach for image companding. In *Proceedings of the 4th WSEAS international conference on Circuits, systems, signal and telecommunications*, pages 19–22. World Scientific and Engineering Academy and Society (WSEAS), 2010.

[102] Ming Yuan Ting and Eve A Riskin. Error-diffused image compression using a binary-to-gray-scale decoder and predictive pruned tree-structured vector quantization. *IEEE Transactions on Image Processing*, 3(6):854–858, 1994.

[103] Gozde Bozkurt Unal and A Enis Çetin. Restoration of error-diffused images using projection onto convex sets. *IEEE Transactions on Image Processing*, 10(12): 1836–1841, 2001.

[104] Robert L Stevenson. Inverse halftoning via map estimation. *IEEE Transactions on Image Processing*, 6(4):574–583, 1997.

[105] Ramesh Neelamani, Robert David Nowak, and Richard G Baraniuk. Winhd: Wavelet-based inverse halftoning via deconvolution. *IEEE Transactions on Image Processing*, 2002.

[106] Yun-Fu Liu, Jing-Ming Guo, and Jiann-Der Lee. Inverse halftoning based on the bayesian theorem. *IEEE Transactions on Image Processing*, 20(4):1077–1084, 2011.

[107] Yuki Minami, Shun-ichi Azuma, and Toshiharu Sugie. Inverse halftoning using super-resolution image processing. *IEEJ Transactions on Electrical and Electronic Engineering*, 7(2):208–213, 2012.

[108] Murat Mese and Palghat P Vaidyanathan. Look-up table (lut) method for inverse halftoning. *IEEE Transactions on Image Processing*, 10(10):1566–1578, 2001.

[109] Chang-Hwan Son. Inverse halftoning based on sparse representation. *Optics letters*, 37(12):2352–2354, 2012.

[110] Chang-Hwan Son and Hyunseung Choo. Local learned dictionaries optimized to edge orientation for inverse halftoning. *IEEE Transactions on Image Processing*, 23(6):2542–2556, 2014.

[111] Pedro G Freitas, Mylène CQ Farias, and Aletéia PF Araújo. Enhancing inverse halftoning via coupled dictionary training. *Signal Processing: Image Communication*, 49:1–8, 2016.

[112] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[113] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[114] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, 35(4):110, 2016.

[115] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision*, pages 577–593. Springer, 2016.

[116] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[117] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[118] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[119] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[120] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[121] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3, 2016.

[122] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*, pages 702–716. Springer, 2016.

[123] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[124] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[125] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[126] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[127] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4829–4837, 2016.

[128] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[129] Bernd Girod. What's wrong with mean-squared error. *Digital images and human vision*, pages 207–220, 1993.

[130] Patrick C Teo and David J Heeger. Perceptual image distortion. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 2, pages 982–986. IEEE, 1994.

[131] Michael P Eckert and Andrew P Bradley. Perceptual quality metrics applied to still image compression. *Signal processing*, 70(3):177–200, 1998.

[132] Zhou Wang and A. C Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002.

[133] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[134] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[135] Alessandro Foi, Vladimir Katkovnik, Karen Egiazarian, and Jaakko Astola. Inverse halftoning based on the anisotropic lpa-ici deconvolution. In *Proceedings of Int. TICSP Workshop Spectral Meth. Multirate Signal Process*, 2004.

[136] A Cengiz Oeztireli and Markus Gross. Perceptually based downscaling of images. *ACM Transactions on Graphics (TOG)*, 34(4):77, 2015.

[137] Mark Grundland and Neil A Dodgson. Decolorize: Fast, contrast enhancing, color to grayscale conversion. *Pattern Recognition*, 40(11):2891–2896, 2007.

[138] Gregory Ward Larson, Holly Rushmeier, and Christine Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, 1997.

[139] Jack Tumblin and Greg Turk. Lcis: A boundary hierarchy for detail-preserving contrast reduction. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 83–90. ACM Press/Addison-Wesley Publishing Co., 1999.

[140] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 249–256. ACM, 2002.

[141] Johannes Kopf, Ariel Shamir, and Pieter Peers. Content-adaptive image downscaling. *ACM Transactions on Graphics (TOG)*, 32(6):173, 2013.

[142] Nicolas Weber, Michael Waechter, Sandra C. Amend, Stefan Guthe, and Michael Goesele. Rapid, detail-preserving image downscaling. *ACM Trans. Graph.*, 35 (6):205:1–205:6, November 2016. ISSN 0730-0301. doi: 10.1145/2980179.2980239. URL `http://doi.acm.org/10.1145/2980179.2980239`.

[143] Amy A Gooch, Sven C Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: salience-preserving color removal. *ACM Transactions on Graphics (TOG)*, 24(3): 634–639, 2005.

[144] Kaleigh Smith, Pierre-Edouard Landes, Joëlle Thollot, and Karol Myszkowski. Apparent greyscale: A simple and fast conversion to perceptually accurate images and video. In *Computer Graphics Forum*, volume 27, pages 193–200. Wiley Online Library, 2008.

[145] Karl Rasche, Robert Geist, and James Westall. Re-coloring images for gamuts of lower dimension. In *Computer Graphics Forum*, volume 24, pages 423–432. Wiley Online Library, 2005.

[146] Codruta Orniana Ancuti, Cosmin Ancuti, and Phillipe Bekaert. Enhancing by saliency-guided decolorization. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 257–264. IEEE, 2011.

[147] Cewu Lu, Li Xu, and Jiaya Jia. Contrast preserving decolorization. In *Computational Photography (ICCP), 2012 IEEE International Conference on*, pages 1–7. IEEE, 2012.

[148] Min Qiu, Graham D Finlayson, and Guoping Qiu. Contrast maximizing and brightness preserving color to grayscale image conversion. In *Conference on Colour in Graphics, Imaging, and Vision*, volume 2008, pages 347–351. Society for Imaging Science and Technology, 2008.

[149] Yibing Song, Linchao Bao, Xiaobin Xu, and Qingxiong Yang. Decolorization: Is rgb2gray () out? In *SIGGRAPH Asia 2013 Technical Briefs*, page 15. ACM, 2013.

[150] Cewu Lu, Li Xu, and Jiaya Jia. Contrast preserving decolorization with perception-based quality metrics. *International journal of computer vision*, 110(2): 222–239, 2014.

[151] Codruta O Ancuti, Cosmin Ancuti, Chris Hermans, and Philippe Bekaert. Image and video decolorization by fusion. In *Asian Conference on Computer Vision*, pages 79–92. Springer, 2010.

[152] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic images. *IEEE Computer graphics and Applications*, 13(6):42–48, 1993.

[153] Greg Ward. A contrast-based scalefactor for luminance display. *Graphics gems IV*, pages 415–421, 1994.

[154] Guoping Qiu, Jiang Duan, and Graham D Finlayson. Learning to display high dynamic range images. *Pattern recognition*, 40(10):2641–2655, 2007.

[155] Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *ACM transactions on graphics (TOG)*, volume 21, pages 257–266. ACM, 2002.

[156] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM transactions on graphics (TOG)*, 21(3): 267–276, 2002.

[157] Jiang Duan, Marco Bressan, Chris Dance, and Guoping Qiu. Tone-mapping high dynamic range images by novel histogram adjustment. *Pattern Recognition*, 43(5): 1847–1862, 2010.

[158] Francesco Banterle, Alessandro Artusi, Kurt Debattista, and Alan Chalmers. *Advanced High Dynamic Range Imaging: Theory and Practice*. AK Peters (CRC Press), Natick, MA, USA, 2011. ISBN 9781568817194.

[159] Fei Zhou and Qingmin Liao. Single-frame image super-resolution inspired by perceptual criteria. *IET Image Processing*, 9(1):1–11, 2014.

[160] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[161] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436, 2015.

[162] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, March 2017. ISSN 2333-9403. doi: 10.1109/TCI.2016.2644865.

[163] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014.

[164] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.

[165] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[166] Diego Nehab and Hugues Hoppe. Generalized sampling in computer graphics. Technical report, Tech. rep., feb, 2011.

[167] Yongjin Kim, Cheolhun Jang, Julien Demouth, and Seungyong Lee. Robust color-to-gray via nonlinear global mapping. In *ACM Transactions on Graphics (TOG)*, volume 28, page 161. ACM, 2009.

[168] Neil DB Bruce. Expoblend: Information preserving exposure blending based on normalized log-domain entropy. *Computers & Graphics*, 39:12–23, 2014.

[169] Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. Interactive local adjustment of tonal values. *ACM Transactions on Graphics (TOG)*, 25(3): 646–653, 2006.

[170] Min H. Kim and Jan Kautz. Consistent tone reproduction. In *Proceedings of the Tenth IASTED International Conference on Computer Graphics and Imaging*, CGIM '08, pages 152–159, Anaheim, CA, USA, 2008. ACTA Press. ISBN 978-0-88986-720-8. URL `http://dl.acm.org/citation.cfm?id=1722302.1722332`.

[171] Zhou Wang and A. C Bovik. Reduced- and no-reference image quality assessment. *Signal Processing Magazine IEEE*, 28(6):29–40, 2011.

[172] M. P. Sampat, Z. Wang, S Gupta, A. C. Bovik, and M. K. Markey. Complex wavelet structural similarity: a new image similarity index. *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, 18(11):2385, 2009.

[173] H.R. Sheikh, M.F. Sabir, and A.C. Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[174] Hamid R Sheikh, Zhou Wang, Lawrence Cormack, and Alan C Bovik. Live image quality assessment database release 2.

[175] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.