Doctoral Dissertation (Shinshu University)

Deep learning for artworks analysis and synthesis

March 2018

TAN WEI REN

SUPERVISORS:
Kiyoshi Tanaka
Hernán Aguirre

Dedicated to my family.

ABSTRACT

With the rapid technology advancement, paintings digitalization has been made possible. Hence, it becomes desired to solve various artwork related problems through computer with the assistance from machine learning-based solutions. Some interesting and important applications include archiving digital artworks, painting analysis, image stylization, etc. However, research in this domain has received limited attention, mainly due to the difficulty in collecting the artworks to create a proper artworks dataset. Fortunately, many digital fine-art paintings have been made available online over the past few years, making the access to these digital artworks easier than ever. Recently, a large scale Wikiart dataset was introduced and released, encouraging more research in this domain.

Among the available machine learning techniques, deep learning has received much attention in the last decades. Evidently, deep learning is currently the state-of-the-art solution to many computer vision problems. More interestingly, deep learning enables automatic features learning, contrast to the traditional methods where the features are human engineered. However, most works are only focusing on datasets with more structural subjects, e.g. digits, faces, and natural objects. Compared to these works, recognition of artworks is more challenging because a lot of art categories require understanding of non-structural cues, e.g. perspective, emotion, history, etc. In addition, many artworks are non-figurative or structured abstract.

Overall, this thesis is interested in the application of deep learning on artworks. In particular, this thesis aim to *understand the features learning of artworks using deep models* because features extraction is the first step to many computer vision problems, including artworks analysis. For this purpose, deep models are trained for artworks *classification* and *synthesis* tasks and analyzed via various visualization methods. This will reveal what kind of features can the deep models learn from the artworks. Furthermore, this thesis also concerned on the *issues arise when embedding deep models into real-world devices*. To this end, this thesis is divided into two parts.

The first part of this thesis first focuses on training and understanding deep *discriminative* models for artworks classification tasks, i.e. categorizing the *genre*, *artist*, or *style* of an artwork. In this work, a modified Wikiart dataset is used and released to encourage better comparative studies in future works. In particular, the dataset is explicitly split into training and validation sets. The best models found employed and finetuned an ImageNet pre-trained model, achieving state-of-the-art results. Then, the neurons' activations are visualized

in this work to analyze the features learned. The visualizations show that the models are able to learn the structural visual cues from the artworks. Meanwhile, it is also found that the differences between the art categories can be extremely abstract and hard to recognize. For instance, the difference between Baroque and Rococo lies on the emotions felt from the paintings. Grasping such abstract concepts is difficult even for a human, while it is unclear that if the models are able to capture such cues with the visualization technique used.

Next, a Generative Adversarial Network (GAN) variant, named **ArtGAN** is proposed and trained on the Wikiart dataset conditioned on the given *genre*, *artist*, or *style*. GAN is a type of *generative* model that learns to simulate true data from the joint distribution, hence able to learn richer representations compared to discriminative model. More importantly, GAN is able to synthesize photo-realistic images compared to other generative models, which is extremely useful when assessing the features learned by the models. The contributions of the proposed ArtGAN are three-fold: 1) The labels are leveraged when calculating the loss function to improve the image quality for each category; 2) Autoencoder is incorporated to compute energy-based loss function for additional complementary information; 3) Most importantly, a novel **magnified learning** is proposed to learn the correlations between neighbouring pixels better, improving image quality. In addition, it allows synthesizing images at higher resolution compared to the maximum resolution available in the dataset. For quantitative assessment, the proposed ArtGAN is trained on CIFAR-10 and STL-10 datasets, and achieves state-of-the-art results on Inception score. Meanwhile, the results show that the proposed ArtGAN is able to generate high resolution photo-realistic images. Furthermore, the synthesized artworks show that deep models seem able to learn some abstract characteristics from the artworks through visual cues.

In the second part of the thesis, the applicability of deep models in real-world devices is explored. In practice, it is infeasible to embed deep models into resource limited hardware, e.g. mobile devices. This is because many deep models are designed to have extremely high memory requirement. To address this problem, it is desired to reduce the memory requirement with minimum compensation on the model performance. In this work, a novel one-shot deep compression method based on the fuzzy quantity space is proposed to remove redundant weights. The experiments demonstrate that the proposed approach is able to compress the deep models up to 14 times with a minimal loss of classification accuracy.

This thesis is concluded with a summary of its contributions. Last but not least, promising directions of future works are outlined.

PUBLICATIONS

The contributions of this thesis have appeared previously in my publications [185, 186, 183]. Meanwhile, the journal [184] is still under review at the time of writing. All source codes for my publications can be found in https://github.com/willtwr. The released updated Wikiart dataset can be downloaded from http://cs-chan.com/.

ACKNOWLEDGEMENTS

First of all I want to thank my parents, Pua Siew Hoo and Tan Hock Guan, for supporting everything I do.

Next, I want to thank Prof. Kiyoshi Tanaka for accepting me to the doctoral programme. I would like to thank all my supervisors, Prof. Kiyoshi Tanaka and Prof. Hernán Aguirre for providing guidance, encouragement, and advice throughout my time as their student. I would like to express my gratitude to the following members of faculty, Mrs. Watanabe, Ms. Keiko Nishizawa and Mr. Takeuchi Masayuki for their administrative supports during my graduate study.

I would also like to thank my ex-supervisor from University of Malaya, Dr. Chan Chee Seng. Although I have left his laboratory to further my studies in Japan, he continued to provide support, guidance, and advice for my research.

Then, I also want to thank my ex-colleague Dr. Fabio Daolio, who is currently a data scientist in ASOS, for recommending me the field of Digital Humanities. Without his recommendation, I might not able to open my eyes to this new and interesting field.

Finally, I want to thank the Japanese taxpayers for their kindness and generosity in funding my studies for these four wonderful years.

CONTENTS

Part I

INTRODUCTION TO DEEP LEARNING FOR ARTWORK

This part starts with the introduction to the broad field of digital humanities. Then, gradually narrow down the focus to machine learning and computer vision, followed by deep learning and artwork related problems. Finally, it describes the motivations and goals of this thesis.

1 INTRODUCTION

1.1 DIGITAL HUMANITIES

With the rapid advancement of technology, many historical artifacts and fine-art paintings have been made available in digital form. This gives rise to the emergence of *digital humanities*, which is an intersection of *digital technologies* and the disciplines of the *humanities*. However, the definition of digital humanities is still undergoing constant reformulation by scholars and practitioners. This is because the field is constantly growing and changing, making specific definitions quickly become outdated and unnecessarily limit the future potential. Currently, digital humanities incorporates both digitalized and born-digital materials, encompassing a wide range of topics: digital archiving; data mining and analysis of large cultural datasets; 3D modelling of historical artifacts; alternate reality games; and much more. Nonetheless, many *computer techniques* are extremely effective for solving many problems in digital humanities. This thesis will focus on problems related to fine-art paintings or artworks (these two terms will be used interchangeably).

1.1.1 *Brief History of Computer Techniques for Paintings Analysis*

The power of computer methods arises from their capability to extract visual features that are hard to be determined by biological eyes. The history of digital paintings analysis can be dated back to 19th-century when x-ray is used to reveal underdrawings[1] and pentimento[2]. Later, more techniques such as infra-red photography, reflectography, multispectra fluorescence, and ultra-violet imaging were developed to reveal different type of composition in a painting [120, 9, 11, 118]. This encourages collaborations between the scholars from various fields, including computer vision, pattern recognition, image processing, computer graphics, and art history to develop rigorous computer methods to address the increasing number of problems in art history.

More interestingly, these days, fine-art paintings can be scanned to produce extremely high quality digital colour images with recent technology. Taking advantage of this technology, most of the artworks have been digitalized and made available on the internet nowadays[3].

1 Underdrawing is a sketch made by painter as a preliminary guide and subsequently covered with layers of paint.

2 Pentimento a visible trace of earlier painting beneath a layer or layers of paint on a canvas.

3 An example is http://www.wikiart.org/, which contains large amount of artworks.

Meanwhile, *machine learning* and *computer vision* techniques have shown promising results and progress in solving various image-related problems. Hence, application of these techniques on the digitalized paintings becomes an interesting and important topic among the scholars to solve related problems.

1.2 MACHINE LEARNING

As data sources proliferate along with the computing power, machine learning has become a first-class ticket to the most exciting careers in data analysis today. According to Arthur Samuel in 1959, machine learning is defined as the "field of study that gives computers the ability to learn without being explicitly programmed" [124]. That is, machine learning explores the study and construction of algorithms that can learn from and make prediction on data.

However, the "No Free Lunch" theorem of Wolpert [204] states that there is no one model or solution that works best for every problem. In general, one may first assess the problem based on its data form. In data analysis, there are fundamentally four types of data forms:

1. *Text and number*, which are available in many sources, e.g. emails, online search engines, client information stored in banks and companies, etc. In addition to the text that can be made sense by human, other data that are represented as a set of characters and/or numbers include protein structure, radio signal, etc.

2. *Image*, which is stored excessively across the internet servers nowadays with the advancement in computer technology. For instance, large internet-related companies such as Google and Baidu support image search. In addition, social media such as Weibo and Facebook allow their users to upload images without limit.

3. *Video*, which is generally defined as the recording of moving images. Video is used for various purposes, including entertainment in streaming media and CCTV for surveillance.

4. *Audio*, which is mainly used in the recording, manipulation, mass-production, and distribution of sound, including recording of songs, instrumental pieces, podcasts, sound effects, and others.

Each data form is usually encoded with different formats in the computer. For instance, *text* is commonly encoded as a set of 1D string arrays. Meanwhile, *colour image* is represented as a 3D matrix in the computer. Under suitable design, the machine learning model is able to automatically analyse these data and build its own logic based on

the data. Although the problems faced by each data form are interesting and important in their own way, this thesis will be concerned entirely with *image*-related problems, which share similar interest with this thesis.

1.3 COMPUTER VISION

As it turned out, one of the very best application areas for machine learning for many years was computer vision. Computer vision can be defined as "an interdisciplinary field that build artificial systems that obtain information from images or videos". As a scientific discipline, it seeks to automate tasks that the human visual system can do. That is, humans use their eyes and their brain to see and visually sense the world around them.

Visual sense is extremely important for solving various problems in many daily activities conducted by humans. For example, when you see someone is holding a gun wandering on the street, your brain will quickly interpret that the person is dangerous and decide to flee the scene immediately. Correspond to this example, one of the open problems in computer vision is to recognize suspicious behaviour of the pedestrians using surveillance camera.

Although computer vision problems include *video* analysis, this thesis will only focus on the problems related to *image*, as aforementioned. In general, many computer vision problems are related to a branch of machine learning. It is called *pattern recognition*, which focuses on the recognition of patterns and regularities in the data. On the other hand, computer vision mostly involves processing and analysing images for applications such as segmentation, object detection, vision based learning, etc. Next, several important applications in computer vision are briefly described.

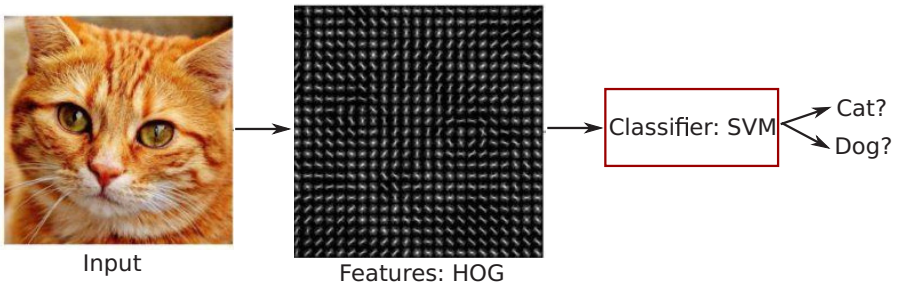


Figure 1: Image classification pipeline. *Image processing* step: The image is first pre-processed to extract relevant features that best summarize the image. *Machine learning* step: Then, the features are used by a classifier to classify the category of the image.

1.3.1 *Image Classification*

Image classification aims to define the category of an image. It has been one of the most explored computer vision problems. Figure 1 summarizes the pipeline for image classification. Many machine learning solutions have been proposed to solve image classification problem. The methods include k-nearest neighbors algorithm [4, 14], Random Forest [16], Bayes classifier [108], Support Vector Machine [206], etc. However, one problem with these solutions is that their performances are highly affected by the *features* used. An image feature is a piece of information which is relevant and best at representing the image in a different high-dimensional space for solving the computational task. Few commonly used image features include points, edges, histogram of oriented gradient (HOG) [31], Scale-invariant feature transform (SIFT) [116], Bag-of-visual-words descriptor [208], etc. Although using these hand-crafted features achieved good performance, it is still far from satisfying. Meanwhile, it is very difficult and complex to design a better feature.

Recently, Krizhevsky et. al. [97] proposed a Convolutional Neural Network (CNN) and achieved the state-of-the-art result for ImageNet dataset [153]. More importantly, CNN demonstrated that it is feasible to learn the relevant features of an image without the need of manual engineering [214]. Hence, unlike traditional algorithms that require an image processing step to extract relevant features followed by a machine learning step to classify the image from the features, CNN provides an one-for-all architecture to solve the same problem. Since then, *deep learning* has been a popular choice for various machine learning and computer vision tasks. Later, many variants of CNN have been proposed to improve the performance [163, 182, 66].

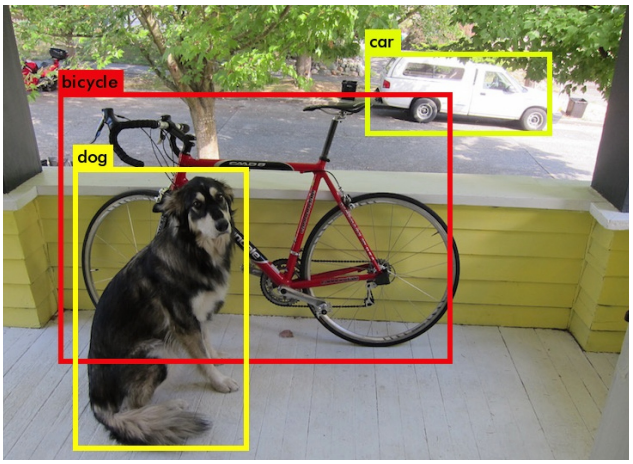


Figure 2: Object detection example. (from [142, 144])

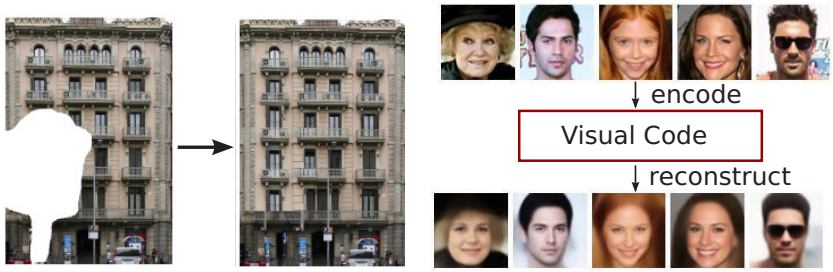
1.3.2 *Object Detection*

Object detection task can be seen as an extension to image classification, which also concerns about the number of known objects in the image and the coordination. In other words, object detection is a more complex task that need to classify and locate all the objects in the scene, as shown in Figure 2. Currently, deep learning is the state-of-the-art method for this task [50, 149, 143].

Figure 3: Image segmentation example. (from [63])

1.3.3 *Image Segmentation*

Image segmentation is a process of partitioning a digital image into multiple segments. A meaningful segmentation typically transforms an image into segments of objects and background by making prediction at every pixel, as shown in Figure 3. Early works used human-engineered features, followed by Conditional Random Field (CRF) [180, 131] or markov random field (MRF) [150], and achieved promising results. With the recent advancement in deep learning, few CNN-based models have been proposed and outperformed previous results [151, 115, 7]. Currently, combining CNN model and CRF achieves state-of-the-art results for image segmentation [22].



(a) Inpainting. (from [29])

(b) Image encoding and reconstruction using Autoencoder. (from [99])

Figure 4: Examples of image synthesis applications.

1.3.4 *Image Synthesis*

Image synthesis is a process of creating new images from some form of image description. It has been a central topic in *computer graphics*, but overlaps with some topics in image processing and computer vision, such as inpainting and image reconstruction as shown in Figure 4. With the advancement in computer vision techniques, many of these techniques have been employed for various image synthesis problems [18]. On the other hand, the intuition behind generative models follow a famous quote from Richard Feynman: "*What I cannot create, I do not understand.*" Hence, generative models are trained by learning to generate true data or synthesize image. One famous example is the Autoencoder [94], which aims to learn the representation for a set of images by setting the target values (e.g. reconstructed images) to the input values (e.g. input images). Recently, Goodfellow et. al. [53] proposed the Generative Adversarial Network (GAN) to train a generative model in a similar way by learning to synthesize random images (or other type of true data). Surprisingly, GAN is able to synthesize *photo-realistic* images. Since then, many works have been published to solve various image synthesis problems using GAN, including text to image synthesis [146, 216], inpainting [137, 211], super-resolution [166], image-to-image translation [81, 219], etc.

1.4 MACHINE LEARNING FOR ARTWORKS

Many works in artworks analysis have been focusing on investigation of certain properties from the paintings by extracting and analysing the relevant features. For instance, the analysis of perspective, scale, and geometry is very important and has a long history in the study of realist art, especially art history of the Renaissance [28, 91]. Meanwhile, estimation of the position and direction of lighting in a tableau are important for some artwork analysis problems [171, 177, 179]. More interestingly, some works explored the analysis of brush strokes and marks that correspond to a particular painter [110, 12, 87]. Furthermore, image synthesis techniques are able to reveal new three-dimensional view from the two-dimensional artwork [148, 176, 165]. Overall, the enhancement from these computer techniques on the study of art should not be overlooked. On the contrary, art scholars are encouraged to exploit the computer techniques to aid their daily tasks for artworks analysis [172, 178, 175, 173, 174].

1.4.1 *Deep Learning for Artworks*

So far, it can be seen that the machine learning techniques used for paintings analysis have been employing human-engineered features. In previous sections, it is revealed that deep learning is currently the

state-of-the-art solution and a popular choice for many computer vision problems. Most importantly, deep learning exhibits some form of *intelligent behaviour* due to its ability to automatically discover key features from images. In fact, study [141] has showed that CNN feature is able to achieve high performance consistently compared to the human-engineered features in their experiments. However, research on deep learning for artworks has been given little attention. Compared to the deep learning works aforementioned, features learning for artworks is more challenging because most of the artworks are structured abstract, opposing the laws of nature. These challenges will be discussed in detail in Chapter 3.2.

Recently, several works related to deep learning for artworks have been published. [155, 88] studied the effects of different features, including CNN and human-engineered features, coupled with different type of metrics for paintings classification task. Meanwhile, some works were focusing on *style transfer* [48, 85, 81, 219, 74], which aims to transfer the artistic style of one image to the target image by matching the Gram matrices of the feature maps. Li et al. [111] demonstrated that such matching is equivalent to minimizing the Maximum Mean Discrepancy (MMD) with second order polynomial kernel. In other words, it is a domain adaptation problem, which statistically matches the feature distribution between the style image and the synthetic image. However, the features used in these works are not learned directly from the artworks, but usually extracted from a pretrained CNN model. Therefore, **features learning of deep model for paintings analysis is still not well studied**.

1.5 GOALS AND OUTLINES

To this end, the main focus of this thesis is to **investigate the capability of deep models in learning features from artworks**. In order to achieve the goal, two types of deep models are investigated for the artworks, i.e. *discriminative* and *generative* approaches. In particular, deep models are trained for artworks *classification* and *synthesis* tasks in the respective approaches. Then, the features learned by the deep models are *visualized* using two different methods, respectively.

In addition, it is desired to deploy the trained models for real-world applications. Hence, this thesis is also interested in **exploring the problems arise when embedding the deep models into real-world devices**. Particularly, a more effective and efficient strategy for *network compression* problem is proposed in order to reduce the network size.

Therefore, this thesis is divided into two parts. The first part will be focusing on the features learning of deep models for artworks, which is covered by two chapters. Meanwhile, the second part of this thesis contains one chapter, which will focus on the network com-

pression problem. In Chapter 2, the history of deep learning and the fundamental of the deep architectures will be briefly reviewed and explained. Next, each task will be described separately.

1.5.1 *Artworks Classification*

Chapter 3 presents the studies on deep learning for *style*, *genre*, and *artist* classification of artworks. First, a slightly modified version of Wikiart dataset [156] is released by splitting the dataset into training and validation sets. This enables better comparative studies in the future compared to the original version, where the dataset was not explicitly split, resulting in different collections of training and validation sets in each experiment. In this work, few AlexNet [97] models with different configurations are trained and compared. Best result is achieved by finetuning an ImageNet pre-trained model and **significantly outperforms the state-of-the-art results** on the dataset. Meanwhile, features learned by the models from the artworks are assessed by **visualizing the averaged neurons' activations**. It is found that the features extracted from the paintings of the same group often vary greatly, in contrast to the features extracted from the images with same object category. However, some categories have extremely similar properties that confuse the deep models, e.g. *analytical cubism* and *synthetic cubism* come from the same root. Such contradiction makes recognition of fine-art paintings an extremely challenging problem. Overall, the deep models are able to learn the structural cues from the artworks. However, it is unclear if the deep models are also able to learn the more abstract cues (e.g. emotion) with the visualization technique used. This leads this thesis to the next work which is summarized in the next section.

1.5.2 *Artworks Synthesis*

In Chapter 4, a new variant of conditional GAN, namely the **ArtGAN** is proposed to synthesize artworks based on the desired *style*, *genre*, or *artist*. In GAN, two networks are trained by competing with each other. First, the *generator* network learns to synthesize images that are difficult to differentiate from real images. Adversarially, the *discriminator* network, typically a CNN model, learns to distinguish the generated images from the real images. With different perspectives, the generator can be seen as a technique that visualizes the learned features by generating photo-realistic images.

Compared to previous experiments in artworks classification, GAN has the following attractive advantages:

1. The generated images visualize the features in a more human-interpretable manner, compared to neurons' activations.

2. The (Art)GAN model can be extended to other applications, for instance image stylization.

In this work, various ways to improve the quality of the generated images are explored. The contributions are three-fold: 1) The labels information are leveraged to compute the loss functions; 2) Autoencoder is incorporated as a complementary information for the loss functions; 3) A novel **magnified learning** is proposed to improve the correlations between neighbouring pixels. Empirically, the experiments show that the proposed ArtGAN **outperforms the state-of-the-art results in terms of Inception score**. Furthermore, the experiments show that ArtGAN is **capable of drawing high quality and realistic artworks**, as well as natural images. Interestingly, the generated artworks show that the network is **able to learn certain abstract characteristics through visual cues**. Overall, these observations show that ArtGAN **has learned rich visual representations** from the artworks.

1.5.3 *Deep Compression Model*

Despite the promising results, it is infeasible to embed the trained deep models into resource-limited hardware, e.g. mobile device, due to its extremely high memory requirement. To address this problem, it is desired to reduce the memory requirement of the deep models with minimum compensation on the performances. In Chapter 5, a novel **one-shot deep compression** method based on the fuzzy quantity space is proposed to remove the redundant CNN weights. In this work, previously trained deep models are employed for artworks classification, in addition to several other popular deep architectures used for other datasets to test the proposed method. The experiments show that the proposed approach is able to **compress the deep models up to 14$\times$ with minimal loss of classification accuracy**.

2 DEEP LEARNING

2.1 INTRODUCTION

Deep learning is an active area of research in machine learning that emphasizes the *hierarchical learning of data representations*. More specifically, it is powered by the deep neural network, which was inspired by the beautiful biological process of brains. Deep learning currently provides the best solution to many problems in computer vision, speech recognition, natural language processing, and so on, such that the performances are comparable to human experts. In this chapter, the history and evolution of deep learning will be briefly explored, focusing on computer vision applications. Then, this thesis proceeds to the technical explanations of the deep models used.

2.2 BRIEF HISTORY OF DEEP LEARNING

According to the survey [159], the earliest deep-learning-like algorithms can be dated back to 1965, where Ivakhnenko and Lapa [82] used thin but deep models with polynomial activation functions. In 1986, Rina Dechter [32] introduced the expression *Deep Learning* to machine learning community. Later, Igor Aizenberg et al. [3] introduced the same term to Artificial Neural Networks in 2000.

Among the deep models, Convolutional Neural Network (CNN) has been the most successful model in solving many computer vision problems. The history of CNN can be traced back to 1982 when the Neocognitron was introduced by Fukushima [47]. Neocognitron was partially trained by *unsupervised* learning with human-merge features. Contrastively, LeCun et al. [100] applied *supervised* backpropagation to a Neocognitron-like architecture to recognize handwritten ZIP codes on mail. Meanwhile, Weng et al. [202] proposed Cresceptron for 3D object recognition from images of cluttered scenes by *automatically learning the features* at each layer through *convolution kernel*. The *max pooling*, that is now often adopted by CNN models, was first used in Cresceptron to provide some form of translation invariance. In 1998, LeNet-5 [101], a 7-layers convolutional network was published to classify digits. The implementation was a success but required long time to train. Fortunately, the training time can be reduced significantly with the recent advancement in GPU technology. The first successful implementation of CNN using GPU was AlexNet [97]. It achieved state-of-the-art results on ImageNet dataset [153], which is the largest object recognition dataset available

to date. Since then, CNN has been the mainstream solution for image recognition [163, 182, 66] and many other image-related problems [149, 211, 85, 81].

Unlike human-engineered features such as contour, HOG [31], and SIFT [116], CNN was viewed as a black box because there was little insight into the internal operation of CNN. One simpler way to access the black box is by visualizing the activations and filters learned [1]. However, these visualizations can be hard to interpret. Zeiler and Fergus [214] proposed a better visualization technique using Deconvolution Network [215], showing that generic features are extracted from CNN. Later, Springenberg et al. [168] proposed a new variant of deconvolution approach to visualize features learned by CNN using a guided backpropagation. Nonetheless, these techniques can only visualize one neuron at a time, which is inefficient and sometimes difficult to interpret. Recently, Dosovitskiy and Brox [40] proposed a new visualization idea by training a *generator* network that inverts the features by reconstructing image that best represents certain features combination. They showed that the reconstructed image is more human-interpretable and can visualize all features through one reconstructed image. Furthermore, the method can also be implemented on other human-engineered features. Later, they introduced an improved version [39] by training the generator network using several complementary loss functions to generate more natural images. They revealed that CNN is able to learn important visual representation, but some details are not preserved.

So far, one can notice that most of the aforementioned deep models are *discriminative*. Deep discriminative models usually require abundance *labelled* data to perform well. However, such data is hard to obtain. On the other hand, with the abundance *unlabelled* data, several *generative* models have been proposed to learn the joint probability distribution. Interestingly, generative models are usually trained by learning to simulate the observed distribution (or learn to draw images), offering a much richer representation. Typically, deep generative models are trained by maximizing the log likelihood, such as Deep Boltzmann Machine [154], its variants [69, 68], and Variational Autoencoder [94, 119]. However, these models generally have intractable likelihood and require numerous approximations. Denoising Autoencoders (DAE) [10] was introduced to overcome the intractable problem, but the reconstructed images are generally blurred. Recently, Goodfellow et al. [53] proposed Generative Adversarial Network (GAN) to estimate the generative model via an adversarial process. They demonstrated that GAN is able to generate more photo-realistic images, which infers that GAN is able to learn richer visual features than other generative models. Later, Salimans et al. [157] and Radford et al. [139] introduced some techniques to stabilize GAN during training. Meanwhile, Arjovsky and Bottou [5] provided an anal-

ysis of the convergence properties in GAN and proposed WGAN [6] for a more stable training of GAN by leveraging Wasserstein distance in the loss function. However, WGAN can still generate low-quality samples or fail to converge in some settings, and requires long time to train. Although an improved version was proposed in [57] and achieved better image quality, it is still extremely computationally expensive to train. Other efforts to stabilize GAN include Energy-based GAN (EBGAN) [217] and Denoising Feature Matching (DFM) [199]. Although WGAN-GP ResNet [57] is the best reported result so far, it is expensive to train and only outperforms DFM by a small margin, in addition to the much larger network size used compared to other GAN variants.

On the other hand, Conditional GAN (CondGAN) [123, 49] enables the control of the attributes in the images, but only demonstrated on faces and digits datasets. Meanwhile, a sequential GAN called LAPGAN [35] was proposed using the Laplacian pyramid framework to generate plausible looking scenes at various image size. Other recursive models built on GAN were demonstrated in [55, 79, 98, 207] and showed some successes with generating images. While, DRAW [55] mimics the drawing process of a human. It is depicted as a sequential model with attention mechanism to draw image recursively. However, it faces challenges when scaling up to large complex images. Other autoregressive approaches including PixelRNN [134] and PixelCNN [191, 158] are able to synthesize images with decent quality, but are computationally expensive to train[1].

2.3 CONVOLUTIONAL NEURAL NETWORK

This section describes the basic structure of a CNN model using mathematical notations. In general, a CNN model $\mathcal{F}$ takes a 4-dimensional matrix of N_b input images $X \in \mathbb{R}^{N_b \times H \times W \times C}$ and outputs a classification vector $\mathbf{c} \in \mathbb{R}^{N_b \times N_c}$, where H, W, and C are height, width, and number of channels of an image, respectively. While, N_c is the number of classes in the dataset. Formally, the operation is denoted as: $\mathcal{F} : X \rightarrow \mathbf{c}$. A CNN model is usually constructed by several *convolution* layers, followed by few *fully-connected* layers. For simplicity, LeNet-5 [101] will be used as an example for explanations.

LeNet-5 is a small convolutional network with two convolution layers (conv1 and conv2) and two fully-connected layers (fc1 and fc2) as shown in Figure 5. The network accepts $28 \times 28 \times 1$ gray-scaled images as input, though it can be extended to RGB images easily by changing the input size to $28 \times 28 \times 3$. In this architecture, the number of feature maps for conv1 and conv2 are set to 20 and 50, respectively.

1 They reported that PixelCNN++ requires approximately 5 days to converge to the reported results using 8 Maxwell TITAN X GPUs in github: https://github.com/openai/pixel-cnn.

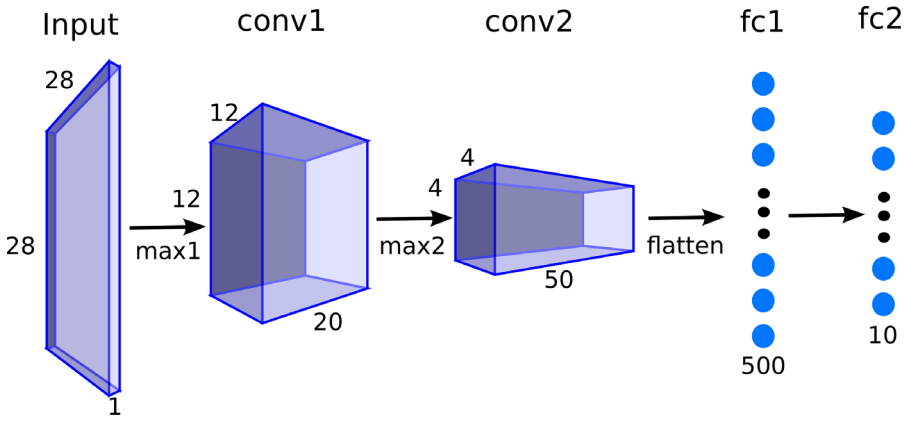


Figure 5: An illustration of LeNet [101].

All convolution layers are followed by an *activation function*, then a *max pooling* layer for downsampling. Meanwhile, the number of neurons in fc1 is set to 500. Similar to the convolution layer, an activation function is computed after fc1. On the other hand, the number of neurons in the last layer fc2 is set to 10, following the number of classes available in the MNIST dataset [104]. Unlike other layers which usually use ReLU [125] as the activation function, the last layer (i.e. fc2) in CNN is commonly followed by a softmax function to transform the outputs to a probability distribution. Next, the key components of CNN will be briefly described. Note that some components are not used in LeNet-5 but is commonly used in other CNN models.

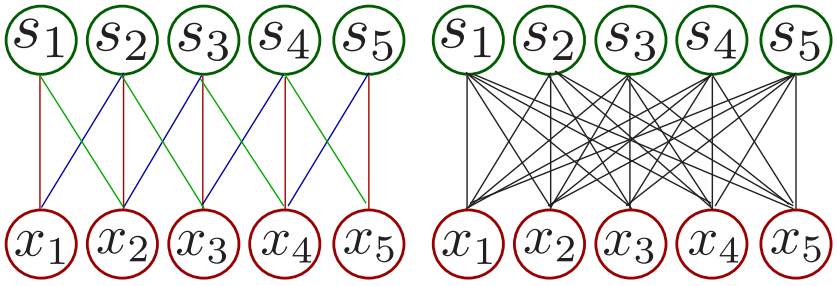
2.3.1 *Convolution and Pooling*

In computer vision, CNN is always favoured over the traditional neural network. This is because *convolution* operation has less number of weights to be learned as shown in Figure 6, resulting in faster computation speed and less memory requirement compared to *fully-connected* linear operation. Furthermore, natural images are stationary [46, 190], which means that the statistics of one part of the image are the same as any other part. For instance, the probability of a ball appears in the top left corner of an image is the same as the probability of it appears in the bottom right corner. Convolution operation is able to handle this property effectively.

Formally, let $\mathcal{X}$ be the input and $\mathcal{Y}$ be its output. The convolution operation "$*$" using the kernel or filter (can be used interchangeably) $\mathcal{K}$ is written as,

$$\mathcal{Y} = \mathcal{X} * \mathcal{K} \tag{1}$$

Figure 7 visualizes a simple convolution operation example. In short, the values in $\mathcal{Y}$ are calculated by applying sliding window that computes the linear combination of the elements in each rounded square



(a) convolution operation (b) fully-connected linear operation

Figure 6: Illustrations of different operations. Each connection requires one trainable weight. Meanwhile, the connections in convolution operation with the same color share the same weight. Therefore, it is very clear that convolution operation requires less number of weights compared to fully-connected linear operation.

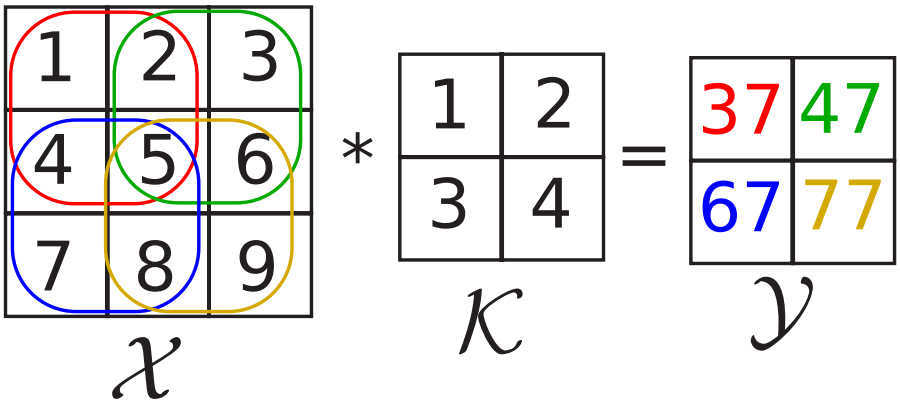
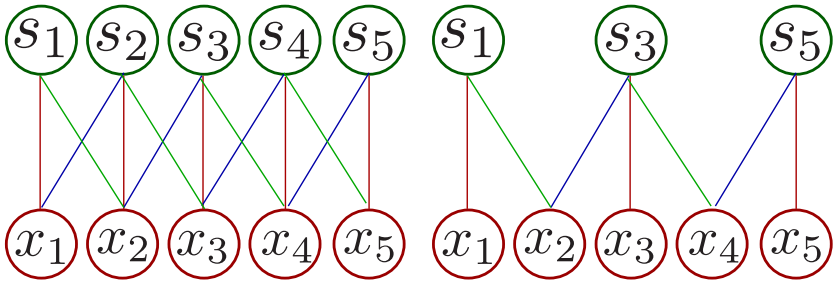


Figure 7: A simple example for convolution operation. The color of the rounded square in $\mathcal{X}$ is correspond to the color of the values in $\mathcal{Y}$.

in $\mathcal{X}$ and all elements of $\mathcal{K}$. In this example, the operation is performed with 2×2 *filter size* and *stride* 1. Commonly, the size of the stride is set to > 1 during downsampling, such that the size of $\mathcal{Y}$ will be reduced by a factor of the stride value. For instance, Figure 8 visualizes the differences between stride 1 and stride 2.

It is also common to use a pooling operation instead of a strided convolution operation for downsampling. Two most commonly used pooling operations are *max pooling* and *average pooling*, which are visualized in Figure 9. Boureau et al. [17] conducted a theoretical analysis on pooling and showed that pooling seems to contribute significantly to improve classification accuracy in object classification task. Most importantly, pooling is used to summarize the features in a region, offering some degree of invariance to input translations. However, the invariant representation of the pooled features offers an incomplete information on the data because the detailed representation of the lower-level features are ignored in the pooling procedure. Re-



(a) Stride 1 (b) Stride 2

Figure 8: Illustrations of using different stride value.

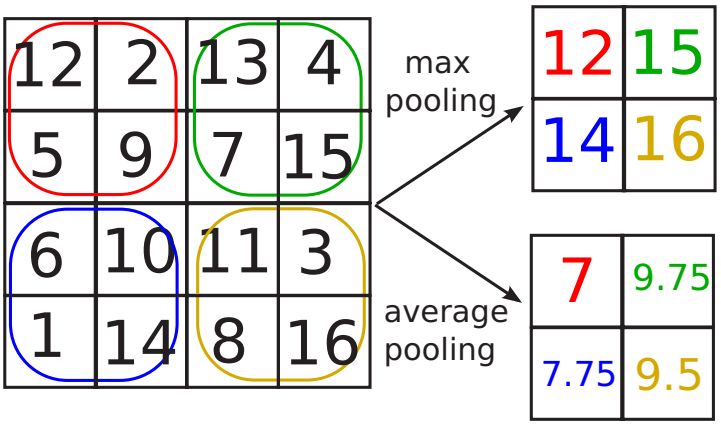


Figure 9: An example for pooling operation. Similar to previous example, the configurations are 2×2 filter size and stride 2.

cently, Springenberg et al. [168] showed that a CNN is able to achieve comparable performance by replacing pooling with strided convolution. This suggests that CNN is able to learn the invariance using the strided convolution. For more details about convolution and pooling, readers are encouraged to read [43].

2.3.2 *Activation Function*

Activation function (or transfer function) defines the output of a neuron given an input through a non-linear function. Crucially, any multiple perceptron could be reduced to a single-layer network when linear activation function is used [203]. Hence, it is important to implement a non-linear activation function to learn the non-linear relationship in order to gain the advantages of a multilayer network. Sigmoid function [58] and tanh function have been the common choice for the activation function, but faced many failures due to the vanishing gradient problem [70]. Fortunately, ReLU [125] was proposed in 2010 to solve this problem. Since then, artificial neural network and deep learning have many successes in solving various problems, as reviewed in previous sections. Later, *leaky ReLU (lReLU)* [117] and

Parametric ReLU (PReLU) [65] are introduced to fix the zero gradient problem occurred in the traditional ReLU. The formula of lReLU is given as,

$$relu(x) = \begin{cases} x & \text{if } x > 0 \\ a_r x & \text{otherwise} \end{cases} \tag{2}$$

where a_r is often set to 0.01. When $a_r = 0$, it becomes the traditional ReLU function. Meanwhile, a_r is a trainable variable in PReLU.

2.3.3 *Normalization*

While normalization layer is not employed in LeNet-5, it has been found to be an important component in CNN to improve the performance. Normalization layer was inspired by a concept in neurobiology called "lateral inhibition". In a nutshell, it is the capacity of an excited neuron to subdue the activity of its neighbours. In other words, normalization layer aims to form significant peaks, creating a contrast in an area, as shown in Figure 10.

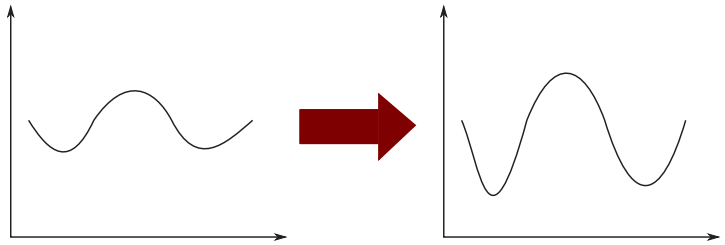


Figure 10: Illustration of forming significant peaks.

For this purpose, Krizhevsky et al. [97] proposed the *Local Response Normalization (LRN)*. Let $a^i_{p,q}$ be the activity of a neuron computed by kernel i at position (p, q), the response-normalize activity $b^i_{p,q}$ is given by the expression,

$$b^i_{p,q} = a^i_{p,q} / \left(k_{lrn} + \alpha_{lrn} \sum_{j=max(0,i-n_{lrn}/2)}^{min(N_{lrn}-1,i+n_{lrn}/2)} (a^j_{p,q})^2 \right)^{\beta_{lrn}} \tag{3}$$

where the sum runs over n_{lrn} "adjacent" kernel maps at the same spatial position, and N_{lrn} is the total number of kernels in the layer. Typically, the configurations are as follow: $k_{lrn} = 2$, $n_{lrn} = 5$, $\alpha_{lrn} = 10^{-4}$, and $\beta_{lrn} = 0.75$. The authors claimed that LRN improves the classification accuracy $1 \sim 2\%$ in their experiments.

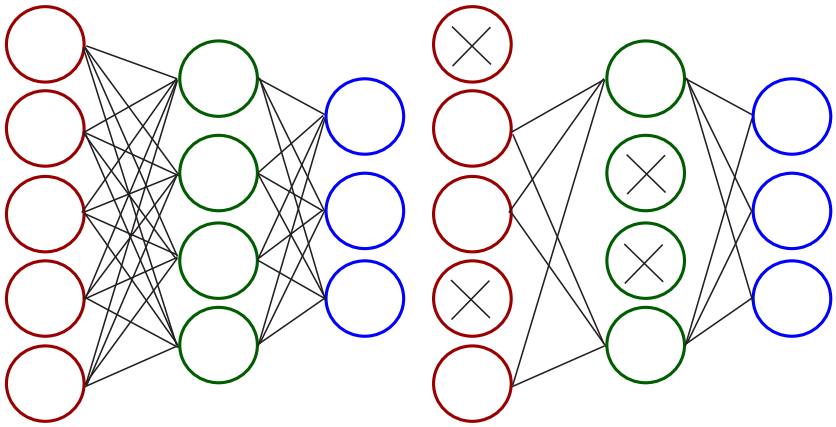
Despite the excellent performance of LRN, it has been replaced by *batch normalization (BN)* [80] in recent works because BN performs better and helps the network to learn faster. The authors of BN also showed that combining both LRN and BN does not gain any benefit. For a layer with d-dimensional input $a = \{a^1, \ldots, a^d\}$, the formula of BN is given as,

$$\hat{a}^i = \frac{a^i - \mathbb{E}[a^i]}{\sqrt{\text{Var}[a^i]}} \tag{4}$$

where $\mathbb{E}[\cdot]$ and $\text{Var}[\cdot]$ are the expectation and variance over the training data. The authors of BN also argued that BN may change the representation of the affected layer. Hence, a pair of parameters γ_{bn} and β_{bn}, which scale and shift the normalized value are introduced,

$$b^i = \gamma_{bn}\hat{a}^i + \beta_{bn} \tag{5}$$

These parameters are learned along the other parameters in the model during training.



(a) Before dropout (b) After dropout

Figure 11: Illustrations of dropout. The inactive neurons (with "×") are set to zero.

2.3.4 *Dropout*

Dropout was introduced by Srivastava et al. [170] as a regularization technique to reduce overfitting problem in neural network. The main idea behind dropout is to prevent complex co-adaptations on training data. During training, dropout is implemented by inactivate some neurons with certain probability $p_d = (0, 1)$, as shown in Figure 11. At test time, all neurons are kept active but all weights are set to $p_d\mathbf{w}$, where $\mathbf{w}$ is the original trained weight. Wan et al. [195] generalized dropout using dropconnect but the performances are generally comparable. More theoretical studies can be found in [8, 193, 67].

2.3.5 *Training*

Generally, a neural network is trained using backpropagation [100], where the derivatives are calculated through chain rule. With large scale dataset available nowadays, it is impossible to fit the whole dataset into the computer's memory. Hence, training is usually done by splitting the dataset into multiple subsets called minibatches and

each minibatch is used iteratively during training. Let $\mathbf{X}$ be the whole dataset, minibatch X_i is the subset of $\mathbf{X}$, such that $\mathbf{X} = \{X_1, \ldots, X_m\}$, where m is the number of minibatches. Given L as the labels of $\mathbf{X}$, where $\{l_i\} \in L$ is the labels of X_i, and $\mathcal{G}$ as the loss function, the pseudocode for training CNN is described as,

Algorithm 1: Pseudocode for training CNN

```
Require: {X_i} ∈ X and {l_i} ∈ L
  while condition not met do
    for i ∈ [1, ..., m] do
      c = F(X_i) # feedforward
      L = G(c, l_i) # calculate loss function
      v = V(w, ∂L/∂w) # backpropagation and calculate velocity
      w = w + v # update w
    end for
  end while
```

where $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ is the derivative of $\mathcal{L}$ w.r.t. the weights $\mathbf{w}$ in the network. While, $\mathcal{V}$ is the weight update rule, typically defined with stochastic gradient descent (SGD) and weight decay:

$$v = \mathcal{V}(\mathbf{w}, \frac{\partial \mathcal{L}}{\partial \mathbf{w}}) = -\epsilon \cdot (\frac{\partial \mathcal{L}}{\partial \mathbf{w}} + \lambda \mathbf{w}) \tag{6}$$

where λ and ϵ are the weight decay rate and learning rate. In AlexNet [97], SGD is used with momentum in order to allow SGD to escape from local optima:

$$v_{t+1} = \gamma \cdot v_t - \epsilon \cdot (\frac{\partial \mathcal{L}}{\partial \mathbf{w}} + \lambda \mathbf{w}) \tag{7}$$

where the velocity at iteration $t+1$ is affected by the velocity at previous iteration t. γ is the momentum, which is usually set to 0.9. More sophisticated SGD variants were proposed in order to improve SGD, such as AdaGrad [41], AdaDelta [213], RMSProp [189], and Adam [93]. Although none of them can claim to outperform other variants in all optimization problems, SGD with momentum (equation 7), RMSProp, and Adam are the most commonly used update rules in recent deep learning works.

Meanwhile, multinomial logistic loss of softmax has been the popular choice for loss function among deep learning models, including AlexNet. The loss function $\mathcal{G}$ is then derived as,

$$\mathcal{G}(\mathbf{c}, l) = -\log\left(\frac{e^{\mathbf{c}_l}}{\sum_{j=1}^{N_c} e^{\mathbf{c}_l}}\right) \tag{8}$$

The derivative of this loss function is minimized during training. Other loss functions are discussed in [90, 152, 121, 83]. Readers who are interested in Energy-based model and loss function may refer to [103].

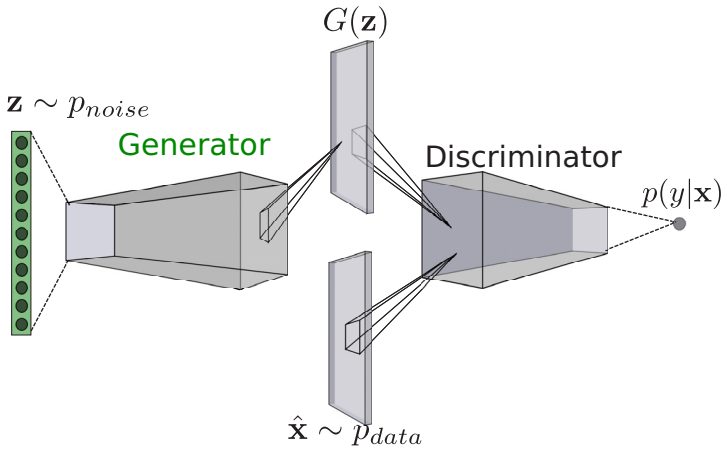


Figure 12: Overview of GAN.

2.4 GENERATIVE ADVERSARIAL NETWORK

Generative Adversarial Networks (GAN) [53] contain two networks (as shown in Figure 12) that are trained by competing with each other. The Generator G aims to learn the true data distribution p_{data} by generating images (or data) $G(\mathbf{z})$ that are difficult to differentiate from real images $\hat{\mathbf{x}} \sim p_{data}$. Traditionally, G generates images from some noise vectors $\mathbf{z} \sim p_{noise}$ that are sampled from a distribution p_{noise} (e.g. uniform distribution). On the other hand, the Discriminator D is trained to distinguish the images generated by G from the real images. Formally, the operation of the generator is written as $G : \mathbf{z} \rightarrow G(\mathbf{z})$. Meanwhile, $D : \mathbf{x} \rightarrow y$ represents the process in the discriminator, where $\mathbf{x}$ is an input image from either $\hat{\mathbf{x}}$ or $G(\mathbf{z})$. Let $p(y|\mathbf{x})$ be the probability distribution function for the binary adversarial prediction, such that $\mathbf{x}$ is from p_{data} when y approaches 1, while $\mathbf{x}$ is from G when y approaches 0.

Overall, the training procedure is a two-player minimax game with the following objective function,

$$\min_G \max_D \mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}}[\log D(\hat{\mathbf{x}})] + \mathbb{E}_{\mathbf{z} \sim p_{noise}}[\log(1 - D(G(\mathbf{z})))] \tag{9}$$

However, this cost function does not perform well when updating G in practice [52]. This is because in the minimax game, D minimizes a cross-entropy while G maximizes the same cross-entropy. When D successfully rejects samples from G with high confidence, the gradient in G will vanish. To overcome this problem, a simple yet effective solution is simply flip the sign of the true label of $G(\mathbf{z})$, such that G is trained to make y approaches 1. Therefore, D is trained by minimizing the following cost function,

$$\mathcal{L}_D = -\mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}}[\log D(\hat{\mathbf{x}})] - \mathbb{E}_{\mathbf{z} \sim p_{noise}}[\log(1 - D(G(\mathbf{z})))] \tag{10}$$

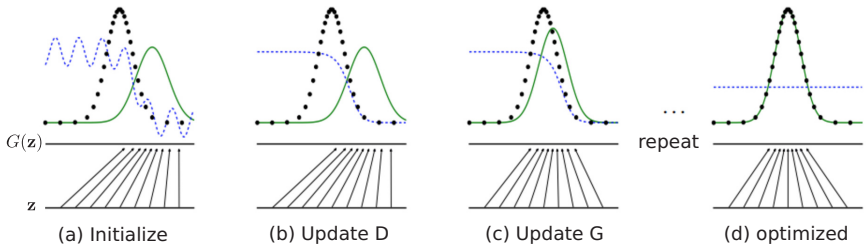


Figure 13: Overview of GAN training. Black dotted lines are true distribution p_{data}. Blue dashed lines are discriminative distribution p_D. Green solid lines are generative distribution p_G. The upward arrows show mapping of $\mathbf{z} \to G(\mathbf{z})$.

Algorithm 2: Pseudocode for training GAN

Require: $\{X_i\} \in \mathbf{X}$
  while condition not met **do**
    for $i \in [1, \ldots, m]$ **do**
      Sample $Z \sim p_{noise}$
      $D_r = D(X_i)$
      $D_f = D(G(Z))$
      Update D by minimizing: $-\mathbb{E}[\log D_r] - \mathbb{E}[\log(1 - D_f)]$
      $D_f = D(G(Z))$ # *Note that the updated D is used here.*
      Update G by minimizing: $-\mathbb{E}[\log D_f]$
    end for
  end while

and G is trained by minimizing the following modified cost function,

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z} \sim p_{noise}}[\log(D(G(\mathbf{z})))] \tag{11}$$

The training process is then consist of simultaneous SGD, where D and G are trained sequentially. Figure 13 visualizes the training process. Let $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_{N_b}\} \in \mathbb{R}^{N_b \times N_z}$, where N_z is the number of dimensions of one sample $\mathbf{z} \in \mathbb{R}^{N_z}$. The pseudocode for the training process is then described in Algorithm 2.

2.4.1 *Deep Convolutional GAN*

Generally, Deep Convolutional GAN (DCGAN) is a GAN model with part of the networks replaced by layers of convolution or deconvolution operations to improve the synthesis and representations learning of image. Although the term "DCGAN" was first introduced in [139], this version of GAN has been implemented by Goodfellow et al. [53] to synthesize CIFAR-10 images when GAN was proposed.

Overall, DCGAN has a similar architecture as shown in Figure 12. Typically, the discriminator is a CNN model described in previous section. On the other hand, the generator aims to reverse the CNN

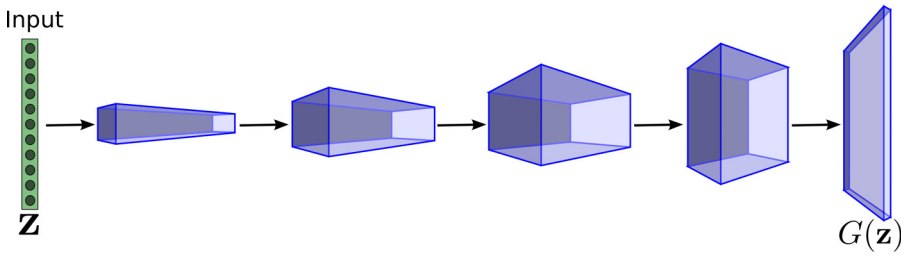


Figure 14: Deconvolution Network.

operations in order to generate realistic images. In most of the implementations [53, 49, 139, 205], the generators employed the Deconvolution network [215] as shown in Figure 14. Generally, it consists of several strided deconvolution layers that are the inverse operation of strided convolution, upsampling the input of the operation by a factor of the strided size. Figure 15 visualizes the difference between the convolution and deconvolution operations. Note that the deconvolution layers used in these networks are not the same as the traditional deconvolution operation described in [200]. To be accurate, the deconvolution layer is called "Transposed convolution" by some researchers [187]. Detailed technical explanations of deconvolution (and convolution) can be read on [187].

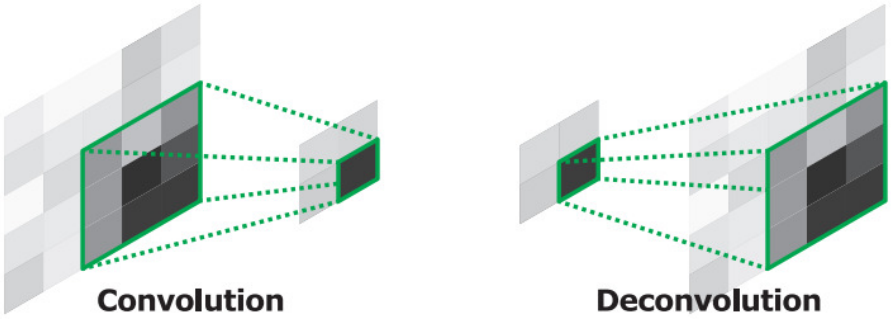


Figure 15: Comparison between convolution and deconvolution. (reproduced from [129])

Meanwhile, Odena et al. [132] blamed that deconvolution has "uneven overlap", putting more of the metaphorical paint in some places than others [49], resulting in "mosaic" pattern in the image. They suggested that a simple alternative is to separate the sequential operations in the deconvolution layer. This can be done by first performing an upsampling to higher resolution using interpolation, followed by a convolution operation of stride 1. They conducted some experiments and showed that using Nearest Neighbour (NN) upsampling instead of strided deconvolution helps avoiding the checkerboard artifacts problem. Similar technique has been used and worked well in image super-resolution [38].

2.4.2 *Convergent Property and Global Optimality*

Many efforts have been done in order to understand the theory behind GAN, mainly revolving around the convergent property of GAN. Intuitively, GAN is a two-player minimax game as aforementioned. Hence, the authors [53] explained the theory behind GAN via game theory using Nash equilibrium. In theory, Nash equilibrium (or global optimality) is reached when the generator matches data distribution and discriminator is unable to distinguish generated samples from real samples [52]. Hence, for a given generator G, the optimal discriminator is:

$$D_G^* = \frac{p_{data}}{p_{data} + p_G} \tag{12}$$

Then, by substituting this into the GAN formula, Goodfellow et al. [53] showed that GAN objective turns out to be equivalent to the Jensen-Shannon divergence (JSD) between p_{data} and p_G. Equation 9 can then be reformulated as,

$$-\log(4) + 2JSD(p_{data}||p_G) \tag{13}$$

In practice, such equilibrium is hard to achieve and GAN only represents a limited family of true distribution via G. However, its excellent performance in practice suggests that GAN is still a reasonable model to use despite its lack of theoretical guarantees. Later, Nowozin et al. [130] further the studies and showed that GAN is a special case of a general variational divergence estimation approach called f-divergence.

In a recent work, Arjovsky and Bottou [5] provided more theoretical investigations on the convergent property of GAN. They showed that it is unlikely that the model manifold and the true distribution's support have a non-negligible intersection. In this case, Kullback-Leibler distance (or the JS distance) cannot be defined, which contributes to the instability of GAN during training. To overcome this problem, they proposed WGAN [6] by reformulating the loss function using Wasserstein distance with Lipschitz constraint enforced on the weights to ensure convergence. However, this version of WGAN can still be unstable. A better version of WGAN is then introduced in [57] with better strategy for the weight constraint. The improved WGAN showed better stability but is slow to train due to the computationally expensive gradient penalty.

In a more recent work, Li et al. [109] proposed the MMD GAN that is trained by using the kernel maximum mean discrepancy (MMD) [56] in their loss function. Theoretically, they showed that training GAN via MMD is continuous and differentiable, which guarantee the model to learn and converge. They connected their work to WGAN by claiming that WGAN can be treated as first-order moment matching,

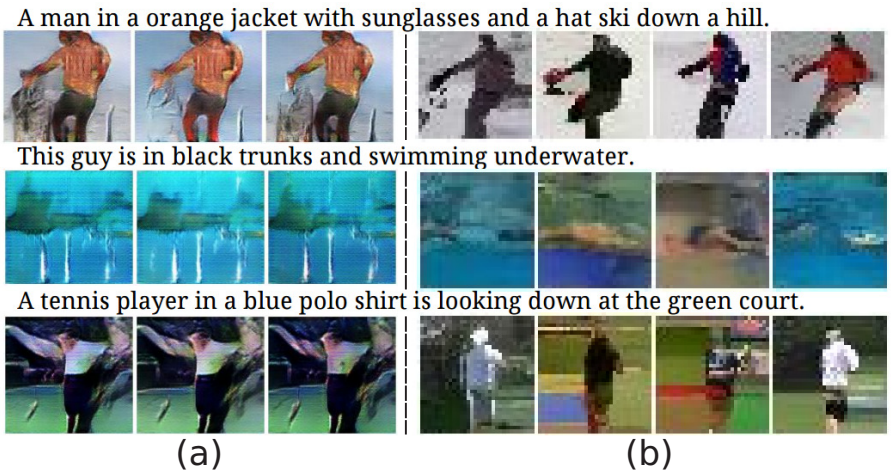


Figure 16: Modified from [147]. (a) Examples of mode collapse experienced in [145]. (b) Extended PixelCNN [147] is able to generate diverse samples from same text.

while MMD GAN aims to match infinite order of moments. Empirically, they showed that MMD GAN is able to achieve good performance, guaranteed convergence, and not as computationally expensive as WGAN. However, Inception scores achieved by MMD GAN does not outperform the state-of-the-art results.

2.4.3 *Mode Collapse*

Mode collapse [52, 145] is another common problem faced when training the GAN models. In general, a model is experiencing mode collapse when the generator in the model synthesizes many images with similar or same texture, as shown in Figure 16. As discussed in [157, 122], this is because the generator has the tendency to learn to synthesize data with high probability in the discriminator. As the training progresses, the generator will eventually maps more data to the highest probability, resulting in producing same data. Salimans et al. [157] then proposed the minibatch discrimination to overcome this problem by allowing the discriminator to compare an example to a minibatch of generated samples and a minibatch of real samples. Experimentally, minibatch discrimination does help in reducing mode collapse problem. Reed et al. [147] showed that their proposed gated conditional PixelCNN is able to generate diverse samples without more explanations about this issue. Meanwhile, Metz et al. [122] proposed Unrolled GAN to address the same problem by unrolling optimization of the discriminator and showed promising results. However, the main drawback of this approach is that the computational cost scales with the number of unrolling steps. Hence, this approach has not yet been scaled up to other problems, e.g. ImageNet.

2.4.4 *Tips and tricks to train GAN*

Some useful tips and tricks were introduced in [157] and listed in https://github.com/soumith/ganhacks to improve the stability when training GAN. Nonetheless, they do not always work well and require trial and error. Few tips and tricks that are found to perform more consistently in this thesis are:

1. Normalize the inputs to $(-1, 1)$ in the discriminator. Meanwhile, use tanh as the activation function for the last layer of generator.
2. Use the loss function trick as described in equation 10 and equation 11.
3. Batch Normalization is still useful in both discriminator and generator.
4. Implementing dropout in the discriminator helps but it is advised to avoid using dropout in the generator.
5. Use leaky ReLU in both discriminator and generator.
6. Use strided convolution and avoid max pooling in the discriminator. Meanwhile, use Nearest Neighbour upsampling instead of strided deconvolution in the generator.
7. One update step of one network for each update step of another network seems to work the best in practice, as suggested in [52].
8. Use labels whenever it is available, which mean a semi-supervised GAN will always works better than the unsupervised counterpart.
9. Apply instance noise [166] in the discriminator.

2.4.5 *Conditional Image Synthesis*

While unconditional image synthesis is an important research area, many practical applications are conditioned on prior information. This prior information comes in many forms, for instance a distorted image for inpainting [134, 137]; natural image for super-resolution [166, 106] or style transfer [48, 81, 197, 74, 111]; text codes for text to image translation [146, 216]; and so on. This work is particularly interested in class-conditioned image generation, as the primary aim is to investigate how a deep model learns the representations of the styles, genres, and artists from the artworks.

An earlier work that employed conditional setting in GAN was the Conditional GAN (CondGAN) [123] by feeding the labels or modes to the generator and discriminator. However, such setting was only demonstrated on less complex images i.e. MNIST and faces [49]. While

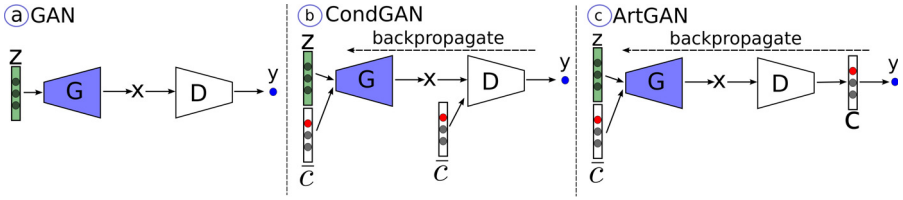


Figure 17: Comparisons between GAN [53] and two conditional GAN variants (CondGAN [123] and ArtGAN [183]).

this website[2] unofficially demonstrated generating images on CIFAR-10 using CondGAN, the objects in the generated images are hardly recognizable. This can be expected because the labels were not fully utilized, as there is no error information backpropagated from the labels. A similar work was introduced in InfoGAN [205] with the discriminator replaced by a multiclass classifier. It is shown that the InfoGAN is able to learn disentangled representations in an unsupervised manner but the meaning of the representations are uncontrollable during training. In addition, their work only demonstrated on the less complex images, i.e. digits and faces.

Figure 17 visualizes the difference between traditional GAN and conditional GAN. Notice that the labels $\bar{c}$ are inputs for both generator and discriminator in CondGAN. During training, the labels are not backpropagated to take advantage of the information provided. On the other hand, the ArtGAN proposed in this thesis leverages the label information by calculating the loss function from the prediction c and the labels. Such architecture performs better than CondGAN. Similar design was introduced in [167, 157], but with unconditional setting. Meanwhile, similar idea was proposed by AC-GAN [133] in the same period when ArtGAN was proposed. However, a key difference is that generated images are not used to compute classification loss of the discriminator in ArtGAN. Contrastively, AC-GAN uses the generated images as positive samples to calculate classification loss of the discriminator. It is counter-intuitive to use the generated images as positive samples for the discriminator.

Built on the Deep Generator Network (DGN) [126], Plug and Play Generative Networks (PPGN) [127] is able to produce high quality image at high resolution. It allows different generators and condition networks to be hacked together without having to re-train the generators. Nonetheless, they showed that adversarial training is crucial to obtain high quality image. However, PPGN differs to the other generative models discussed, where image is generated in one-shot from the latent code in the traditional generative models. In PPGN, image is generated by optimizing the latent code to produce image that highly activate target neuron in the condition network. The sampling procedure is formalized as an approximate Langevin Markov

2 http://soumith.ch/eyescream/

chain Monte Carlo sampler to ensure diversity. Like other sequential approaches, such gradient-based recursive approach may cause unwanted overhead when deployed in some real-world applications, e.g. mobile devices.

Part II

UNDERSTANDING FEATURES LEARNING OF DEEP MODELS FOR ARTWORKS

This part aims to understand what kind of features are learned by the deep models from the artworks. Two types of deep models are trained for artworks *classification* and *synthesis* problems, respectively. In the first task, the proposed deep discriminative models achieve state-of-the-art results for classification. Then, the features learnt are visualized via neurons' activations. In the second task, a new deep generative model is proposed and trained to learn richer visual representations. The learnt features can then be visualized via the trained generator that can synthesize high resolution photo-realistic images.

3 ARTWORKS CLASSIFICATION

Figure 18: Samples of famous artworks in the Wikiart dataset [156].

3.1 INTRODUCTION

In recent years, vast digital collections have been made available across the Internet and museums due to the rapid advancement of digital acquisition of fine-art paintings. With such massive digital artwork collections, automated paintings classification in terms of style, artist, etc. are now possible and desired. In addition, representations learning of fine-art paintings is also an important topic for assisting the curators in their daily work routine, such as paintings analysis, archiving artworks, etc. However, most of the research have been focusing on recognition of more structural images [149, 107, 64, 164, 71]. On the other hand, little attention was given to the classification of fine-art paintings.

This work presents a study on large-scale *style*, *genre*, and *artist* classification of fine-art paintings by training CNN models on the Wikiart paintings dataset [155]. The main objective in this work is to investigate the capability of the CNN in learning the representations of the fine-art paintings. As a summary, the contributions of this work are as follows:

1. An ImageNet pre-trained AlexNet [97] model is finetuned on the Wikiart dataset for artworks classification task. It has been proven that transferring the well-learned knowledge from one source to a target domain improves the deep model accuracy significantly [37, 212, 135, 141]. However, the features learned by the pre-trained model may not suit the target dataset. Hence, finetuning is essential for learning a better model for the target domain while using the pre-trained weights as a better initialization. In the experiments, the trained model achieves state-of-the-art results (68% accuracy), in comparison to [155] (56% accuracy) in the Wikiart dataset.

2. The features of the artworks extracted by the trained AlexNet model are visualized by picturing the *overall* responses of neurons. The visualizations show that the trained deep models are able to extract structural visual cues from the artworks. However, it is unclear if deep models are able to learn the more abstract cues from the artworks. Meanwhile, those features extracted from the paintings of the same group could vary greatly, in contrast to object or face recognition where features extracted from the same class are somehow very similar. At the same time, the differences between the categories can be small and abstract. Therefore, it is extremely challenging to recognize the categories of an artwork.

3. A modified Wikiart dataset is released to provide a better platform for future comparative studies.

3.2 CHALLENGES IN ARTWORKS CLASSIFICATION

3.2.1 *Large-scale Artworks Dataset*

Most of the studies have been revolving around datasets with more structural categories. Hence, one of the challenges in artworks analysis is the availability of a decent fine-art paintings dataset for evaluation. As evidence, in the object recognition, there are PASCAL VOC [45], CIFAR-10 [95] and ImageNet [153] datasets. While in the scene recognition, there is Places dataset [218]. In fine-grained image classification task, there are Caltech UCSD Birds-200 [201], Flowers-102 [128], and Wild Faces [73] datasets. These datasets contain ten thousands to millions of images. Contrary, only a few, very small paintings datasets have been made publicly available. For instance, Khan et al. [92] proposed a dataset consists of 4,266 paintings only. Whereas, dataset used in [86, 160, 161] have less than 1,000 paintings. Until recently, [155] provided a new dataset, namely the Wikiart paintings dataset[1] that consists of more than 80,000 of paintings. Examples of paintings in this dataset are illustrated in Figure 18. To date, [155, 88] are the only works that have used a fine-art paintings dataset of this size. In this work, a modified version of Wikiart dataset is provided by randomly splitting the dataset into training and validation sets. Compared to this newly released dataset, researchers are required to randomly split the existing Wikiart dataset, resulting in different collections of training and validation sets in each experiments. Hence, readers are encouraged to use the newly released Wikiart dataset for more comparable studies. Detail of the Wikiart dataset is described in Appendix B.

1 Paintings are collected from http://www.wikiart.org/

3.2.2 *Abstractionism*

Compared to recognizing the more structural objects such as dog, cat, or human faces [209], classification of fine-art collections is extremely difficult. One will require strong background in art domain for recognizing paintings as this mode of creative expression comes in different kinds of forms. In general, individuals could differentiate simple paintings categories, such as *portrait* or *landscape*. On the other hand, many other forms are **non-representational nor figurative**, and might require imagination to recognize them. For instance, *Impressionism* marked the beginning of a gradual departure from *Realism*, using loose brush stroke, sketchy lines, and blotches of colors that blend together to create the feeling of impression, giving less attention to the details. *Color Field paintings* are characterized primarily by large fields of flat, solid color spread across the canvas, making them non-figurative. While *Cubism* may contains object, it is broken up and reassembled in a structured abstracted form.

As an example, the second last painting in Figure 18 namely "*The nightingales song at midnight and morning rain*" is a piece of the 23 small paintings on paper (Constellations series), initiated by the great artist Joan Miró in 1939, is belong to this category. The Constellations is Miró's most luminous and affecting series of painting as it captures and represents the most vibrant expressions of Miró's inner universe during the outbreak of the Second World War. He explained the genesis in a letter to a friend: "*I had always enjoyed looking out of the windows at night and seeing the sky and the stars and the moon, but now we werent allowed to do this any more, so I painted the windows blue and I took my brushes and paint, and that was the beginning of the Constellations.*" Hence, a question arose is that does a machine able to capture "imagination" in paintings? One way to find out is train a CNN and then visualize the low-level to high-level features learned by the CNN. More samples for each category can be seen in Appendix B.

3.2.3 *Confusing Categories*

Furthermore, paintings categorizations are often confusing, especially in *style* classification. One example can be seen in the Renaissance arts. In general, most artworks that were created during the Renaissance period (14th to 17th centuries) are categorized as Renaissance art. In this period, many artists were influenced by the developments of science, humanism and ideas about depicting perspective. Hence, these artworks are usually looked highly realistic. However, such properties can also be noticed in other styles, such as *Realism*, *Baroque*, *Rococo*, *Romanticism*, etc. Another important cue to recognize Renaissance arts is the Greek mythology depicted in the artworks due to the influence from the newly-allowed study of Ancient Greek and Roman literature.

(a) Song of the Angels, William-Adolphe Bouguereau, 1881. *Neoclassicism*

(b) The Archangel Michael defeating Satan, Guido Reni, 1635. *Baroque*

Figure 19: Examples of artworks that contain angels. Sub-captions refers to: Name of artworks, artist, year. *style*

Still, Renaissance arts are not the only style that have mythological component in it, as shown in Figure 19. To further complicate matters, Renaissance arts are generally categorized into several different styles, differentiated by the periods and locations, which also affect the ideals, painting movements, subjects, and media of the artworks. These include *Proto-Renaissance*, *Early Renaissance*, *High Renaissance*, *Mannerism (Late Renaissance)*, and *Northern Renaissance*. Recognizing the correlations and differences between these styles is awfully difficult even for human, let alone machine.

3.3 METHODOLOGY

This work employed the AlexNet [97] for empirical studies. AlexNet is a 8-layers CNN model with 5 convolutional layers (conv1-5) and 3 fully-connected layers (fc6-8), as shown in Figure 20. For each convolution layer, the filter configurations are as follow. In conv1, the filter size is 11×11 with stride of 4. Filter size of 3×3 is used for other convolution layers with stride of 1, except for conv2 that uses filter size of 5×5. LRN is employed after conv1 and conv2 to aid generalization. Meanwhile, max pooling layers with size of 3×3 and stride of 2 are used after each LRN and conv5 for downsampling. Dropout [170] is implemented after fc6 and fc7 for regularization, as most of the parameters are concentrated in these layers. ReLU [125] is used in all layers except fc8 as the activation function.

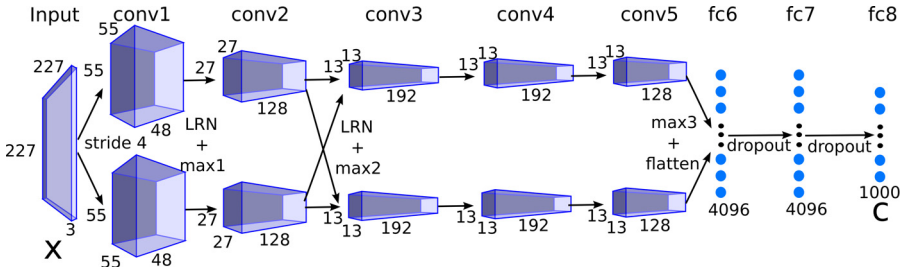


Figure 20: An illustration of AlexNet [97]. The network accepts 227×227 images with 3 channels (RGB) as input. Each convolutional layer yields total of 96, 256, 384, 384, and 256 feature maps, respectively. Meanwhile, fc6 and fc7 outputs 4096 neurons. Finally, the network yields 1000-way softmax output in fc8 as the classification confidence scores. In general, the scores are maximized to predict the class of the input image. When AlexNet is trained or finetuned on Wikiart dataset, the number of outputs is adjusted to match the number of classes in the recognition tasks.

Several models are trained under different configurations in order to investigate the effects of these configurations on the performance. The configurations are as follows:

- CNN - Trained from scratch on Wikiart dataset.
- CNN-nofine - Pre-trained on ImageNet [153], then replace the last layer (fc8) with a new softmax layer and only finetune the new layer.
- CNN-SVM - Similar to CNN-nofine, except that Support Vector Machine (SVM) [26] is used instead of softmax layer on fc8.
- CNN-1000 - Pre-trained on ImageNet [153], then stack a new softmax layer after fc8. Finetuning is only performed on the new layer.
- CNN-finetune - Similar to CNN-nofine, except that finetuning is conducted on all layers.
- CNN-fc6 - Similar to CNN-finetune, but fc7 is removed in this model. In other words, the new layer is stacked on top of fc6.
- CNN-1024 - Similar to CNN-finetune, except that fc6 and fc7 are replaced by new layers with the number of neurons reduced from 4096 to 1024, in addition to the new softmax layer at fc8.

CNN is used as the baseline benchmark during comparisons. Razavian et al. [141] showed that replacing last layer of CNN with SVM during transferring performs better. Hence, the CNN-SVM is trained to verify this claim. Meanwhile, the performance of CNN-1000 can be used to deduce if the high level semantic information provided by fc8 helps. On the other hand, the performances of CNN-fc6 and

CNN-1024 is useful to assess the overparametrization problem that occurred in many deep learning models [33]. The rest of the settings are as follows:

Weights initialization: When the pre-trained weights are not available (e.g. train from scratch or on the new layers), all weights are initialized from the zero-mean Gaussian distribution with standard deviation of 0.01. Biases are initialized to 1, except for the first and third layers are set to zero. This is because setting biases to nonzero in some layers provide ReLUs with positive inputs to improve training [97].

Learning rate configurations: Learning rates were initialized to 0.01 for all weights and 0.02 for bias. However, learning rates of the fine-tuning layers are reduced by a factor of 10 to avoid tampering the already well-learned weights and biases. For every 5000 iterations prior to the termination, all learning rates are reduced by a factor of 10.

Data augmentation: The networks were trained on pro-processed images by subtracting the mean activity over the training set from each pixel. In order to overcome the overfitting problem, few data augmentation techniques were employed in the experiments. First, image translation is performed by randomly cropping 227×227 patches from the 256×256 images. Each iteration will only crop one random patch for each image. Then, the patches are randomly mirrored under the horizontal reflection technique. Note that different patches are cropped in each iteration. During validation, the images from the validation set are centered cropped without horizontal reflection.

Other settings: The models are trained using the SGD with momentum as described in Chapter 2.3.5 with batch size of 128. The momentum is set to 0.9, while weight decay is set to 0.0005. All models are trained for 20,000 iterations. It is found that further training does not improve the results. The experiments were carried out using Caffe [84] on NVIDIA GTX 980 4GB GPU.

3.4 EXPERIMENTS AND DISCUSSIONS

Table 1 summarizes the classification results of the trained models. The *overall* results infer the averaged accuracy of all classification tasks of the same model. Based on the results, a few interesting deductions can be made:

- By comparing CNN with other models trained in this work, it can be seen that transfer learning helps improve the accuracy.
- Meanwhile, the accuracies between softmax (CNN-nofine) and SVM (CNN-SVM) are comparable. In fact, CNN-nofine outper-

Table 1: Comparisons between different models on Wikiarts dataset for styles, genres, artists classification.

Model	Accuracy (%)				Size
	Style	Genre	Artist	Overall	
CNN	42.96	65.45	54.39	54.27	61M
CNN-nofine	45.95	69.24	67.02	60.74	61M
CNN-SVM	44.17	69.18	67.17	60.17	61M
CNN-1000	43.56	68.38	64.55	58.83	61M
CNN-finetune	**54.50**	**74.14**	**76.11**	**68.25**	61M
CNN-fc6	51.51	72.11	74.26	65.96	44M
CNN-1024	53.38	73.75	76.02	67.72	48M
CNN-PCA-SVM [156]	21.99	49.98	33.62	35.20	-
Saleh and Elgammal [156]	45.97	60.28	63.06	56.44	-

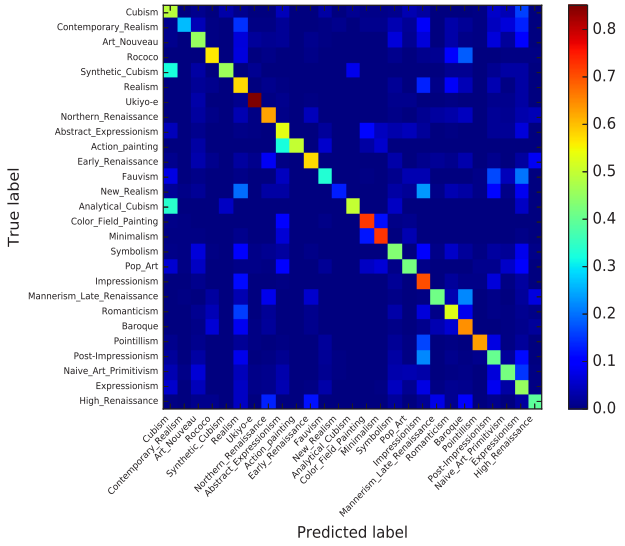
formed CNN-SVM by a small margin in the experiments. In other words, there is no guarantee that softmax or SVM is better.

- On the other hand, CNN-1000 performed worse than other fine-tuned models. Note that the last layer of the pre-trained model computes the confidence scores of the existence of certain objects in the image. Hence, the results can be expected because recognizing a painting's style, genre, and artist require other knowledge such as shape, perspective, emotion, etc.

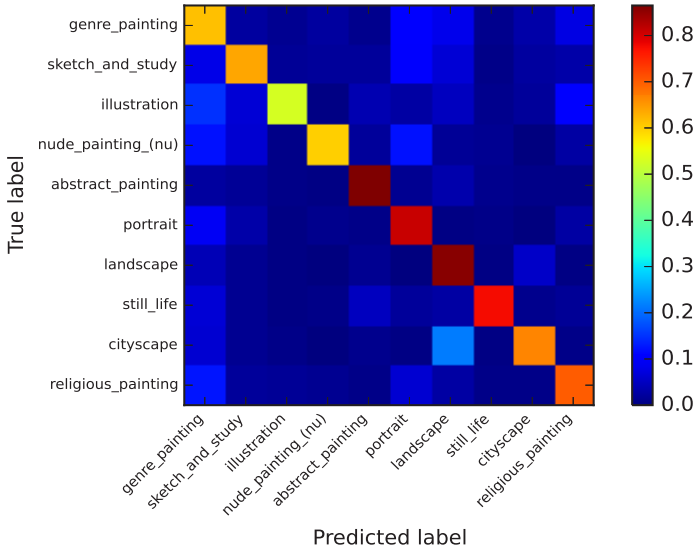
- CNN-finetune outperformed all other models in the experiments with overall accuracy of 68%, compared to the state-of-the-art result (56%).

- Last but not least, reducing the number of parameters by $20 \sim 30\%$ in CNN-fc6 and CNN-1024 only deteriorated the accuracy by $1 \sim 2\%$. The main insight here is that, it seems that a better pruning strategy might be able to compress the network further without affecting the system accuracy substantially, as proven in a recent study by Han et al. [60].

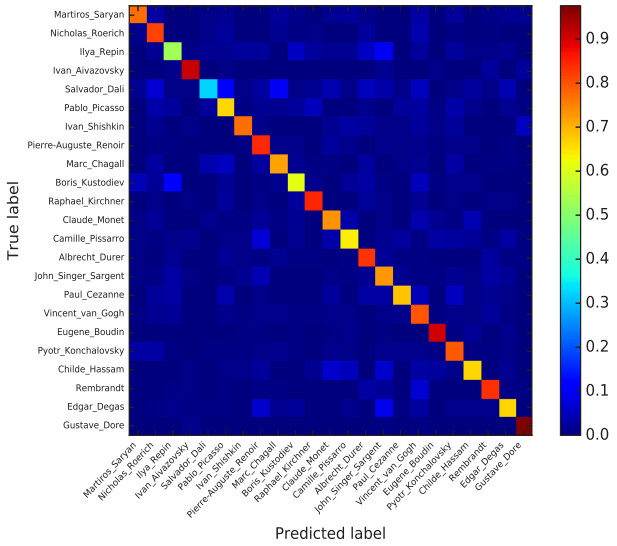
In order to further assess the performance of the CNN, the confusion matrices are analysed using the results from the CNN-finetune model; as well as visualizing the CNN features by the neurons' responses to the paintings in the next subsections.



(a) Style



(b) Genre



(c) Artist

Figure 21: Confusion matrix of the style, genre and artist classification tasks using the CNN-finetune model. The colorbar shows the normalized intensity. Best viewed in color and enlarged pdf.

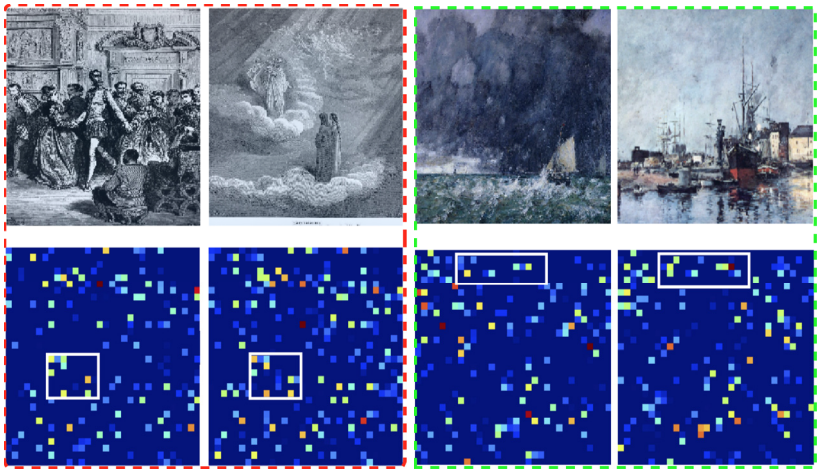
3.4.1 *Confusion Matrix*

Figure 21 shows the confusion matrix for each classification task. Among all, there are a few observations that worth attention.

Style: It shows that the deep model is able to distinctly differentiate *Ukiyo-e* (85%) from other styles. This is because most of the paintings in other styles are from the regions of Europe. Meanwhile, Ukiyo-e is a type of art that flourished in Japan that is very distinctive from other styles. However, the model performed very poorly in differentiating many other styles due to the confusions as discussed before. For instance, differentiating *synthetic cubism* (46%) and *analytical cubism* (50%) is extremely difficult as these styles are from the same root, i.e. *Cubism*. Similarly, *Rococo* (56%) and *Baroque* (64%) are historically related. This explains why the style classification has the poorest performance among the three classification tasks.

Genre: It is not surprised to see that CNN performed very well in genre classification. This is because most of the categories in genres can be related to other recognition problems. For instance, recognizing *portrait* (81%) and *landscape* (86%) can be related to face detection [209] and scene recognition [218], where CNN has been very successful in these tasks.

Artist: It is interesting to see that the CNN performed the best in artist classification task. To uncover the factors behind this, the best and worst performances are investigated. It is discovered that the artists that CNN can recognize with high precision usually prefer certain techniques or subjects in their paintings. For example, *Gustave Dore* has been using mostly engraving, etching, and lithography techniques, which result in greyish images as shown in Figure 22a. *Eugene Boudin* has many paintings that depict outdoor scenes, and most of his paintings were rendering marine and seashore (Figure 22b). Meanwhile, the CNN failed miserably (33%) in recognizing artworks from Salvator Dalí, a prominent Spanish best known for the striking and bizarre images, and confuses his works to the greatest and most influential artists of the 20th century, *Pablo Picasso*. The most interesting part of this finding is, it is found that historically, Salvator Dalí made a number of works heavily influenced by Picasso, which is not known before this result. Further investigations found that a recent exhibit at the Salvador Dalí Museum in Florida (Feb. 2015) examined how these two artists influenced each other and Dr. William Jeffett, a Chief Curator of Exhibitions of the museum said "*The paintings look good together. The pieces really complement each other, which I think says a lot about the artists and their works, for example Picasso's Portrait of Olga, or Dalí's Portrait of My Sister. This wasn't just a contextual show where we were looking for academic or documentary links. There's a visual component here evident in the art, which supports what were trying to say*". This seems like a hint



(a) Gustave Dore (b) Eugene Boudin

Figure 22: The first row shows the artists paintings and the second row shows the corresponding features in the feature space, where paintings from similar artists are close with each other.

that CNN is able to semantically link artists together. This hypothesis is left for future investigation.

3.4.2 *Visualizing Neurons' Responses*

Figure 23 visualizes the neurons' responses in the genre classification task. The visualization is done by averaging the neurons' value over the feature maps. Let $l_i \in \mathbb{R}^{W \times H \times C}$ be the features extracted at layer i, that is a 3D matrix with $W \times H$ positions and C feature maps, the averaged neurons' responses $\mathcal{R}_i^{(w,h)}$ at position (w, h) is given as,

$$\mathcal{R}_i^{(w,h)} = \frac{1}{C} \sum_{n=0}^{C} l_{i,n}^{(w,h)} \tag{14}$$

As shown in Figure 23, the network learned to recognize simple edges/blobs (low level features) at lower layers (e.g. layer 1). As the layer goes higher, the network learned to recognize texture pattern to complex object parts, such as face in Portrait. According to the observation, training a CNN for paintings classification task is tougher as paintings from the same group does not necessarily have similar low to high level features, e.g *Illustration*. Meanwhile, Illustration is defined by its published media instead of some visual properties[2]. Hence, the content of an Illustration may also depicts a landscape or genre paintings, causing more confusions when recognizing these

2 Illustration is defined as "a decoration, interpretation or visual explanation of a text, concept or process, designed for integration in published media, such as posters, flyers, magazines, books, teaching materials, animations, video games and films" in Wikipedia.

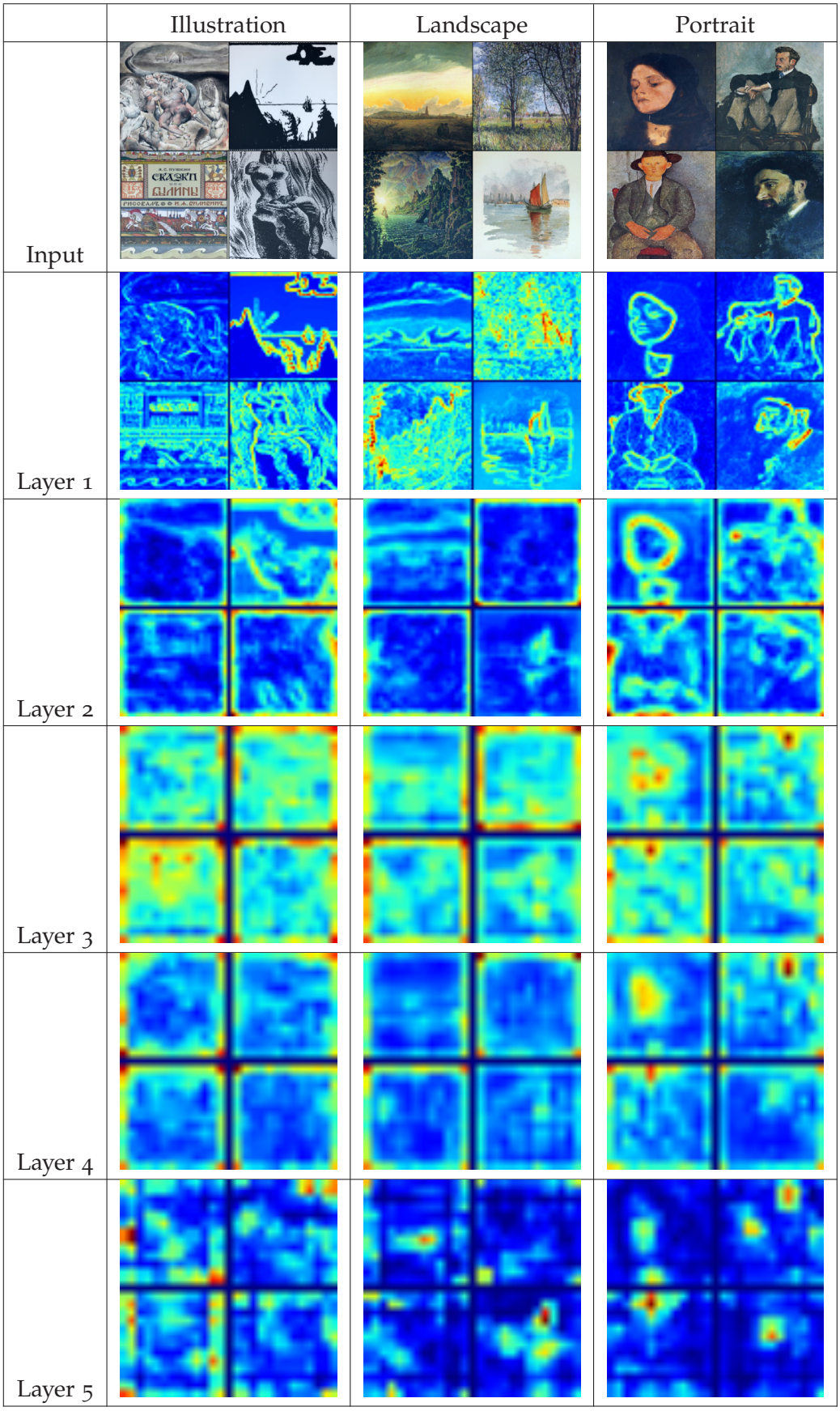


Figure 23: The visualization of neurons' responses in the genre classification for layers conv1-5.

categories while the published media is not known. Hence, it is reasonable that CNN performed poorly in recognizing Illustration since it is hard to learn the visual cues from Illustration. For paintings that are more structured, the visualizations also show that CNN tends to find key objects or shapes for cues. No clue indicates if the deep models are able to learn the more abstract cues.

3.5 CONCLUSIONS

This work presented a study on features learning of fine-art paintings by training deep models for style, genre, and artist classification tasks. It is found that finetuning an ImageNet pre-trained CNN yields the best result and achieves state-of-the-art results. However, the confusion matrices and the visualization from the neurons' responses showed two contradictory reasons that lead to many misclassifications. First, similar properties can be exhibited by many different styles. Secondly, the features extracted from the lower level to higher level layers can be very different among the same category. Unfortunately, such contradiction is reasonable based on the definition of the art categories, as discussed. Nonetheless, the visualizations also showed that deep models are able to learn to extract structural cues from the artworks. However, it is unclear if any abstract cue has been learned. In addition, it is also found that the CNN somehow could relate different artists together based on their painting style. For future works, the author is interested in investigating other loss functions that can help develop and learn other visual cues that are useful in paintings analysis. Meanwhile, different visualization technique will be presented in next chapter for better understanding of how CNN extracts features from paintings.

4

ARTWORKS SYNTHESIS

4.1 INTRODUCTION

Generative models have been a fundamental interest and challenging problem in the field of computer vision and machine learning. Traditionally, discriminative models (e.g. CNN models used in Chapter 3) learn the dependencies of the unobserved variables y (e.g. classes) on the observed variables $\mathbf{x}$ (e.g. images) by modelling the conditional probability distribution $p(y|\mathbf{x})$. Unlike discriminative models, generative models learn the dependencies from the joint distribution $p(\mathbf{x}, y)$, which allow them to exploit the unlabelled data and generate samples from the observed distribution. Hence, generative models usually require lesser or no labelled data to train, compared to discriminator models which require large amount of labelled data. Such low labelled data requirement is extremely helpful and important to many real-world problem where labelled data are hard to collect. In addition, the observed data simulated by the generative models can be visualized to assess the performance of the learned model. Compared to quantitative assessment such as accuracy, visualization of the synthetic data helps understand what kind of features have been learned by the model better in a qualitative manner.

However, training generative model is extremely difficult due to the difficulty in estimating the intractable likelihood. Recently, Goodfellow et al. [53] introduced an interesting way to bypass the likelihood estimation. They proposed the Generative Adversarial Network (GAN) that is trained via adversarial training, as described in Chapter 2.4. Since then, many extensions of GAN [123, 139, 167, 35, 133, 146, 216] have been proposed and showed promising results in generating *structural* images using the MNIST [104], CIFAR-10 [96], CUB-200 [194] and LFW [73] datasets. As discussed in the literature, GAN has shown excellent performance in generating random *photo-realistic* images, inferring that GAN is able to learn better visual representations compared to other generative models. However, limited attention has been given to learn visual representations and generate samples from the more *abstract* artworks.

To this end, this work proposes a novel GAN variant called ArtGAN[1]. This work anticipates that a good way to look at this problem is to understand how human learns to draw. An artist teacher

1 The network is named as ArtGAN since the nature of this work is to synthetically generate artworks

wrote an online article[2] and pointed out that an effective learning requires focus on a particular type of skill at a time, e.g. practice to draw a particular object or one kind of movement. Following this intuition, the generator in ArtGAN takes a randomly chosen label and a noise vector as inputs. The chosen label is then used as the true label in the discriminator when computing the loss function for the generated image. Hence, the discriminator is modified, resembling a CNN classifier and is named as "categorical discriminator" in this context. Overall, the idea is to **allow the generator to learn better by leveraging the feedback information from the labels**. In this regard, ArtGAN can be seen as a mixture of discriminative and generative models, which utilizes the advantages from them. Inspired by recent works [217, 13], a **categorical autoencoder-based discriminator** that incorporates an autoencoder into the categorical discriminator for additional complementary information is also introduced. Rather than deploying two separate computationally expensive networks (a categorical discriminator and an autoencoder), the categorical autoencoder-based discriminator partly shares the same architecture and weights. In specific, the encoder in the autoencoder is shared by the categorical discriminator.

Additionally, this work proposed a novel **magnified learning** strategy so that ArtGAN can synthetically generate high-resolution artworks. The motivation behind this approach is to generate a set of pixels that vote for a single pixel. One may naively train an ensemble of GANs to achieve this goal. However, training multiple networks explicitly is computationally expensive and unnecessary to achieve similar performance gain [198, 72]. Hence, this work proposes an alternative approach by generating images with higher resolution compared to the available image size in the dataset, e.g. 64×64 pixels instead of 32×32 pixels for CIFAR-10 samples. Then, the generated images are downsampled to the original size (i.e. 32×32 as accordance with previous example), which resembles a form of voting scheme. An advantage of this approach is that the correlations between the pixels within the same downsampling block can be learned better.

In summary, the key contributions are 1) A novel conditional GAN variant is proposed, namely as the ArtGAN to emulate the concept of effective learning to generate very challenging images (i.e. artwork). To the best of our knowledge, no existing empirical research has addressed the implementation of a generative model on a large scale paintings dataset. 2) A novel magnified learning is proposed to synthesize better quality images. 3) Empirically, this work shows that the proposed model is capable of generating high quality artworks that exhibit similar visual representations within genre, artist, or style. At the same time, the proposed model is also able to generate samples

2 http://www.learning-to-see.co.uk/effective-practice

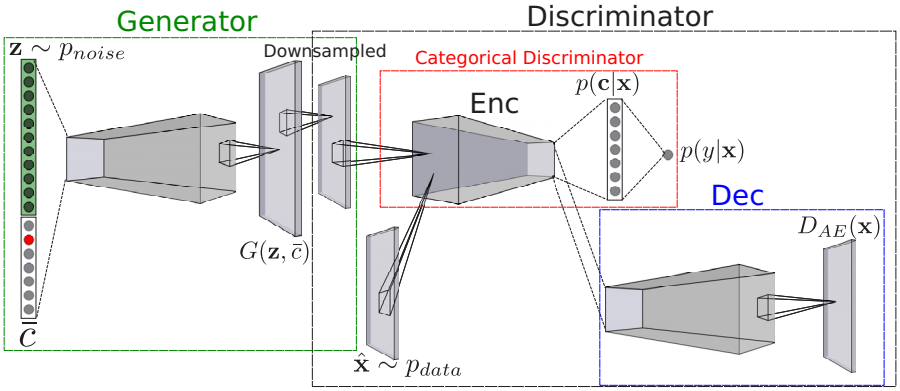


Figure 24: Overview of the ArtGAN architecture. $\mathbf{z}$ and $\bar{c}$ are concatenated and fed to the generator to produce synthetic image $G(\mathbf{z}, \bar{c})$. Either the downsampled generated image $G(\mathbf{z}, \bar{c})$ or real data $\hat{\mathbf{x}}$ is used as the input $\mathbf{x}$ to the (categorical autoencoder-based) discriminator. The discriminator produces three outputs: the class prediction $p(\mathbf{c}|\mathbf{x})$, adversarial prediction $p(y|\mathbf{x})$, and the reconstructed image $D_{AE}(\mathbf{x})$.

on CIFAR-10 [96], STL-10 [24], Oxford-102 [128], and CUB-200 [194] that look natural and contain clear object structures in them.

4.2 METHODOLOGY

This section describes the proposed method in detail, while Figure 24 summarizes the overall architecture of the proposed ArtGAN. Detailed architecture and pseudocode are presented in Appendix C.

4.2.1 *ArtGAN*

The basic structure of ArtGAN is similar to GAN, such that it consists of a discriminator and a generator that are simultaneously trained using the minimax formulation of GAN, as described in equation 9. The key innovation of ArtGAN is to allow feedback from the labels given to each generated image through the loss function. That is, additional label information is fed to the generator to draw a specific subject based on the information, imitating how human learns to draw. This is in contrast to CondGAN [123] that does not fully utilize the labels during training. In order to leverage the labels information, the discriminator is modified and named as *categorical discriminator* to output $K + 1$ logistic predictions with K actual categories following the dataset used, and $K + 1$th output as the adversarial class (denoted as Fake category).

Formally, the formulation of a categorical discriminator is written as $D : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{K+1}$, where H, W, and C are the height, width, and number of channels of an image, respectively. This is somehow

similar to Salimans et al. [157], except that the conditional setting is not implemented in their work. While, the notations of the conditional generator is written as $G : (\mathbf{z}, \bar{c}) \rightarrow \mathbb{R}^{H \times W \times C}$, where $\bar{c}$ is the randomly chosen label for the generated sample in the form of one-hot vector. This allows the generator to learn from the feedback labels information better. Following Salimans et al. [157], the categorical discriminator is modified such that D becomes the standard supervised classifier with K outputs, $D : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^{K}$. Let $l_k(\mathbf{x}) \in D(\mathbf{x})$ be the output of $D(\mathbf{x})$ at class k without activation function and $\mathbf{x}$ is an input image (either from real data or generator). The probability distribution over K classes is given as $p(\mathbf{c}|\mathbf{x})$, such that the predicted probability for each class k is defined as a Softmax function,

$$p(c_k|\mathbf{x}) = \frac{e^{l_k}}{\sum_{i=1}^{K} e^{l_i}} \tag{15}$$

The binary adversarial prediction y, which is the inverse of $K+1$th output, is then formulated as the probability distribution function $p(y|\mathbf{x})$:

$$p(y|\mathbf{x}) = \frac{Z(\mathbf{x})}{Z(\mathbf{x}) + 1} \tag{16}$$

where $Z(\mathbf{x}) = \sum_{i=1}^{K} e^{l_i}$ and $p(y|\mathbf{x}) = 1$ infers that the image $\mathbf{x}$ is real. The benefit of such setting is that the number of parameters can be reduced to relax the overparametrization problem without changing the output of the Softmax, conceptually. Detailed explanation is available in Appendix A. The D is then trained by minimizing the following discriminator loss function $\mathcal{L}_D$,

$$\begin{aligned} \mathcal{L}_D = &- \mathbb{E}_{(\hat{\mathbf{x}}, \hat{c}) \sim p_{data}} \left[\sum_{i=1}^{K} \hat{c}_i \log p(c_i|\hat{\mathbf{x}}) + \log p(y|\hat{\mathbf{x}}) \right] \\ &- \mathbb{E}_{\mathbf{z} \sim p_{noise}, \bar{c}} \left[\log(1 - p(y|G(\mathbf{z}, \bar{c}))) \right] \end{aligned} \tag{17}$$

where $\hat{c}$ is the ground truth one-hot label of the given real image $\hat{\mathbf{x}}$. The generator loss function $\mathcal{L}_G$ to be minimized for training G is defined as,

$$\mathcal{L}_G = - \mathbb{E}_{\mathbf{z} \sim p_{noise}, \bar{c}} \left[\sum_{i=1}^{K} \bar{c}_i \log p(c_i|G(\mathbf{z}, \bar{c})) + \log(p(y|G(\mathbf{z}, \bar{c}))) \right] \tag{18}$$

Inspired by recent works [199, 217, 13], another discriminator variant named *categorical autoencoder-based discriminators* is proposed by incorporating an autoencoder into the categorical discriminator for additional complementary information. The core idea of implementing autoencoder in the discriminator is that reconstruction-based out-

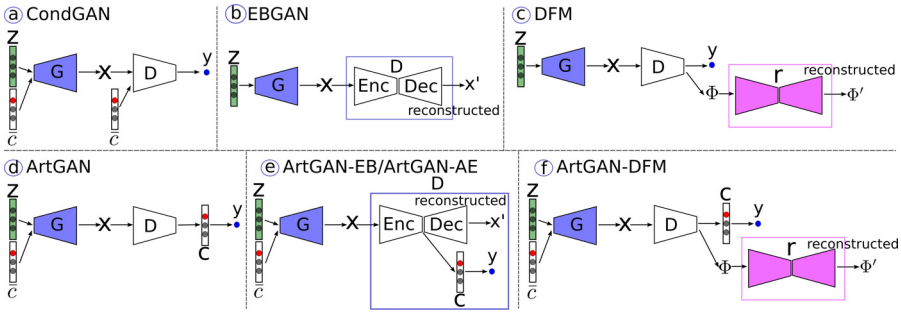


Figure 25: Different ArtGAN variants (bottom) compared to the state-of-the-art models (top). The discriminator in ArtGAN outputs the class predictions and the loss function is computed from the true labels, instead of taking the true labels as input as depicted in the CondGAN. Hence, the true labels can be leveraged to train the discriminator and generator. Meanwhile, ArtGAN-EB and ArtGAN-AE share the same model, which combines ArtGAN and EBGAN by sharing the encoder. However, decoder in the ArtGAN-AE is not trained using the generated samples, contrast to the ArtGAN-EB. The ArtGAN-DFM depicts the extension from DFM with conditional settings.

put offers diverse targets and produces very different gradient directions within the minibatch. This conceptually improves the efficiency and effectiveness when training a GAN model. As shown in Figure 24, the categorical discriminator and the *encoder* in the autoencoder share the same weights, rather than deploying two separate computationally expensive networks. The formulations of the categorical autoencoder-based discriminators are described in three ways. The first two types, namely as the ArtGAN-EB and the ArtGAN-AE are implemented using the pixel-level autoencoder, similar to the EBGAN [217]. However, these two models are differed by the discriminator loss functions formulation. The third type, namely the ArtGAN-DFM is an extension of the Denoising Feature Matching (DFM) [199] to a conditional setup, forming a *Conditional DFM*. The ArtGAN variants are summarized in Figure 25 and the formulation of the loss functions will be described next in detail. Meanwhile, analysis and comparisons between these ArtGAN variants will be discussed in the experiment section.

ArtGAN-EB: EBGAN [217] is formulated according to the energy-based models by replacing the discriminator with an autoencoder, such that $D_{AE}(\cdot) = Dec(Enc(\cdot))$, where Dec and Enc are the decoder and encoder, respectively. The discriminator loss $\mathcal{L}_{Deb}$ in the EBGAN is given as,

$$\mathcal{L}_{Deb} = \mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}} \Big[||D_{AE}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}|| \Big] + \mathbb{E}_{\mathbf{z} \sim p_{noise}} \Big[\max(0, m - ||D_{AE}(G(\mathbf{z})) - G(\mathbf{z})||) \Big] \tag{19}$$

where $||\cdot||$ is an Euclidean norm, and m as a positive margin. While the generator loss $\mathcal{L}_{Geb}$ is formulated as,

$$\mathcal{L}_{Geb} = \mathbb{E}_{\mathbf{z} \sim p_{noise}} \Big[||D_{AE}(G(\mathbf{z})) - G(\mathbf{z})|| \Big] \tag{20}$$

In order to formulate a conditional energy-based loss function, ArtGAN-EB propose a novel discriminator loss function $\mathcal{L}_{Debc}$ as,

$$\mathcal{L}_{Debc} = \mathcal{L}_D + \mathcal{L}_{Deb} \tag{21}$$

While, the new generator loss $\mathcal{L}_{Gae}$ is defined as,

$$\mathcal{L}_{Gae} = \mathcal{L}_G + \mathcal{L}_{Geb} \tag{22}$$

ArtGAN-AE: The discriminator loss is similar to $\mathcal{L}_{Debc}$, except that the generated images are not used as adversarial samples to update the decoder. This was inspired by DFM [199] to use the autoencoder as a source of *complementary information* when updating the generator, instead of using the autoencoder as an adversarial function (as in [217]). Hence, the discriminator loss $\mathcal{L}_{Dae}$ of ArtGAN-AE is formulated as,

$$\mathcal{L}_{Dae} = \mathcal{L}_D + \mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}} \Big[||D_{AE}(\hat{\mathbf{x}}) - \hat{\mathbf{x}}|| \Big] \tag{23}$$

Meanwhile, ArtGAN-AE uses equation 22 described in ArtGAN-EB as the generator loss function.

ArtGAN-DFM: In DFM [199], an additional denoising autoencoder (or denoiser) $r(\cdot)$ is employed to update the generator. The denoiser is trained separately from the discriminator. In specific, the denoiser is trained on the discriminator's hidden state when evaluated on the training data. Formally, D is updated according to Eq. 9. Given that $\Phi(\cdot)$ is a hidden state from $D(\cdot)$, the denoiser is trained by minimizing the following loss function $\mathcal{L}_r$,

$$\mathcal{L}_r = \mathbb{E}_{\hat{\mathbf{x}} \sim p_{data}} \Big[||\Phi(\hat{\mathbf{x}}) - r(\Phi(\hat{\mathbf{x}}))|| \Big] \tag{24}$$

Then, the generator is trained with the loss function $\mathcal{L}_{Gdfm}$,

$$\mathcal{L}_{Gdfm} = \mathbb{E}_{\mathbf{z} \sim p_{noise}} \Big[\lambda_{denoise} ||\Phi(G(\mathbf{z})) - r(\Phi(G(\mathbf{z})))|| - \lambda_{adv} \log D(G(\mathbf{z})) \Big] \tag{25}$$

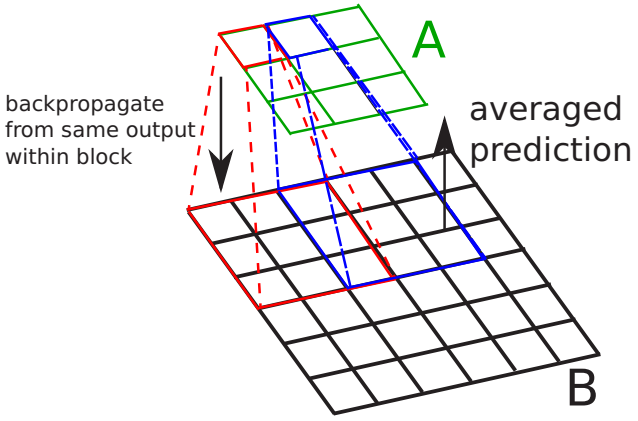


Figure 26: **Magnified learning** using *overlapped average pooling*. Pixels in the same block from B (e. g.$[B_1, ..., B_9]$) will vote for one pixel in A (i.e. A_1) through averaging. During training, the same gradient (i.e. from A_1) will be backpropagated to all pixels in the block (i.e. $[B_1, ..., B_9]$), which helps the pixels in B to learn better correlations between themselves. Similarly, the neighbouring pixels in A will also be correlated due to the *overlapped* pooling operation.

The authors [199] suggested to fix $\lambda_{adv} = 1$ and set $\lambda_{denoise} = 0.03/n_h$, where n_h is the number of discriminator hidden units fed to the denoiser as input. Here, the modification is straightforward using the categorical discriminator as the discriminator network. Hence, the discriminator loss is same as Eq. 17, and the denoiser loss remains unchanged (Eq. 24). While, the generator loss $\mathcal{L}_{Gdfmc}$ for the conditional DFM is defined as,

$$\mathcal{L}_{Gdfmc} = \mathbb{E}_{\mathbf{z}\sim p_{noise}}\Big[\lambda_{denoise}||\Phi(G(\mathbf{z})) - r(\Phi(G(\mathbf{z})))||\Big] + \mathcal{L}_G \tag{26}$$

4.2.2 *Magnified Learning*

In order to synthetically generate high quality image, this work proposes a novel learning scheme called magnified learning. The motivation behind the proposed magnified learning is to generate a set of pixel values to vote for a single pixel. In this context, the proposed magnified learning is defined as "*a process of training a generative model by generating samples at higher dimensional feature space, while evaluate the samples at original dimension (i.e. same as the true data)*".

In specific, suppose a generator in the traditional GAN trained on CIFAR-10 usually generates 32×32 pixels images, $G : \mathbf{z} \rightarrow \mathbb{R}^{32\times32\times C}$. By using magnified learning, the generator is formulated to generate 64×64 pixels images, $G : \mathbf{z} \rightarrow \mathbb{R}^{64\times64\times C}$. This can be done by adding an extra upsampling layer between the existing layers in the generator. Meanwhile, the input image size of the discriminator remains the

same as to the original size, such that $D : \mathbb{R}^{32\times32\times C} \rightarrow \mathbb{R}^{K}$. When evaluating the generated samples with the discriminator, the samples are downsampled, such that $\pi : \mathbb{R}^{64\times64\times C} \rightarrow \mathbb{R}^{32\times32\times C}$ where $\pi(\cdot)$ is a downsampling operation.

In this work, overlapped average pooling is used as the downsampling operation. This pooling operation can be viewed as a form of voting system, as shown in Figure 26. An advantage of this design is that the correlations between the neighbouring pixels can be learned better. This helps the model to learn to draw finer details, resulting in improved image quality. Overlapping the pooling operations discourages the generator to compute pixel values that collapse within the same block, i.e. pixels of the same pooling block have exactly same value. Overall, by using overlapped average pooling in magnified learning, the generator is regularized with two seemingly contradictory constraints: 1) the pixels within the same pooling block should have similar values so that the generated image looks smooth across same color (i.e. the blue coloured sky looks smooth); 2) these pixels must not be naively computed to produce exactly same value that could cause excessive artefacts in the image.

In addition, the magnified learning also supports generating images at higher resolution without the need of using dataset that contains high resolution images. Unlike PPGN [127] which requires a computationally expensive recurrent sampling procedure to generate one image, ArtGAN with magnified learning, namely as the ArtGAN-M is able to generate high quality images at high resolution in *one-shot*. It can be argued that one-shot procedure is more desired when a model is deployed in a real-world application as not all embedded systems has high computational power, for instance a mobile device.

4.3 EXPERIMENTS AND DISCUSSIONS

4.3.1 *Experimental settings*

This section describes the settings that are shared by all experiments in this chapter, unless stated otherwise. All networks are trained with the Adam optimizer [93] with initial learning rate of 0.0002, $\beta_1 = 0.5$, and minibatch size of 100. The learning rate is decreased once by a factor of 10 after iteration $30,000$. Input noise vector $\mathbf{z}$ is a 100-dimensional multivariate random variable sampled using an i.i.d. uniform distributed random generator $U(-1, 1)$. Instance noise [166] is implemented in all discriminators for better training stability. During magnified learning, all samples are generated at twice as high as the resolution of the non-magnified learning counterparts. For fair comparisons, this work run one gradient descent step for each player in each iteration. This usually works better than running more steps of one player than the other [52]. In practice, it is very difficult to de-

termine how many more steps to run as the performance is usually inconsistent with the same setting on different datasets. The rest of the settings will be described in other sections. The experiments were conducted using Tensorflow [2] with one Titan X (Maxwell) GPU.

In order to assess the performance of the proposed models, Inception score is adopted [157] for quantitative measurement. Intuitively, Inception score measures the *objectness* by minimizing entropy per-sample posterior (i.e. each sample is classified with high certainty), as well as the *class diversity* by maximizing the entropy aggregate posterior (i.e. the classifier used in Inception score identifies wide variety of classes among the samples). However, the *class diversity* metric becomes meaningless in the conditional setting as the conditional generative models will almost always generate visually different images for different modes. In addition, the *class diversity* metric can be misleading, i.e. it can be maximized (higher is better) and fooled when the generated samples have uniform distribution across all classes prediction. Hence, the measurements are split (*objectness* and *class diversity* metrics) when the scores are reported for a better performance assessments.

Since Inception score is measured mainly based on the objectness of an image, therefore it is unsuitable for assessing the model performance on the artworks. Meanwhile, evaluation of generative model based on the state-of-the-art log-likelihood estimates can be misleading [188]. Hence, the comparative studies are first conducted using the *objectness* metric from Inception score on CIFAR-10 [96] and STL-10 [24] datasets. The experiments report state-of-the-art results on these datasets based on the *objectness* and Inception scores. Then, Wikiart dataset [156, 185] is used to train the best model found for artworks synthesis based on the genres, artists, and styles. In addition, this work also trained the model on Oxford-102 [128] and CUB-200 [194] for additional performance assessments.

Similar design to BEGAN [13] is used by employing nearest neighbour upsampling instead of strided deconvolution layer in the generator as suggested by Odena et al. [132] in order to avoid checkerboard artifacts. Between the upsampling layers are at least one layer of convolutional layer. The discriminator has the same design as to the traditional GAN with multiple layers of strided convolutional layers. Batch normalization and leaky ReLU are used for both the discriminator and generator. Detailed network descriptions and additional generated samples are available in the Appendix C and Appendix D. Codes are made available in: https://github.com/willtwr/ArtGAN. The list of models to be evaluated are as follows:

1. ArtGAN - Baseline model.

2. ArtGAN-EB - The first type of categorical autoencoder-based discriminator.

3. ArtGAN-AE - The second type of categorical autoencoder-based discriminator.
4. ArtGAN-DFM - The third type of categorical autoencoder-based discriminator.
5. ArtGAN-M - ArtGAN with **magnified learning**.
6. ArtGAN-D - ArtGAN with deeper architecture (more layers and number of parameters). This is implemented to verify that network size is not the main factor that contributes to the improvements observed in the experiments when using magnified learning.
7. ArtGAN-AEM - ArtGAN-AE with **magnified learning**.
8. ArtGAN-AEMT - Huang et al. [75] employed a trick by updating more steps for the generator per each discriminator update step. Although it is hard to determine number of steps, their setup seems to work well for CIFAR-10. Hence, the same setting is employed in the CIFAR-10 experiment.

4.3.2 *Evaluation and quantitative metric*

Evaluation of a generative model is extremely difficult as it is still not clear how to quantitatively evaluate a generative model. This is due to the difficulty in estimating the intractable log-likelihood in many models [188]. The most widely used log-likelihood estimator is the Parzen window estimates [136]. However, Theis et al. [188] convincingly argued that this estimator can be quite misleading for high-dimensional data. Recently, Salimans et al. [157] proposed a different way to assess image quality by using the Inception score (higher is better). The formulation of the Inception score is defined as:

$$\begin{aligned} I(\{\mathbf{x}\}_1^N) &= \exp(\mathbb{E}[D_{KL}(p(y|\mathbf{x})||p(y))]) \\ &\approx \exp(-\mathbb{E}[H(p(y|\mathbf{x}))] + \mathbb{E}[H(\mathbb{E}_{\mathbf{x}}(p(y|\mathbf{x})))]) \end{aligned} \tag{27}$$

where $H(\cdot)$ is the Shannon entropy and $D_{KL}(\cdot)$ is the KullbackLeibler divergence. As aforementioned, this metric measures the *objectness* in the first term (lower is better) and *class diversity* in the second term (higher is better) of the samples, which can be misleading when the *class diversity* metric is fooled. One of this case can be seen in the experiments when comparing ArtGAN (baseline) and ArtGAN-EB in Table 2. Although ArtGAN-EB performed better than ArtGAN with higher Inception score (8.26 by ArtGAN-EB compared to 8.21 by ArtGAN), it has worse *objectness* score (33.51 by ArtGAN-EB compared to 33.24 by ArtGAN). It is clear that the *class diversity* score in the

ArtGAN-EB unreliably affected its Inception score as higher *class diversity* score can also imply that the objects in the generated images are hard to recognize. This is especially true when the model has the worst score in the *objectness* metric. Nonetheless, Inception score is still a preferred metric due to the lack of a better alternative for the quantitative measurement. Hence, this work adopts Inception score but the performance assessment is done mainly based on the *objectness* score since it is a more reliable metric.

In addition, the generated images will be displayed for visual inspection as human evaluation is always more accurate when accessing the image quality, though can be subjective at times. Furthermore, latent space interpolation on the proposed ArtGAN-AEM is performed to "probe" the structure of the latent space $\mathbf{z}$. The smooth transitions between samples when the latent space is interpolated usually indicates how well the generative models understand the structure of the images.

4.3.3 *CIFAR-10*

CIFAR-10 [96] is a small, well-studied dataset consisting 32×32 pixels RGB images. It is split into 50,000 training images and 10,000 test images from 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The models are trained on full image size, i.e. 32×32 pixels, while the generator generates 64×64 pixels images during magnified learning. The models are trained for 70,000 iterations and saved every 1,000 iterations. As shown by Gulrajani et al. [57], the Inception scores of the generative models continue to oscillate with non-negligible amplitude at convergence. Hence, best models found based on the *objectness* score are reported in Table 2 along with the state-of-the-art results. Generated samples are shown in Figure 27.

4.3.4 *STL- 10*

STL-10 [24] is a dataset inspired by CIFAR-10 with higher image resolution of 96×96 pixels. It contains fewer labelled training examples and a very large set of unlabelled examples. Although STL-10 is primarily used for developing unsupervised features learning, this work stays focus on the goal of this thesis and use STL-10 for conditional image synthesis in a supervised fashion. In particular, this work only used the labelled examples during training, which contains only 5,000 samples from 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck, which makes STL-10 a more challenging dataset than CIFAR-10.

During training, 84×84 pixels images are randomly cropped from the 96×96 pixels images, then the images are resized and trained at

Table 2: Inception scores on CIFAR-10 evaluated at 32×32 pixels. Scores are reported in the form of *mean score*$\pm$*std*. In the proposed methods, other scores are reported in the form of *objectness*(*class diversity*).

Model	Scores
Unlabelled	
Infusion training [15]	4.62 ± 0.06
ALI [42] (as reported in [199])	5.34 ± 0.05
BEGAN [13]	5.62
GMAN [44]	6.00 ± 0.19
EGAN-Ent-VI [30]	7.07 ± 0.10
LR-GAN [207]	7.17 ± 0.07
Denoising feature matching [199]	7.72 ± 0.13
Labelled	
SteinGAN [196]	6.35
DCGAN (as reported [196])	6.58
Improved GAN [157]	8.09 ± 0.07
AC-GAN [133]	8.25 ± 0.07
SGAN [75]	8.59 ± 0.12
Proposed methods	
ArtGAN (baseline)	8.21 ± 0.08 33.24 (272.90)
ArtGAN-EB	8.26 ± 0.10 33.51 (276.60)
ArtGAN-AE	8.43 ± 0.09 31.09 (262.04)
ArtGAN-DFM	8.25 ± 0.09 33.34 (274.99)
ArtGAN-M	8.50 ± 0.06 30.19 (256.62)
ArtGAN-D	8.29 ± 0.10 33.30 (276.15)
ArtGAN-AEM	8.53 ± 0.09 **30.07** (256.42)
ArtGAN-AEMT	**8.81** ± 0.14 30.65(269.83)
Real data	11.24 ± 0.12 24.32 (271.76)

(a) ArtGAN (**32** × **32**) (b) ArtGAN-AE (**32** × **32**)

(c) ArtGAN-M (**64** × **64**) (d) ArtGAN-AEM (**64** × **64**)

Figure 27: Generated CIFAR-10 images. From top to bottom: (1) Airplane, (2) Automobile, (3) Bird, (4) Cat, (5) Deer, (6) Dog, (7) Frog, (8) Horse, (9) Ship, (10) Truck.

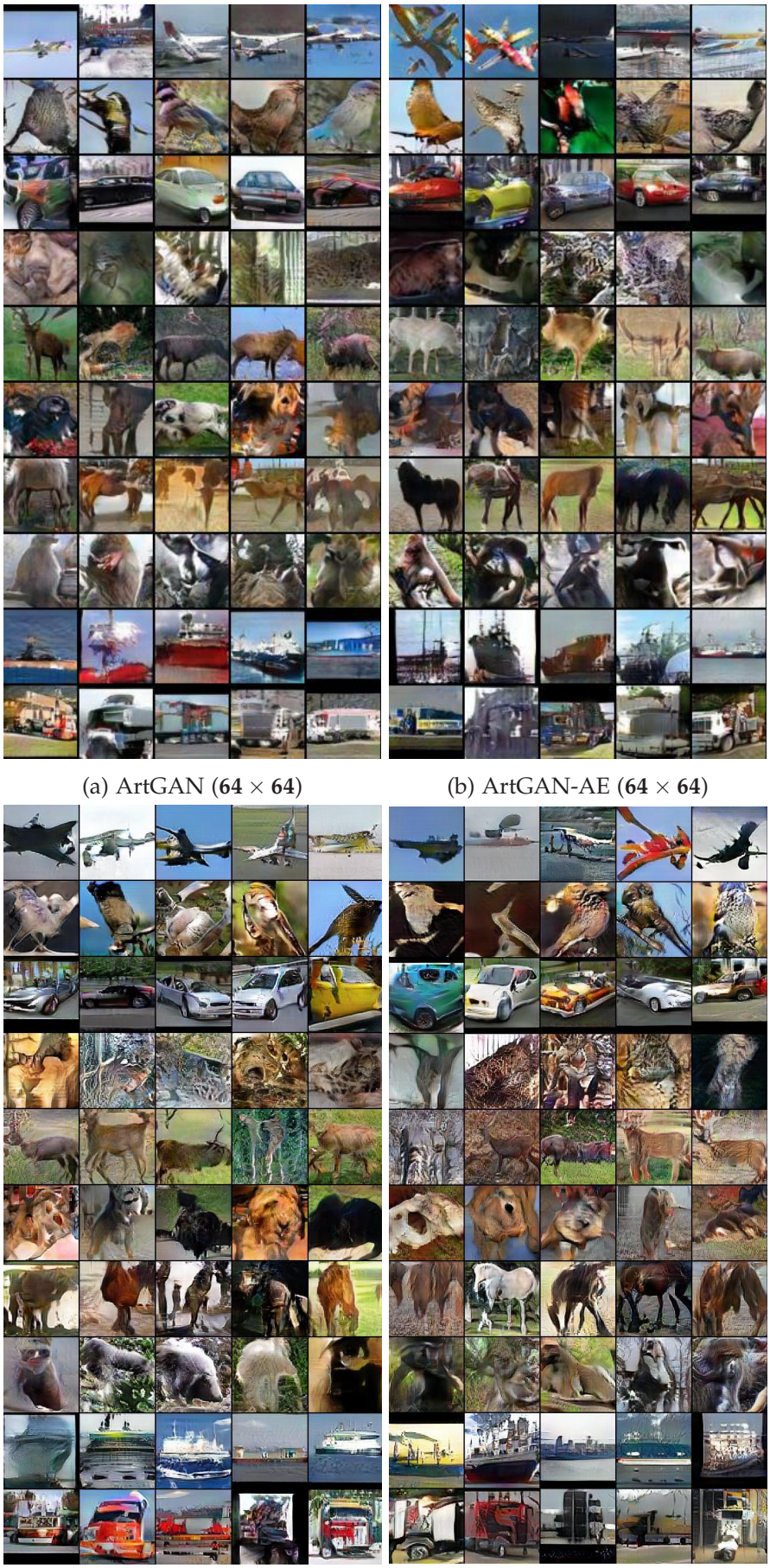
64×64 resolution. Note that different areas are cropped in different iterations. The models trained using magnified learning will generate samples at 128×128 pixels resolution. All models are trained for 50,000 iterations. Similar to CIFAR-10, models are saved every 1,000 iterations and the best models are reported. Generated samples are shown in Figure 28, while the Inception scores are reported in Table 3.

Table 3: Inception scores on STL-10 evaluated at 64×64 pixels. Readers may refer to Table 2 for scores descriptions.

Model	Scores
ArtGAN (baseline)	9.72 ± 0.14 31.03 (301.63)
ArtGAN-EB	9.73 ± 0.12 30.22 (293.89)
ArtGAN-AE	9.65 ± 0.08 31.04 (299.50)
ArtGAN-DFM	9.63 ± 0.09 31.25 (300.89)
ArtGAN-M	$\mathbf{10.12} \pm 0.09$ 29.05 (293.90)
ArtGAN-D	9.87 ± 0.09 31.03 (306.39)
ArtGAN-AEM	10.07 ± 0.09 **28.18**(283.81)
Real data	15.48 ± 0.76 15.04 (232.17)

4.3.5 *Empirical studies and analysis*

This section focuses on evaluating and comparing the performances of the proposed models on CIFAR-10 (Table 2) and STL-10 (Table 3) datasets. Note that the performances are evaluated based on the *objectness* metric, unless specified otherwise. First, the performance of magnified learning can be assessed by comparing the ArtGAN-M and the ArtGAN-D. Note that the ArtGAN-D is not overfit since its performance is similar to the baseline (ArtGAN). In addition, the ArtGAN-D has more number of parameters than the ArtGAN-M. Hence, it is clear that the additional parameter numbers are not the main factor that contribute to the improvement in the ArtGAN-M. These results also show that magnified learning improves the gener-



(a) ArtGAN (**64** × **64**) (b) ArtGAN-AE (**64** × **64**)

(c) ArtGAN-M (**128** × **128**) (d) ArtGAN-AEM (**128** × **128**)

Figure 28: Generated STL-10 images. From top to bottom: (1) Airplane, (2) Bird, (3) Car, (4) Cat, (5) Deer, (6) Dog, (7) Horse, (8) Monkey, (9) Ship, (10) Truck.

ated image quality with significantly better *objectness* and Inception score when compared to the ArtGAN baseline.

On the other hand, the ArtGAN-DFM performed slightly worse than the baseline. In the ArtGAN-DFM, the features fed to the denoiser are extracted from the discriminator which is still in training. Hence, it can be speculated that measuring the loss using these underdeveloped features can cause instability when training the denoiser and generator. Therefore, it is encouraged to compute the losses by leveraging the true data directly. Meanwhile, performance inconsistency can be seen in the ArtGAN-EB, where it performed better on one dataset but worse on the other. This suggests that additional adversarial loss does not always complement a model. This is because the underdeveloped adversarial samples may provide noisy information that can hamper the training process. Hence, extra adversarial loss can aggravate this problem. Without the aforementioned problems, the ArtGAN-AE exhibited more consistent performances with either better or comparable scores. Readers should note that this work tried to train another model using only the Energy-based adversarial loss (traditional adversarial loss is removed). The results showed that this model failed to learn, producing collapsed and meaningless images. This hints that traditional adversarial loss is still a better choice for adversarial training in the settings. Hence, ArtGAN-AE model is used for the rest of the experiments.

Nonetheless, ArtGAN-AEM, which is the ArtGAN-AE with magnified learning, achieved the best results with consistent and significant improvements. Using the trick aforementioned, ArtGAN-AEMT achieved Inception score of **8.81** on CIFAR-10, outperforming SGAN [75] (8.59) and AC-GAN [133] (8.25) and achieves state-of-the-art result. However, it performed worse than ArtGAN-AEM in terms of *objectness* score. This again shows the unreliability of the Inception score. The trick is not used in STL-10 as this work found out that the trick leads to mode collapse in this dataset.

Unlike other datasets such as CUB-200 and Oxford-102 where their classes are meta-classes of a same object category (i.e. different types of birds in CUB-200 and different types of flowers in Oxford-102), CIFAR-10 and STL-10 contain different complex objects. Learning these complex representations is difficult and becomes more challenging when the number of samples are as small as in STL-10. Interestingly, the proposed models are able to produce many samples with high visual fidelity, especially when magnified learning is employed as shown in Figure 27-28. Particularly in CIFAR-10, the details are drawn finer, e.g. cats are more recognizable with better ears shape, most of the frogs are drawn with clear contour, etc. When examining the images at higher resolution (i.e. 64×64 for CIFAR-10 and 128×128 for STL-10), it can be seen that the images are clearer and

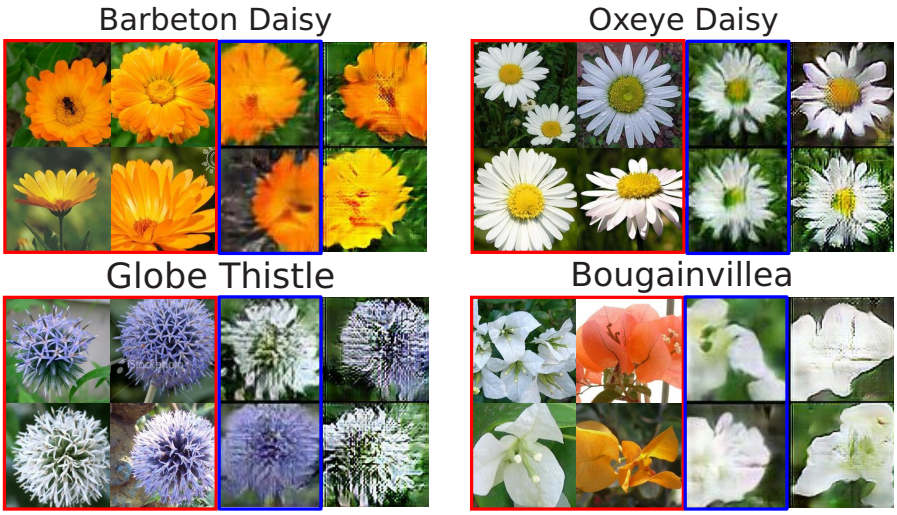


Figure 29: Sample generated images on Oxford-102 flowers. Left (red box): Real samples; Middle (blue box): Generated samples (64×64); Right: Generated samples (128×128).

sharper without much artefacts. No mode collapse is observed in this part of the experiments.

4.3.6 *Oxford-102 flowers*

Oxford-102 [128] consists of 102 flower categories, with around 40 and 258 images in each class. The images have large scale, pose, and light variations and some categories are very similar to each other. The model was trained for 30,000 iterations with learning rate reduced after iteration 15,000. The images were saved at resolution of 256×256 and randomly cropped to 224×224. Then, the cropped images are resized to 64×64 for training.

Two experiments were conducted where the first experiment trained with batch size of 102. In the generator, one sample is drawn for each class during the training. This work found out that the image quality is high but it experienced mode collapse, i.e. generated images look almost the same within same class. In the second experiment, 20 classes are randomly chosen in each iteration and hence, 5 samples are drawn for each class during the training. This solves the mode collapse problem, which suggests that more adversarial images should be sampled for each class in the same iteration to learn more diverse correlations between the latent codes and the image space. Sample generated images can be visualized in Figure 29. Although the discriminator performed poorly on classifying the flower species ($\sim 50\%$ accuracy), the figures show that ArtGAN-AEM is able to generate high quality flower images that look natural with distinctive species-typical features, i.e. color and shape. More samples are presented in Appendix D.

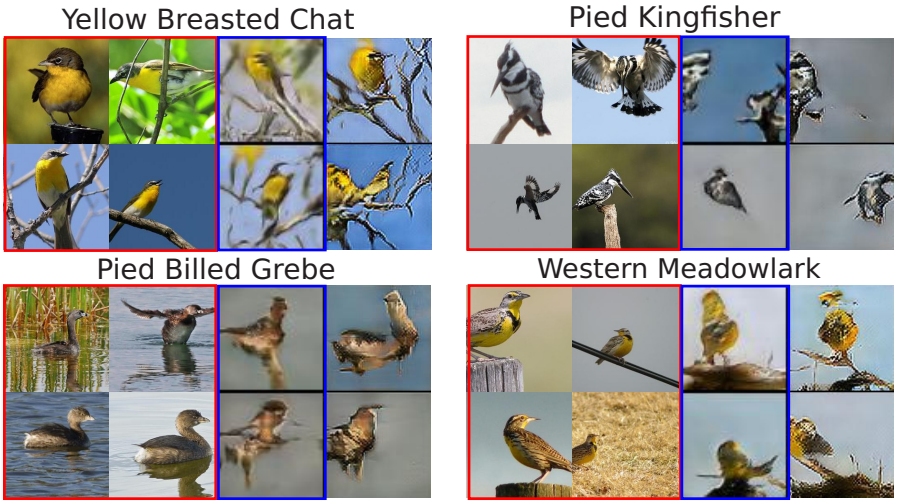


Figure 30: Sample generated images on CUB-200 birds. Left (red box): Real samples; Middle (blue box): Generated samples (64×64); Right: Generated samples (128×128).

4.3.7 *CUB-200*

Caltech-UCSD Birds-200-2011 (CUB-200) [194] is an image dataset with 200 bird species and a total of 11,788 images. It also has the annotations for 15 part locations, 312 binary attributes, and one bounding box per image. However, these attributes are not used in this work as the design of the proposed model is not suitable for multi-label problem in its current state. Hence, the ArtGAN-AEM is trained based on the 200 bird species classes.

The images are pre-processed in the similar way as to the experiments on Oxford-102, i.e. model is trained at 64×64 resolution after cropping and resizing. In order to avoid mode collapse as to experimenting the Oxford-102, the model follows the same settings by randomly choosing 20 classes in each iteration with 5 samples per class. Generated images sample are shown in Figure 30. Similar to the Oxford-102 dataset, the discriminator has a poor performance on bird species classification ($\sim$ 20% accuracy). Interestingly, the figures show that the proposed model is still able to recognize and draw the characteristics of different bird species, i.e. colors, shape, and body size. However, the body structures of the birds are not well learned. More samples can be seen in Appendix D.

4.3.8 *WikiArt*

Wikiart is a fine-art paintings dataset first introduced by Saleh et al. [156]. The paintings were obtained from the wikiart.org website. Currently, Wikiart is the largest public dataset available that contains around 80,000 annotated paintings for genres, artists and styles clas-

sification tasks. However, not all paintings are used in all tasks. To be specific, all paintings are used for 27 *styles* classification. But, there are only 60,000 paintings annotated for 10 *genres*, and only around 20,000 paintings are annotated for 23 *artists*. The same extended version of Wikiart dataset released from previous work in this thesis is used.

The paintings were stored in 256×256 resolution and randomly cropped to 224×224 resolution, then resized to 64×64 resolution for training. Three different ArtGAN-AEM models were trained for different tasks (*styles*, *genres*, and *artists*) for 50,000 iterations. The results are reported using the last updated models (i.e. models at iteration 50,000). In general, ArtGAN-AEM is able to learn artistic representations and generate high quality paintings. Detailed discussions are as follows:

4.3.8.1 *Genre*

Generated paintings based on genre are visualized in Figure 31. Out of the three tasks, categorizing genres is the easiest task [185]. Hence, it is expected that ArtGAN-AEM is able to draw many meaningful paintings based on the genre. For instance, anyone should be able to differentiate *abstract paintings*, *cityscape*, *landscape*, and *portraits* from other classes. The synthesized paintings show that ArtGAN-AEM is able to recognize and draw high quality paintings on these genres. An interesting observation can also be seen in *genre painting*. Not to be confused with "genre", "genre paintings" is the pictorial representation of scenes or events from everyday life, such as markets, parties, etc. Hence, a group of people can usually be seen in this kind of paintings. Figure 31 shows that ArtGAN-AEM is able to draw several human-like figures in a few synthetic paintings. The model may not be able to understand the true meaning of *genre paintings*, but it shows that the model is able to find some similarities in genre paintings at semantic level.

4.3.8.2 *Artist*

Figure 32 shows the synthetic paintings based on artist. Learning visual representations in this task is not impossible as artists usually have their own preferences when deciding what to draw, what kind of styles to use, etc. Hence, many visual similarities can be found between artworks of the same artist. First example can be seen in the paintings of *Nicholas Roerich*. He is a Russian who settled in Himachal Pradesh, India (a mountainous state) for a long time. Hence, many of his famous masterpieces depict the beauty of the mountains with expressive colors and fluid brushwork, as shown in the synthesized paintings. Meanwhile, the figures also clearly show *Gustave Dore*'s primary approach in engraving, etching, and lithography, which result

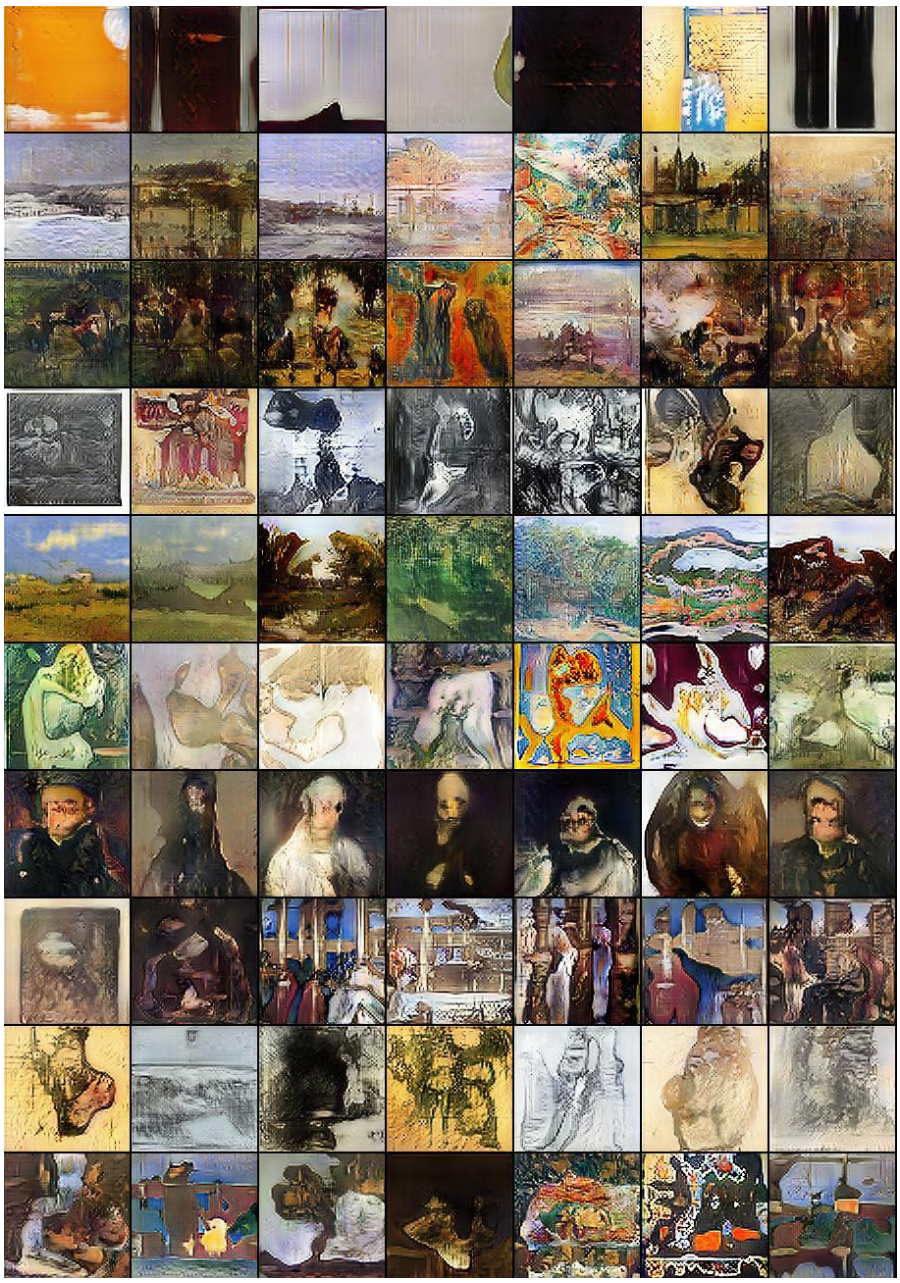


Figure 31: Generated *genres* images at **128** × **128** pixels. From top to bottom: (1) Abstract, (2) Cityscape, (3) Genre painting, (4) Illustration, (5) Landscape, (6) Nude, (7) Portrait, (8) Religious, (9) Sketch and study, (10) Still life.

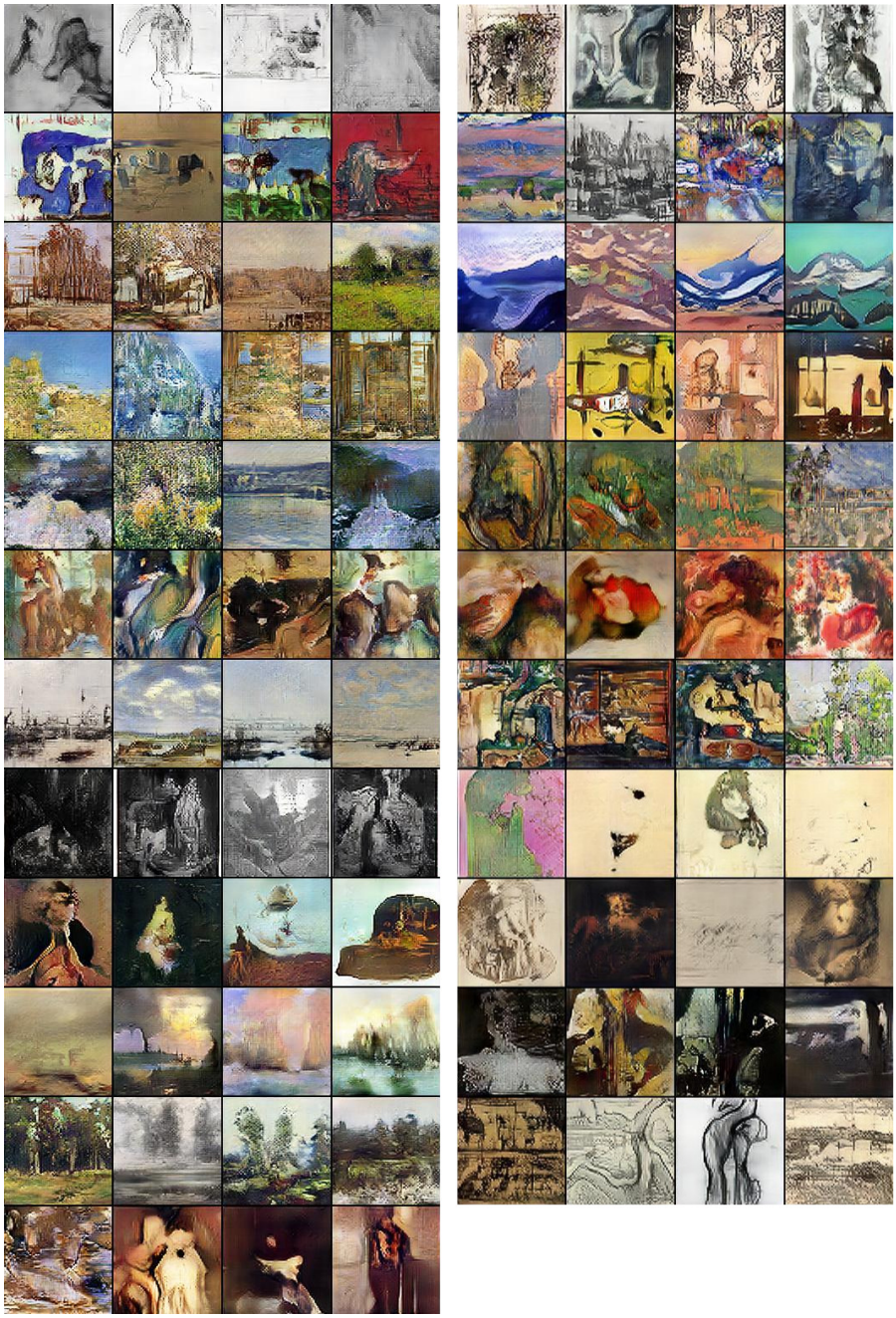


Figure 32: Generated *artists* images at **128** × **128** pixels. From top to bottom: (1) Albrecht Durer, (2) Boris Kustodiev, (3) Camille Pissarro, (4) Childe Hassam , (5) Claude Monet, (6) Edgar Degas, (7) Eugene Boudin, (8) Gustave Dore, (9) Ilya Repin, (10) Ivan Aivazovsky, (11) Ivan Shishkin, (12) John Singer Sargent, (13) Marc Chagall, (14) Martiros Saryan, (15) Nicholas Roerich, (16) Pablo Picasso, (17) Paul Cezanne, (18) Pierre Auguste Renoir, (19) Pyotr Konchalovsky, (20) Raphael Kirchner, (21) Rembrandt, (22) Salvador Dali, (23) Vincent van Gogh.

in greyish artworks. However, the model generated many colourless paintings when conditioned on *Vincent van Gogh*. After some investigations, an interesting fact is found that more than half of his artworks were annotated as *sketch and study* genre. By including all his artworks, most *Van Gogh*s palette consisted mainly of sombre earth tones, particularly dark brown, and showed no sign of the vivid colours that distinguish from his later work, e.g the famous The Starry Night masterpiece. This explains the behaviour of the trained model, but is not desired as the striking colour, emphatic brushwork, and the contoured forms of his work that powerfully influenced the current of Expressionism in modern art is not well-learned by the model. *Eugene Boudin* is a marine painter and has always favoured rendering the sea and along its shores in his artworks. Meanwhile, *Ivan Shishkin* became famous for his forest landscapes. These preferences can be seen in the synthesized paintings.

4.3.8.3 ***Style***

Synthetic paintings based on style is shown in Figure 33. Out of the three task, styles classification is seem to be the most difficult task. In addition to the difficulty in recognizing *Renaissance* art as explained previously, differentiating *Baroque* and *Rococo* is also challenging as they are historically related. They are generally differentiated by the "feelings" they give to their viewers. *Baroque* art often depicts violence, darkness, and the nudes more plump than in *Rococo* works. During mid-1700s, artists gradually moved away from *Baroque* into the modern *Rococo* style. *Rococo* art was often light-hearted, pastoral, and a rosy-tinted view of the world. A subjective observation can be seen in Figure 33 such that *Baroque* synthetic arts are drawn with darker color than the *Rococo* counterparts. This suggests that ArtGAN-AEM do learned some characteristics in these styles through the color intensity. Meanwhile, *Ukiyo-e* is a type of Japanese art flourished from the 17th through 19th centuries. Generally, *Ukiyo-e* is produced using the woodblock printing for mass production and a large portion of these paintings appear to be yellowish due to the paper material. Such characteristic can be seen in the generated Ukiyo-e style paintings.

4.3.9 *Latent space interpolation*

Walking on the manifold of the latent space $\mathbf{z}$ can examines the signs of memorization, i.e. sharp image transitions along the latent space indicates high probability that the model is memorizing the true data space. This is an undesired property as it also implies that the relation between the latent codes and image space is not well learned. Figure 34 shows that the samples generated have smooth semantic changes and look plausible. This indicates that the model is not memorizing and has learned relevant, interesting, and rich visual representations.

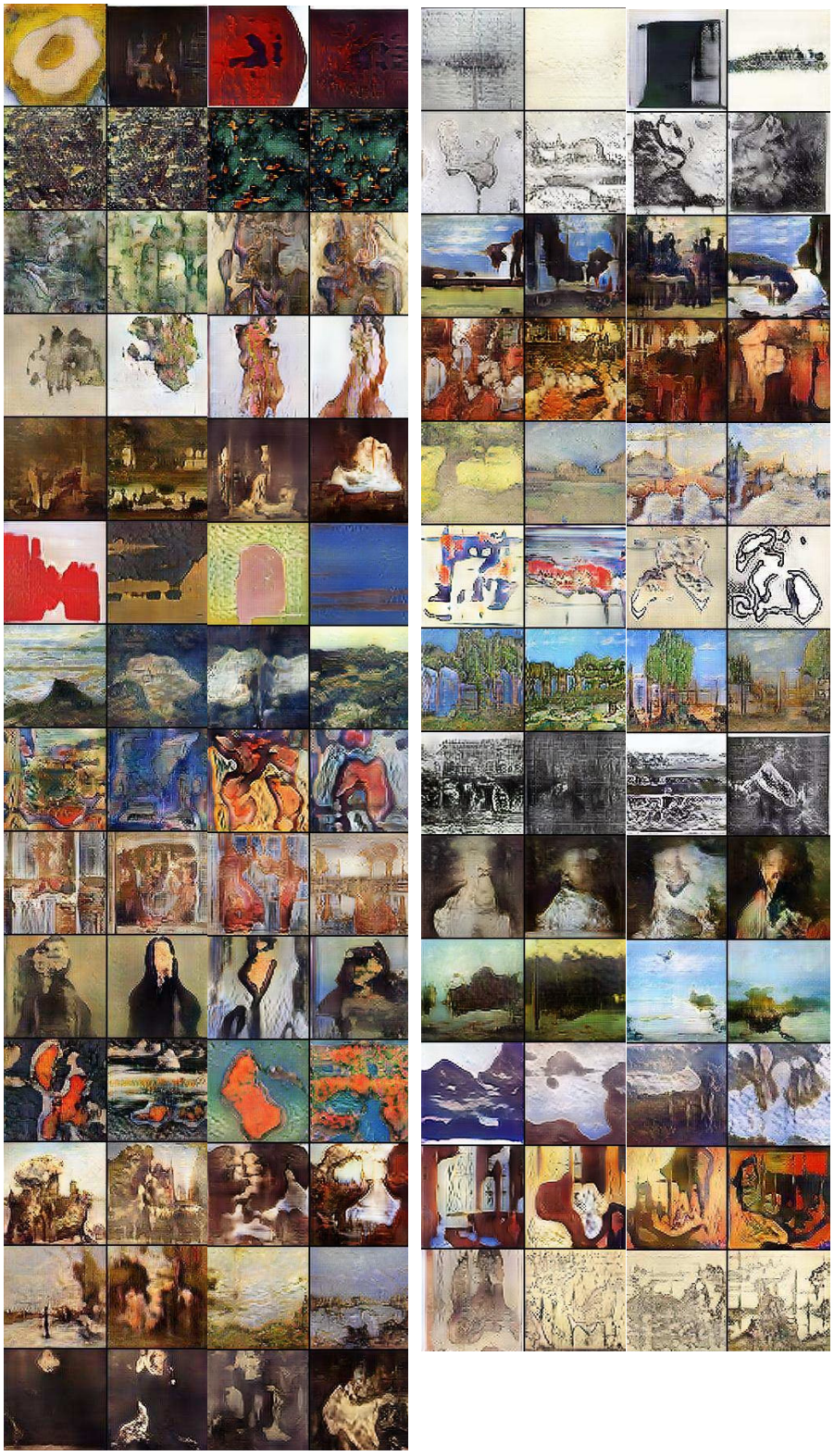


Figure 33: Generated *styles* images at **128** × **128** pixels. From top to bottom, **Left**: (1) Abstract Expressionism, (2) Action painting, (3) Analytical Cubism, (4) Art Nouveau, (5) Baroque, (6) Color Field Painting, (7) Contemporary Realism, (8) Cubism, (9) Early Renaissance, (10) Expressionism, (11) Fauvism, (12) High Renaissance, (13) Impressionism, (14) Mannerism Late Renaissance; **Right**: (15) Minimalism, (16) Naive Art Primitivism, (17) New Realism, (18) Northern Renaissance, (19) Pointillism, (20) Pop Art, (21) Post Impressionism, (22) Realism, (23) Rococo, (24) Romanticism, (25) Symbolism, (26) Synthetic Cubism, (27) Ukiyo-e.

Figure 34: Interpolations over the latent space **z**. Samples show smooth transitions and each image looks plausible.

4.4 CONCLUSIONS

This work proposed a novel GAN variant called **ArtGAN** by leveraging the labels information for better representation learning and image quality. In addition, its extension called ArtGAN-AEM was introduced by employing the proposed categorical autoencoder-based discriminator and magnified learning. Empirical experiments showed that the proposed ArtGAN-AEM is able to achieve state-of-the-art results on CIFAR-10 and STL-10 in terms of Inception score. Furthermore, the figures visualized in this work showed the superiority of the proposed ArtGAN-AEM in generating high quality and plausibly looking images. More importantly, the generated paintings showed that ArtGAN-AEM is able to learn artistic representations from the Wikiart paintings that are usually non-figurative and structured abstract. ArtGAN-AEM is able to generate many high quality fine-art paintings based on the given *style*, *genre*, or *artist*. For future work, the author is looking forward to extend this work for other interest-

ing applications, such as natural to artistic image translation based on a desired semantic-level mode, e.g. *style*.

Part III

COMPRESSION OF DEEP MODELS

This part explores the model compression problem which aims to reduce the network size in order to deploy the trained deep models to real-world devices. More specifically, this work is interested in finding a better way for weights pruning to reduce the number of weights in a model, which are usually redundant.

5 DEEP COMPRESSION MODEL

5.1 INTRODUCTION

As the performance of the CNNs become more and more promising, it is desirable to deploy them on embedded systems for practical use. However, CNN requires intensive memory and computational cost, where many embedded systems may not be able to cope with, specifically mobile devices. For example, the well-known CNN architecture AlexNet [97] has 61 million parameters, and it requires more than 200MB memory space. Meanwhile, the VGG-16 model [163] has 138 million parameters and requires over 500MB memory. Downloading any of these networks into the mobile device from digital distribution platforms (e.g. Google Play or App Store) may be restricted due to the large binary file size[1]. As a result, it is a great challenge to incorporate CNNs into mobile apps.

Inspired by [34, 36] that demonstrated there is a significant redundancy in the parameterization of deep learning models recently, network compression has become a hot topic in the deep learning community. In this sense, it is possible to compress the CNN (i.e. pruning the redundant parameters), while preserving its original accuracy. The goal of this work is to study an efficient strategy to compress the CNN model so that its memory and computational requirement can be reduced significantly, while at the same time preserving its accuracy. Interestingly, research in neuroscience has shown that the development of the mammalian brain involves synaptic over-growth followed by selective pruning [140, 138]. This phenomenon is also found in human brain between early childhood and puberty [76, 77, 78]. Under limited metabolic energy resources restricting the amount and strength of synapses, this phenomenon maximizes the memory performance [20].

Inspired by this phenomenon, the CNN can also be compressed by removing the weak connections in the network. However, determining the strength of the connections is difficult because the weights have variable ranges in different layers, which will be discussed later. While one can solve this problem by trial and error as in Han et al. [60], it is extremely time consuming since the ranges are continuous and hence are infinite sets. This work introduces an alternative solution such that the main idea is to discretize the ranges so that they

1 App Store has the restriction where any apps above 100MB may not be downloaded until the device is connected to Wi-Fi.

are reduced to a finite set. While there are many ways to do so, this work will design the solution from the fuzzy logic perspective.

To this end, this work proposes a novel one-shot compression method based on the fuzzy quantity space [162] to prune the redundant CNN parameters. Similar to Han et al. [60], the procedure starts with learning the connectivity via normal network training. Next, all the weights in the CNN model are fuzzified in the fuzzy quantity space, and re-represented as fuzzy qualitative states. As such, similar weights will be grouped together in the same qualitative states and this simplifies the choice of threshold. The pruning process is performed according to the chosen fuzzy qualitative states. It is important to note that after the pruning process, the pruned weights will be removed permanently. This is different from dropout [170] where connections are only randomly dropped during the training stage to avoid correlations of weights. Finally, the network is retrained to learn the final weights for the remaining sparse connections. The proposed strategy is one-shot retraining, as opposed to [60] that used an iterative process. Three representative CNNs (i.e. LeNet-5, VGGnet, and AlexNet) are used on MNIST [102], CIFAR-10 [95] and ImageNet [153] datasets. Experimental results show that the proposed one-shot fuzzy compression method managed to compress all these CNN representatives up to 14× with a minimal loss of classification accuracy.

5.2 RELATED WORKS

Recently, deep learning has become the-state-of-the-art technique for image classification problems [97, 181, 164] but the model is memory and computational intensive. For example, running a CNN requires a lot of memory bandwidth to fetch the network weights and a lot of computations to perform the dot products that in turn consumes considerable energy. Mobile devices are battery constrained, making power hungry applications such as deep neural networks hard to deploy. Energy consumption is dominated by memory access. Under 45nm CMOS technology, a 32 bit floating point ADD consumes 0.9pJ, a 32bit SRAM cache access takes 5pJ, while a 32bit DRAM memory access takes 640pJ, which is 3 orders of magnitude of an ADD operation. Large networks do not fit in on-chip storage and hence require the more costly DRAM accesses. Running a 1 billion connection neural network, for example, at 20fps would require (20Hz)(1G)(640pJ) = 12.8W just for the DRAM access - well beyond the power envelope of a typical mobile device.

Studies on neural networks can always be related to the biological processes in the brain as neural networks were inspired by the human's central nervous systems. Hence, the overparameterization in neural networks can be seen as a resemblance in the brain development, where the over-growth of synapses occurs [140, 138]. How-

ever, the limited energy resources in the brain restricted the amount of synapses, resulting in the necessity of synapses pruning [20]. In this regard, an interesting study was shown by Chechik et al. [21], where weaker synapses are removed for optimal synaptic modification during massive synaptic pruning. Inspired by this phenomenon, pruning in the neural network has been studied in early 90s to reduce the complexity of the neural network. Early approaches include biased weight decay [61] and shadow array [89]. Optimal Brain Damage [105] and Optimal Brain Surgeon [62] were proposed to prune the networks based on Hessian of loss function for more accurate pruning. However, second-order derivative requires additional computation and calculation of Hessian matrix becomes infeasible when the network is very large.

Meanwhile, other attempts for network compression that are not focusing on weights pruning were introduced. Gong et al. [51] proposed vector quantization to compress the CNN parameters. HashNet [23] is a recent work that implements hashing tricks so that weights are approximated and shared, limiting memory overhead. [59] compressed a CNN model up to 49× without loss of accuracy. This is done by incorporating parameters pruning [60] with network quantization (reducing number of bits required [27]), weight sharing [23] and Huffman coding [192]. The work shows that different type of compression methods complement each other when used together. Data-free parameter pruning [169] was proposed to prune the neurons instead of weights by wiring similar neurons together. The approach achieved 1.5× compression rate with some loss of accuracy. Deep Fried Convnets [210] prunes the network to less than 4× compression rates with no loss of accuracy but only works on fully-connected layers. Collins and Kohli [25] proposed a memory bounded CNN that reduces the memory consumption by a factor of four with minimal accuracy loss.

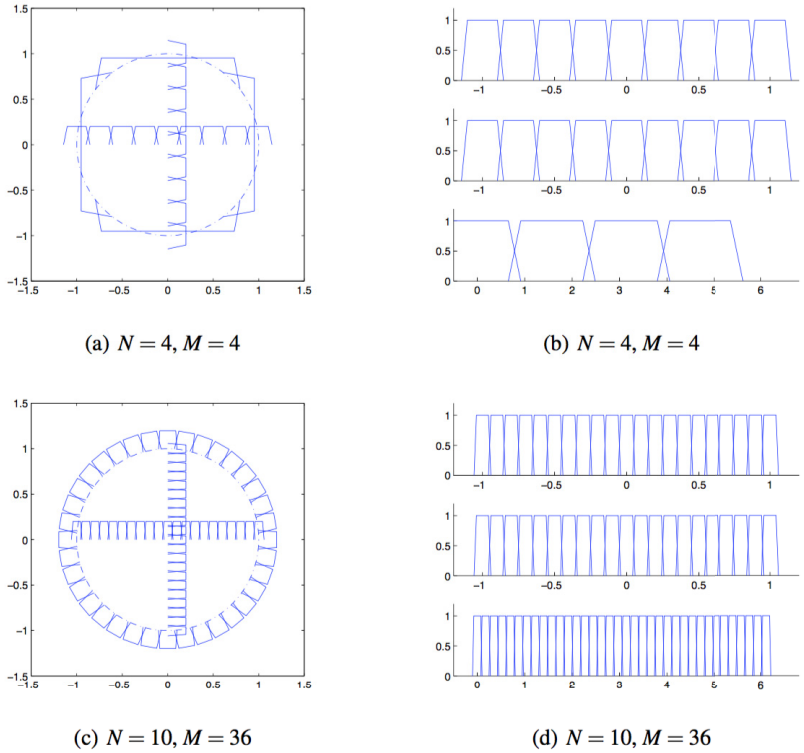
Focusing on weights pruning, a recent and very successful pruning method was proposed by Han et al. [60]. They pruned the CNN representatives using a simple thresholding method, i.e. prune those CNN weights that are lower than a pre-determined threshold. The work achieved a reduction in the network size up to 13× without loss of accuracy. Although they achieved excellent performance with massive weights pruning, a trial-and-error process (i.e. iterative pruning process) is required to find the optimal threshold for pruning. In contrast to their work, this work aims to find a simple yet effective way for parameters pruning without the need of excessive trial-and-error step.

5.3 FUZZY QUANTITY SPACE REVISIT

In general, fuzzy quantity space (FQS) [114] was introduced to replace the conventional Cartesian space into fuzzy qualitative Cartesian space. A FQS is generated by a finite discrimination of the underlying range of each variable of a system being modelled. The FQS will have the desirable properties of finiteness and coverage, as long as the system contains a finite number of variables. Granularity in the FQS is obtained by the arbitrariness of the discrimination of the numeric ranges of system variables that are assumed to be of interest. Hence, a subset of a numeric range can be translated to one qualitative value according to what is needed in a particular modelling process, such that the extensions of a single qualitative intention may be rather different. The adoption of fuzzy subsets has a direct distinct advantage over the traditional crisp representations when considering granularity.

In fact, if one intends to describe the qualitative values of system variables only in terms of the crisp subsets of the underlying real range of the variables, the mapping from the real range to a quantity space will result in the search for the limits of the real numbers served as the boundaries between (dis-jointly) adjacent qualitative values within the quantity space. This usually incurs severe difficulties in determining these limits [162]. The fuzzy representation of qualitative values is more general than ordinary (crisp) interval representations, since it can represent not only the information stated by a well-determined real interval but also the knowledge embedded in the soft boundaries of the interval. Thus, FQS removes, or largely weakens (if not completely resolving), the boundary interpretation problem, achieved through the description of a gradual rather than an abrupt change in the degree of membership of which a physical quantity is mapped onto a particular qualitative value. It is, therefore, closer to the common sense intuition of the description of a qualitative value.

This definition on a FQS is given in a general form such that the operations performed within such a quantity space, consisting of normal and convex fuzzy numbers with arbitrary forms of distribution. As a matter of fact, operations on fuzzy qualitative values are based upon the extension principle outlined in [162]. This principle is invoked every time an arithmetic operation is performed and requires expensive calculation. Also, the computational implementation of the calculation with arbitrary membership distributions of fuzzy numbers can only be done in a discrete domain obtained by sampling the original continuous distribution. The use of the extension principle with sampled membership distributions generates a considerable increase in the discrete samples of the result, and furthermore, only some of the resulting samples are correct. Fortunately, more effi-



(a) $N = 4, M = 4$ (b) $N = 4, M = 4$

(c) $N = 10, M = 36$ (d) $N = 10, M = 36$

Figure 35: Examples of the fuzzy quantity space with different number of components N and M.

cient approaches to characterise fuzzy numbers have been developed. These approaches utilise a parametric approximation of the membership function where the membership distribution of a normal convex fuzzy number is approximated by the 4-tuples, $[a, b, \alpha, \beta]$. The membership function for FQS is then defined as:

$$\mu_A(x) = \begin{cases} 0 & , x < a - \alpha \\ \alpha^{-1}(x - a + \alpha) & , x \in [a - \alpha, a] \\ 1 & , x \in [a, b] \\ \beta^{-1}(b + \beta - x) & , x \in [b, b + \beta] \\ 0 & , x > b + \beta \end{cases} \tag{28}$$

In the recent trends, 4-tuples fuzzy numbers have been utilized in constructing the fuzzy quantity space that endowed with the capability to model the uncertainties. As aforementioned, it is replacing the conventional Cartesian space into fuzzy qualitative Cartesian space and has been contributed in many ways [114, 19, 112]. Here, a brief explanation on the construction of FQS will be provided in terms of its architecture and the advantages. To begin with, lets denote FQS with Q which is composed from an orientation component Q^o and the translation component Q^t,

$$Q = \{Q^o, Q^t\} \tag{29}$$

where

$$Q^o = \{QS_o(\theta_m)\}, \text{where } m = 1, 2, 3, \cdots, M$$
$$Q^t = \{QS_t(l_n)\}, \text{where } n = 1, 2, 3, \cdots, N$$

$QS_o(\theta_m)$ denotes the state of an angle m, $QS_t(l_n)$ denotes the state of a distance l_n , M and N are the number of the elements of the two components. Figure 35 shows some examples of the Q_o and Q_t with different number of respective components. The position measurement of $P(QS_o(\theta_m), QS_t(l_n))$ is determined by both the characteristics of the fuzzy tuple of $QS_o(\theta_m)$ and $QS_t(l_n)$. For example, an origin is represented as $P_0 = (X_0, Y_0) = ([0\,0\,0\,0], [0\,0\,0\,0])$.

5.4 METHODOLOGY

This section describes the implementation of the proposed method. This work employed AlexNet [97], VGG-8, and LeNet-5 [102] to test the proposed method. First, the networks are trained through the standard training procedure as described in Chapter 2.3. Then, *Fuzzy Qualitative Pruning* is performed on these pre-trained deep models to compress the network size. Finally, the networks are retrained in one shot to finetune the remaining weights. Next, the proposed pruning procedure will be described in detail.

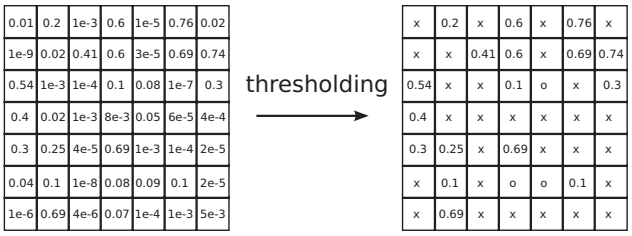
5.4.1 *Fuzzy Qualitative Pruning*

In this phase, redundant weights are pruned in order to compress the network. The idea is to prune small-weight connections because these connections are usually too weak to contribute to the computations. This is similar to the biological phenomenon mentioned in Section 5.2, where weak synapses are removed. Specifically, let $W^l = \{w_1^l, \cdots, w_n^l\}$ be n weights at layer l. The weights will be clustered into k FQSs and assigned with an index $c \in \{1, \cdots, k\}$, such that $n \gg k$. This can be done by checking if the weight w_i^l is in a predefined range of c. However, setting this interval is a difficult task because weights have variable ranges in different layers.

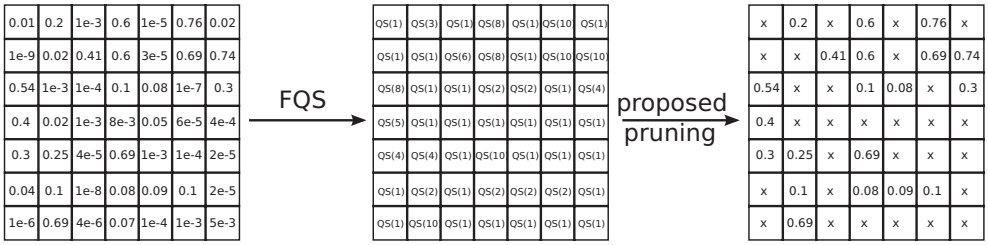
In order to tackle this problem, a simple and effective way is to normalize the weights:

$$\hat{w}_i^l = \frac{|w_i^l|}{\max |W^l|} \tag{30}$$

where $\hat{w}_i^l$ is the normalized weight of i weight and $|\cdot|$ denotes element-wise absolute value. Note that in the convolution layer, the denominator is obtained by maximizing all weights across all feature maps



(a) Classicial solution: Pruning by Weights ($\tau = 0.1$)



(b) The proposed: Fuzzy pruning ($\tau = QS(1)$)

Figure 36: Diagram for two different pruning approaches. The 'x' and 'o' infer the pruned weights. 'o' refers to the differences between the pruning approaches.

in each layer. This allows all weights in the same feature map to be pruned if they are all very small with respect to the maximum weight in the layer. Then, the weight can be assigned with a FQS index according to the following:

$$c_i^l = \begin{cases} j, & \frac{j-1}{k} \leq \hat{w}_i^l < \frac{j}{k}, \forall j \in \{1, \ldots, k\} \\ k, & \hat{w}_i^l = 1 \end{cases} \tag{31}$$

This is equivalent to a 4-tuple representation of $[\frac{j-1}{k}, \frac{j}{k}, 10^{-5}, -10^{-5}]$ for all $\mu_j(\hat{w})$, except $[\frac{k-1}{k}, 1, 10^{-5}, 10^{-5}]$ when $j = k$.

The next step is to prune the weights according to their FQS indices. Specifically, for all weights w_i^l where $c_i^l \leq \tau$, the weights or connections are removed from the network. For example in Figure 36, given the learned weights in a layer with one feature map, the proposed method first represents the weights in FQS, e.g. $0.1 \rightarrow QS(2)$, $0.69 \rightarrow QS(10)$, etc. Then, with $\tau = QS(1)$, all weights in $QS(1)$ are removed from the network and the remaining weights are retained as to their original weight values. Notice that the weights in this example vary from very small value (10^{-9}) to larger value (0.76). Consider when the range is between 10^{-9} and 10^{-2}, pruning by weights with the same setup as illustrated in Figure 36a ($\tau = 0.1$) will result in pruning all weights in the layer. However, this will not happen in the proposed fuzzy pruning strategy because all the weights will be associated in different FQS indices according to the new maximum weight value.

5.4.2 *One-Shot Pruning*

Finally, the network is retrained to learn the final weights for the remaining sparse connections. Here, the iterative pruning [60] is not used but a more aggressive approach is employed by doing one time massive pruning (one-shot pruning). This is because in the iterative pruning process, an iteration consists of a pruning step followed by fine-tuning, where a list of values for τ are chosen in each layer. Then, trade-off curves are drawn based on the τ values after finetuning and the best performing model is kept for the next iteration. This process is repeated until the minimum number of parameters is found with minimal accuracy loss. Although this approach is useful to find the best model with maximal pruning, it is computationally expensive [2].

5.5 EXPERIMENTS AND DISCUSSIONS

Three representative CNN models are used as the baseline architectures in the experiments. AlexNet [97] as described in Chapter 3 is employed on both the ImageNet and Wikiart datasets. Instead of training the model from scratch, the trained model from the Caffe model zoo [84] is utilized. This work also employed LeNet-5 [102] on the MNIST dataset and designed a small VGG-like network [164] on the CIFAR-10 dataset. All experiments were conducted in Ubuntu 14.04 with a Tesla K40 GPU.

5.5.1 *MNIST Dataset*

MNIST is a handwritten digits dataset containing 10 classes, i.e. handwritten digits between 0 and 9. This dataset has 60,000 examples for training and 10,000 examples for validation. The digits have been centered in the 28×28 fixed-size images. In the experiments, the black and white images from the MNIST dataset were size-normalized and anti-aliased, introducing grayscale levels in the images.

LeNet-5 [102] is a small convolutional network with two convolutional layers (conv1 and conv2) and two fully-connected layers (fc3 and fc4) as described in Chapter 2.3. This architecture serves as the baseline and achieved 0.83% error for MNIST. Then, this LeNet-5 is pruned using the proposed method (i.e. a reduction of 14 times) and achieved error rate of 0.84%, which has comparable results to the original network as shown in Table 4. Although the state-of-art approach [60] achieved better performance, the compression rate is higher than [60] ($14\times$ vs. $12\times$). Compensating a small error margin (0.06%) in return for a higher compression rate, the proposed method has shown its effectiveness in the overall performance. Figure 37 shows some of the misclassification examples using the proposed method. It can

2 The work [60] reported a total of 173 hours was used for iterative pruning

Table 4: Results comparison between the original and pruned LeNet-5 [102] in MNIST dataset. Note that the proposed implementation of the LeNet-5 has a slightly different result compared to [60] due to different random seed used.

Deep Architecture	Compression Approach	# weights (in thousand, K)	Compression Rate	Error
LeNet-5 [60]	None	431K	1×	0.80%
LeNet-5 [60]	Deep compression	36K	12×	**0.77%**
LeNet-5 (Ours)	None (baseline)	431K	1×	0.83%
LeNet-5 (Ours)	Fuzzy Pruning	**30K**	**14**×	0.84%

Table 5: Number of parameters (in thousand, K) retained in LeNet-5 for MNIST after the proposed fuzzy pruning strategy.

Layer	# weights (original)	τ	# weight retained (%)
conv1	0.5K	5	66.2
conv2	25K	10	20.0
fc3	400K	25	5.32
fc4	5K	10	70.1
Total	431K	-	6.99

Figure 37: Examples of classification results in MNIST dataset. (Top) Correctly predicted classes (0-9 from left to right). (Bottom) Misclassified examples. The true classes are [5, 6, 7, 5, 9, 9, 5, 2, 3, 8], respectively.

be seen that those misclassifications indeed are very fuzzy, and even human might have difficulty to identify them correctly. For example, the "3" that the proposed system misclassified as "5", the "5" $\rightarrow$ "3", "6" $\rightarrow$ "0" and "4" $\rightarrow$ "9". The number of parameters pruned in each layer is shown in Table 5.

5.5.2 *CIFAR-10 Dataset*

CIFAR-10 is a dataset with 32×32 pixels colour images. It contains 60,000 images in 10 classes, with 6,000 images per classes. During the experiment, the dataset is split into 50,000 training images and 10,000 validation images, randomly. In this experiment, in order to avoid network overfitting, data augmentation (i.e. horizontal reflection) that

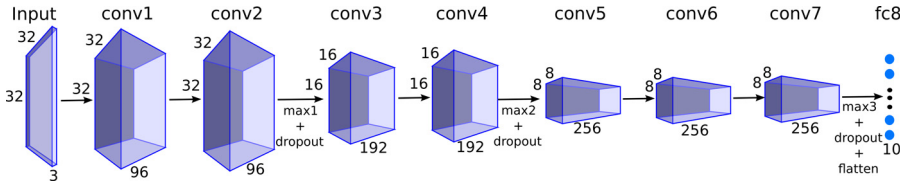


Figure 38: An illustration of the VGG-8 architecture.

Table 6: Results comparison between the original and pruned VGG-8 in CIFAR-10 dataset. (In bracket) for the compression rate column dedicates compared to.

Deep Architecture	Compression Approach	# weights (in Million, M)	Compression Rate	Accuracy
VGG-8	None (baseline)	2.2M	1×	91.0%
VGG-9	None	2.5M	1×	90.9%
VGG-10	None	2.6M	1×	90.7%
AllCNN [168]	None	43M	1×	95.6%
Fractional Max-Pooling [54]	None	79M	1×	**96.5%**
VGG-8 (Ours)	Fuzzy Pruning	**0.44M**	**5**× (baseline) **97**× ([168]) **180**× ([54])	90.7%

randomly flips the images horizontally during training is employed; however data augmentation is not performed during the validation stage.

This work implemented a small VGG-like network for CIFAR-10, which has 8 layers with 7 convolutional layers (conv1-7) and a fully-connected layer (fc8). For convenience, this network is named as VGG-8. Similar to the VGGnet [164], the kernel size used in all convolutional layers are 3×3. Meanwhile, the numbers of feature maps are set to 96, 192, and 256 for conv1-2, conv3-4, and conv4-7, respectively. Downsampling is performed using max-pooling after conv2, conv4, and conv7. No LRN is used in this architecture. In addition, dropout is performed after each pooling layers for regularization. Figure 38 illustrates the near-optimal VGG-8 architecture. As shown in Table 6 (VGG-8 vs. VGG-9 and VGG-10), it is found that more layers degrade the performance because VGG-like architecture overfits easily as it goes deeper due to overparametrization [34].

As shown in Table 6, VGG-8 achieved accuracy of 91.0% in the CIFAR-10, which is comparable to the state-of-the-art results [168, 54] that have 20× to 40× higher parameters. The proposed method is able to compress the VGG-8 to only 440K parameters (5× compression compared to the baseline) and achieved similar results to the original VGG-8 (90.7%). Compared to [168, 54], the proposed model is poorer by $\sim$ 5% but the model is much compressed (97× to 180×) to be deployed in mobile platform. The number of parameter of the

Table 7: Number of parameters (in thousand, K) retained for VGG-8 in CIFAR-10 after the proposed fuzzy pruning strategy.

Layer	# weights (original)	τ	# weight retained (%)
conv1	2.6K	5	71.6
conv2	83K	10	26.5
conv3	166K	15	41.2
conv4	332K	15	25.3
conv5	442K	20	15.8
conv6	590K	20	7.7
conv7	590K	20	20.6
fc8	41K	20	33.1
Total	2200K	-	19.0

original network and the number of parameters pruned is recorded in Table 7.

5.5.3 *ImageNet Dataset*

Further evaluations are conducted to examine the performance of the proposed fuzzy pruning strategy with AlexNet [97] architecture on the ImageNet ILSVRC-2012 dataset. This dataset has 1.2 million training examples and 50,000 validation examples. Images in the ImageNet consist of variable resolutions but AlexNet requires fixed input dimensionality. Therefore, the images are resized to a fixed resolution of 256×256 pixels. This is done by first, rescaling the shorter side of the image to length of 256. Then, the central 256×256 patch is cropped from the resulting image. The images were pre-processed by subtracting the mean activity over the training set from each pixel.

Following the original work [97], data augmentation is performed to avoid overfitting. The first form of data augmentation is translation, which can be done by randomly cropping 227×227 patch from the image. During validation, the center patch is cropped from each image for evaluation. The second form of data augmentation is horizontal reflection. The images are randomly flipped horizontally during training, but did not perform this step during validation. Contrast to Krizhevsky et. al [97], the intensities of the RGB channels are not altered as a form of data augmentation.

The validation performance of the original AlexNet for Top-1 accuracy is 57.1% and for Top-5 accuracy is 80.2%. The proposed method managed to prune this network by a factor of 11 using the proposed method and obtained Top-1 accuracy of 56.1% and Top-5 accuracy of 79.2%, which are comparable to the original network's performance. Table 8 summarizes and compares the results along with the best re-

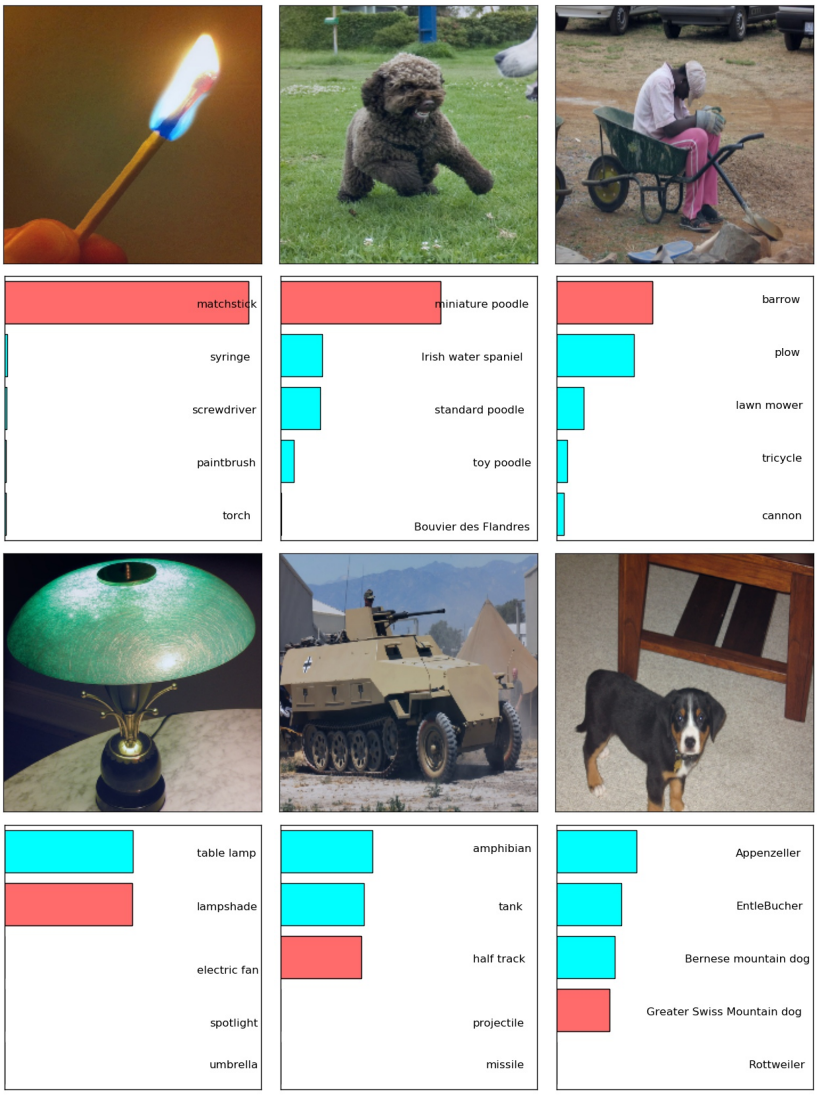


Figure 39: Top-5 predicted classes of the classification model for six sample images. The red bar implies the ground truth class of the image.

sults in the state-of-the-art. Qualitatively, the Top-5 predicted classes of the proposed method are showed in Figure 39. This figure illustrates some predictions, suggesting that the model may account for data variations in a different modality to a certain extent. Table 9 shows the number of parameters pruned in each layer.

5.5.4 *Wikiart Paintings Dataset*

Last but not least, this work examines the performance of the proposed fuzzy pruning strategy on the Wikiart Paintings dataset. In this part of the work, the models trained in Chapter 3 are employed and pruned. Similarly, the tasks are to classify style, genre, or artist of a painting. The results for the paintings classification are summa-

Table 8: Results in ImageNet dataset with state-of-the-art methods.

Deep Architecture	Compression Approach	Accuracy		# weights (in Million, M)	Compression
		Top-1	Top-5		
AlexNet [97]	None (baseline)	57.1%	80.2%	61M	-
AlexNet	Deep compression [60]	**57.2%**	**80.3%**	6.7M	9×
AlexNet (Ours)	Fuzzy Pruning	56.1%	79.2%	**5.3M**	**11**×

Table 9: Number of parameters (in Million, M) retained in AlexNet for ImageNet after Fuzzy Pruning.

Layer	# weights (original)	τ	# weight retained (%)
conv1	0.035M	1	88.4
conv2	0.307M	3	39.6
conv3	0.885M	3	32.8
conv4	0.663M	4	28.1
conv5	0.442M	5	33.2
fc6	38M	20	7.8
fc7	17M	20	2.6
fc8	4M	15	28.1
Total	61M	-	8.7

rized in Table 10 along with the state-of-the-art results. It can be seen that the pruned models (average of 68.1% accuracy over three tasks) outperformed Saleh and Elgammal [155] (average of 56.4% accuracy over three tasks) in all aspects. Meanwhile, the pruned models also achieved comparable results to the original AlexNet architecture (average of 68.2% accuracy over three tasks). That is, the averaged accuracies have only 0.1% difference with 12× to 13× reduction in number of weights. The pruning details are listed in Table 11.

Table 10: Results in Wikiart paintings dataset along with state-of-the-art methods.

Task	Network	Accuracy	# weights (in Million, M)	Compression
Styles	Original	54.5%	61M	-
	Fuzzy Pruning	**54.5%**	**4.9M**	**12**×
	Saleh and Elgammal [155]	45.9%	-	-
Genres	Original	**74.1%**	61M	-
	Fuzzy Pruning	74.0%	**4.5M**	**13**×
	Saleh and Elgammal [155]	60.3%	-	-
Artists	Original	**76.1%**	61M	-
	Fuzzy Pruning	75.7%	**4.4M**	**13**×
	Saleh and Elgammal [155]	63.0%	-	-

Table 11: Number of parameters retained in AlexNet for Wikiart paintings dataset after Fuzzy Pruning. The same configurations for threshold τ is used for all tasks.

Layer	# weights (original) (in Million, M)	τ	# of weights retained (%)		
			Styles	Genres	Artists
conv1	0.035M	1	88.4	88.7	88.7
conv2	0.307M	3	39.1	39.8	39.9
conv3	0.885M	3	32.2	33.6	33.5
conv4	0.663M	4	28.4	28.8	30.8
conv5	0.442M	5	32.5	33.9	34.4
fc6	38M	20	8.5	7.5	7.2
fc7	17M	20	3.4	3.3	3.5
fc8	4M	15	42.4	53.1	36.8
Total	61M	-	8.0	7.4	7.3

Table 12: Comparison of results on Wikiarts dataset for styles, genres, artists classification by reducing parameters in fc7.

Model	Accuracy (%)				# weights (in Million, M)
	Style	Genre	Artist	Overall	
AlexNet	**54.50**	**74.14**	**76.11**	**68.25**	61M
AlexNet-1024	53.38	73.75	76.02	67.72	48M
AlexNet-fc6	51.51	72.11	74.26	65.96	**44M**

5.5.5 *Discussions*

One important finding from all these experiments is that, most of the parameters in the lower layer of the CNN are retained. For example, in the AlexNet architecture, less than 12% of the parameters are pruned in conv1. While, more parameters are pruned at the higher layers, e.g. more than 90% of the parameters in fc6 and fc7 are pruned in the AlexNet. This corresponds to the experiments reported in [60], where the trade-off between accuracy and number of parameters pruned has worse disadvantages at the lower layers. To further support this finding, two additional experiments were carried out using the AlexNet on paintings classification tasks. First, fc7 in the AlexNet is reduced to 1024 neurons, compared to the original 4096 neurons. Then, it is finetuned and the results are reported in Table 12 as AlexNet-1024. Similarly, for the second experiment, fc7 is totally removed, then finetuned. This model is denoted as AlexNet-fc6 in the table. The results show that the network size is reduced by up to 17 million parameters (27% reduction) without affecting the accuracy significantly. This implies that most of the redundant param-

eters are concentrated in the fully-connected layers. Table 5, 7, 9 and 11 also show this phenomenon where only a very small amount of weights in the fully-connected layers are retained compared to other lower layers. Similar conclusion was drawn by Lin et al. [113], where they substituted the fully-connected layers by global averaging pooling and achieved better accuracy.

5.6 CONCLUSIONS

This work aims to reduce the memory requirement of the CNN and propose a one-shot fuzzy qualitative deep compression method to prune the redundant parameters. This is because research has found out that deep models are typically filled with redundant parameters. This results in unnecessary large memory requirement and lead to great bottleneck to implement the CNN in real-world embedded systems, such as mobile devices. The proposed approach first clusters similar weights into an associated FQS. Then, the redundant connections are removed by performing thresholding based on the weighted FQS indices. This avoids the variable range problem when performing thresholding based on the weight values directly. Experimental results have shown that the proposed approach is able to prune the CNN up to 14 times with a minimal loss of accuracy. For future work, the author is interested in developing better pruning strategy.

Part IV

CONCLUSIONS

6

CONCLUSIONS AND FUTURE WORKS

6.1 CONCLUSIONS

As a summary, this thesis explored two main goals. In the first part, the features learning of deep models for artworks is studied. While in the second part, an alternative method to improve network compression is investigated. The summary of the contributions of this thesis are as follow:

Features Learning of Deep Models for Artworks

1. Several CNN networks are trained under different configurations for artworks classification of *style*, *genre*, and *artits*. It is found out that finetuning the Imagenet pre-trained network achieves the best result and outperforms the state-of-the-art results. Then, the averaged neurons' responses are visualized in order to understand the features learned by the network. It is found that deep models are able to learn the structural cues from the artworks. However, it is unclear if the networks are able to learn the more abstract cues. Meanwhile, training deep models for artworks-related tasks is extremely challenging because of two contradictory reasons: 1) The visualizations showed that features extracted from artworks of same category may vary a lot; 2) Many categories are defined with only subtle difference, hence they have very similar visual properties.

2. Next, a novel GAN variant called ArtGAN is proposed to learn and visualize the features of the artworks via generative approach. The advantages of generative approach are that the visualization is done by generating photo-realistic images and all relevant features can be visualized in one image. The contributions of this work are three-fold: 1) ArtGAN leverages the labels information by backpropagating the information through derivative of the loss function, improving image quality and fidelity; 2) Inspired by recent works, autoencoder is employed in the ArtGAN for additional energy-based complementary information, further improving the performance; 3) Most importantly, a novel magnified learning approach is proposed to learn the correlations between neighbouring pixels better. In addition, it also allows ArtGAN to generate images at higher resolution while training with samples with lower resolution. Combining all the proposed modifications, ArtGAN achieves state-of-the-art results on CIFAR-10 and STL-10 datasets in terms of Incep-

tion score. The generated images showed that the proposed ArtGAN is able to generate images of high quality and fidelity, hence rich visual features are learned. Furthermore, through the synthetic artworks, it seems that deep models are able to learn more abstract cues via visual properties.

Deep Compression Network

This work proposed a one-shot fuzzy qualitative deep compression method to prune the redundant weights in a network. The experiments have shown that the proposed method is able to prune the networks up to 14 times with minimal loss of classification accuracy.

6.2 FUTURE WORKS

With the introduction of GAN, generative models have shown excellent performance in learning rich representations and synthesizing photo-realistic images. Hence, the author is interested in pursuing this direction for future works. One problem with generative model is that there is still no good way to evaluate it. Hence, developing a good evaluation metric for generative model is still an ongoing work. Meanwhile, better understanding of the GAN model and its stability are also important topics for future works. Application wise, the author is interested in developing algorithms for various artwork synthesis problems, e.g. semantic-level style transfer. Finally, the author is also interested in finding a way to train and compress the network at the same time, instead of a two-phase approach as discussed. This will reduce the computational cost, while aiming to minimize the network size and performance loss.

Part V

APPENDICES

A

EXPLANATIONS FOR MODIFICATION IN ARTGAN

In the original formulation, the discriminator D in ArtGAN (Chapter 4) outputs $K+1$ predictions. These include K actual categories and the adversarial class (Fake category) for $K+1$th output. However, Salimans et al. [157] suggested that it is over-parameterised. In their work, they briefly explained their solution to this problem, which avoids the need of learning weights for $K+1$th output. This appendix will explain the solution in detail, mathematically.

Let D outputs $K+1$ predictions. Then, the probability distribution function for each output c_k conditioned on an input image x is given,

$$p(c_k|x) = \frac{e^{l_k(x)}}{\sum_{i=0}^{K+1} e^{l_i(x)}} \tag{32}$$

where $l_k(x) \in D(x)$ is the output of $D(x)$ at class k without activation function. Here, it is desired to remove $l_{K+1}(x)$ from the network. This can be done by first, let $p(y|x)$ be the probability distribution function of the adversarial prediction, such that x is real when $y = 1$. Hence, $p(y|x)$ is the inverse of $p(c_{k+1}|x)$ and is equivalent to the joint distribution $p(c_{1,\dots,K}|x)$:

$$p(y|x) \equiv 1 - p(c_{k+1}|x) \equiv p(c_{1,\dots,K}|x) = \frac{\sum_{i=0}^{K} e^{l_i(x)}}{\sum_{j=0}^{K+1} e^{l_j(x)}} \tag{33}$$

Then, equation 33 can be reformulated by dividing the numerator and denominator by $e^{l_{K+1}(x)}$:

$$\begin{aligned}
p(y|x) &= \frac{\sum_{i=0}^{K} e^{l_i(x)}}{\sum_{j=0}^{K+1} e^{l_j(x)}} \\
&= \frac{\sum_{i=0}^{K} e^{l_i(x)} / e^{l_{K+1}(x)}}{\sum_{j=0}^{K+1} e^{l_j(x)} / e^{l_{K+1}(x)}} \\
&= \frac{\sum_{i=0}^{K} e^{l_i(x) - l_{K+1}(x)}}{\sum_{j=0}^{K+1} e^{l_j(x) - l_{K+1}(x)}} \\
&= \frac{\sum_{i=0}^{K} e^{l_i(x) - l_{K+1}(x)}}{\sum_{i=0}^{K} e^{l_i(x) - l_{K+1}(x)} + e^{l_{K+1}(x) - l_{K+1}(x)}} \\
&= \frac{\sum_{i=0}^{K} e^{l_i(x) - l_{K+1}(x)}}{\sum_{i=0}^{K} e^{l_i(x) - l_{K+1}(x)} + 1}
\end{aligned} \tag{34}$$

Assuming that $l_{K+1}(x)$ is removed, it becomes a constant and will not affect the outcome of equation 34. Hence, the network can be trained

by setting $l_i(x) \leftarrow l_i(x) - l_{K+1}(x)$. Equation 34 can then be rewritten as:

$$\begin{aligned} p(y|x) &= \frac{\sum_{i=0}^{K} e^{l_i(x)}}{\sum_{i=0}^{K} e^{l_i(x)} + 1} \\ &= \frac{Z(x)}{Z(x) + 1} \end{aligned}$$

where $Z(x) = \sum_{i=0}^{K} e^{l_i(x)}$. This proves that the adversarial prediction can be calculated only from the K actual categories, indicates that $l_{K+1}(x)$ is redundant. Using similar derivation, equation 32 is then reformulated as

$$p(c_k|x) = \frac{e^{l_k(x)}}{\sum_{i=0}^{K} e^{l_i(x)} + 1} \tag{35}$$

when calculating the probability distribution function for other categories. In practice, GAN variants can also be trained by calculating the classification and adversarial loss functions separately [167, 133]. Hence, the standard probability distribution function for classification task can be used instead:

$$p(c_k|x) = \frac{e^{l_k(x)}}{\sum_{i=0}^{K} e^{l_i(x)}} \tag{36}$$

B

WIKIART DATASET

Table 13 details the members in each of the annotation class in the Wikiart dataset. Figure 40 and 41 show one sample painting for each genre and style, respectively.

	List of Members		
Style	(1) Abstract Expressionism	(2) Action Painting	(3) Analytical Cubism
	(4) Art Nouveau-Modern Art	(5) Baroque	(6) Colour Field Painting
	(7) Contemporary Realism	(8) Cubism	(9) Early Renaissance
	(10) Expressionism	(11) Fauvism	(12) High Renaissance
	(13) Impressionism	(14) Mannerism-Late-Renaissance	(15) Minimalism
	(16) Primitivism-Naive Art	(17)New Realism	(18) Northern Renaissance
	(19) Pointillism	(20) Pop Art	(21) Post Impressionism
	(22) Realism	(23) Rococo	(24) Romanticism
	(25) Symbolism	(26) Synthetic Cubism	(27) Ukiyo-e
Genre	(1) Abstract Painting	(2) Cityscape	(3) Genre Painting
	(4) Illustration	(5) Landscape	(6) Nude Painting
	(7) Portrait	(8) Religious Painting	(9) Sketch and Study
	(10) Still Life		
Artist	(1) Albrecht Durer	(2) Boris Kustodiev	(3) Camille Pissarro
	(4) Childe Hassam	(5) Claude Monet	(6) Edgar Degas
	(7) Eugene Boudin	(8) Gustave Dore	(9) Ilya Repin
	(10) Ivan Aivazovsky	(11) Ivan Shishkin	(12) John Singer Sargent
	(13) Marc Chagall	(14) Martiros Saryan	(15) Nicholas Roerich
	(16) Pablo Picasso	(17) Paul Cezanne	(18) Pierre-Auguste Renoir
	(19) Pyotr Konchalovsky	(20) Raphael Kirchner	(21) Rembrandt
	(22) Salvador Dali	(23) Vincent van Gogh	

Table 13: List of *Style*, *Genre*, and *Artist* in the Wikiart Dataset

(a) Abstract Painting (b) Cityscape (c) Genre Painting (d) Illustration (e) Landscape

(f) Nude Painting (g) Portrait (h) Religious Painting (i) Sketch and Study (j) Still Life

Figure 40: Examples of artworks in the Wikiart dataset according to *genre*. In total, there are 10 *genre* categories.

Figure 41: Examples of the artworks in the Wikiart dataset according to *style*. In total, there are 27 *style* categories.

C

OTHER DETAILS OF ARTGAN

This appendix provides additional details which are not covered in the main context. First, the pseudocode that is used to train the ArtGAN is presented. Then, the detailed model configurations of the Generator and Discriminator used in this thesis are listed to facilitate future reimplementation.

C.1 ALGORITHM

Algorithm 3 illustrates the training process in the ArtGAN models. The notations are consistent with the thesis. In addition, $\mathbf{K} = \{1, \ldots, K\}$ is denoted as the set of indices of the classes. Then, the one-hot vector of a sample $\bar{c}_k$ is randomly sampled, where $k \in \mathbf{K}$ and value at position k is set to one while the rest of the elements are set to zero. Given n samples in a minibatch, $\mathbf{y} = \{y_1, \ldots, y_n\}$ is a vector of the computed adversarial outputs. While, $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ is a set of class prediction.

Algorithm 3: Pseudocode for training ArtGAN

Require: Minibatch size, n, learning rate, λ, and $\mathbf{z}$ vector size, d
Require: Randomly initialize θ_D and θ_G
while condition not met **do**
  Sample $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n] \sim \mathcal{N}(0, 1)^{n \times d}$
  Randomly set $\bar{\mathbf{C}} = [\bar{c}_{k_1}, \ldots, \bar{c}_{k_n}]$
  Sample minibatch $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n]$
  $\mathbf{C}, \mathbf{y} = D(\hat{\mathbf{X}})$
  $\bar{\mathbf{X}} = G(\mathbf{Z}, \bar{\mathbf{C}})$
  $\mathbf{C}', \mathbf{y}' = D(\bar{\mathbf{X}})$
  if use magnified learning **then**
    $\mathbf{R} = Dec(\Phi(\hat{\mathbf{X}}))$
    $\mathbf{R}' = Dec(\Phi(\bar{\mathbf{X}}))$
    $\theta_D = \theta_D - \lambda \frac{\partial \mathcal{L}_{Dae}}{\partial \theta_D}$, $\mathcal{L}_{Dae} \leftarrow \mathbf{y}, \mathbf{C}, \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}', \mathbf{R}$
    $\theta_G = \theta_G - \lambda \frac{\partial \mathcal{L}_{Gae}}{\partial \theta_G}$, $\mathcal{L}_{Gae} \leftarrow \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}', \mathbf{R}'$
  else
    $\theta_D = \theta_D - \lambda \frac{\partial \mathcal{L}_D}{\partial \theta_D}$, $\mathcal{L}_D \leftarrow \mathbf{y}, \mathbf{C}, \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}'$
    $\theta_G = \theta_G - \lambda \frac{\partial \mathcal{L}_G}{\partial \theta_G}$, $\mathcal{L}_G \leftarrow \bar{\mathbf{C}}, \mathbf{C}', \mathbf{y}'$
  end if
end while

C.2 NETWORK ARCHITECTURES

1. *CIFAR-10*: This section describes the network architectures used on CIFAR-10. Table 14 shows the architecture for the discriminator. Table 15 shows the architectures for the generators.

2. *STL-10*: Table 16 and Table 17 show the network architectures of the discriminator and generator used on STL-10.

3. *Wikiart*: All tasks in Wikiart (i.e. genres, styles, and artists) used the same architectures described in Table 18 and Table 19.

4. *Oxford-102 flowers* and *CUB-200 birds*: Oxford-102 and CUB-200 datasets share the same network architectures as described in Table 20 and Table 21.

Table 14: Network architectures of the discriminator used on CIFAR-10, which contains a classifier and a decoder.

Classifier	Decoder
conv(96, 3, 1)	convBN(256, 3, 1)
convBN(96, 3, 2)	
Dropout(0.2)	
convBN(192, 3, 1)	NNupsample(16)
convBN(192, 3, 2)	convBN(128, 3, 1)
Dropout(0.2)	convBN(128, 3, 1)
convBN(256, 3, 1)	NNupsample(32)
convBN(256, 1, 1)	convBN(64, 3, 1)
convBN(512, 1, 1)	conv(3, 3, 1)
fc(10)	

Table 15: Network architectures for the generators (with and without magnified learning) used on CIFAR-10.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$)	fcBN($512 \times 4 \times 4$)
NNupsample(8)	NNupsample(8)
convBN(256, 3, 1)	convBN(256, 3, 1)
NNupsample(16)	NNupsample(16)
convBN(128, 3, 1)	convBN(128, 3, 1)
convBN(128, 3, 1)	convBN(128, 3, 1)
NNupsample(32)	NNupsample(32)
convBN(64, 3, 1)	convBN(64, 3, 1)
convBN(3, 3, 1)	convBN(64, 3, 1)
	NNupsample(64)
	convBN(32, 3, 1)
	conv(3, 3, 1)

Table 16: Network architectures for discriminator (containing a classifier and a decoder) used on STL-10.

Classifier	Decoder
conv(64, 3, 1)	convBN(512, 3, 1)
convBN(64, 3, 2)	
Dropout(0.2)	
convBN(128, 3, 1)	NNupsample(16)
convBN(128, 3, 2)	convBN(256, 3, 1)
Dropout(0.2)	
convBN(256, 3, 1)	NNupsample(32)
convBN(256, 3, 2)	convBN(128, 3, 1)
Dropout(0.2)	convBN(128, 3, 1)
convBN(512, 3, 1)	NNupsample(64)
convBN(512, 3, 1)	convBN(64, 3, 1)
	conv(3, 3, 1)
fc(10)	

Table 17: Network architectures for the generators (with and without magnified learning) used on STL-10.

Generator	Generator with Magnified Learning
fcBN(512 × 4 × 4)	fcBN(512 × 4 × 4)
NNupsample(8)	NNupsample(8)
convBN(512, 3, 1)	convBN(512, 3, 1)
NNupsample(16)	NNupsample(16)
convBN(256, 3, 1)	convBN(256, 3, 1)
NNupsample(32)	NNupsample(32)
convBN(128, 3, 1)	convBN(128, 3, 1)
convBN(128, 3, 1)	convBN(128, 3, 1)
NNupsample(64)	NNupsample(64
convBN(64, 3, 1)	convBN(64, 3, 1)
convBN(3, 3, 1)	convBN(64, 3, 1)
	NNupsample(128)
	convBN(32, 3, 1)
	conv(3, 3, 1)

Table 18: Network architectures for discriminator (containing a classifier and a decoder) used on Wikiart.

Classifier	Decoder
conv(128, 3, 2)	convBN(512, 3, 1)
Dropout(0.2)	
convBN(256, 3, 2)	NNupsample(8)
Dropout(0.2)	convBN(256, 3, 1)
convBN(512, 3, 2)	NNupsample(16)
convBN(512, 3, 1)	convBN(128, 3, 1)
Dropout(0.2)	
convBN(1024, 3, 2)	NNupsample(32)
	convBN(64, 3, 1)
fc(10)	NNupsample(64)
	convBN(32, 3, 1)
	conv(3, 3, 1)

Table 19: Network architectures for the generators (with and without magnified learning) used on Wikiart.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$)	fcBN($512 \times 4 \times 4$)
NNupsample(8)	NNupsample(8)
convBN(512, 3, 1)	convBN(512, 3, 1)
NNupsample(16)	NNupsample(16)
convBN(256, 3, 1)	convBN(256, 3, 1)
NNupsample(32)	NNupsample(32)
convBN(128, 3, 1)	convBN(128, 3, 1)
NNupsample(64)	NNupsample(64
convBN(64, 3, 1)	convBN(64, 3, 1)
convBN(3, 3, 1)	convBN(64, 3, 1)
	NNupsample(128)
	convBN(32, 3, 1)
	conv(3, 3, 1)

Table 20: Network architectures for discriminator (containing a classifier and a decoder) used on Oxford-102 and CUB-200.

Classifier	Decoder
conv(64, 3, 2)	convBN(512, 3, 1)
Dropout(0.2)	
convBN(128, 3, 2)	NNupsample(8)
Dropout(0.2)	convBN(256, 3, 1)
convBN(256, 3, 2)	NNupsample(16)
convBN(256, 3, 1)	convBN(128, 3, 1)
Dropout(0.2)	
convBN(512, 3, 2)	NNupsample(32)
convBN(512, 3, 1)	convBN(64, 3, 1)
fc(K)	NNupsample(64)
	conv(32, 3, 1)
	conv(3, 3, 1)

Table 21: Network architectures for the generators (with and without magnified learning) used on Oxford-102 and CUB-200.

Generator	Generator with Magnified Learning
fcBN($512 \times 4 \times 4$)	fcBN($512 \times 4 \times 4$)
NNupsample(8)	NNupsample(8)
convBN(512, 3, 1)	convBN(512, 3, 1)
NNupsample(16)	NNupsample(16)
convBN(256, 3, 1)	convBN(256, 3, 1)
NNupsample(32)	NNupsample(32)
convBN(128, 3, 1)	convBN(128, 3, 1)
NNupsample(64)	NNupsample(64
convBN(64, 3, 1)	convBN(64, 3, 1)
convBN(3, 3, 1)	convBN(64, 3, 1)
	NNupsample(128)
	convBN(32, 3, 1)
	conv(3, 3, 1)

D

MORE SYNTHETIC IMAGES

This appendix shows the full samples for generated *CUB-200 birds* (Figure 42, 43, 44, 45, 46, 47, 48) and *Oxford-102 flowers* (Figure 49, 50, 51, 52) images. Each row represents one of the bird or flower species. Additional *CIFAR-10* and *STL-10* synthetic images can be seen in Figure 53 and 54, respectively. Furthermore, Figure 55, 56, 57 show the extra synthetic artworks based on the *genre*, *artist*, and *style*, respectively. It can be seen that the generated images are clear and has high quality.

Figure 42: More generated images on CUB-200 birds at **128** × **128** pixels.

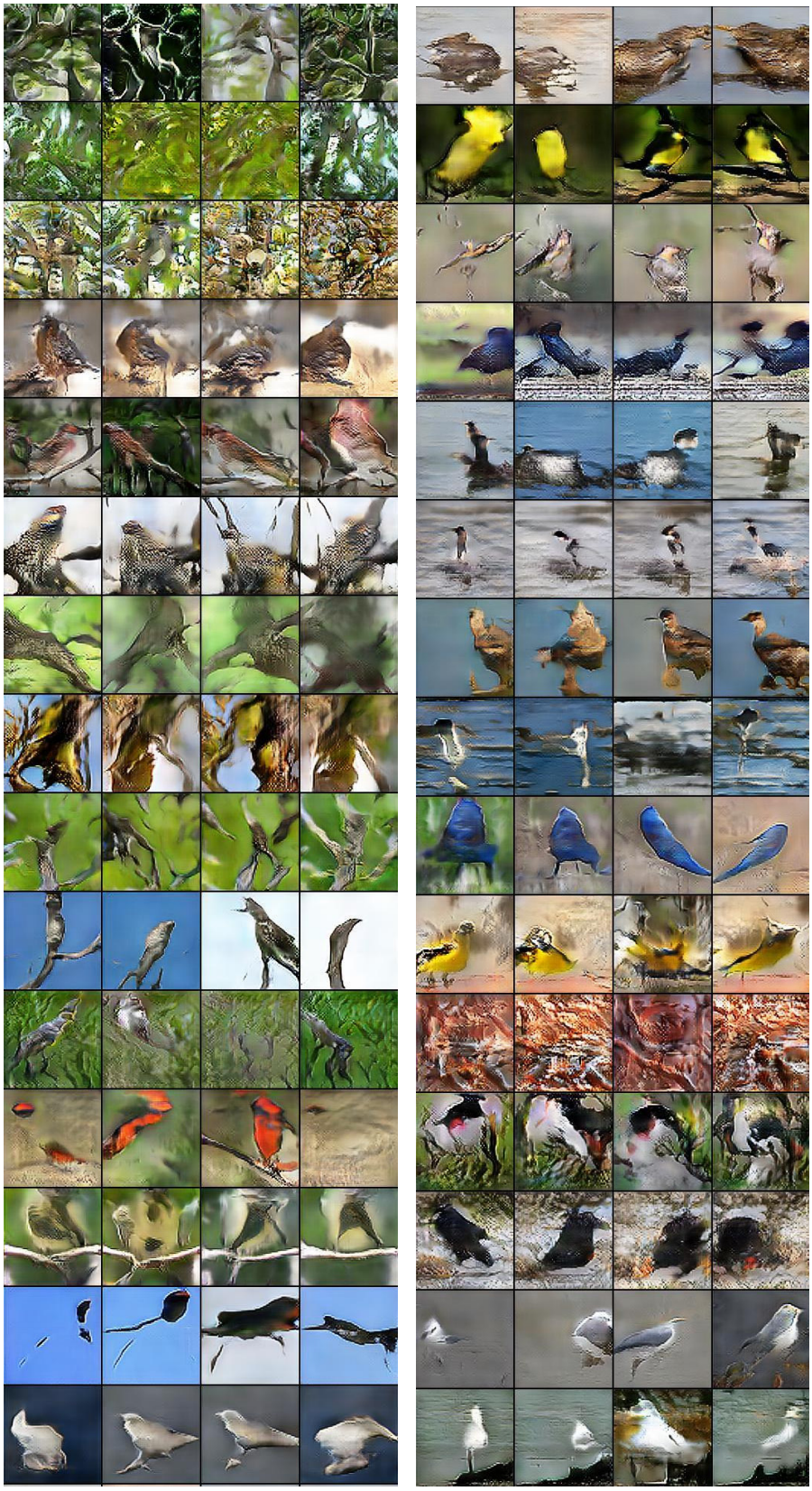


Figure 43: More generated images on CUB-200 birds at **128** × **128** pixels.

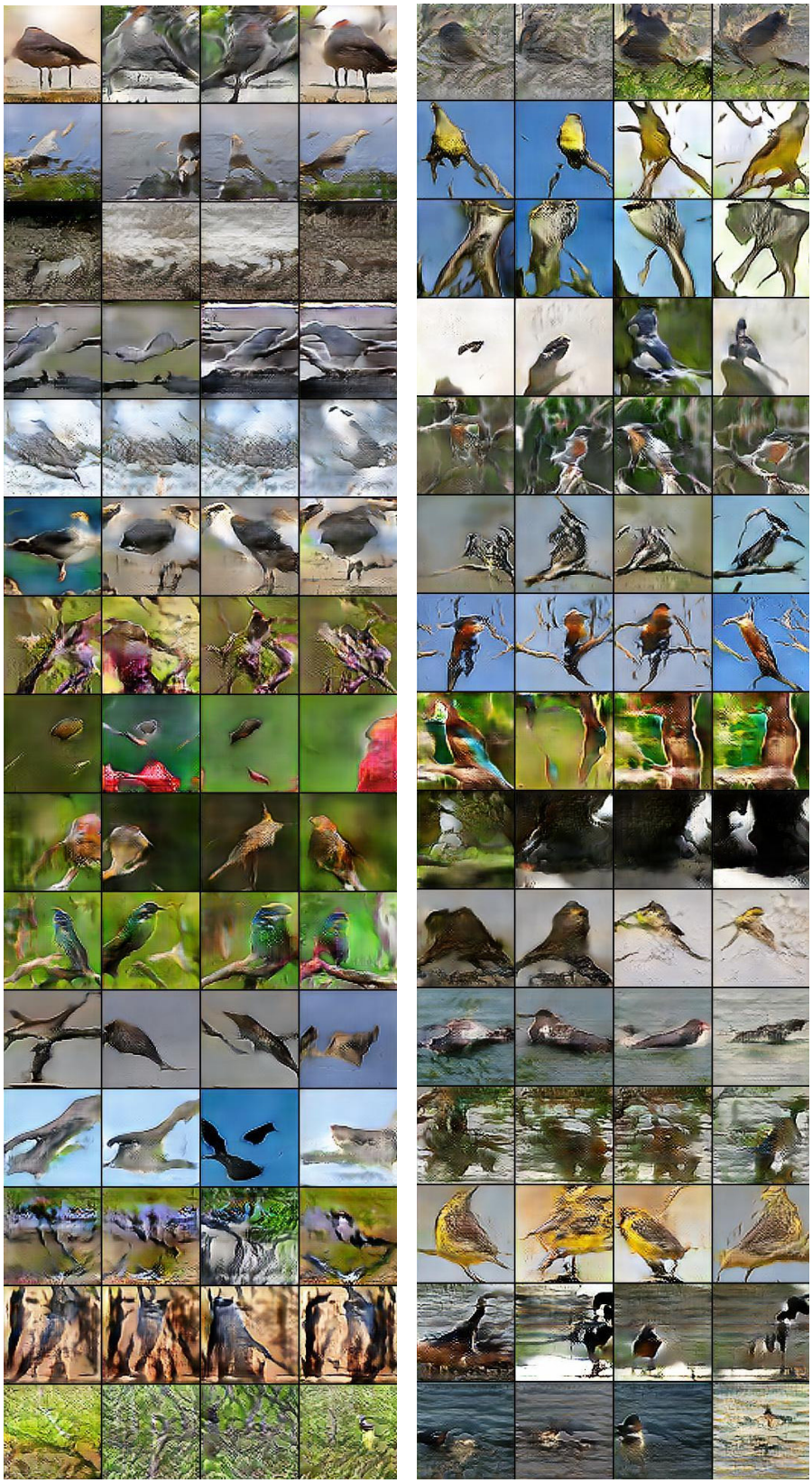


Figure 44: More generated images on CUB-200 birds at **128** × **128** pixels.

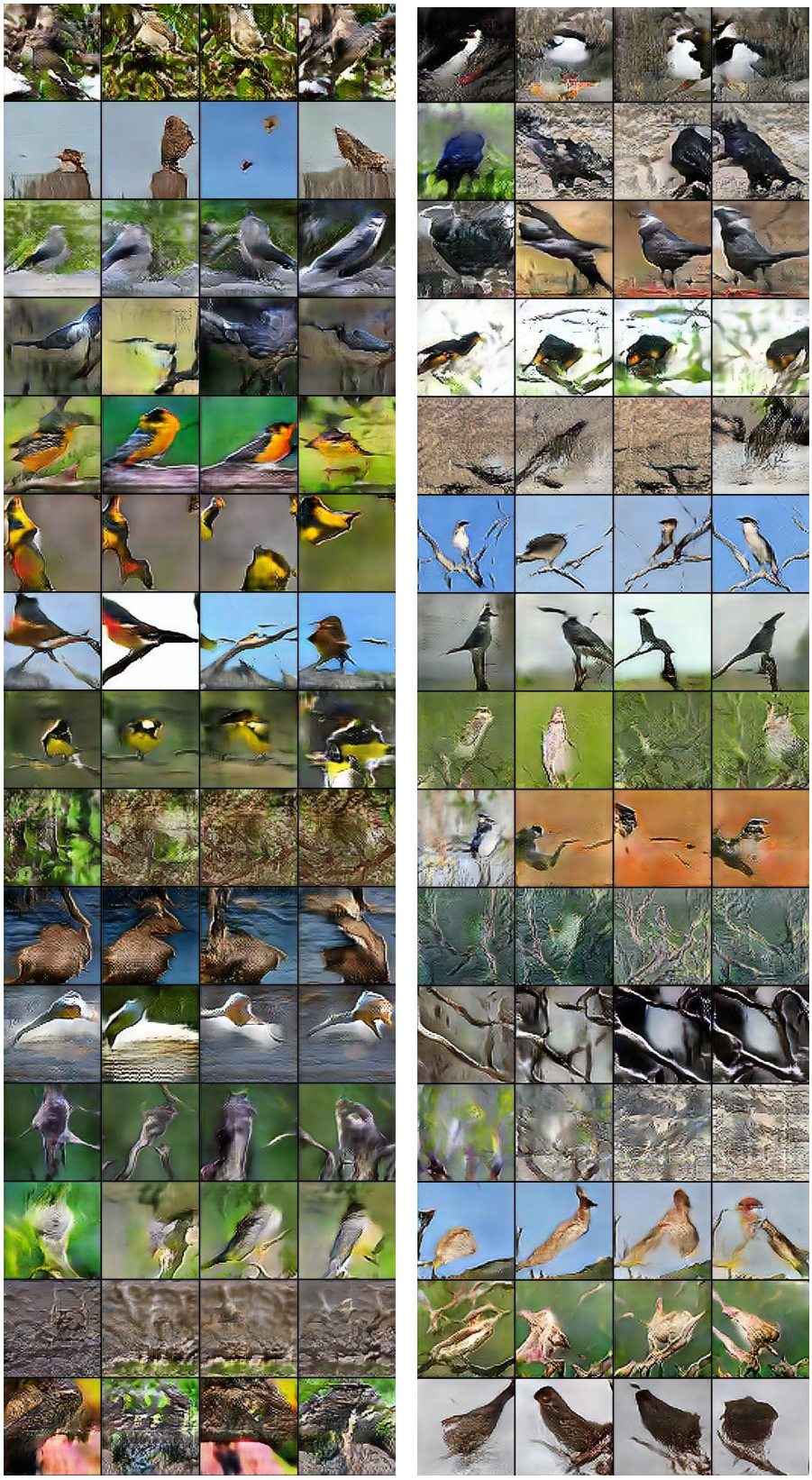


Figure 45: More generated images on CUB-200 birds at **128** × **128** pixels.

Figure 46: More generated images on CUB-200 birds at **128** × **128** pixels.

Figure 47: More generated images on CUB-200 birds at **128** × **128** pixels.

Figure 48: More generated images on CUB-200 birds at **128** × **128** pixels.

Figure 49: More generated images on Oxford-102 flowers at **128** × **128** pixels.

Figure 50: More generated images on Oxford-102 flowers at $\mathbf{128} \times \mathbf{128}$ pixels.

Figure 51: More generated images on Oxford-102 flowers at **128** $\times$ **128** pixels.

Figure 52: More generated images on Oxford-102 flowers at **128** $\times$ **128** pixels.

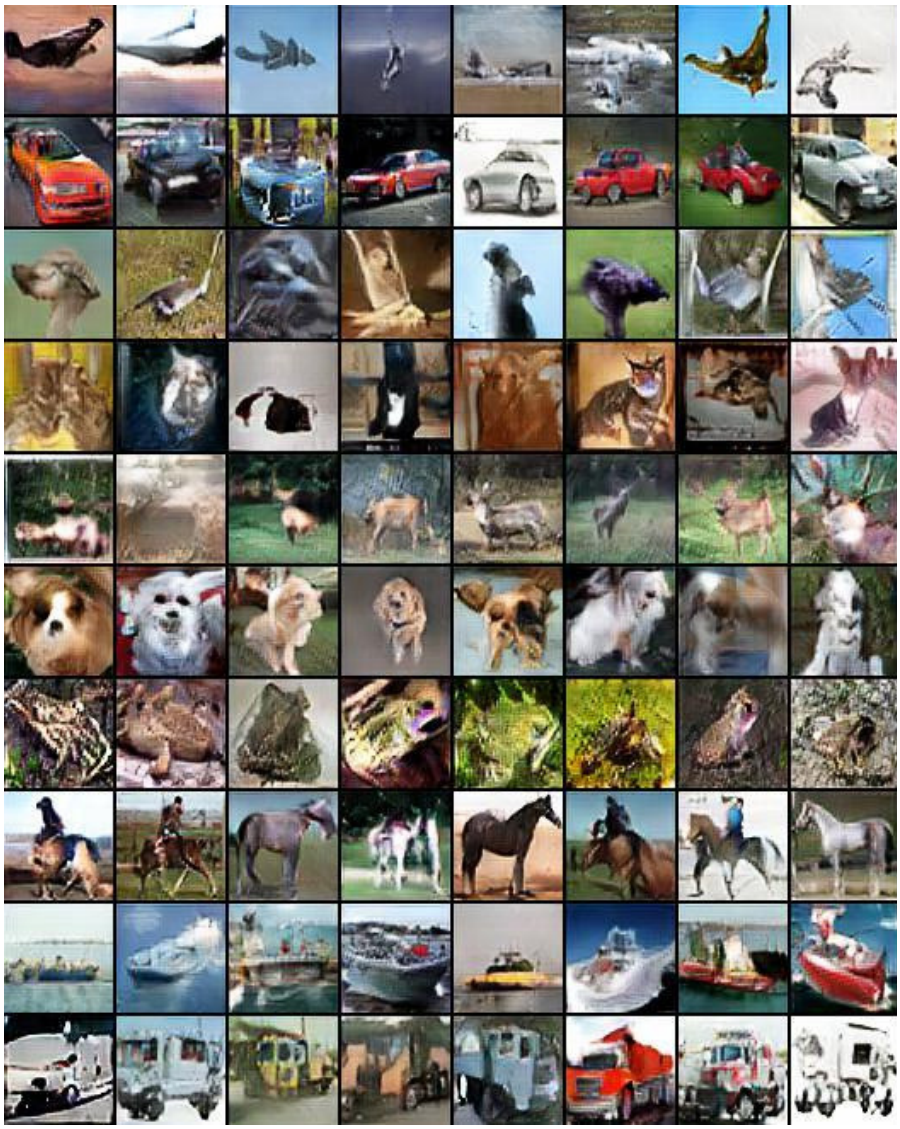


Figure 53: More generated images on CIFAR-10 at **64** $\times$ **64** pixels. From top to bottom: (1) Airplane, (2) Automobile, (3) Bird, (4) Cat, (5) Deer, (6) Dog, (7) Frog, (8) Horse, (9) Ship, (10) Truck.

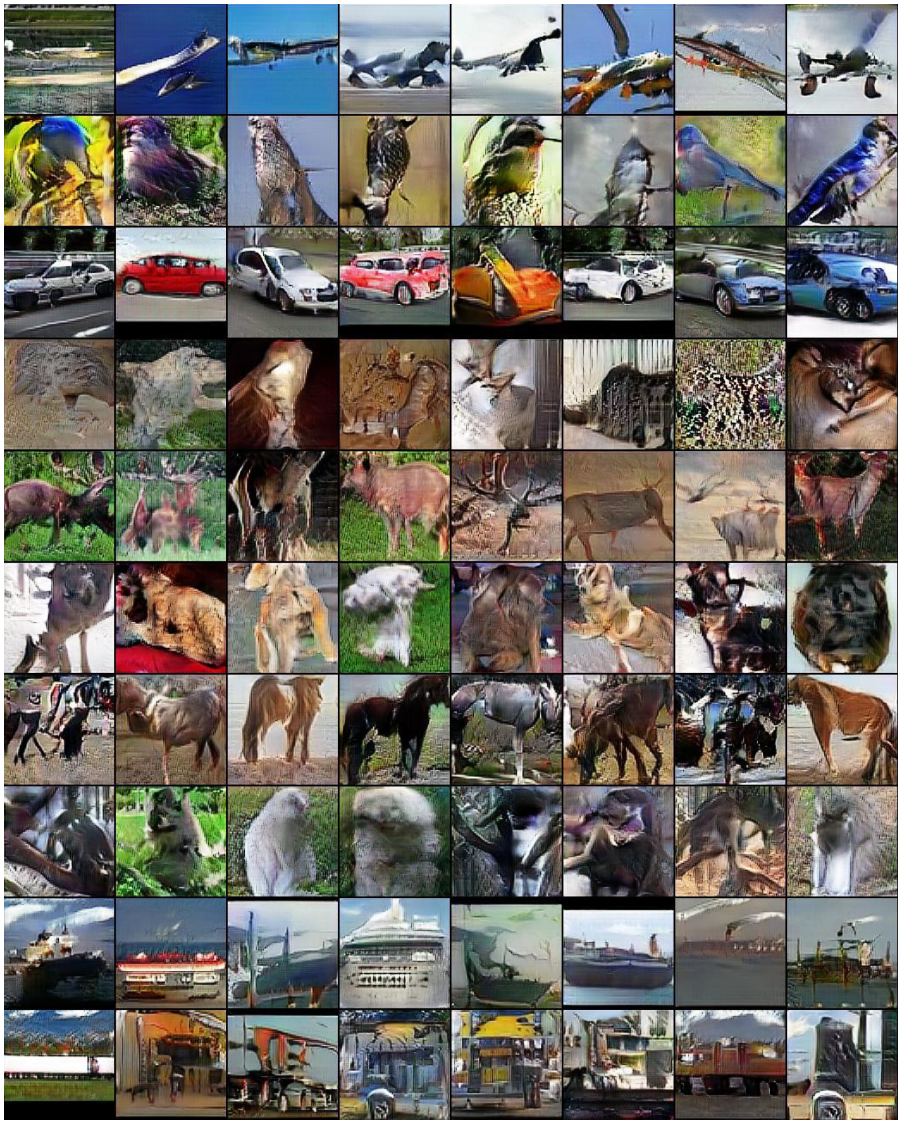


Figure 54: More generated images on STL-10 at **128** $\times$ **128** pixels. From top to bottom: (1) Airplane, (2) Bird, (3) Car, (4) Cat, (5) Deer, (6) Dog, (7) Horse, (8) Monkey, (9) Ship, (10) Truck.

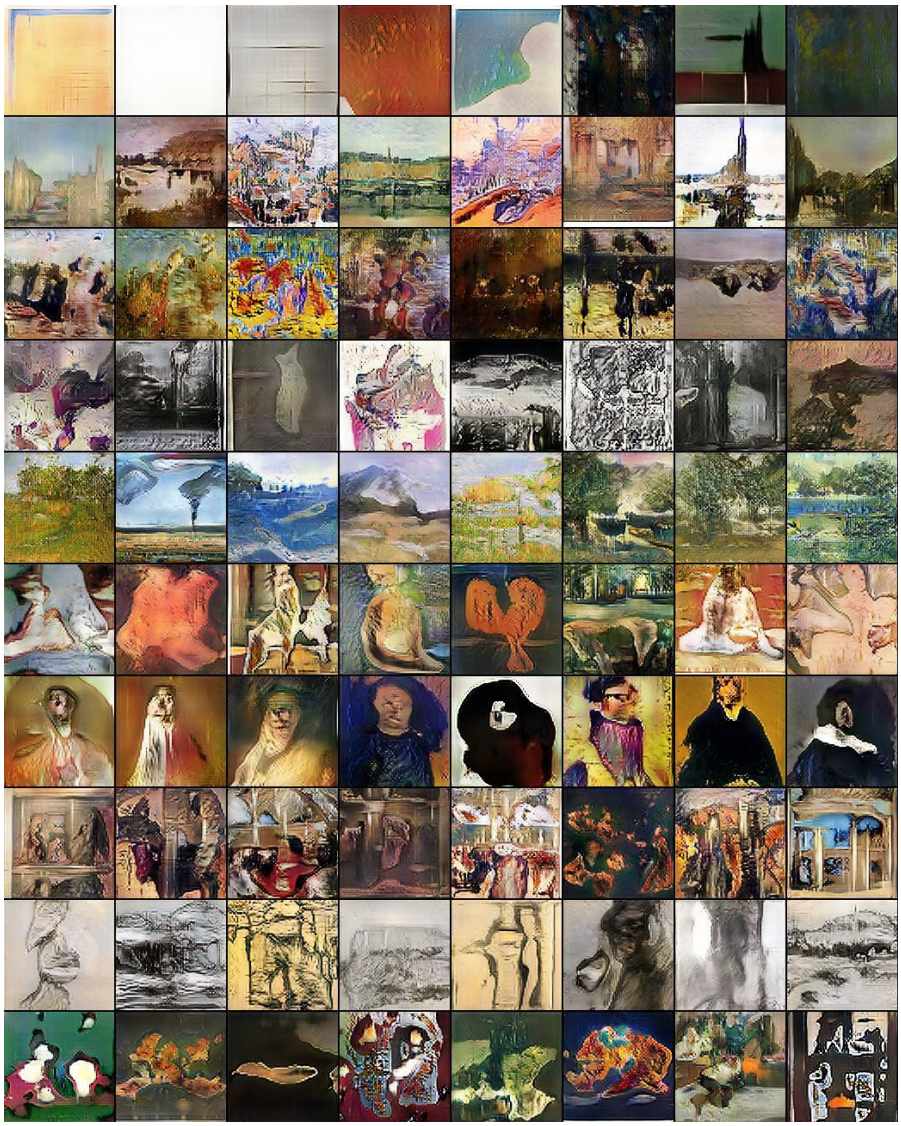


Figure 55: More generated images on Wikiart *Genre* at **128** $\times$ **128** pixels. From top to bottom: (1) Abstract, (2) Cityscape, (3) Genre painting, (4) Illustration, (5) Landscape, (6) Nude, (7) Portrait, (8) Religious, (9) Sketch and study, (10) Still life.

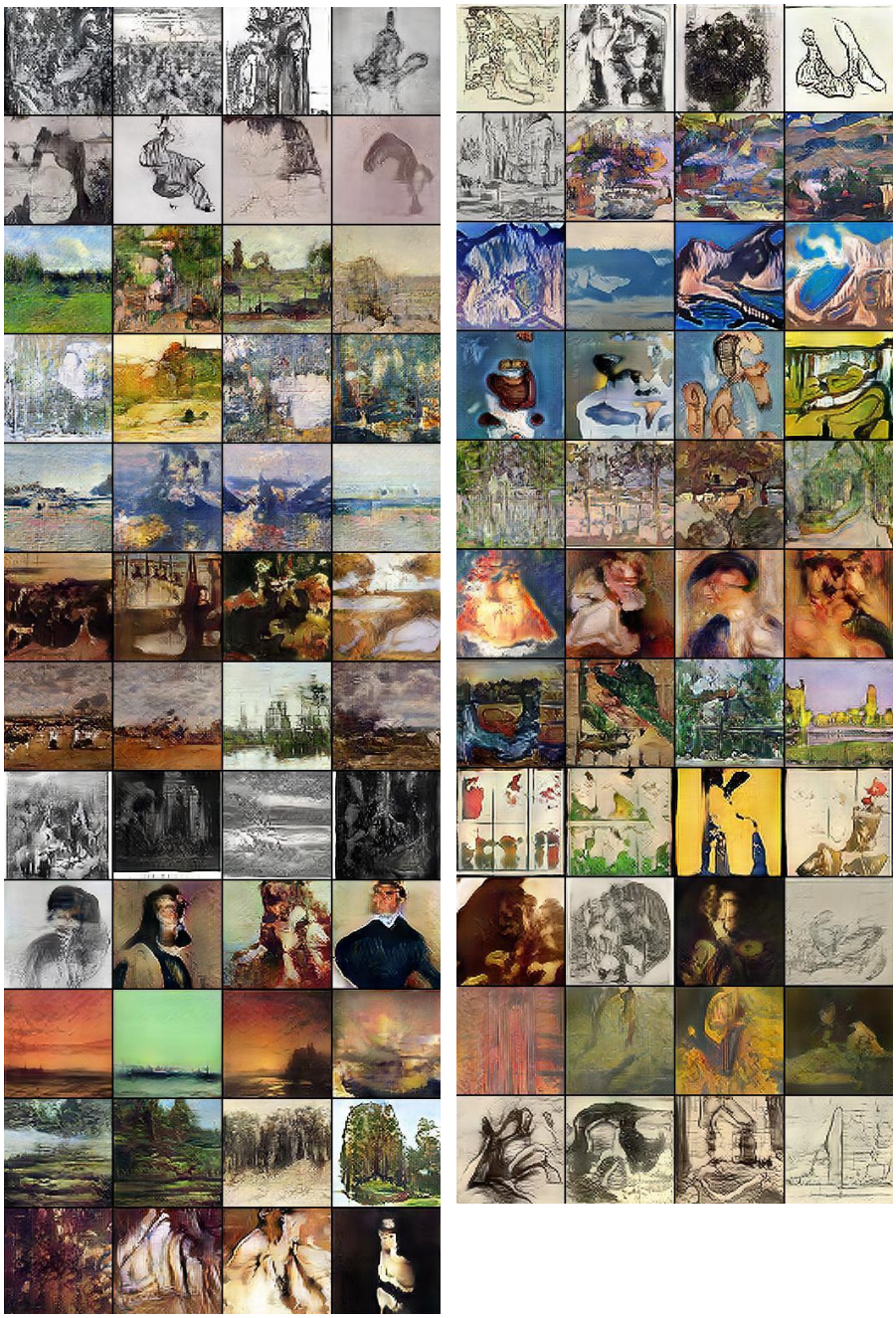


Figure 56: More generated images on Wikiart *Artist* at $\mathbf{128 \times 128}$ pixels. From top to bottom: (1) Albrecht Durer, (2) Boris Kustodiev, (3) Camille Pissarro, (4) Childe Hassam , (5) Claude Monet, (6) Edgar Degas, (7) Eugene Boudin, (8) Gustave Dore, (9) Ilya Repin, (10) Ivan Aivazovsky, (11) Ivan Shishkin, (12) John Singer Sargent, (13) Marc Chagall, (14) Martiros Saryan, (15) Nicholas Roerich, (16) Pablo Picasso, (17) Paul Cezanne, (18) Pierre Auguste Renoir, (19) Pyotr Konchalovsky, (20) Raphael Kirchner, (21) Rembrandt, (22) Salvador Dali, (23) Vincent van Gogh.

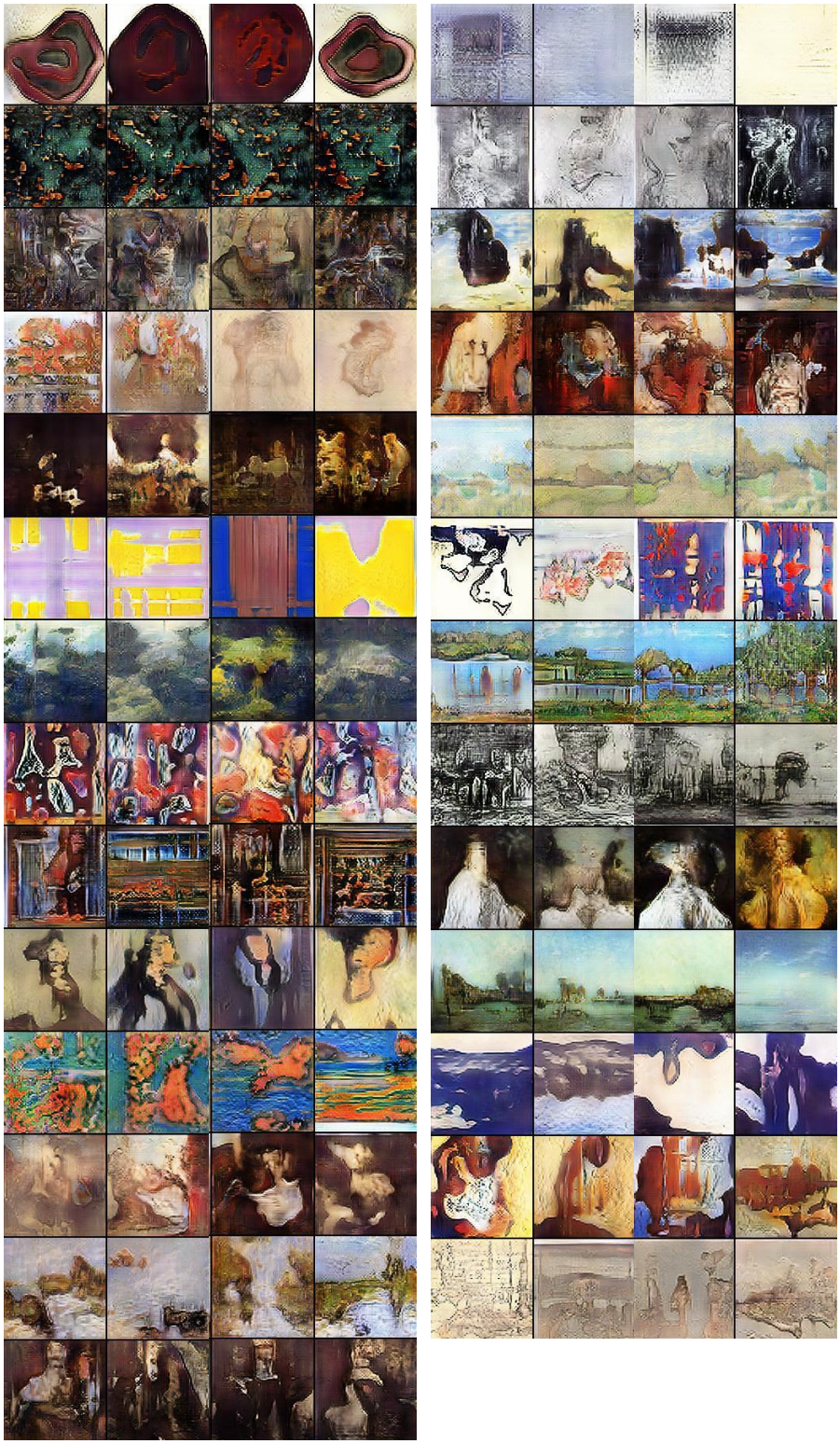


Figure 57: More generated images on Wikiart *Style* at **128** × **128** pixels. From top to bottom, **Left**: (1) Abstract Expressionism, (2) Action painting, (3) Analytical Cubism, (4) Art Nouveau, (5) Baroque, (6) Color Field Painting, (7) Contemporary Realism, (8) Cubism, (9) Early Renaissance, (10) Expressionism, (11) Fauvism, (12) High Renaissance, (13) Impressionism, (14) Mannerism Late Renaissance; **Right**: (15) Minimalism, (16) Naive Art Primitivism, (17) New Realism, (18) Northern Renaissance, (19) Pointillism, (20) Pop Art, (21) Post Impressionism, (22) Realism, (23) Rococo, (24) Romanticism, (25) Symbolism, (26) Synthetic Cubism, (27) Ukiyo-e.

BIBLIOGRAPHY

[1] Understanding and visualizing cnn. http://cs231n.github.io/understanding-cnn/.

[2] 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.

[3] Igor Aizenberg, Naum N Aizenberg, and Joos PL Vandewalle. *Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications*. Springer Science & Business Media, 2000.

[4] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[5] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[8] Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.

[9] Mauro Barni, Anna Pelagotti, and Alessandro Piva. Image processing for the analysis and conservation of paintings: opportunities and challenges. *IEEE Signal processing magazine*, 22(5):141–144, 2005.

[10] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[11] Igor Berezhnoy, Eric Postma, and Jaap van den Herik. Computer analysis of van goghs complementary colours. *Pattern Recognition Letters*, 28(6):703–709, 2007.

[12] Igor E Berezhnoy, Eric O Postma, and H Jaap van den Herik. Authentic: computerized brushstroke analysis. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 1586–1588. IEEE, 2005.

[13] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[14] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[15] Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *arXiv preprint arXiv:1703.06975*, 2017.

[16] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[17] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[18] Susan S Bowen. Computer vision as a tool for image synthesis. *SPIE Visual Data Exploration and Analysis II*, 2410:346–357, 1995.

[19] Chee Seng Chan and Honghai Liu. Fuzzy qualitative human motion analysis. *IEEE Transactions on Fuzzy Systems*, 17(4):851–862, 2009.

[20] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: a computational account. *Neural computation*, 10(7):1759–1777, 1998.

[21] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Neuronal regulation: A mechanism for synaptic pruning during brain maturation. *Neural Computation*, 11(8):2061–2080, 1999.

[22] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[23] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. *arXiv preprint arXiv:1504.04788*, 2015.

[24] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[25] Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

[26] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[27] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[28] Antonio Criminisi, Martin Kemp, and Andrew Zisserman. Bringing pictorial space to life: computer techniques for the analysis of paintings. *Digital art history: A subject in transition*, 1:5, 2005.

[29] Dengxin Dai, Hayko Riemenschneider, Gerhard Schmitt, and Luc Van Gool. Example-based facade texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1065–1072, 2013.

[30] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. *arXiv preprint arXiv:1702.01691*, 2017.

[31] 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.

[32] Rina Dechter. *Learning while searching in constraint-satisfaction problems*. University of California, Computer Science Department, Cognitive Systems Laboratory, 1986.

[33] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

[34] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2148–2156, 2013.

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

[36] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1269–1277, 2014.

[37] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[38] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[39] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[40] 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.

[41] 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.

[42] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[43] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[44] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[45] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[46] David J Field. Wavelets, vision and the statistics of natural scenes. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 357(1760):2527–2542, 1999.

[47] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[48] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[49] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014.

[50] 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.

[51] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[52] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[53] 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.

[54] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[55] 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.

[56] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[57] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[58] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. *From Natural to Artificial Neural Computation*, pages 195–201, 1995.

[59] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[60] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015.

[61] Stephen José Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 177–185, 1989.

[62] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[63] Caner Hazirbas. Feature selection and learning for semantic segmentation. Master's thesis, Technical University Munich, Germany, June 2014.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision–ECCV 2014*, pages 346–361. Springer, 2014.

[65] 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.

[66] 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.

[67] David P Helmbold and Philip M Long. Surprising properties of dropout in deep networks. *arXiv preprint arXiv:1602.04484*, 2016.

[68] Geoffrey E Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

[69] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[70] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[71] Wai Lam Hoo and Chee Seng Chan. Zero-shot object recognition system based on topic model. *IEEE T. Human-Machine Systems*, 45(4):518–525, 2015.

[72] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[73] Gary B. Huang and Erik Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014.

[74] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *arXiv preprint arXiv:1703.06868*, 2017.

[75] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

[76] Peter R Huttenlocher and Chr De Courten. The development of synapses in striate cortex of man. *Human neurobiology*, 6(1):1–9, 1986.

[77] Peter R Huttenlocher, Christian de Courten, Laurence J Garey, and Hendrik Van der Loos. Synaptogenesis in human visual cortex—evidence for synapse elimination during normal development. *Neuroscience letters*, 33(3):247–252, 1982.

[78] Peter R Huttenlocher et al. Synaptic density in human frontal cortex—developmental changes and effects of aging. *Brain Res*, 163(2):195–205, 1979.

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

[80] 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.

[81] 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.

[82] Alekseĭ Grigor'evich Ivakhnenko and Valentin Grigorévich Lapa. Cybernetic predicting devices, 1965.

[83] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[84] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[85] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[86] C Richard Johnson Jr, Ella Hendriks, Igor J Berezhnoy, Eugene Brevdo, Shannon M Hughes, Ingrid Daubechies, Jia Li, Eric Postma, and James Z Wang. Image processing for artist identification. *IEEE Signal Processing Magazine*, 25(4):37–48, 2008.

[87] Katherine Jones-Smith and Harsh Mathur. Fractal analysis: revisiting pollock's drip paintings. *Nature*, 444(7119):E9–E10, 2006.

[88] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.

[89] Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[90] Andrej Karpathy. Linear classification: Loss function. http://cs231n.github.io/linear-classify/#loss.

[91] M Kemp and A Criminisi. Computer vision and painters vision in italian and nederlandish art of the fifteenth century. *Perspective, projections and design technologies of architectural representation, M. Carpo and F. Lemerle, eds*, pages 31–46.

[92] Fahad Shahbaz Khan, Shida Beigpour, Joost van de Weijer, and Michael Felsberg. Painting-91: a large scale database for computational painting categorization. *Machine Vision and Applications*, 25(6):1385–1397, 2014.

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

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

[95] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images, 2009.

[96] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[97] 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.

[98] Hanock Kwak and Byoung-Tak Zhang. Generating images part by part with composite generative adversarial networks. *arXiv preprint arXiv:1607.05387*, 2016.

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

[100] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[101] 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.

[102] 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.

[103] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006.

[104] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[105] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *NIPs*, volume 2, pages 598–605, 1989.

[106] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *arXiv preprint arXiv:1609.04802*, 2016.

[107] Sue Han Lee, Chee Seng Chan, Paul Wilkin, and Paolo Remagnino. Deep-plant: Plant identification with convolutional neural networks. In *IEEE International Conference on Image Processing ICIP*, pages 452–456, 2015.

[108] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[109] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. *arXiv preprint arXiv:1705.08584*, 2017.

[110] Jia Li and James Ze Wang. Studying digital imagery of ancient paintings by mixtures of stochastic models. *IEEE Transactions on Image Processing*, 13(3):340–353, 2004.

[111] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[112] Chern Hong Lim, Anhar Risnumawan, and Chee Seng Chan. A scene image is nonmutually exclusivea fuzzy qualitative scene understanding. *IEEE Transactions on Fuzzy Systems*, 22(6):1541–1556, 2014.

[113] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[114] Honghai Liu, David J Brown, and George M Coghill. Fuzzy qualitative robot kinematics. *IEEE Transactions on Fuzzy Systems*, 16(3):808–822, 2008.

[115] 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.

[116] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

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

[118] Henri Maitre, Francis Schmitt, and Christian Lahanier. 15 years of image processing and the fine arts. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 1, pages 557–561. IEEE, 2001.

[119] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *arXiv preprint arXiv:1511.02793*, 2015.

[120] Kirk Martinez, John Cupitt, David Saunders, and Ruven Pillay. Ten years of art imaging research. *Proceedings of the IEEE*, 90(1):28–41, 2002.

[121] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009.

[122] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

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

[124] Andres Munoz. Machine learning and optimization. *URL: https://www. cims. nyu. edu/˜ munoz/files/ml_optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]*, 2014.

[125] 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.

[126] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in Neural Information Processing Systems*, pages 3387–3395, 2016.

[127] Anh Nguyen, Jason Yosinski, Yoshua Bengio, Alexey Dosovitskiy, and Jeff Clune. Plug & play generative networks: Conditional iterative generation of images in latent space. *arXiv preprint arXiv:1612.00005*, 2016.

[128] 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.

[129] 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.

[130] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[131] Sebastian Nowozin, Peter V Gehler, and Christoph H Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *European conference on computer vision*, pages 98–111. Springer, 2010.

[132] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[133] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[134] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[135] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1717–1724, 2014.

[136] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[137] 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.

[138] Dale Purves and Jeff W Lichtman. Elimination of synapses in the developing nervous system. *Science*, 210(4466):153–157, 1980.

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

[140] JP Rauschecker. Neuronal mechanisms of developmental plasticity in the cat's visual system. *Human neurobiology*, 3(2):109–114, 1983.

[141] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 512–519, 2014.

[142] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[143] 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.

[144] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[145] Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *NIPS*, 2016.

[146] 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.

[147] Scott Reed, Aäron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt Botvinick, and Nando de Freitas. Generating interpretable images with controllable structure. 2016.

[148] Fabio Remondino, Sabry F El-Hakim, Armin Gruen, and Li Zhang. Turning images into 3-d models. *IEEE Signal Processing Magazine*, 25(4), 2008.

[149] 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.

[150] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012.

[151] 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.

[152] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[153] 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.

[154] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *AISTATS*, volume 1, page 3, 2009.

[155] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

[156] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

[157] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[158] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[159] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[160] Lior Shamir, Tomasz Macura, Nikita Orlov, D Mark Eckley, and Ilya G Goldberg. Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art. *ACM Transactions on Applied Perception*, 7(2):8, 2010.

[161] Lior Shamir and Jane A Tarakhovsky. Computer analysis of art. *Journal on Computing and Cultural Heritage*, 5(2):7, 2012.

[162] Qiang Shen and Roy Leitch. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 23(4):1038–1061, 1993.

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

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

[165] B Smith, David G Stork, and Li Zhang. Three-dimensional reconstruction from multiple reflected views within a realist painting: An application to scott frasers three way vanitas. *Electronic imaging: 3D imaging metrology*, 7239:72390U1–10, 2009.

[166] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

[167] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[168] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[169] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[170] Nitish Srivastava, Geoffrey 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.

[171] David G Stork. Color and illumination in the hockney theory: A critical evaluation. In *Color and Imaging Conference*, volume 2003, pages 11–15. Society for Imaging Science and Technology, 2003.

[172] David G Stork. Computer vision, image analysis, and master art: part 1. *IEEE MultiMedia*, 13(3):16–20, 2006.

[173] David G Stork. Imaging technology enhances the study of art. *Vision Systems Design*, 12(10):69–71, 2007.

[174] David G Stork. Computer vision and computer graphics analysis of paintings and drawings: An introduction to the literature. In *International Conference on Computer Analysis of Images and Patterns*, pages 9–24. Springer, 2009.

[175] David G Stork and Marco F Duarte. Computer vision, image analysis, and master art: Part 3. *IEEE MultiMedia*, 14(1), 2007.

[176] David G Stork and Yasuo Furuichi. Image analysis of paintings by computer graphics synthesis: An investigation of the illumination in georges de la tour's christ in the carpenter's studio. In *Electronic Imaging 2008*, pages 68100J–68100J. International Society for Optics and Photonics, 2008.

[177] David G Stork and M Kimo Johnson. Estimating the location of illuminants in realist master paintings computer image analysis addresses a debate in art history of the baroque. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 255–258. IEEE, 2006.

[178] David G Stork and Micah K Johnson. Computer vision, image analysis, and master art: Part 2. *IEEE MultiMedia*, 13(4):12–17, 2006.

[179] David G Storka and Micah K Johnsonb. Lighting analysis of diffusely illuminated tableaus in realist paintings: An application to detecting compositing in the portraits of garth herrick. *algorithms*, 10:2, 2009.

[180] Paul Sturgess, Karteek Alahari, Lubor Ladicky, and Philip HS Torr. Combining appearance and structure from motion features for road scene understanding. In *BMVC 2012-23rd British Machine Vision Conference*. BMVA, 2009.

[181] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[182] 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.

[183] Wei Ren Tan, Chee Seng Chan, Hernán Aguirre, and Kiyoshi Tanaka. Artgan: Artwork synthesis with conditional categorial gans. In *Image Processing (ICIP), 2017 IEEE International Conference on*. IEEE, 2017.

[184] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Learning a generative adversarial network for high resolution artwork synthesis. *arXiv preprint arXiv:1708.09533*, 2017.

[185] Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka. Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 3703–3707. IEEE, 2016.

[186] Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka. Fuzzy qualitative deep compression network. *Neurocomputing*, (251):1–15, 2017.

[187] Theano. Convolution arithmetic tutorial. http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html.

[188] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[189] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magni-

tude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[190] Antonio Torralba and Aude Oliva. Statistics of natural image categories. *Network: computation in neural systems*, 14(3):391–412, 2003.

[191] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[192] Jan Van Leeuwen. On the construction of huffman trees. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 382–410, 1976.

[193] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

[194] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[195] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1058–1066, 2013.

[196] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized mle for generative adversarial learning. *arXiv preprint arXiv:1611.01722*, 2016.

[197] Nannan Wang, Xinbo Gao, Leiyu Sun, and Jie Li. Bayesian face sketch synthesis. *IEEE Transactions on Image Processing*, 26(3):1264–1274, 2017.

[198] Yaxing Wang, Lichao Zhang, and Joost van de Weijer. Ensembles of generative adversarial networks. *arXiv preprint arXiv:1612.00991*, 2016.

[199] D Warde-Farley and Y Bengio. Improving generative adversarial networks with denoising feature matching. *ICLR submissions*, 8, 2017.

[200] Eric W. Weisstein. Deconvolution. http://mathworld.wolfram.com/Deconvolution.html.

[201] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[202] Juyang Weng, Narendra Ahuja, and Thomas S Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 1, pages 576–581. IEEE, 1992.

[203] Wikipedia. Artificial neuron: Types of transfer functions. https://en.wikipedia.org/wiki/Artificial_neuron#Types_of_transfer_functions.

[204] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

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

[206] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. IEEE, 2009.

[207] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*, 2017.

[208] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval*, pages 197–206. ACM, 2007.

[209] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *International Conference on Computer Vision (ICCV)*, pages 3676–3684, 2015.

[210] Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1476–1483, 2015.

[211] Raymond Yeh, Chen Chen, Teck Yian Lim, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with perceptual and contextual losses. *arXiv preprint arXiv:1607.07539*, 2016.

[212] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NIPS)*, pages 3320–3328, 2014.

[213] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[214] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[215] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[216] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint arXiv:1612.03242*, 2016.

[217] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016.

[218] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495, 2014.

[219] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.