Learning Through Text-Image Pairs and Image Sequences

QICHENG LAO

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

OCTOBER 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Qicheng Lao**

Entitled: **Learning Through Text-Image Pairs and Image Sequences**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. William E. Lynch

_____ External Examiner
Dr. Ismail Ben Ayed

_____ Examiner
Dr. Maria Amer

_____ Examiner
Dr. Adam Krzyzak

_____ Examiner
Dr. Tien D. Bui

_____ Supervisor
Dr. Thomas Fevens

Approved by: _____
Dr. Leila Kosseim, Graduate Program Director
Department of Computer Science and Software Engineering

October 7th, 2019

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Learning Through Text-Image Pairs and Image Sequences

Qicheng Lao, Ph.D.
Concordia University, 2019

Many machine learning systems for artificial intelligence are biologically inspired, for example, the artificial neural networks (ANNs) have similar architecture as human brains, and convolutional neural networks (CNNs) are inspired by the observations from early study on animal's visual cortex system. The above two examples (ANNs and CNNs) are inspirations at the level of creating fundamental tools (*e.g.*, neural networks) for a machine learning system. Another level of inspirations can come from the way human learn or respond that builds on top of the existing powerful learning tools, *i.e.*, brains. In this thesis, we will focus on another type of inspiration that also belongs to the second level. It is based on the common practice that for an efficient learning or an optimal decision, human integrate all sources of available information in multiple views and leverage the reasoning of the underlying connections among them, *i.e.*, multi-view learning. We address several problems in both medical and non-medical domains, including text-to-image synthesis, cell phenotype classification, histopathological malignancy diagnosis and disease progression learning, from the perspective of multi-view learning with an emphasis on learning the underlying connections among the multiple distinct feature sets representing the given multi-view data (*i.e.*, image sequences in a unimodal setting and text-image pairs in a multi-modal setting).

Acknowledgments

I would like to take this opportunity to thank again my supervisor Dr. Thomas Fevens, who has been extremely supportive, patient and thoughtful during my PhD study. This is my second try for a PhD after I quit a previous PhD program in experimental medicine, and the new journey would not have been started without him.

I would also like to thank other members of my thesis committee, Drs. William E. Lynch, Maria Amer, Adam Krzyzak, Tien D. Bui and Ismail Ben Ayed for their supportive efforts and constructive remarks.

I should also acknowledge open innovation team from Imagia, especially Mohammad Havaei, Francis Dutil and Lisa Di Jorio, without whose collaboration, part of this thesis work would not have been possible.

Finally, my deepest gratitude goes to my parents for their unconditional support throughout my study and entire life, and all of my best friends for their accompany, encouragement and help. I wish them all the best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Many machine learning systems for artificial intelligence are biologically inspired, for example, the artificial neural networks (ANNs) have similar architecture as human brains, where a neuron takes input signal from many other neurons, processes the information and fires its own signal after activation to downstream neurons; And convolutional neural networks (CNNs) are inspired by the observations from early study on animal's visual cortex system [8], which contains different types of cells that are sensitive to their specialized receptive fields, with different sizes from small to large, systematically covering the entire visual field.

The above two examples (ANNs and CNNs) are inspirations at the level of creating fundamental tools (*e.g.*, neural networks) for a machine learning system. Another level of inspirations can come from the way human learn or respond that builds on top of the existing powerful learning tools, *i.e.*, brains. In fact, this level of inspirations widely exists with or without awareness, and also results in various machine learning topics that are currently under active research, such as transfer learning [9, 10, 11] and multi-task learning [12, 13, 14], respectively based on the fact that human usually learn a new task based on previous knowledge, and can respond towards multiple tasks simultaneously. Unlike the first level of inspiration, the second one is more intuitive and implicit, and it also relies on the success of the first level. In other words, a complete machine learning system includes a learning tool and how this tool is used. Quite often the latter also shapes the tool itself, and the two together can form a positive feedback loop for an overall better system, similar to the evolution of human intelligence. One great example of the second level inspirations reshaping the learning tool is the integration of attention mechanism into many neural networks such as recurrent neural networks (RNNs), where the attention reflects how human visual perception differentiates high resolution information from low resolution information.

This thesis will focus on another type of inspiration that also belongs to the second level. It is based on the common practice that for an efficient learning or an optimal decision, human integrate all sources of available information in multiple views and leverage the reasoning of underlying connections among them, *i.e.*, multi-view learning [4, 15, 16]. The available information can be present within a single modality or across multiple modalities, such as image, text and audio data. For example, in a unimodal setting (*e.g.*, all views in image modality), the multi-view data would comprise sequences of images for a certain kind of object or concept in multiple views, with each view encoding different feature sets, either independent or complementary, such as different viewpoints of a 3D object, or different magnification levels of a 2D image; in a multi-modal setting (*e.g.*, views in both image and text modality), the multi-view data could be given as pairs of image and its corresponding text descriptions, and the challenge is to transform both image and text data into the same latent feature space for the subsequent tasks. This is illustrated in Figure 1.1 with one example for learning through image sequences (left, Figure 1.1) and one example for learning through text-image pairs (right, Figure 1.1). As shown on the left side of the figure, the malignancy diagnosis of histopathological images can be transformed into a multi-view learning problem, where each view is at a different magnification level from the same whole-side image, and as a result, the images in multiple views can form image sequences; on the right side of the figure, the text-image matching problem essentially deals with the joint learning of the two modalities, and the learned representations could later on be used in the similarity analysis for image ranking or retrieval, image captioning and text-to-image synthesis tasks.

Multi-view learning has been widely applied in many domains such as cross-media retrieval, natural language processing and video analysis [4], where most applications have explicit multi-view data, for example, in language translation, the text representations in different languages naturally agree with the concept of multi-view. In other cases, however, the unstructured data often makes the concept of multi-view not as obvious, and it is required that one reshape the original data into associated sequences or pairs of data for the multi-view learning approaches to be applicable. Another challenge concerns the learning of underlying connections among data originated from multiple sources or multi-view data originated from the same source. Since we are still at the very early stage of machine learning as compared to human intelligence, there exists no such a unified framework that fits universally all the task scenarios; therefore, the solutions to many machine learning systems are often problem-based at the current time.

In this thesis, we address several problems in both medical and non-medical domains, using a multi-view learning approach with the emphasis on learning the underlying connections among the multiple distinct feature sets representing the given multi-view data (*i.e.*, image sequences in a unimodal setting and image-text pairs in a multi-modal setting).

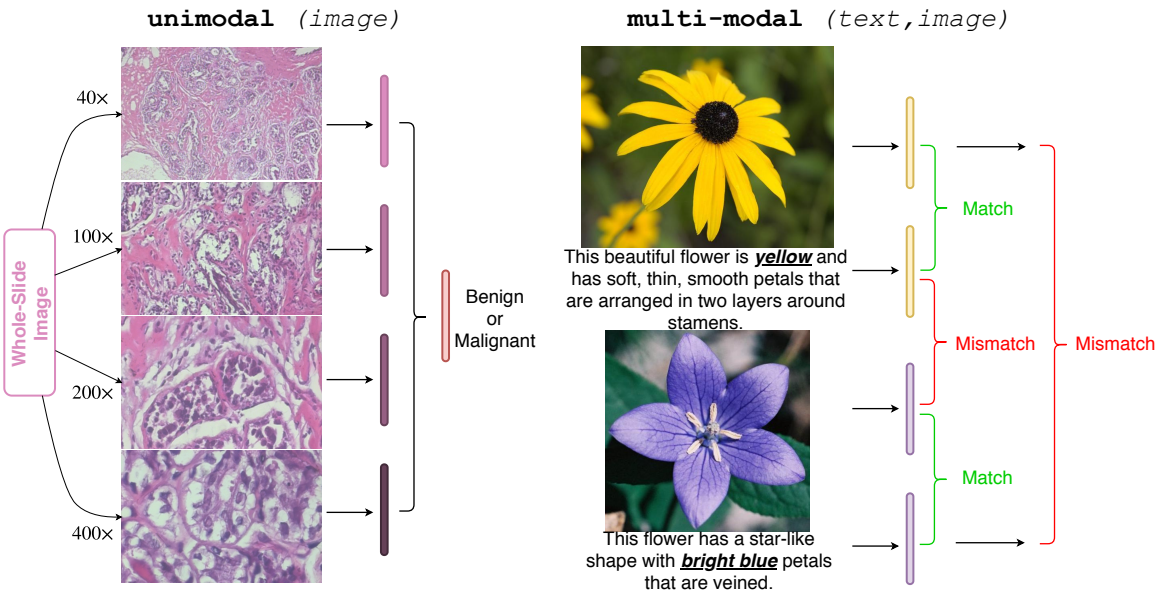


Figure 1.1: Examples of learning through image sequences (*i.e.*, all views in the image modality in a unimodal setting) (left) and text-image pairs (*i.e.*, views in both text and image modalities in a multi-modal setting) (right). On the left side, the histopathological malignancy diagnosis problem can be transformed into a multi-view learning problem, where each view is at a different magnification level from the same whole-side image; on the right side, the text-image matching problem is a multi-view learning problem with each view in a different modality.

1.2 Thesis Outline

The outline of this thesis is listed as follows.

Chapter 2 gives a general background on several deep learning methods that are used in this thesis, including convolutional neural network, long short-term memory network and generative adversarial neural network. It also summarizes in a high level the two general categories of current multi-view learning techniques, namely multi-view representation alignment and multi-view representation fusion.

Learning through text-image pairs

- **Chapter 3** points out an unrecognized problem of current state-of-the-art methods for text-to-image synthesis, and introduces a dual adversarial inference mechanism to learn representations that are aligned among multiple views in a multi-modal setting (*i.e.*, text and image modalities), given the paired text-image data. The learned representations include not only the content information that is present in both views, but also the style information that is present in the image view while missing in the text view. The dual adversarial inference is demonstrated to improve the performance of the text-to-image synthesis task.

Learning through image sequences

- **Chapter 4** provides an example application of multi-view learning for cell phenotype classification problem, in which each view is explicitly defined by each channel capturing independent features of cell phenotypes (*i.e.*, cell components). In this case, the image sequences are constructed by combining images of each individual channel.

- **Chapter 5** presents a case-based approach for histopathological malignancy diagnosis, where a case is defined as a sequence of histopathological images at multiple magnification levels. Given that each magnification level represents a set of features that is complementary to each other, it is shown that a better representation of histopathological images can be learned through the joint learning of image sequences from multiple magnification levels.

- **Chapter 6** studies the disease progression problem that is commonly exist in medical image recognition. To better solve this problem, disease progression learning is introduced to emphasize the learning of underlying connections among multiple stages of a disease, with each stage being a sequential view of the disease. Disease progression learning is evaluated on a diabetic retinopathy staging problem, and shows significant improvement in the staging accuracy.

Chapter 7 concludes this thesis with a review of the main contributions, and also provides discussions on the possible directions for future research work.

In summary, we address several problems in both medical and non-medical domains, including text-to-image synthesis, cell phenotype classification, histopathological malignancy diagnosis and disease progression learning, from the perspective of multi-view learning. Based on the number of modalities that the data is presented in, we divide this thesis into two main parts: (1) learning through text-image pairs in a multi-modal setting (*i.e.*, views in both text and image modalities), and (2) learning through image sequences in a unimodal setting (*i.e.*, all views in the image modality). In addition, we cover examples in both categories of multi-view representation alignment and multi-view representation fusion. More specifically, the text-to-image synthesis requires the representations to be aligned as matched or mismatched text-image pairs while the disease progression learning aligns the representations from different disease stages in a sequential order. For the cell phenotype classification problem and the histopathological malignancy diagnosis problem on the other hand, we focus on the fusion of the learned representations from multiple views for a better generalized representation of the data.

1.3 Summary of Contributions

The main contributions of this thesis are as follows.

In Chapter 3, we propose dual adversarial inference for the text-to-image synthesis problem. This is the first time an attempt has been made to explicitly learn two variables that are disentangled for content and style in the context of text-to-image synthesis using inference. By learning disentangled representations of content and style, we can generate images that respect the content information from a text source while controlling style by inferring the style information from a style source. We show that capturing these subtleties is important to learn richer representations of the data, and by incorporating inference we improve on the state-of-the-art in image quality while maintaining comparable variability and visual-semantic similarity when evaluated on the Oxford-102, CUB and COCO datasets. The related paper:

- **Qicheng Lao**, Mohammad Havaei, Ahmad Pesaranghader, Francis Dutil, Lisa Di Jorio and Thomas Fevens. Dual Adversarial Inference for Text-to-Image Synthesis. In IEEE *International Conference on Computer Vision*, ICCV 2019.

In Chapter 4, we propose to solve the image-based cell phenotype classification problem using deep residual network and its variants via multi-channel learning. Instead of using segmented images of individual cell as data samples, our approach uses the raw image with multiple cells directly, thus avoids the time-consuming segmentation step and improves the efficiency. We demonstrate the potential of applying deep residual network and its variants in high-content screening (*i.e.*, cell phenotype classification) that can overcome issues associated with analyzing high-content screening data, such as exhaustive preprocessing and inefficient learning. The related papers:

- **Qicheng Lao** and Thomas Fevens. Cell Phenotype Classification using Deep Residual Network and Its Variants. *International Journal of Pattern Recognition and Artificial Intelligence*, IJPRAI 2019, doi: 10.1142/S0218001419400172.

- **Qicheng Lao**, Haoran Sun and Thomas Fevens. Segmentation-Free Cell Phenotype Classification using Deep Residual Neural Networks. In *International Conference on Pattern Recognition and Artificial Intelligence*, ICPRAI 2018.

In Chapter 5, we transform a histopathological malignancy diagnosis problem into a multi-view learning problem by constructing the image sequences from multiple magnification levels. Our proposed case-based approach makes a diagnosis decision based on features learned in combination at multiple magnification levels, which can help to build a more reasonable and reliable computer aided diagnosis system. The related paper:

- **Qicheng Lao** and Thomas Fevens. Case-Based Histopathological Malignancy Diagnosis using Convolutional Neural Networks. In *British Machine Vision Conference*, BMVC 2017.

In Chapter 6, we propose to leverage disease progression learning to emphasize the learning of underlying connections among multiple stages of a disease for medical image recognition problems. We experiment with long short-term memory network to model the disease progression after feature extraction using a shared vision model (*i.e.*, convolutional neural network) for the images from each stage. We show that by leveraging disease progression learning, the disease staging accuracy can be improved. The related paper:

- **Qicheng Lao**, Thomas Fevens and Boyu Wang. Leveraging Disease Progression Learning for Medical Image Recognition. In IEEE *International Conference on Bioinformatics and Biomedicine*, BIBM 2018.

1.4 Contribution from Authors

The contribution from authors is listed as follows based on the published papers in a chronicle order.

Case-Based Histopathological Malignancy Diagnosis using Convolutional Neural Networks

Qicheng Lao	Conception and design of the study, experimental work, data analysis, writing the original draft, editing and proofing
Thomas Fevens	Research supervision, funding, editing and proofing

Segmentation-Free Cell Phenotype Classification using Deep Residual Neural Networks

Qicheng Lao	Conception and design of the study, experimental work, data analysis, writing the original draft, editing and proofing
Haoran Sun	Editing and proofing
Thomas Fevens	Research supervision, funding, editing and proofing

Leveraging Disease Progression Learning for Medical Image Recognition

Qicheng Lao	Conception and design of the study, experimental work, data analysis, writing the original draft, editing and proofing
Boyu Wang	Editing and proofing
Thomas Fevens	Research supervision, funding, editing and proofing

Cell Phenotype Classification using Deep Residual Network and Its Variants

Qicheng Lao	Conception and design of the study, experimental work, data analysis, writing the original draft, editing and proofing
Thomas Fevens	Research supervision, funding, editing and proofing

Dual Adversarial Inference for Text-to-Image Synthesis

Qicheng Lao	Conception and design of the study, experimental work, data analysis, writing the original draft, editing and proofing
Mohammad Havaei	Brainstorming, experimental work on Bernoulli distribution, critical revisions, editing and proofing
Ahmad Pesaranghader	Brainstorming, editing and proofing
Francis Dutil	Brainstorming, editing and proofing
Lisa Di Jorio	Administrative support, funding, brainstorming, editing and proofing
Thomas Fevens	Research supervision, funding, brainstorming, editing and proofing

Chapter 2

Background

In this chapter, we first give a general background on multiple deep learning methods including convolutional neural network, long short-term memory network and generative adversarial neural network in Section 2.1. Next, in Section 2.2, we present two categories of multi-view learning methods: multi-view representation alignment and multi-view representation fusion.

2.1 Deep Learning Methods

Deep learning is a group of machine learning algorithms that involve much more levels of composition, *i.e.*, learned functions or learned concepts, compared to traditional machine learning [17]. Learning such great amount of levels of composition typically requires big data and powerful computation. Recently, the availability of large datasets and the emergence of Graphics Processing Units (GPUs) with library supports (CUDA, OpenCL) have fueled the development of deep learning, making it the state-of-the-art for visual object detection, speech recognition, language understanding and many other artificial intelligent tasks. More details about the development of deep learning can be found in the survey paper [18]. In this section, we introduce three popular deep learning methods that are used in this thesis: convolutional neural network (Section 2.1.1), long short-term memory network(Section 2.1.2) and generative adversarial neural network (Section 2.1.3).

2.1.1 Convolutional neural network

Convolutional neural network (CNN) is one of the deep learning approaches specialized in image-based tasks (*e.g.*, classification, segmentation), which has been widely demonstrated to outperform traditional state-of-the-art machine learning algorithms [1]. As a special type of multi-layer neural network, CNN contains from several up to hundreds of convolutional layers inside the network. It was first introduced by LeCun as LeNet for hand-written digit recognition, and the purpose is

to recognize visual patterns directly from images with minimal preprocessing [19]. Indeed, one of the biggest advantages of CNN, compared to other traditional approaches, is that for CNN, all features used for classification are automatically learned by the network itself. Thus there is no need to extract predefined hand-crafted features, which is a quite difficult and time-consuming step for most practical problems.

While LeNet started the era of convolutional neural networks, AlexNet [1] is the first popular CNN model that was used on a large-scale dataset of natural images for general purpose classification tasks. Compared to five-layer LeNet, AlexNet has eight layers, thus is slightly deeper, and besides that, it also uses new techniques such as Local Response Normalization, Dropout, Rectified Linear Unit (ReLU) for the nonlinearity activation function, and data augmentation techniques to improve the performance. The techniques are explained in details as follows.

Local Response Normalization Inspired by lateral inhibition in neuroscience where the activation of a neuron is also affected by its neighboring neurons, local response normalization is implemented in AlexNet by the following formula:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}, \tag{2.1}$$

where $a^i_{x,y}$ is the source output of kernel i at position (x, y), $b^i_{x,y}$ is the response-normalized output, and the constants k, n, α and β are hyper-parameters. The summation runs over n size neighborhood, where N is the total number of kernels in that layer. Despite its usage in AlexNet, local response normalization has been replaced by Batch Normalization [20] or other normalization techniques nowadays .

ReLU Nonlinearity Instead of using traditional saturating nonlinearities such as Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

or Tanh:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{2.3}$$

AlexNet works with Rectified Linear Unit (ReLU) (Figure 2.1(a)):

$$f(x) = \max(0, x). \tag{2.4}$$

It is shown that not only ReLU speed up the training, but also has a great impact on the model performance especially for deep models trained on large datasets. As demonstrated in Figure 2.1(b),

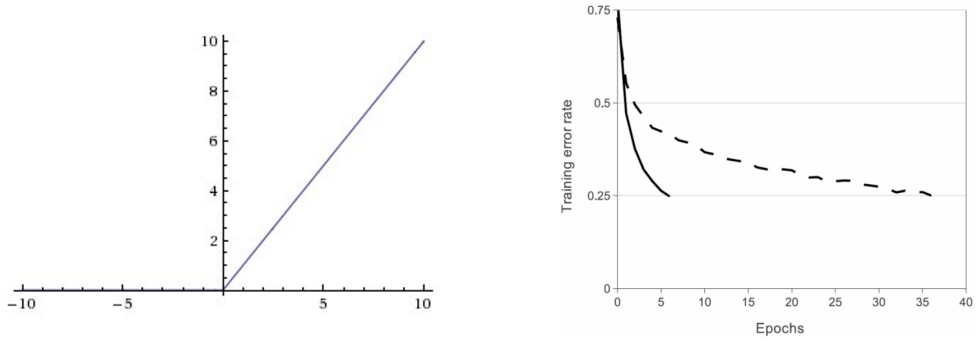
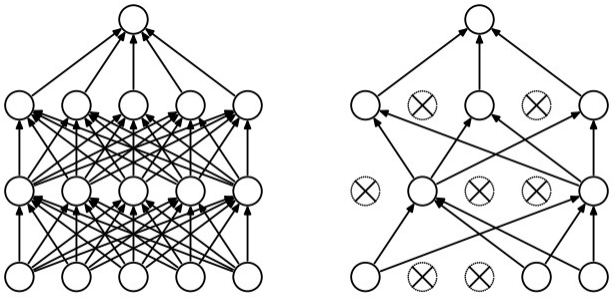


Figure 2.1: (a) Rectified Linear Unit activation function; (b) Convergence comparison between ReLU and Tahn activation (Extracted from [1]).



(a) Standard Neural Net (b) After applying dropout.

Figure 2.2: Neural networks before and after applying dropout. Extracted from [2].

a convolutional neural network with ReLUs learns six times faster than an equivalent network with Tanh activation.

Dropout Dropout is another effective regularization technique to prevent over-fitting [2]. AlexNet also adopts Dropout with a probability of 0.5, which means half of the hidden neurons will be dropped out and not participating the forward-pass or back-propagation. Figure 2.2 illustrates the idea of dropout.

After AlexNet, the CNN models tend to be deeper and deeper in order to achieve even better performance, among which VGGNet [21] and GoogleLeNet [22] are examples. Most of the recent successful applications in visual recognition tend to exploit very deep models, suggesting that the depth of convolutional neural networks has a crucial importance in the model performance [21, 23, 24]. However, deep models are difficult to train. One notorious problem is known as gradient vanishing or exploding, which has been largely alleviated by various normalization techniques, such as Batch Normalization [20]. When deep models finally start to converge, another problem has also been noticed on multiple datasets, where both the training and testing accuracy degrade rapidly with

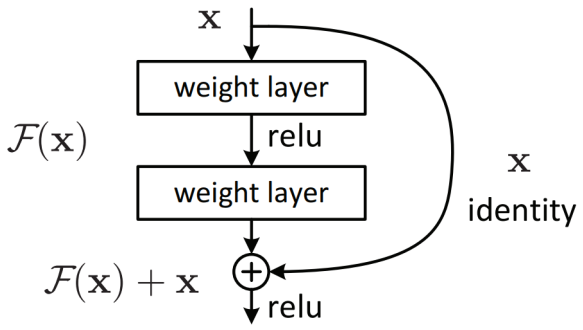


Figure 2.3: A unit of residual learning block with a residual mapping $\mathcal{F}(x)$ and an identity mapping x. Extracted from [3].

the increase of model depth.

To solve the accuracy degradation problem with deep models, residual network (ResNet) has been developed by He *et al*. [3], where the network contains residual units parallel to normal convolutional layers for residual learning (Figure 2.3). Denoting $\mathcal{H}(x)$ as the desired underlying mapping to be fit by a neural network, where x is the input to the network, a residual function could be defined as $\mathcal{F}(x) := \mathcal{H}(x) - x$ (assuming that the dimensions of the input x and output $y = \mathcal{H}(x)$ are the same). Thus the original mapping $\mathcal{H}(x)$ becomes $\mathcal{F}(x) + x$. This reformulation divides the original mapping into two parts: the residual mapping $\mathcal{F}(x)$ and the identity mapping x. Instead of optimizing the original mapping $\mathcal{H}(x)$ directly, it could be easier to optimize the residual mapping $\mathcal{F}(x)$, as the degradation problem suggests that the difficulties may come from the approximation of identity mapping. In the extreme cases, for example, if the identity mapping is optimal already, it would be much easier for the residual to approximate zero than to fit an identity mapping by stacked nonlinear layers. The residual learning can be mathematically formulated as the following:

$$y = \mathcal{F}(x, \{W_i\}) + x, \tag{2.5}$$

where $\mathcal{F}$ denotes the residual mapping function. In cases where the dimensions of x and y mismatch, a linear projection W_s can be performed on the input x in order to match the dimensions:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x. \tag{2.6}$$

New designs of the residual unit have also been proposed to improve the generalization thus giving improved accuracy [25]. And more recently, several ResNet based architectures are developed, each of which improves previous state-of-the-art performance on the benchmark datasets. By simply widening the residual blocks, wide residual network (WRN) has been proven much more effective than deep residual network, for example, a 16-layer WRN can achieve similar performance as a

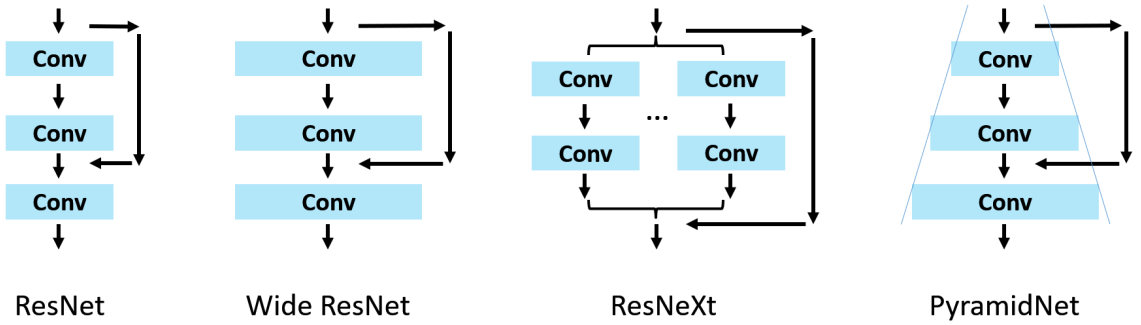


Figure 2.4: Comparison of the original residual network (ResNet) and its variants, including wide residual network (WRN), aggregated deep residual network (ResNeXt) and pyramidal residual network (PyramidNet).

1000-layer regular ResNet while the former is much faster to train [26]. Another way to increase the width of residual network is through aggregated residual transformation, resulting in aggregated deep residual network (ResNeXt) [27]. Deep pyramidal residual network (PyramidNet), on the other side, uses the strategy to increase the number of filters gradually like a pyramid as the layer goes deeper [28]. The comparison of ResNet and its variants (WRN, ResNeXt and PyramidNet) is shown in Figure 2.4.

2.1.2 Recurrent neural network

Recurrent neural network (RNN) is another type of neural networks that has cycles designed to remember past information for time series data. The carrying memory can be represented by hidden state denoted as h, and at the time step t, the hidden state can be computed as:

$$h_t = \sigma(W_h X_t + W_r h_{t-1}), \tag{2.7}$$

where X_t is the input for the current time step, and h_{t-1} is the hidden state from the previous time step. The network is parameterized by two weight matrices W_h and W_r (note that the bias terms are ignored for simplicity). The output y at time step t is then given by:

$$y_t = \sigma(W_o h_t). \tag{2.8}$$

Figure 2.5 illustrates the computation graph of the unfolded RNN. The problem of vanilla RNN lies in its inability to capture long-term dependency as the time step t goes large, due to the gradient vanishing or exploding problem when the network is trained by the back propagation through time (BPTT) algorithm [29, 30]. Consider the gradient of the error $\mathcal{E}_t$ at time step t concerning the

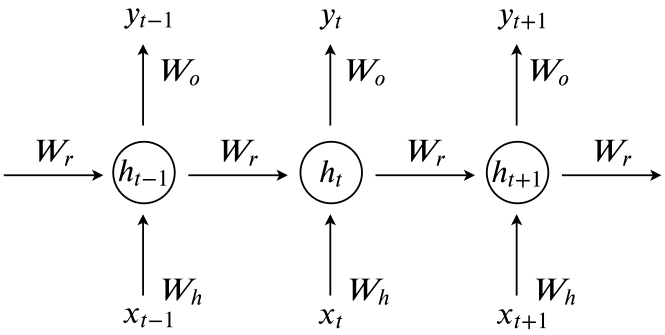


Figure 2.5: Recurrent neural network. x: inputs; y: outputs; h; hidden states; W: weights

weights $W = [W_h, W_r]$, it can be written as:

$$\frac{\partial \mathcal{E}_t}{\partial W} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \right) \tag{2.9}$$

$$= \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \left(\prod_{k < i \le t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W} \right) \tag{2.10}$$

$$= \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \left(\prod_{k < i \le t} diag(\sigma'(W_h X_i + W_r h_{i-1})) W_r \right) \frac{\partial h_k}{\partial W} \right) \tag{2.11}$$

$$= \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \left(W_r^{t-k} \prod_{k < i \le t} diag(\sigma'(W_h X_i + W_r h_{i-1})) \right) \frac{\partial h_k}{\partial W} \right). \tag{2.12}$$

If W_r is small, then the term W_r^{t-k} gets vanished as the gap between time step t and k gets large, and as a result, $\frac{\partial \mathcal{E}_t}{\partial W} \to 0$. In such case, the networks' weights W stop getting updated when using the gradient descent (GD) algorithm:

$$W \leftarrow W - \alpha \frac{\partial \mathcal{E}_t}{\partial W} \approx W. \tag{2.13}$$

To overcome this problem and stabilize the gradients, multiple strategies have been proposed, including truncated back-propagation (*i.e.*, stop the forward-pass and back-propagation at certain time steps), input reversal (*i.e.*, reverse the order of the input sequence), identity initialization [31] and gradient clipping (*i.e.*, set a maximum value for the gradient, shown in Algorithm 2.1) [30]. Among them, long short-term memory (LSTM), which has been proposed by Hochreiter and Schmidhuber [32], is one of the most popular solutions. LSTM is a variant of recurrent neural network, where memory cell (shown in Figure 2.6) is specially designed with three different gates (input gate i,

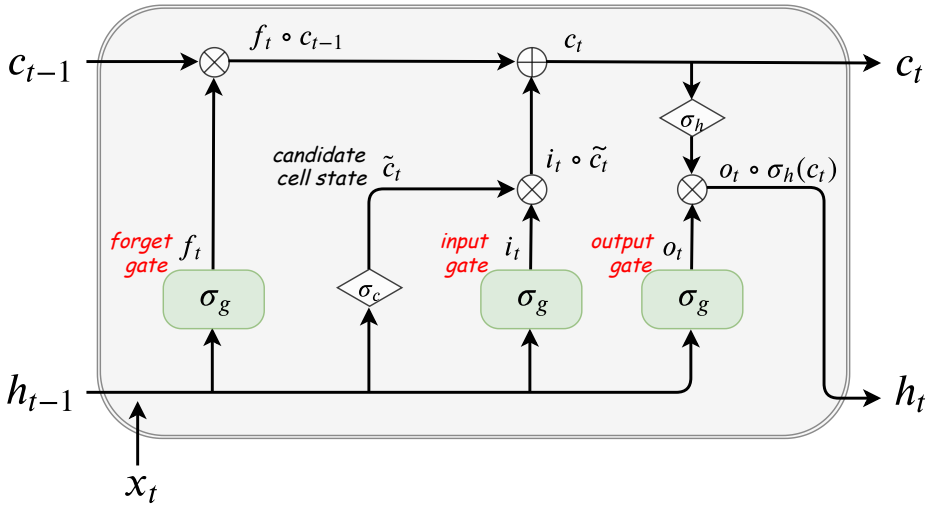


Figure 2.6: A cell block of long short-term memory. The information flow is controlled by three gates: input gate i, forget gate f and output gate o. All variables are defined through equations from 2.14 to 2.19.

Algorithm 2.1: Gradient clipping by norm

$\hat{\boldsymbol{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\boldsymbol{g}}\| \geq threshold$ **then**
    $\hat{\boldsymbol{g}} \leftarrow \frac{threshold}{\|\hat{\boldsymbol{g}}\|}\hat{\boldsymbol{g}}$
end

forget gate f and output gate o) to control the information flow:

$$i_t = \sigma_g(W_i X_t + U_i h_{t-1}), \tag{2.14}$$

$$f_t = \sigma_g(W_f X_t + U_f h_{t-1}), \tag{2.15}$$

$$o_t = \sigma_g(W_o X_t + U_o h_{t-1}). \tag{2.16}$$

Based on the above three gates, the candidate cell state $\hat{c}$, cell state c and hidden state h are then given as the following:

$$\hat{c}_t = \sigma_c(W_c X_t + U_c h_{t-1}), \tag{2.17}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t, \tag{2.18}$$

$$h_t = o_t \circ \sigma_h(c_t), \tag{2.19}$$

where σ_g is sigmoid function (Equation 2.2), and σ_c, σ_h can be hyperbolic tangent function (Equation 2.3). With the integration of cell state, the gradient at time step t now turns into:

$$\frac{\partial \mathcal{E}_t}{\partial W} = \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial c_t} \frac{\partial c_t}{\partial c_k} \frac{\partial c_k}{\partial W} \right) \tag{2.20}$$

$$= \sum_{1 \leq k \leq t} \left(\frac{\partial \mathcal{E}_t}{\partial c_t} \left(\prod_{k < j \leq t} \frac{\partial c_j}{\partial c_{j-1}} \right) \frac{\partial c_k}{\partial W} \right), \tag{2.21}$$

where the recursive derivative part is changed to:

$$\frac{\partial c_j}{\partial c_{j-1}} = f_j + i_j \sigma_c'(\cdot) U_c \frac{\partial h_{j-1}}{\partial c_{j-1}} + \frac{\partial c_j}{\partial i_j} \frac{\partial i_j}{\partial h_{j-1}} \frac{\partial h_{j-1}}{\partial c_{j-1}} + \frac{\partial c_j}{\partial f_j} \frac{\partial f_j}{\partial h_{j-1}} \frac{\partial h_{j-1}}{\partial c_{j-1}}. \tag{2.22}$$

Therefore, the network can learn to adjust the value of f_j to be close to 1, in order to bring the value of $\frac{\partial c_j}{\partial c_{j-1}}$ to be close to 1, thus preventing the gradients from vanishing or exploding.

Although there are several further variants of LSTM have been proposed such as gated recurrent unit (GRU) [33], a large-scale experiment has shown that none of those variants can significant improve the standard LSTM [34].

2.1.3 Generative adversarial network

The generative adversarial network (GAN) [35] framework has been proven to be one of the most successful generative models with numerous promising results for image generation. Inspired by the game theory, it is designed with two adversarial components: (1) a generator G that generates plausible images $\tilde{\boldsymbol{x}}$ by mapping a prior distribution (*e.g.*, $\mathcal{N}(0, 1)$) to the generated data distribution p_g in the image space:

$$\tilde{\boldsymbol{x}} = G(\boldsymbol{z}), \boldsymbol{z} \sim p_z(\boldsymbol{z}); \tag{2.23}$$

and (2) a discriminator D that tries to distinguish the generated images from real images $\boldsymbol{x}$ ($\boldsymbol{x} \sim p_{\text{data}}$). The overall framework is shown in Figure 2.7. The two models are trained alternatively to compete with each other as a minimax game by optimizing the following objective function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z}[\log(1 - D(G(\boldsymbol{z})))], \tag{2.24}$$

where the generator is trained to minimize the above objective function while the discriminator is trained to maximize it. Theoretically, the GAN models converge when the discriminator and the generator reach a Nash equilibrium. The optimal discriminator for any given G is:

$$D_G^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2.25}$$

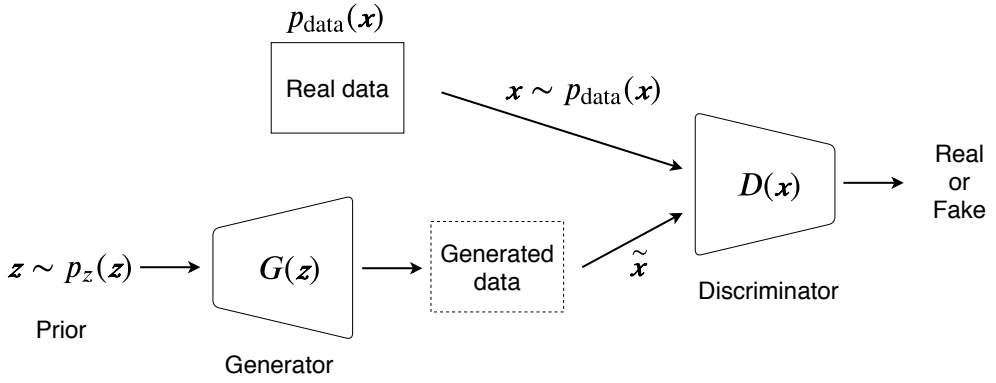


Figure 2.7: Generative adversarial network. A generator G generates plausible images $\tilde{\boldsymbol{x}}$ from a prior distribution $p_z(\boldsymbol{z})$, and a discriminator D distinguishes between the generated images $\tilde{\boldsymbol{x}}$ and real images $\boldsymbol{x}$.

that maximizes the objective function in Equation 2.24, which can be rewritten as:

$$V(D,G) = \int_x p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_z(\boldsymbol{z}) \log(1 - D(G(\boldsymbol{z})))dz \tag{2.26}$$

$$= \int_x p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx. \tag{2.27}$$

Replacing $D(\boldsymbol{x})$ by $D^*_G(\boldsymbol{x})$ and integrating Equation 2.25, the objective function can be further formulated as:

$$C(G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D^*_G(G(\boldsymbol{z})))] \tag{2.28}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))] \tag{2.29}$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log \frac{p_{\text{g}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}] \tag{2.30}$$

$$= -\log(4) + KL\left(p_{\text{data}} \| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \| \frac{p_{\text{data}} + p_g}{2}\right) \tag{2.31}$$

$$= -\log(4) + 2 \cdot JSD(p_{\text{data}} \| p_g). \tag{2.32}$$

Therefore, minimizing the GAN objective function (Equation 2.24) is equivalent to minimizing the Jensen-Shannon divergence between p_{data} and p_g, given the condition that the discriminator is trained well. Alternative divergence measures have also been proposed by defining a general form of divergences [36]:

$$D_f(P\|Q) = \int_\chi q(x) f\left(\frac{p(x)}{q(x)}\right) dx, \tag{2.33}$$

where the divergence is determined by function f. When f is defined as:

$$f(u) = u \log u - (u+1)\log(u+1), \tag{2.34}$$

$D_f(p_{\text{data}} \| p_g)$ recovers Equation 2.32. In practice, in order for the generator to get gradients in the early stage of the training, it is required to train the generator to maximize $\log(D(G(\boldsymbol{z})))$ instead of minimizing $\log(1 - D(G(\boldsymbol{z})))$. However, despite this, GAN training is often unstable [37], and various techniques have been proposed to improve GAN training, such as injecting noise to the generated images [37] and label smoothing [38].

Arguing that the training difficult may come from the design of the original GAN objective function, Wasserstein GAN (WGAN) has been proposed to use Wasserstein distance $W(p, q)$ as the objective [39]. The minmax optimization is then updated to:

$$\min_G \max_{D \in \mathbb{F}} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[D(\boldsymbol{x})] - \mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_g}[D(\tilde{\boldsymbol{x}})], \tag{2.35}$$

where $\mathbb{F}$ is the set of 1-Lipschitz functions. To enforce the Lipschitz constraint on D, WGAN applies a clipping strategy to keep the weights of the discriminator within a certain range. An alternative way to enforce the Lipschitz constraint is to use gradient penalty as been proposed in WGAN-GP [40], updating the objective to:

$$\mathbb{E}_{\tilde{\boldsymbol{x}} \sim p_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[D(\boldsymbol{x})] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim p_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2]. \tag{2.36}$$

This is based on the fact that a differentiable function is 1-Lipschtiz if and only if it has gradients with norm at most 1 everywhere.

Other popular and fundamental GAN models include (1) conditional GAN (CGAN), where a conditioning factor encoding the desired properties of generated images is given as additional input to the generator G, and (2) auxiliary classifier GAN (ACGAN), where the discriminator D is trained to give the probability distribution over the class labels in addition to the source labels (*i.e.*, fake or real). The objective function of CGAN is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{c})))], \tag{2.37}$$

where $\boldsymbol{c}$ is the conditioning factor. And the objective function of ACGAN is:

$$L_D^{\text{ACGAN}} = L_D^{\text{GAN}} + \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{c})}[-\log p(\boldsymbol{c}|\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z}[-\log p(\boldsymbol{c}|G(\boldsymbol{z}, \boldsymbol{c}))], \tag{2.38}$$

$$L_G^{\text{ACGAN}} = L_G^{\text{GAN}} + \mathbb{E}_{\boldsymbol{z} \sim p_z}[-\log p(\boldsymbol{c}|G(\boldsymbol{z}, \boldsymbol{c}))], \tag{2.39}$$

where L_D^{GAN} and L_G^{GAN} are the original GAN objective functions that are defined in Equation 2.24.

2.2 Multi-View Representation Learning

Multi-view representation learning aims to learn the representations of the given multi-view data that are useful to the subsequent tasks. Due to the great success of deep learning methods in single-view representation learning, approaches based on deep neural networks have also been explored in multi-view representation learning settings. Based on the use case of learned representations, current multi-view representation learning approaches can be divided into two main categories: multi-view representation alignment and multi-view representation fusion [4]. As seen from Figure 2.8, the goal of multi-view representation alignment is to align the features that are learned from different views, while in multi-view representation fusion, the model aims to learn a unified representation that contains features from different views. In this section, we introduce several multi-view representation learning approaches in both categories (multi-view representation alignment in Section 2.2.1 and multi-view representation fusion in Section 2.2.2).

2.2.1 Multi-view representation alignment

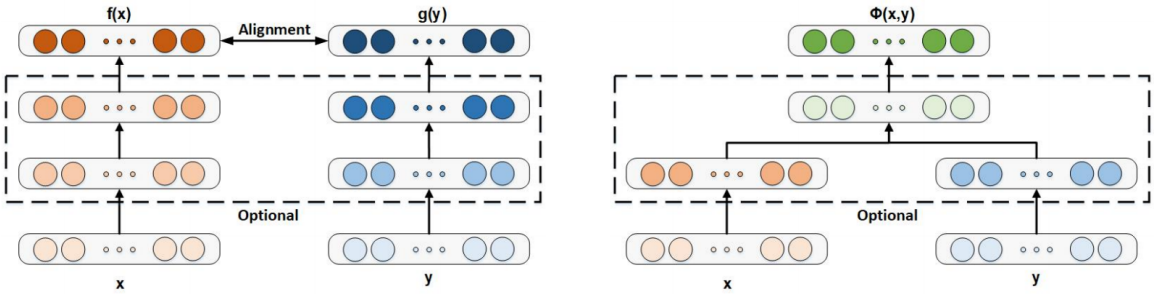
In multi-view learning, a group of methods are based on canonical correlation analysis (CCA) [41], which can be used for the modeling of the relationships between different sets of variables. Given a pair of data in two views with N samples, $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$, where $\boldsymbol{x}_i \in \mathbb{R}^{D_x}$ and $\boldsymbol{y}_i \in \mathbb{R}^{D_y}$ for $i = 1, \ldots, N$, CCA computes two linear projections $\boldsymbol{w}_x \in \mathbb{R}^{D_x}$ and $\boldsymbol{w}_y \in \mathbb{R}^{D_y}$, such that the correlation between the two views are maximized. The correlation coefficient is given by:

$$\rho = \frac{\boldsymbol{w}_x^T \boldsymbol{X}\boldsymbol{Y}^T \boldsymbol{w}_y}{\sqrt{(\boldsymbol{w}_x^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{w}_x)(\boldsymbol{w}_y^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{w}_y)}}. \tag{2.40}$$

Alternatively, it can also be written as:

$$\begin{aligned} \max_{\boldsymbol{w}_x, \boldsymbol{w}_y} \quad & \boldsymbol{w}_x^T \boldsymbol{X}\boldsymbol{Y}^T \boldsymbol{w}_y \\ \text{s.t.} \quad & \boldsymbol{w}_x^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{w}_x = 1, \boldsymbol{w}_y^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{w}_y = 1. \end{aligned} \tag{2.41}$$

The maximization of the correlations between the two views in CCA approach can be viewed as the alignment of the two views. An extension of CCA with deep neural networks (referred as deep CCA) has been proposed in [42], where two deep neural networks f and g are used to extract features from each view, and with additional regularization, the canonical correlation formula is



(a) Muti-view representation alignment (b) Muti-view representation fusion

Figure 2.8: Multi-view representation learning approaches can be divided into two main categories: multi-view representation alignment (a) and multi-view representation fusion (b). Multi-view representation alignment aims to align the features that are learned from different views, while multi-view representation fusion focuses on learning a unified representation that contains features from different views. Extracted from [4].

proposed as:

$$\begin{aligned} \max_{\boldsymbol{W}_f, \boldsymbol{W}_g, \boldsymbol{U}, \boldsymbol{V}} \quad & \frac{1}{N}\text{tr}(\boldsymbol{U}^T f(\boldsymbol{X}) g(\boldsymbol{Y})^T \boldsymbol{V}) \\ \text{s.t.} \quad & \boldsymbol{U}^T \left(\frac{1}{N} f(\boldsymbol{X}) f(\boldsymbol{X})^T + r_x \boldsymbol{I} \right) \boldsymbol{U} = \boldsymbol{I}, \\ & \boldsymbol{V}^T \left(\frac{1}{N} g(\boldsymbol{Y}) g(\boldsymbol{Y})^T + r_y \boldsymbol{I} \right) \boldsymbol{V} = \boldsymbol{I}, \\ & \boldsymbol{u}_i^T f(\boldsymbol{X}) g(\boldsymbol{Y})^T \boldsymbol{v}_j = 0, \text{for } i \neq j, \end{aligned} \tag{2.42}$$

where $\boldsymbol{W}_f$ and $\boldsymbol{W}_g$ are the weights for networks, $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_L]$ are the CCA directions, with L denoting the feature length. The performance of deep CCA can be further improved by adding an autoencoder regularization term (*i.e.*, reconstruction loss), resulting in the following objective:

$$\begin{aligned} \min_{\boldsymbol{W}_f, \boldsymbol{W}_g, \boldsymbol{U}, \boldsymbol{V}} \quad & -\frac{1}{N}\text{tr}(\boldsymbol{U}^T f(\boldsymbol{X}) g(\boldsymbol{Y})^T \boldsymbol{V}) \\ & + \frac{\lambda}{N} \sum_{i=1}^{N} (\|\boldsymbol{x}_i - p(f(\boldsymbol{x}_i))\|^2 + \|\boldsymbol{y}_i - q(g(\boldsymbol{y}_i))\|^2) \\ \text{s.t.} \quad & \boldsymbol{U}^T \left(\frac{1}{N} f(\boldsymbol{X}) f(\boldsymbol{X})^T + r_x \boldsymbol{I} \right) \boldsymbol{U} = \boldsymbol{I}, \\ & \boldsymbol{V}^T \left(\frac{1}{N} g(\boldsymbol{Y}) g(\boldsymbol{Y})^T + r_y \boldsymbol{I} \right) \boldsymbol{V} = \boldsymbol{I}, \\ & \boldsymbol{u}_i^T f(\boldsymbol{X}) g(\boldsymbol{Y})^T \boldsymbol{v}_j = 0, \text{for } i \neq j, \end{aligned} \tag{2.43}$$

where p and g are the decoders for f and g respectively.

In addition to the CCA based alignment, multi-view features can also be simply aligned with a distance metric such as Euclidean distance as the objective:

$$\min_{\boldsymbol{W}_f, \boldsymbol{W}_g} \frac{1}{N} \sum_{i=1}^{N} \|f(\boldsymbol{x}_i) - g(\boldsymbol{y}_i)\|_2^2. \tag{2.44}$$

To maximize the similarity between unmatched pairs and minimize the similarity between matched pairs, the following objective function based on similarity measure has also been proposed for visual-semantic representation learning [43]:

$$\sum_{i \neq j} \max(0, m - \textit{sim}(\boldsymbol{x}_i, \boldsymbol{y}_i) + \textit{sim}(\boldsymbol{x}_i, \boldsymbol{y}_j)), \tag{2.45}$$

where $\textit{sim}(\cdot)$ is the function for similarity measurement and m is the margin distance.

2.2.2 Multi-view representation fusion

In multi-view representation fusion, the goal is to learn a unified representation $\boldsymbol{h}$ from all the given views, such that $\boldsymbol{h}$ is optimal for the subsequent task. For example, given n views $[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$, the model learns:

$$\boldsymbol{h} = \phi(\boldsymbol{x}_1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n), \tag{2.46}$$

and $\boldsymbol{h}$ integrates all knowledge of the data.

Similar to multi-view representation alignment, deep autoencoder has also been applied in the case of multi-view representation fusion. In order for the features to be fused, multiple layers in the autoencoder network are designed to be shared among different views so that shared representations can be learned. Given paired data in two views $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$, where $i = 1, 2, \ldots, N$, the loss function is defined as follows:

$$L = \sum_i \Big(L_{\textit{recon}}(\boldsymbol{x}_i, g(f(\boldsymbol{x}_i, \boldsymbol{y}_i))) + L_{\textit{recon}}(\boldsymbol{y}_i, g'(f(\boldsymbol{x}_i, \boldsymbol{y}_i))) \Big), \tag{2.47}$$

where $L_{\textit{recon}}$ is the reconstruction loss, *e.g.*, $\|\boldsymbol{x}_i - g(f(\boldsymbol{x}_i, \boldsymbol{y}_i))\|^2$ in the case of l_2 loss, f denotes the feature encoder and g is the decoder. The fused representation $\boldsymbol{h}$ can be obtained by $\boldsymbol{h} = f(\boldsymbol{x}, \boldsymbol{y})$.

In a supervised learning setup where the labels are available, multi-view representation fusion has also been widely explored using convolutional neural networks, especially for tasks such as 3D object recognition [44] and action recognition in the video [45]. It has been shown that multi-view representation learning has advantages in performance, since it integrates different feature sets from the multi-view data, therefore giving a more general and robust representation compares to the

single-view learning. In general, there are several ways to fuse the features from multiple views, common strategies include average fusion (Equation 2.48), sum fusion (Equation 2.49), max fusion (Equation 2.50) and concatenation fusion (Equation 2.51):

$$\boldsymbol{h}^{average} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i, \tag{2.48}$$

$$\boldsymbol{h}^{sum} = \sum_{i=1}^{N}\boldsymbol{x}_i, \tag{2.49}$$

$$\boldsymbol{h}^{max} = \max(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N), \tag{2.50}$$

$$\boldsymbol{h}^{cat} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]. \tag{2.51}$$

Some of the above strategies fit naturally with the pooling layers that are commonly used in CNN architectures, such as averaging pooling and max pooling.

Chapter 3

Dual Adversarial Inference for Text-to-Image Synthesis

In this chapter, we address the text-to-image synthesis problem with the emphasis on learning information that is present in one view (*e.g.*, image) but missing in another view (*e.g.*, text), given the paired image-text data. Synthesizing images from a given text description involves engaging two types of information: the content, which includes information explicitly described in the text (*e.g.*, color, composition, etc.), and the style, which is usually not well described in the text (*e.g.*, location, quantity, size, etc.). However, in previous works, it is typically treated as a process of generating images only from the content, *i.e.*, without considering learning meaningful style representations. We aim to learn two variables that are disentangled in the latent space, representing content and style respectively. We achieve this by augmenting current text-to-image synthesis frameworks with a dual adversarial inference mechanism. Through extensive experiments, we show that our model learns, in an unsupervised manner, style representations corresponding to certain meaningful information present in the image that are not well described in the text. The new framework also improves the quality of synthesized images when evaluated on Oxford-102, CUB and COCO datasets.

3.1 Introduction

The problem of text-to-image synthesis is to generate diverse yet plausible images given a text description of the image and a general data distribution of images and matching descriptions. In recent years, generative adversarial networks (GANs) [35] have asserted themselves as perhaps the most effective architecture for image generation, along with their variant Conditional GANs [46], wherein the generator is conditioned on a vector encompassing some desired property of the generated image.

A common approach for text-to-image synthesis is to use a pre-trained text encoder to produce a text embedding from the description. This vector is used as the conditioning factor in a conditional GAN-based model. The very first GAN model for the text-to-image synthesis task [47] uses a noise vector sampled from a normal distribution to capture image style features left out of the text representation, enabling the model to generate a variety of images given a certain textual description. StackGan [48] introduces conditioning augmentation as a way to augment the text embeddings, where a text embedding can be sampled from a learned distribution representing the text embedding space. As a result, current state-of-the-art methods for text-to-image synthesis generally have two sources of randomness: one for the text embedding variability, and the other (noise $\boldsymbol{z}$ given a normal distribution) capturing image variability.

Having two sources of randomness is, however, only meaningful if they represent different factors of variation. Problematically, our empirical investigation of some previously published methods reveals that those two sources can overlap: due to the randomness in the text embedding, the noise vector $\boldsymbol{z}$ then does not meaningfully contribute to the variability nor the quality of generated images, and can be discarded. This is illustrated qualitatively in Figure 3.1 and quantitatively in Figure 3.5 (Section 3.4).

In this work we aim to learn a latent space that represents meaningful information in the context of text-to-image synthesis. To do this, we incorporate an inference mechanism that encourages the latent space to learn the distribution of the data. To capture different factors of variation, we construct the latent space through two independent random variables, representing content ('$\boldsymbol{c}$') and style ('$\boldsymbol{z}$'). Similar to previous work [47], '$\boldsymbol{c}$' encodes image content which is the information in the text description. This mostly includes color, composition, etc. On the other hand, '$\boldsymbol{z}$' encodes style which we define as all other information in the image data that is not well described in the text. This would typically include location, size, pose, and quantity of the content in the image, background, etc. This new framework allows us to better represent information found in both text and image modalities, achieving better results on Oxford-102 [49], CUB [50] and COCO [51] datasets at 64×64 resolution.

The main goal of this work is to learn disentangled representations of style and content through an inference mechanism for text-to-image synthesis. This allows us to use not only the content information described in the text descriptions but also the desired styles when generating images. To that end, we only focus on the generation of low-resolution images (*i.e.*, 64×64). In the literature, high-resolution images are generally produced by iterative refinement of lower-resolution images and thus we consider it a different task, more closely related to generating super-resolution images.

To the best of our knowledge, this is the first time an attempt has been made to explicitly separate the learning of style and content for text-to-image synthesis. We believe that capturing these subtleties is important to learn richer representations of the data. As shown in Figure 3.2, by learning

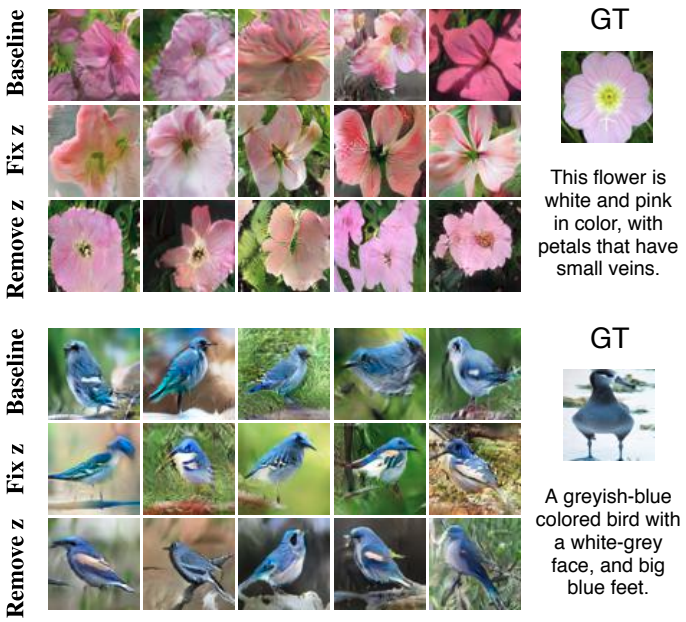


Figure 3.1: Generated images from previous state-of-the-art method (Baseline), fixing the noise vector z in the baseline method (Fix z) and removing the noise vector z in the baseline method (Remove z). The removal of the randomness from the noise source by either fixing z or removing z does not affect the variability nor the quality of generated images, indicating that the noise vector z has no contribution in the synthesis process.

disentangled representations of content and style, we can generate images that respect the content information from a text source while controlling style by inferring the style information from a style source. It is worth noting that although we hope to learn the style from the image modality, the style information could possibly be connected to (or leaked into) some text instances. Despite this, the integration of the style in the model eventually depends on how well it is represented in both modalities. For example, if certain types of style information are commonly present in the text, then according to our definition, those types of information are considered as content. If only a few text instances describe that information however, then it would not be fully representative of a shared commonality among texts and therefore would not be captured as content, and whether it can be captured as style depends on how well it is represented in the image modality. On the other hand, we would also like to explore modalities other than *text* as the content in our future work using the proposed method, which may bring us closer to image-to-image translation [52] if we choose both modalities to be *image*.

The contributions of this work are twofold: (i) we are the first to learn two variables that are disentangled for content and style in the context of text-to-image synthesis using inference; and (ii) by incorporating inference we improve on the state-of-the-art in image quality while maintaining

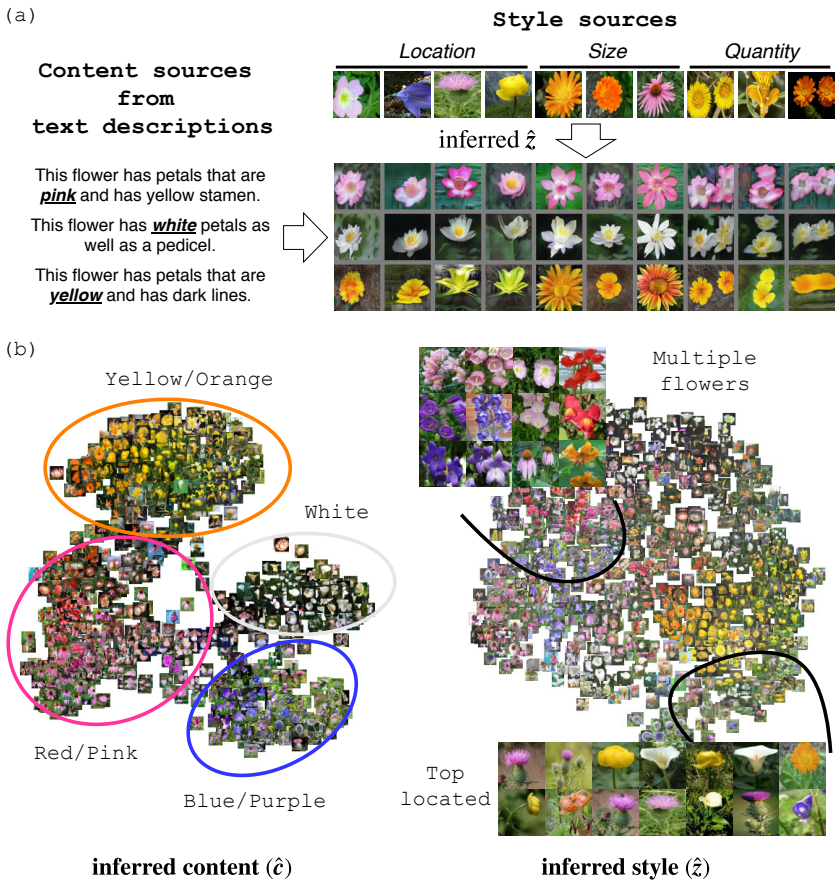


Figure 3.2: (a) Controlling the style (in columns) of generated images given a text description as the content (in rows). Columns 1-4 show locations (*e.g.*, left, right and top) of the content in the image; Columns 5-7 and columns 8-10 represent size and quantity of the content respectively. (b) The learned content and style features through our dual adversarial inference, visualized by t-SNE. The inferred content is clustered solely on color (one dominant factor that is described in the text), while the inferred style shows a more diffused cluster pattern, with local clusters such as multiple flowers and top-located flowers.

comparable variability and visual-semantic similarity when evaluated on the Oxford-102, CUB and COCO datasets.

3.2 Literature Review

Text-to-image synthesis methods Text-to-image synthesis has been made possible by Reed *et al*. [47], where a conditional GAN-based model is used to generate text-matching images from the text description. Zhang *et al*. [48] use a two-stage GAN to first generate low-resolution images in stage I and then improve the image quality to high-resolution in stage II. By using a hierarchically-nested GAN (HDGAN) which incorporates multiple loss functions at increasing levels of resolution,

Zhang *et al*. [53] further improve the state-of-the-art on this task in an end-to-end manner. Several attempts have been made to leverage additional available information, such as object location [54], class label [55, 56], attention extracted from word features [57, 58] and text regeneration [58]. Hong *et al*. [59] propose another approach by providing the image generator with a semantic structure that is sequentially constructed with a box generator followed by a shape generator; however, their approach would not be applicable for single-object image synthesis. Compared to all previous work, our method incorporates the inference mechanism into the current framework for text-to-image synthesis, and by doing so, we explicitly force the model to simultaneously learn separate representations of content and style.

Reed *et al*. [47] have also investigated the separation of content and style information. However, their learning of style is detached from the text-to-image framework, and the parameters of the image generator are fixed during the style learning phase. Therefore, their concept of content and style separation is not actually leveraged during the training of the image generator. In addition, their work uses a deterministic text embedding, which cannot plausibly cover all content variations, and as a result, one can assume that information belonging to the content could severely contaminate the style. In our work, we learn style from the data itself as opposed to the generated images. This allows us to learn style while updating the generator and effectively incorporate style information from the data into the generator.

Adversarial inference methods Various papers have explored learning representations through adversarial training. Notable mentions are BiGANs [60, 61] where a bidirectional discriminator acts on pairs $(\boldsymbol{x}, \boldsymbol{z})$ of data and generated points. While these models assume that a single random variable $\boldsymbol{z}$ encodes data representations, in this work we extend the adversarial inference to two random variables that are disentangled with each other. Our model is also closely related to [62], where the authors incorporate an adversarial reconstruction loss into the BiGAN framework. They show that the additional loss term results in better reconstructions and more stable training. Although Dumoulin *et al*. [61] show results for conditional image generation, in their model the conditioning factor is discrete, fully observed and not inferred through the inference model. In our model however, '$\boldsymbol{c}$' can be a continuous conditioning variable that we infer from the text and image.

Relation to InfoGAN While the matching-aware loss (Section 3.3.1) used in many text-to-image works can also be viewed as maximizing mutual information between the two modalities (*i.e*., *text* and *image*), the way it is approximated is different. InfoGAN [63] uses the variational mutual information maximization technique, whereas the matching-aware loss uses the concept of matched and mismatched pairs. In addition, InfoGAN concentrates all semantic features on the latent code $\boldsymbol{c}$, which contains both content and style, whereas in this work, we only maximize mutual information

on the content since we consider *text* as our content.

3.3 Methods

3.3.1 Preliminaries

We start by describing text-to-image synthesis. Let φ_t be the text embedding of a given text description associated with image $\boldsymbol{x}$. The goal of text-to-image synthesis is to generate a variety of visually-plausible images that are text-matched. Reed *et al*. [47] first propose a conditional GAN-based framework, where a generator G_x takes as input a noise vector $\boldsymbol{z}$ sampled from $p(\boldsymbol{z}) = \mathcal{N}(0, 1)$ and φ_t as the conditioning factor to generate an image $\tilde{\boldsymbol{x}} = G_x(\boldsymbol{z}, \varphi_t)$. A *matching-aware* discriminator D_{x,φ_t} is then trained to not only judge between real and fake images, but also discriminate between matched and mismatched image-text pairs. The minimax objective function for text-to-image (subscript denoted as $t2i$) framework is given as:

$$\begin{aligned}\min_G \max_D V_{t2i}(D_{x,\varphi_t}, G_x) = \mathbb{E}_{(\boldsymbol{x}_a,t_a)\sim p_{\text{data}}}[\log D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_a})] \\ + \frac{1}{2}\big\{\mathbb{E}_{(\boldsymbol{x}_a,t_b)\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_b}))] \\ + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),t_a\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(G_x(\boldsymbol{z}, \varphi_{t_a}), \varphi_{t_a}))]\big\},\end{aligned} \tag{3.1}$$

where $(\boldsymbol{x}_a, t_a)$ is a matched pair and $(\boldsymbol{x}_a, t_b)$ is a mismatched pair.

To augment the text data, Zhang *et al*. [48] replace the deterministic text embedding φ_t in the generator with a latent variable $\boldsymbol{c}$, which is sampled from a learned Gaussian distribution $p(\boldsymbol{c}|\varphi_t) = \mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t))$, where μ and Σ are functions of φ_t parameterized by neural networks. For simplicity in notation, we denote $p(\boldsymbol{c}|\varphi_t)$ as $p(\boldsymbol{c})$. As a result, the objective function (3.1) is updated to:

$$\begin{aligned}\min_G \max_D V_{t2i}(D_{x,\varphi_t}, G_x) = \mathbb{E}_{(\boldsymbol{x}_a,t_a)\sim p_{\text{data}}}[\log D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_a})] \\ + \frac{1}{2}\big\{\mathbb{E}_{(\boldsymbol{x}_a,t_b)\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_b}))] \\ + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{c}\sim p(\boldsymbol{c}),t_a\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(G_x(\boldsymbol{z}, \boldsymbol{c}), \varphi_{t_a}))]\big\}.\end{aligned} \tag{3.2}$$

In addition to the matching-aware pair loss that guarantees the semantic consistency, Zhang *et al*. [53] propose another type of adversarial loss that focuses on the image fidelity (*i.e*., image loss),

further updating (3.2) to:

$$\begin{aligned}\min_G \max_D V_{t2i}(D_x, D_{x,\varphi_t}, G_x) = \mathbb{E}_{\boldsymbol{x}_a \sim p_{\text{data}}}[\log D_x(\boldsymbol{x}_a)] \\ + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{c}\sim p(\boldsymbol{c})}[\log(1 - D_x(G_x(\boldsymbol{z}, \boldsymbol{c})))] \\ + \mathbb{E}_{(\boldsymbol{x}_a, t_a)\sim p_{\text{data}}}[\log D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_a})] \\ + \frac{1}{2}\big\{\mathbb{E}_{(\boldsymbol{x}_a, t_b)\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(\boldsymbol{x}_a, \varphi_{t_b}))] \\ + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{c}\sim p(\boldsymbol{c}),t_a\sim p_{\text{data}}}[\log(1 - D_{x,\varphi_t}(G_x(\boldsymbol{z}, \boldsymbol{c}), \varphi_{t_a}))]\big\}, \quad (3.3)\end{aligned}$$

where D_x is a discriminator distinguishing between images sampled from p_{data} and those sampled from the distribution parameterized by the generator (*i.e.*, p_{model}).

Consider two general probability distributions $q(\boldsymbol{x})$ and $p(\boldsymbol{z})$ over two domains $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{z} \in \mathcal{Z}$, where $q(\boldsymbol{x})$ represents the empirical data distribution and $p(\boldsymbol{z})$ is usually specified as a simple random distribution, *e.g.*, a standard normal $\mathcal{N}(0, 1)$. Adversarial inference [60, 61] aims to match the two joint distributions $q(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{x})$ and $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$, which in turn implies that $q(\boldsymbol{z}|\boldsymbol{x})$ matches $p(\boldsymbol{z}|\boldsymbol{x})$. To achieve this, an encoder $G_z(\boldsymbol{x}) : \hat{\boldsymbol{z}} = G_z(\boldsymbol{x}), \boldsymbol{x} \sim q(\boldsymbol{x})$ is introduced in the generation phase, in addition to the standard generator $G_x(\boldsymbol{z}) : \tilde{\boldsymbol{x}} = G_x(\boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})$. The discriminator D is trained to distinguish joint pairs between $(\boldsymbol{x}, \hat{\boldsymbol{z}})$ and $(\tilde{\boldsymbol{x}}, \boldsymbol{z})$. The minimax objective of adversarial inference can be written as:

$$\begin{aligned}\min_G \max_D V(D, G_x, G_z) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}),\hat{\boldsymbol{z}}\sim q(\boldsymbol{z}|\boldsymbol{x})}[\log D(\boldsymbol{x}, \hat{\boldsymbol{z}})] \\ + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p(\boldsymbol{x}|\boldsymbol{z}),\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(\tilde{\boldsymbol{x}}, \boldsymbol{z}))]. \quad (3.4)\end{aligned}$$

3.3.2 Dual adversarial inference

As described in Section 3.3.1, the current state-of-the-art methods for text-to-image synthesis can be viewed as variants of conditional GANs, where the conditioning is initially on φ_t itself [47] and later on updated to the latent variable $\boldsymbol{c}$ sampled from a distribution learned through φ_t [48, 53, 57, 58]. The generator then has two latent variables $\boldsymbol{z}$ and $\boldsymbol{c}$: $\boldsymbol{z} \sim p(\boldsymbol{z})$, $\boldsymbol{c} \sim p(\boldsymbol{c})$ (left, Figure 3.3). The priors can be Gaussian or non-Gaussian distributions such as the Bernoulli distribution [1]. To learn disentangled representations for style ($\boldsymbol{z}$) and content ($\boldsymbol{c}$) and to enforce the separation between these two variables, we incorporate dual adversarial inference into the current framework for text-to-image synthesis (right, Figure 3.3). In this dual inference process, we are interested in matching the conditional $q(\boldsymbol{z}, \boldsymbol{c}|\boldsymbol{x})$ to the posterior $p(\boldsymbol{z}, \boldsymbol{c}|\boldsymbol{x})$, which under the independence assumption can

[1] In this work, we experiment with both Gaussian and Bernoulli distributions for $p(\boldsymbol{c})$ (More details in Section 3.4).

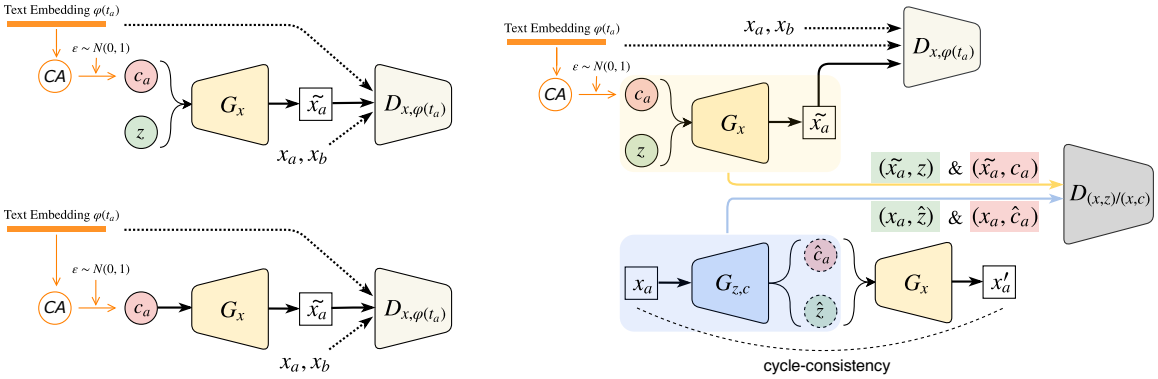


Figure 3.3: Overview of the current state-of-the-art methods (left top) and our proposed method (right) for text-to-image synthesis at low-resolution scale. By default, the current state-of-the-art methods adopt *conditioning augmentation* (CA), which introduces variable $\boldsymbol{c} \sim p(\boldsymbol{c}|\varphi_t)$, in addition to variable $\boldsymbol{z} \sim \mathcal{N}(0,1)$ as the inputs for the image generator G_x. The removal of $\boldsymbol{z}$ (left bottom) does not affect the model performance (*viz.* Figure 3.5 for quantitative evaluations). In our method (right), we incorporate the inference mechanism, where $G_{z,c}$ encodes both $\boldsymbol{z}$ and $\boldsymbol{c}$, and the discriminator $D_{(x,z)/(x,c)}$ distinguishes between joint pairs. For the cycle consistency, sampled $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{c}}$ are also used to reconstruct $\boldsymbol{x}'$.

be factorized as follows:

$$q(\boldsymbol{z}, \boldsymbol{c} \mid \boldsymbol{x}) = q(\boldsymbol{z} \mid \boldsymbol{x}) q(\boldsymbol{c} \mid \boldsymbol{x}), \tag{3.5}$$

$$p(\boldsymbol{z}, \boldsymbol{c} \mid \boldsymbol{x}) = p(\boldsymbol{z} \mid \boldsymbol{x}) p(\boldsymbol{c} \mid \boldsymbol{x}). \tag{3.6}$$

This formulation allows us to match $q(\boldsymbol{z}|\boldsymbol{x})$ with $p(\boldsymbol{z}|\boldsymbol{x})$ and $q(\boldsymbol{c}|\boldsymbol{x})$ with $p(\boldsymbol{c}|\boldsymbol{x})$, respectively. Similar to previous work [60, 61], we achieve this by matching the two pairs of joint distributions:

$$q(\boldsymbol{z}, \boldsymbol{x}) = p(\boldsymbol{z}, \boldsymbol{x}), \tag{3.7}$$

$$q(\boldsymbol{c}, \boldsymbol{x}) = p(\boldsymbol{c}, \boldsymbol{x}). \tag{3.8}$$

The encoder for our dual adversarial inference then encodes both $\boldsymbol{z}$ and $\boldsymbol{c}$: $\hat{\boldsymbol{z}}, \hat{\boldsymbol{c}} = G_{z,c}(\boldsymbol{x}), \boldsymbol{x} \sim q(\boldsymbol{x})$, while the generator decodes $\boldsymbol{z}$ and $\boldsymbol{c}$ sampled from their corresponding prior distributions into an image: $\tilde{\boldsymbol{x}} = G_x(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{z} \sim p(\boldsymbol{z}), \boldsymbol{c} \sim p(\boldsymbol{c})$. To compete with G_x and $G_{z,c}$, the discrimination phase also has two components: the discriminator $D_{x,z}$ is trained to discriminate $(\boldsymbol{x}, \boldsymbol{z})$ pairs sampled from either $q(\boldsymbol{x}, \boldsymbol{z})$ or $p(\boldsymbol{x}, \boldsymbol{z})$, and the discriminator $D_{x,c}$ for the discrimination of $(\boldsymbol{x}, \boldsymbol{c})$ pairs sampled from either $q(\boldsymbol{x}, \boldsymbol{c})$ or $p(\boldsymbol{x}, \boldsymbol{c})$. Given the above setting, the original adversarial inference

objective (3.4) is updated as:

$$\min_G \max_D V_{dual}(D_{x,z}, D_{x,c}, G_x, G_{z,c}) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}),\hat{\boldsymbol{z}},\hat{\boldsymbol{c}}\sim q(\boldsymbol{z},\boldsymbol{c}|\boldsymbol{x})}[\log D_{x,z}(\boldsymbol{x},\hat{\boldsymbol{z}}) + \log D_{x,c}(\boldsymbol{x},\hat{\boldsymbol{c}})] + \mathbb{E}_{\tilde{\boldsymbol{x}}\sim p(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{c}),\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{c}\sim p(\boldsymbol{c})}[\log(1 - D_{x,z}(\tilde{\boldsymbol{x}},\boldsymbol{z})) + \log(1 - D_{x,c}(\tilde{\boldsymbol{x}},\boldsymbol{c}))]. \tag{3.9}$$

3.3.3 Cycle consistency

In unsupervised learning, *cycle-consistency* refers to the ability of the model to reconstruct the original image $\boldsymbol{x}$ from its inferred latent variable $\boldsymbol{z}$. It has been reported that bidirectional adversarial inference models often have difficulties in reproducing faithful reconstructions as they do not explicitly include any reconstruction loss in the objective function [60, 61, 62]. The cycle-consistency criterion, as having been demonstrated in many previous works such as CycleGAN [64], DualGAN [65], DiscoGAN [66] and augmented CycleGAN [67], enforces a strong connection between domains (here $\boldsymbol{x}$ and $\boldsymbol{z}$) by constraining the models (*e.g.*, encoder and decoder) to be consistent with one another. Li *et al*. [62] show that the integration of the cycle-consistency objective stabilizes the learning of adversarial inference, thus yielding better reconstruction results.

With the above in mind, we integrate cycle-consistency in our dual adversarial inference framework in a similar fashion to [62] . More concretely, we use another discriminator $D_{x,x'}$ to distinguish between $\boldsymbol{x}$ and its reconstruction $\boldsymbol{x}' = G_x(\hat{\boldsymbol{z}},\hat{\boldsymbol{c}})$, where $\hat{\boldsymbol{z}},\hat{\boldsymbol{c}} = G_{z,c}(\boldsymbol{x})$, by optimizing:

$$\min_G \max_D V_{cycle}(D_{x,x'}, G_x, G_{z,c}) = \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}[\log D_{x,x'}(\boldsymbol{x},\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x}),(\hat{\boldsymbol{z}},\hat{\boldsymbol{c}})\sim q(\boldsymbol{z},\boldsymbol{c}|\boldsymbol{x})}[\log(1 - D_{x,x'}(\boldsymbol{x}, G_x(\hat{\boldsymbol{z}},\hat{\boldsymbol{c}})))]. \tag{3.10}$$

We later show in an ablation study (Section 3.4.7) that using l_2 loss for cycle-consistency leads to blurriness in the generated images, which agrees with previous studies [65, 68].

3.3.4 Full objective

Taking (3.3), (3.9), (3.10) into account, our full objective is:

$$\begin{aligned}\min_G \max_D V_{full}(D, G) = {} & V_{t2i}(D_x, D_{x,\varphi_t}, G_x) \\ & + V_{dual}(D_{x,z}, D_{x,c}, G_x, G_{z,c}) \\ & + V_{cycle}(D_{x,x'}, G_x, G_{z,c}),\end{aligned} \tag{3.11}$$

where G and D are the sets of all generators and discriminators in our method: $G = \{G_x, G_{z,c}\}$ and $D = \{D_x, D_{x,\varphi_t}, D_{x,z}, D_{x,c}, D_{x,x'}\}$.

Note that in addition to the latent variable $\boldsymbol{c}$, the encoded $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{c}}$ in our method are also sampled from the inferred posterior distributions through the reparameterization trick [69], *i.e.*, $\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}|\boldsymbol{x})$ and $\hat{\boldsymbol{c}} \sim q(\boldsymbol{c}|\boldsymbol{x})$. In order to encourage smooth sampling over the latent space, we regularize the posterior distributions $q(\boldsymbol{z}|\boldsymbol{x})$ and $q(\boldsymbol{c}|\boldsymbol{x})$ to match their respective priors by minimizing the KL divergence. We apply a similar regularization term to $p(\boldsymbol{c})$, *e.g.*, $\lambda D_{KL}(p(\boldsymbol{c}) \,||\, \mathcal{N}(0,1))$ for a normal distribution prior, as done in previous text-to-image synthesis works [48, 53]. Our preliminary experiments [2] showed that without the above regularization, the training became unstable and the gradients typically explode after certain number of epochs.

3.4 Experiments and Results

3.4.1 Proof-of-concept study

To evaluate the effectiveness of our proposed dual adversarial inference on the disentanglement of content and style, we first validate our proposed method on a toy dataset: MNIST-CB [70], where we formulate the digit generation problem as a text-to-image synthesis problem by considering the digit identity as the text content. In this setup, digit font and background color represent styles learned in an unsupervised manner through adversarial inference. We add a cross-entropy regularization term to the content inference objective since our content in this case is discrete (*i.e.*, one-hot vector for digit identity). As shown in Figure 3.4 (a), the content and style are disentangled in the generation phase, where the generator has learned to assign the same style to different digit identities when the same $\boldsymbol{z}$ is used. More importantly, the t-SNE visualizations (Figure 3.4 (b)) from our inferred content and style ($\hat{\boldsymbol{c}}$ and $\hat{\boldsymbol{z}}$) indicate that our dual adversarial inference has successfully separated the information on content (digit identity) and style (font and background color). This is further validated in Figure 3.4 (c) where we show our model's ability to infer style and content from different image sources and fuse them to generate hybrid images, using content from one source and style from the other.

3.4.2 Text-to-image setup

Once validated on the toy example, we move to the original text-to-image synthesis task. We evaluate our method based on model architectures similar to HDGAN [53], one of the current state-of-the-art methods for text-to-image synthesis, making HDGAN our baseline method. The architecture designs are the same as described in [53], keeping in mind that we only consider the 64×64 resolution. More implementation, dataset and evaluation details are described as follows.

[2] We also experimented with minimizing the cosine similarity between $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{c}}$, but did not observe improved performance in terms of the inception score and FID.

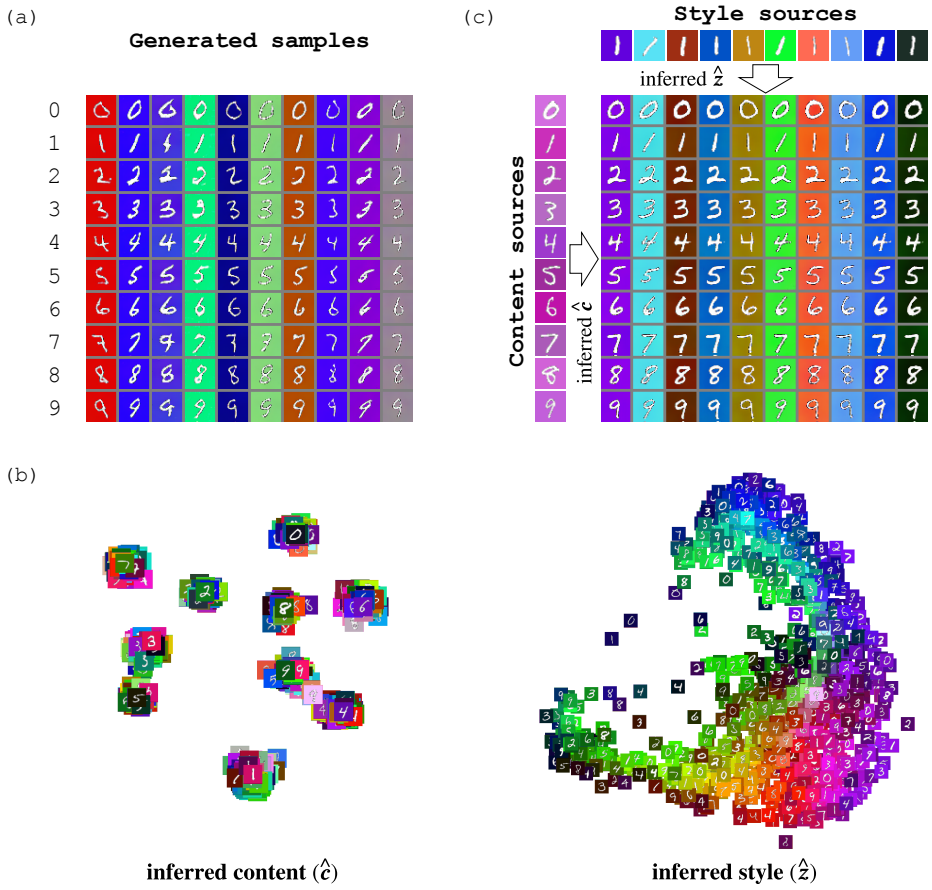


Figure 3.4: Disentangling content and style on MNIST-CB dataset. (a) Generated samples given digit identities as the content $\boldsymbol{c}$. Each column uses the same style $\boldsymbol{z}$ sampled from $\mathcal{N}(0, 1)$. (b) The t-SNE visualizations of inferred content $\hat{\boldsymbol{c}}$ and inferred style $\hat{\boldsymbol{z}}$. (c) Reconstructed samples using inferred content $\hat{\boldsymbol{c}}$ (in rows) and inferred style $\hat{\boldsymbol{z}}$ (in columns) from image sources.

Implementation details For the encoder $G_{z,c}$, we first extract a 1024-dimension feature vector from a given image $\boldsymbol{x}$ (note that the text embeddings are also 1024-dimension vectors), and apply the reparameterization trick to sample $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{c}}$. To reduce the complexity of our models, we use the same discriminator for both $D_{x,z}$ and $D_{x,c}$ in our experiments, thus re-denoted as $D_{(x,z)/(x,c)}$, and the weights are shared between D_x and D_{x,φ_t}. By default, $\lambda = 4$. For the training, we iteratively train the discriminators D_{x,φ_t}, $D_{(x,z)/(x,c)}$, $D_{x,x'}$ and then the generators G_x, $G_{z,c}$ for 600 epochs (Oxford-102 and CUB) or 200 epochs (COCO), with Adam optimization [71]. The initial learning rate is set to 0.0002 and decreased to half of the previous value for every 100 epochs.

Datasets The Oxford-102 dataset [49] contains 8,189 flower images from 102 different categories, and the CUB dataset [50] contains 11,788 bird images belonging to 200 different categories. For both datasets, each image is annotated with 10 text descriptions provided by [72]. Following the

same experimental setup as used in previous works [47, 48, 53], we preprocess and split both datasets into disjoint training and test sets: 82 + 20 classes for the Oxford-102 dataset and 150 + 50 classes for the CUB dataset. For the COCO dataset [51], we use the 82,783 training images and the 40,504 validation images for our training and testing respectively, with each image given 5 text descriptions. We also use the same text embeddings pretrained by a char-CNN-RNN encoder [72].

Evaluation metrics Three quantitative metrics are used to evaluate our method: Inception score [38], Fréchet inception distance (FID) [73] and Visual-semantic similarity [53]. Inception score focuses on the diversity of generated images. We compute inception score using the fine-tuned inception models for both Oxford-102 and CUB datasets provided by [48]. FID is a metric which calculates the distance between the generated data distribution and the real data distribution, through the feature representations of the inception network. Similar to previous works [48, 53], we generate $\sim$ 30,000 samples when computing both inception score and FID. Visual-semantic similarity measures the similarity between text descriptions and generated images. We train our neural distance models for 64$\times$64 resolution images similar to [53] for 500 epochs. In the testing phase, we generate $\sim$ 30,000 images at 64$\times$64 resolution and compute the visual-semantic similarity scores based on the trained neural distance models. The real images are also used to compute the visual-semantic similarity scores as the ground truth.

It has been noticed in our experiments and also reported by others [74] that, due to the variations in the training of GAN models, it is unfair to draw a conclusion based on one single experiment that achieves the best result; therefore, in our experiments, we perform three independent experiments for each method, with averages reported as final results.

3.4.3 Quantitative results

Inception score and FID

To get a global overview of how our method, the baseline method and its variants (by either fixing or removing the noise vector $\boldsymbol{z}$) behave throughout training, we evaluate each model in 20 epoch intervals. Figure 3.5 shows inception score (left axis) and FID (right axis) for both Oxford-102 and CUB datasets. Consistent with the qualitative results presented in Figure 3.1, we quantitatively show that by either fixing or removing $\boldsymbol{z}$, the baseline models retain unimpaired performance, suggesting that $\boldsymbol{z}$ has no contribution in the baseline models. However, with our proposed dual adversarial inference, the model performance is significantly improved on FID scores for both datasets (red curves, Figure 3.5), indicating the proposed method's ability to produce better-quality images. Table 3.1 (Inception score) and 3.2 (FID) summarize the comparison of the results of our method to the

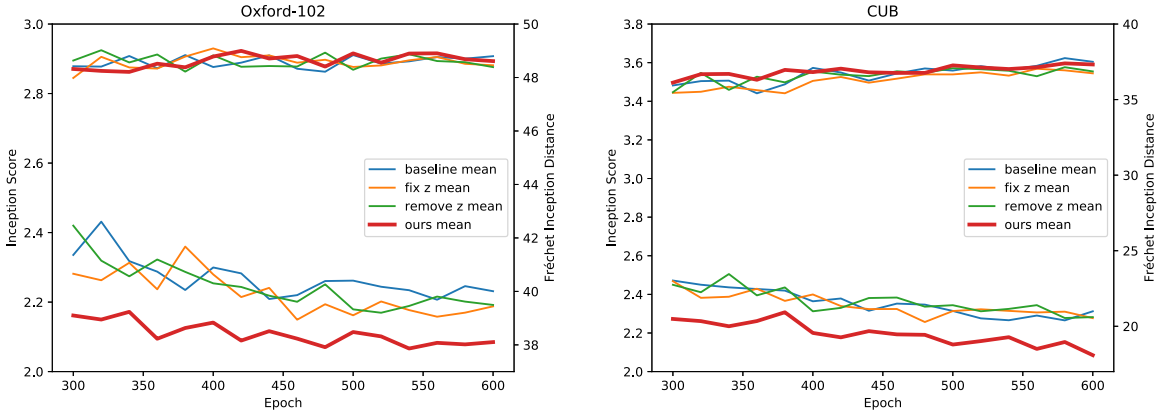


Figure 3.5: Inception score (left axis, top curves) and FID (right axis, bottom curves) for the baseline method, its variants (fix $\boldsymbol{z}$ and remove $\boldsymbol{z}$) and our method on Oxford-102 (left) and CUB (right) datasets. Each curve is the mean of three independent experiments. Higher inception score and lower FID mean better performance.

Table 3.1: Comparison of inception score at 64×64 resolution scale. Higher inception score means better performance.

Method	Inception Score		
	Oxford-102	CUB	COCO
GAN-INT-CLS [47]	2.66 ± 0.03	2.88 ± 0.04	7.88 ± 0.07
GAWWN [54]	—	3.10 ± 0.03	—
StackGAN [48, 75]	2.73 ± 0.03	3.02 ± 0.03	8.35 ± 0.11
HDGAN [53]	—	3.53 ± 0.03	—
HDGAN mean*	$\mathbf{2.90 \pm 0.03}$	$\mathbf{3.58 \pm 0.03}$	8.64 ± 0.37
Ours mean*	$\mathbf{2.90 \pm 0.03}$	$\mathbf{3.58 \pm 0.05}$	$\mathbf{8.94 \pm 0.20}$

* mean calculated on three experiments at five different epochs (600, 580, 560, 540, 520), or three different epochs (200, 190, 180) for COCO dataset

baseline method and also other reported results of previous state-of-the-art methods for the 64 × 64 resolution task on the three benchmark datasets: Oxford-102, CUB and COCO. Our method achieves the best performance based on the mean scores for both metrics on all datasets; on the FID score, it shows a 5.2% improvement (from 40.02 to 37.94) on the Oxford-102 dataset, and a 10.6% improvement (from 20.60 to 18.41) on the CUB dataset.

Visual-semantic similarity

Table 3.3 shows the comparison of the baseline method, our method and the ground truth. Similar to inception score and FID, we provide mean scores for visual-semantic similarity metric based on three independent experiments at five different epochs (600, 580, 560, 540 and 520). As shown in

Table 3.2: Comparison of FID at 64×64 resolution scale. Lower FID means better performance.

Method	FID		
	Oxford-102	CUB	COCO
GAN-INT-CLS [47]	79.55	68.79	60.62
GAWWN [54]	—	53.51	—
StackGAN [48, 75]	43.02	35.11	33.88
HDGAN [53]	—	—	—
HDGAN mean*	40.02 ± 0.55	20.60 ± 0.96	29.13 ± 3.76
Ours mean*	$\mathbf{37.94 \pm 0.39}$	$\mathbf{18.41 \pm 1.07}$	$\mathbf{27.07 \pm 2.55}$

* mean calculated on three experiments at five different epochs (600, 580, 560, 540, 520), or three different epochs (200, 190, 180) for COCO dataset

Table 3.3: Visual-semantic similarity. Higher visual-semantic similarity score means better performance.

Method	Dataset	
	Oxford-102	CUB
Ground Truth	0.422 ± 0.117	0.382 ± 0.154
HDGAN mean*	0.248 ± 0.136	0.263 ± 0.164
Ours mean*	0.246 ± 0.139	0.251 ± 0.168

* mean calculated on three experiments at five different epochs

the table, our method achieves comparable results for visual-semantic similarity compared to the baseline method.

3.4.4 Qualitative results

In this subsection, we present qualitative results on text-to-image generation and interpolation analysis based on inferred content ($\hat{\boldsymbol{c}}$) and inferred style ($\hat{\boldsymbol{z}}$).

Generation First, we visually compare the quality and diversity of images generated from our method against the baseline. Figure 3.6 shows one example for each dataset, illustrating that our method is able to generate better-quality images compared to the baseline method, which agrees with our quantitative results in Table 3.1 and 3.2.

We provide more text-to-image generation examples in Figure 3.7. For each method, we generate three groups of images given a certain text description: 1) use a fixed $\boldsymbol{z}$ and sample $\boldsymbol{c}$ from $p(\boldsymbol{c})$

Figure 3.6: Examples of generated images on Oxford-102 (top), CUB (middle) and COCO (bottom) datasets.

Figure 3.7: Examples of generated images on Oxford-102 dataset compared with the baseline method using three different strategies: 1) use a fixed $\boldsymbol{z}$ and sample $\boldsymbol{c}$ from $p(\boldsymbol{c})$ to show the role of $\boldsymbol{c}$ in the generated images; 2) use a fixed $\boldsymbol{c}$ and sample $\boldsymbol{z}$ from $p(\boldsymbol{z})$ to examine how $\boldsymbol{z}$ contributes to image generation; 3) sample both $\boldsymbol{z}$ and $\boldsymbol{c}$ from $p(\boldsymbol{z})$ and $p(\boldsymbol{c})$ respectively to evaluate the overall performance. Note that $p(\boldsymbol{c})$ is conditioned on the text description. The conditioning text descriptions and their corresponding images are shown in the left column.

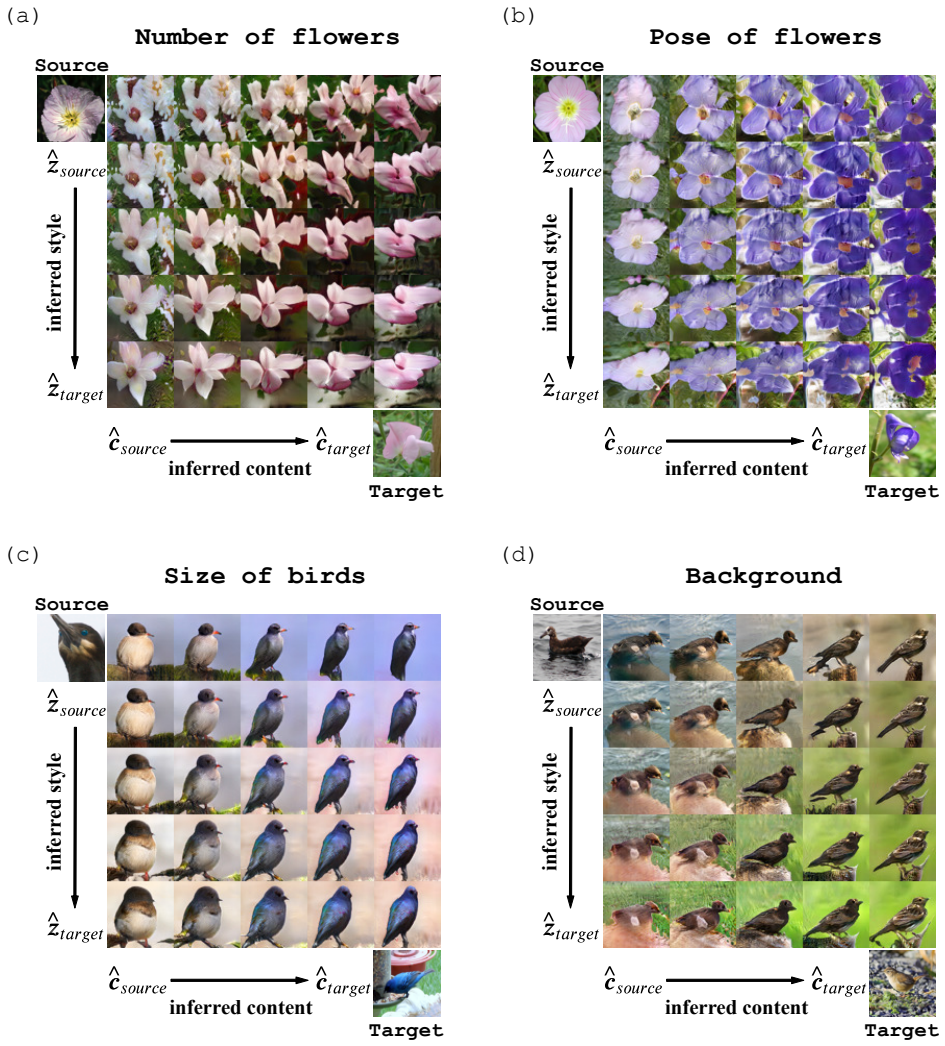


Figure 3.8: Examples of reconstructed images by interpolation of inferred content $\hat{\boldsymbol{c}}$ and inferred style $\hat{\boldsymbol{z}}$ from sources to targets. The learned style information includes: (a) quantity, (b) pose, (c) size and (d) background.

to show the role of $\boldsymbol{c}$ in the generated images; 2) use a fixed $\boldsymbol{c}$ and sample $\boldsymbol{z}$ from $p(\boldsymbol{z})$ to examine how $\boldsymbol{z}$ contributes to image generation; 3) sample both $\boldsymbol{z}$ and $\boldsymbol{c}$ from $p(\boldsymbol{z})$ and $p(\boldsymbol{c})$ respectively to evaluate the overall performance. Note that $p(\boldsymbol{c})$ is conditioned on the text description as defined in Section 3.3.1. As seen in Figure 3.7, our model generates better-quality images and gives more meaningful variability in style compared to the baseline, when we keep the content information constant and only sample from the latent variable $\boldsymbol{z}$. This is not surprising due to the fact that the generator has learned to match changes in $\boldsymbol{z}$ with style. Note that for most of the 'easy' tasks (images that are easy for the generator to synthesize), visual comparison does not reveal significant differences between the baseline and our method.

Interpolation To make sure we are not overfitting, and to investigate whether we have learned a representative latent space, we look at interpolations of projected locations in the latent space. Interpolations also enable us to examine whether the model has indeed learned to separate style from content in an unsupervised way. To do this, we provide the trained inference model with two images: the source image and the target image, and extract their projections $\hat{z}$ and $\hat{c}$ for interpolation analysis. As shown in Figure 3.8, the rows correspond to reconstructed images of linear interpolations in $\hat{c}$ from source to target image and the same for $\hat{z}$ as displayed in columns. The smooth transitions of both the content represented by $\hat{c}$ from the left to right and the style represented by $\hat{z}$ from the top to bottom indicate a good generalization of our model representing both latent spaces, and more interestingly, we find promising results showing that $\hat{z}$ is indeed controlling some meaningful style information, *e.g.*, the number and pose of flowers, the size of birds and the background (Figure 3.8).

3.4.5 Disentanglement constraint

Despite promising results evidenced by many such examples as shown in Figure 3.8, we notice that the information captured by inferred style ($\hat{z}$) is not always consistent and faithful when we use Gaussian priors for both content and style. Inspired by the theories from independent component analysis (ICA) for separating a multivariate signal into additive subcomponents [76], we use a Bernoulli distribution for the content representation to satisfy the non-Gaussian constraint. This provides us with a better disentanglement of content and style. Note that an alternative approach for ICA has also recently been explored in [77]. As shown in Figure 3.9 and Figure 3.10, our models learn to synthesize images by combining content and style information from different sources while preserving their respective properties (*e.g.*, color for the content; and location, pose, quantity, etc. for the style), which suggests the disentanglement of content and style. Note that the content information can either directly come from a text description (top, Figure 3.9 and Figure 3.10) or be inferred from an image source (bottom, Figure 3.9 and Figure 3.10). More examples and discussions are provided in Section 3.4.6.

Higgins *et al*. [78] and Zhang *et al*. [79] have proposed quantitative metrics for the disentanglement analysis which involve classification of the style attributes or comparison of the distance between generated style and true style. However, in our case, the dataset does not contain any labeled attribute that can be used to evaluate a captured style. As a result, their proposed metrics would not be suitable in our case. One possible solution would be to artificially create a new dataset that has the same content over multiple known styles. We leave this exploration for future work.

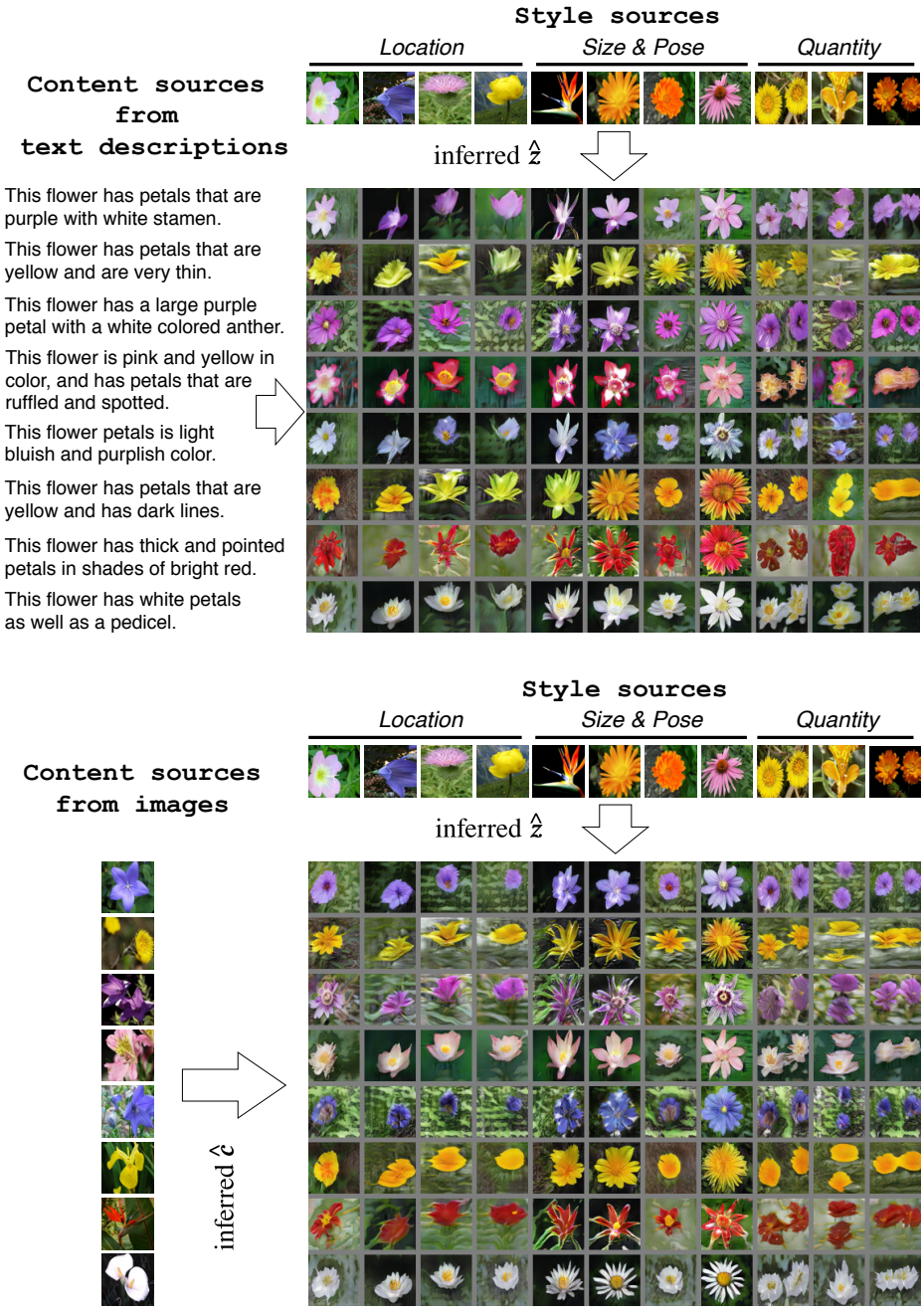


Figure 3.9: Disentangling content (in rows) and style (in columns) on Oxford-102 dataset by using content sources either from text descriptions (top) or images (bottom). More results are provided in Figure 3.15 (Section 3.4.6).

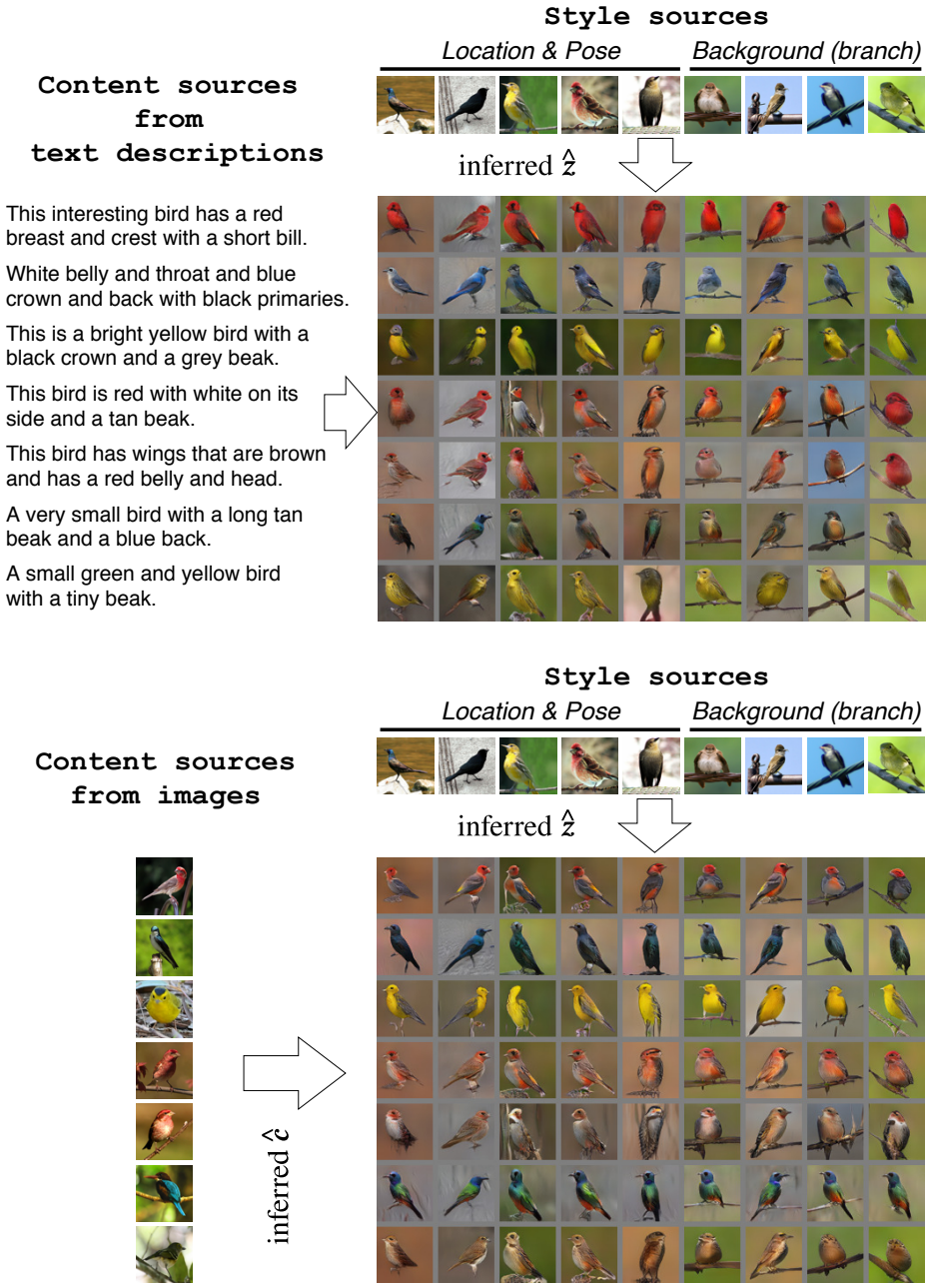


Figure 3.10: Disentangling content (in rows) and style (in columns) on CUB dataset by using content sources either from text descriptions (top) or images (bottom). More results are provided in Figure 3.16 (Section 3.4.6).

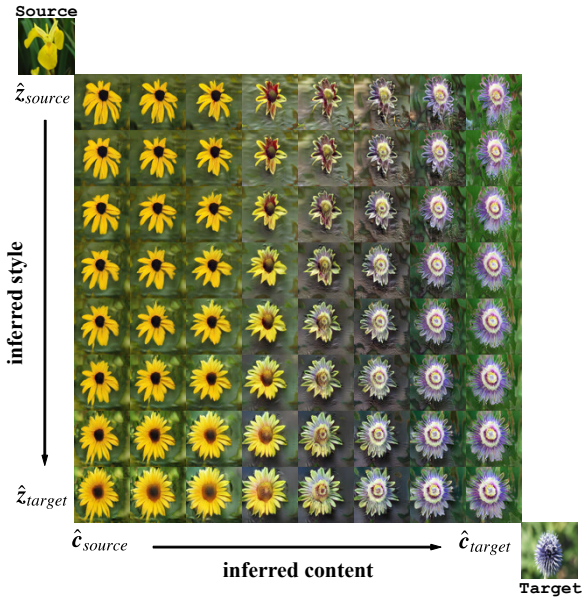


Figure 3.11: Example of inferred style controlling the number of petals, and the pose of flowers from facing towards upright to facing front.

3.4.6 More examples on disentanglement analysis

In this subsection, we provide more results and discussions on the disentanglement analysis with the Bernoulli constraint (see Section 3.4.5). This includes more interpolation results based on inferred content ($\hat{c}$) and inferred style ($\hat{z}$), and style transfer results based on real images and synthetic images with specifically engineered styles.

Interpolations on content and style

For interpolations, we provide our trained inference model with two images: the source image and the target image, to extract their projections $\hat{z}$ and $\hat{c}$ in the latent space. As shown in Figure 3.11, Figure 3.12, Figure 3.13 and Figure 3.14, the rows correspond to reconstructed images of linear interpolations in $\hat{c}$ from source to target image and the same for $\hat{z}$ as displayed in columns. The source images are shown in the top left corner and the target images are shown in the bottom right corner. The figures demonstrate that our proposed dual adversarial inference is able to separate the learning of content and style, and moreover, the learned style $\hat{z}$ indeed represents certain meaningful information. In particular, Figure 3.11 shows an example of style controlling the number of petals, and the pose of flowers from facing towards upright to facing front; Figure 3.12 shows a smooth transition of style from a single flower to multiple flowers; Figure 3.13 shows the changing of the bird pose from sitting to flying; and finally Figure 3.14 shows the switch of the bird pose from facing towards right to facing towards left and the emergence of tree branches in the background.

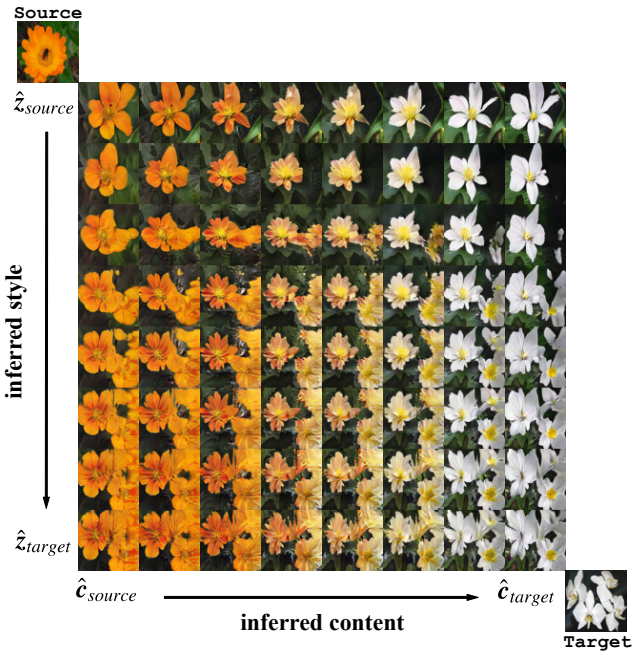


Figure 3.12: Example of inferred style controlling the number of flowers, from a single flower to multiple flowers.

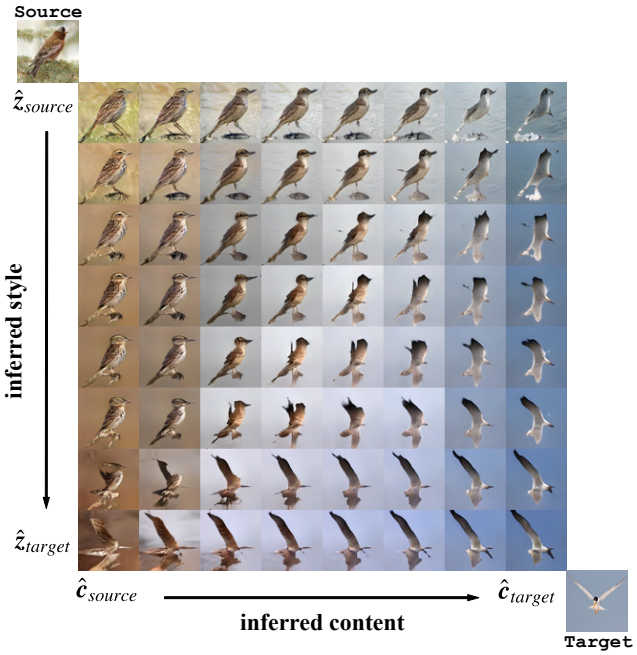


Figure 3.13: Example of inferred style controlling the pose of birds from sitting to flying.

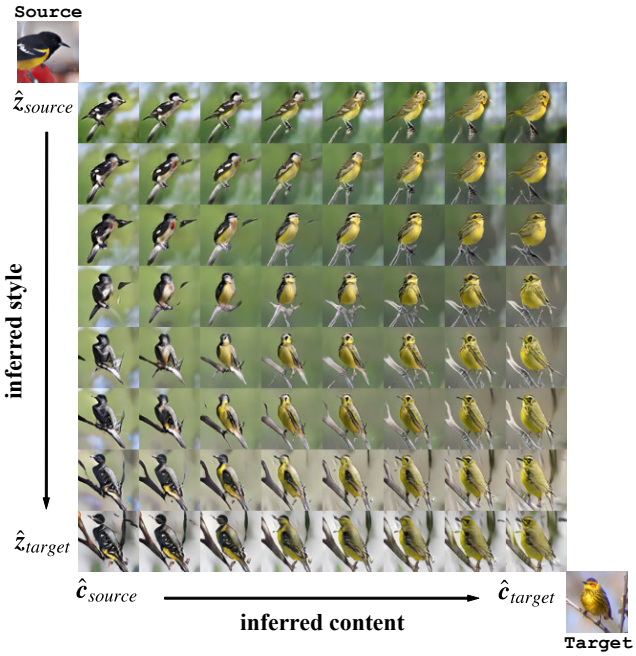


Figure 3.14: Example of inferred style controlling the pose of birds from facing towards right to facing towards left and the emergence of tree branches in the background.

More style transfer results

Here, we provide more style transfer results in Figure 3.15 (Oxford-102) and Figure 3.16 (CUB). Each column uses the same style inferred from a style source, and each row uses the same text description as the content source. The style sources that are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

Style interpolations with synthetic style sources

To further validate the disentanglement of content and style, we artificially synthesize images that have certain desired known styles, *e.g.*, a flower or multiple flowers located in different locations (top left, top right, bottom left or bottom right), and perform linear interpolation of $\hat{z}$ from two different style sources. As shown in Figure 3.17, the flower can move smoothly from one location to another, and the number of flowers can grow smoothly from one to two, suggesting that a good representation of style information has been captured by our inference mechanism.

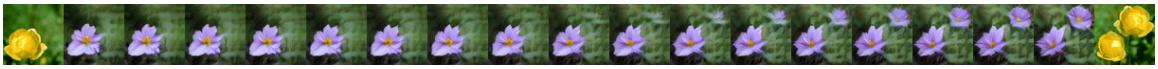
Figure 3.15: Disentangling content (in rows) and style (in columns) on Oxford-102 dataset by using content sources from text descriptions. The style sources are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

Figure 3.16: Disentangling content (in rows) and style (in columns) on CUB dataset by using content sources from text descriptions. The style sources are shown in the top row, and the corresponding real images for the content sources are shown in the leftmost column.

This flower is white and yellow in color, with only one large petal.

Flower is big and round petals are orange and the pistil is orange.

This flower has light, purple petals and a bee on it's stigma.



This flower has petals that are white and are bunched together.

This flower has a light purple bottom petal with three other petals that are dark purple with green streaks.

The petals on this flower are orange with a purple pistil.

Figure 3.17: Style interpolations with synthetic style sources: the moving flowers and the growing quantities of flowers.

Table 3.4: Ablation study on CUB dataset. Note that the ablation on V_{dual} requires us to remove V_{cycle} as well, and it eventually turns into V_{t2i} only, which is the baseline method (results shown in Table 3.1 and 3.2).

Method	Inception Score	FID
ours	3.58 ± 0.05	18.41 ± 1.07
ours without V_{t2i}	3.31 ± 0.04	20.65 ± 0.47
ours without V_{cycle}	3.53 ± 0.06	19.29 ± 0.90
l_2 loss for V_{cycle}	1.73 ± 0.15	149.8 ± 16.4

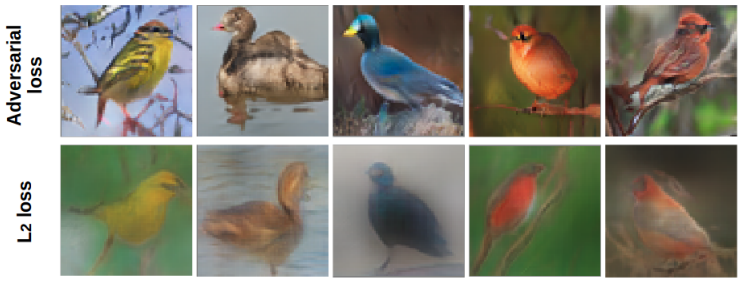


Figure 3.18: Examples of generated images by using either adversarial loss or l_2 loss for the cycle consistency.

3.4.7 Ablation study

In our method, we have multiple components, each of which is optimized by its corresponding objective. The previous works [47, 48, 53] for text-to-image synthesis use the discriminator D_{x,φ_t} to discriminate whether the image $\boldsymbol{x}$ matches its text embedding φ_t. However, with the integration of adversarial inference, where a new discriminator $D_{x,c}$ is designed to match the joint distribution of $(\boldsymbol{x}, \hat{\boldsymbol{c}})$ and $(\tilde{\boldsymbol{x}}, \boldsymbol{c})$, we now question whether the discriminator D_{x,φ_t} is still required, given the fact that $\boldsymbol{c}$ is learned from φ_t. To answer this question, we remove the objective $V_{t2i}(D, G)$ from our method, and as seen in Table 3.4, the performance on the CUB dataset significantly drops for both inception score and FID, indicating that D_{x,φ_t} is not redundant in our method by providing strong supervision over the text embeddings. Similarly, we examine the role of cycle-consistency loss in our method by removing $V_{cycle}(D, G)$ from the objective. We observe a slight drop in both inception score and FID (Table 3.4), suggesting that cycle-consistency can further improve the learning of adversarial inference, which is in agreement with [62]. It is also worth mentioning that our method without cycle-consistency still achieves better FID scores than the baseline method on the CUB dataset (Table 3.2 and Table 3.4), which additionally supports our proposal to integrate the inference mechanism in the current text-to-image framework. Note that, in this work, we use three terms in our final objective function: V_{t2i}, V_{dual} and V_{cycle}. Without V_{dual}, we would not be

able to learn an inference procedure on the stochastic variables $\boldsymbol{z}$ and $\boldsymbol{c}$, which are required for the reconstruction when computing V_{cycle} in our setup. Therefore, the ablation on V_{dual} requires us to remove V_{cycle} as well, and it eventually turns into V_{t2i} only, which is the baseline method (results shown in Table 3.1) and 3.2.

We also examine the model performance by using l_2 loss for cycle-consistency instead of the adversarial loss $V_{cycle}(D, G)$ defined in Section 3.3.3. The resulting degradation in quality is unexpectedly dramatic (Table 3.4). Figure 3.18 shows the generated images using adversarial loss compared with those using l_2 loss, and it is clear that the latter gives blurrier images. A similar effect is observed when using l_1 loss. The quantitative results are shown in Table 3.4.

3.5 Conclusion

In this chapter, we incorporate a dual adversarial inference procedure in order to learn disentangled representations of content and style in an unsupervised way, which we show improves text-to-image synthesis. It is worth noting that the content is learned both in a supervised way through the text embedding and in an unsupervised way through the adversarial inference. The style, however, is learned solely in an unsupervised manner. Despite the challenges of the task, we show promising results on interpreting what has been learned for style. With the proposed inference mechanism, our method achieves improved quality and comparable variability in generated images evaluated on Oxford-102, CUB and COCO datasets.

Chapter 4

Multi-Channel Learning for Cell Phenotype Classification

In this chapter, we aim to solve the cell phenotype classification problem using deep residual network (ResNet) and its variants via multi-channel learning, in which each view is explicitly defined by each channel capturing independent features of cell phenotypes (*i.e.*, cell components), and we can construct the image sequences by simply combining images from each individual channel. Cell phenotype classification is an image-based method that can be used for drug high-content screening, in which complex cell states associated with chemical compound treatment can be characterized. Previous work on cell phenotype classification typically requires a routine yet cumbersome step of single cell segmentation before the classification task. In this work, we present a segmentation-free method for image-based cell phenotype classification using ResNet and its variants. The cell images are samples treated with annotated compounds that can be mainly grouped into three clusters, giving three classes to be classified. Instead of single-cell phenotype classification, we use the raw images without segmentation for our training and evaluation directly. Compared to previous reference work, we significantly simplify the data preprocessing step and accelerate the training while still achieving high accuracy. Our trained models achieve a 98.2% accuracy rate on the three classes classification problem (three compound clusters only), and a 93.8% accuracy rate on the four classes classification problem (three compound clusters plus the mock class) based on five-fold cross-validation.

4.1 Introduction

Cell phenotypes are complexes of morphological features that are present in cell microscopy images. Multiplex cytological profiling assay provides a way to capture a wide range of cell phenotypes by painting the cells in multiple channels with as many fluorescent labels as possible [80]. This image-based cell profiling has great potential in applications such as chemical compound characterization and future drug discovery through a high-content screening approach. However, with the huge amount of image data acquired and high dimensional features to represent each image, it still remains a big challenge for computer vision algorithms to process the massive image data and perform cell phenotype classification with high accuracy but still within an appropriate computation time.

With recent emerging algorithms and technologies in deep learning, object recognition and image classification have made impressive breakthroughs for various applications. Convolutional neural network (CNN) is one of the deep learning approaches specialized in image-based classification tasks, which has been widely demonstrated to outperform traditional state-of-the-art machine learning algorithms, not only for natural images [1] but also for high-content microscopy images [81, 82]. Among various architectures that have been proposed, deep residual network (ResNet) is currently the basis of many popular state-of-the-art convolutional neural network models for image recognition, and its recent variants include wide residual network (WRN), aggregated deep residual network (ResNeXt) and deep pyramidal residual network (PyramidNet). The comparison of different ResNet-based architectures is shown in Figure 2.4 (Chapter 2). In this work, we demonstrate the potential application of deep residual network and its variants in high-content screening (i.e. cell phenotype classification) that can overcome issues associated with analyzing high-content screening data, such as exhaustive preprocessing and inefficient learning.

4.2 Literature Review

Deep learning has been applied to various tasks in high-content screening, such as cell-cycle reconstruction [83], protein subcellular localization [82], and cell phenotype classification [81]. However, none of the previous works considers residual learning in their model selection. The first CNN application for cell phenotype classification in high-content drug screening was reported in 2016 by Dürr and Sick [81], where the authors segmented individual cells from the raw microscopy images and then used an AlexNet based CNN model to classify single cells into different phenotypes (see Figure 4.1(a)). Note that while the authors only focused on single-cell phenotype classification, an applicable high-content screening system would still require a final voting algorithm in order to determine the whole-image cell phenotype. The performance of CNN has also been compared to

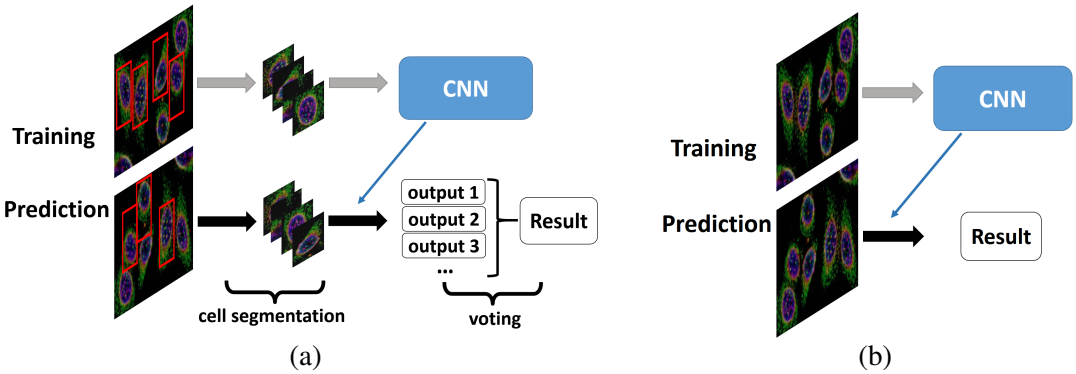


Figure 4.1: Pipelines for cell phenotype classification. (a) Previous method with single cell segmentation; (b) Our proposed segmentation-free method.

various traditional methods, and it has been shown that not only CNN could save time and costs because of its feature self-learning ability, but also achieve the best performance with overall accuracy of 93.4% for four classes classification problem (three compound clusters plus the mock class). It is also mentioned in the paper that for three classes classification problem (three compound clusters only), CNN gives a performance of 97.3%.

To the best of our knowledge, it is also the only CNN application to this problem so far. However, the authors only worked on single-cell phenotype classification but did not consider CNN performance on whole-image cell phenotype classification.

4.3 Methods

In this section, we first present our motivation for this work in Section 4.3.1, and then introduce the dataset that we use for cell phenotype classification (Section 4.3.2). The details of data preprocessing steps are shown together with data augmentation in Section 4.3.3. Finally, we describe in Section 4.3.4, 4.3.5 and 4.3.6, the architectures of the CNN models used in this work, including AlexNet, ResNet and several latest variants of ResNet (WRN, ResNeXt and PyramidNet).

4.3.1 Our motivation

The previous work by Dürr and Sick [81] requires a crucial step of single cell segmentation before the classification task, which can take a considerable amount of time and resources. In addition, although the previous method gives quite good performance, we argue that there are still at least three potential problems:

- First, single cell phenotype may sometimes be mislabeled. This is because the cells are highly diverse in response to compound treatment, therefore, compound treatment may induce

changes in certain cells but not necessary all the cells in the same plate. For example, a cell plate treated with fenbendazole for a certain period of time does not necessarily mean that all of the single cells segmented from that plate will render fenbendazole-like phenotype. Considering the high variability in biological systems, it is possible that certain cells may not respond or be resistant to fenbendazole, thus the actual phenotype of those cells remains the same as the mock class, rather than the fenbendazole-like phenotype.

- Second, some compound induced phenotypes may only be reflected in the interactions of cells, *e.g.*, cell aggregation, in which case single-cell phenotype based method will fail to work.

- Third, the treatment of compounds that are lethal to cells can result in massive cell death, making single cell segmentation impossible.

With the above concerns in mind, we propose here a segmentation-free method for image-based cell phenotype classification using CNN models with residual learning. As is shown in Figure 4.1(b), we use the raw images with multiple cells as our samples directly for the training and evaluation. Note that although CNN models generally do not require a segmentation step for the classification task, for biomedical image classification tasks, it is quite common to preprocess the data before the training of a CNN model for better performance, *e.g.*,, single cell segmentation for cell phenotype classification or the use of patch strategy for high resolution image classification. Our work sheds lights on the use of more advanced networks as an alternative choice to avoid the time-consuming data preprocessing step.

4.3.2 BBBC022v1 dataset

We use image set BBBC022v1 [80], available from the Broad Bioimage Benchmark Collection [84]. The image set provides images of cultured U2OS cells (Human Bone Osteosarcoma Epithelial Cell Line) treated with 1600 known bioactive chemical compounds for a certain period of time. The images were acquired in five channels, with each channel representing one or two cell components respectively: Hoechst 33342 (Nucleus), Concanavalin A (Endoplasmic Reticulum), SYTO 14 (Nucleoli), WGA + Phalloidin (Golgi and Actin) and MitoTracker Deep Red (Mitochondria). A different chemical compound treatment can induce a specific pattern change in cell morphology, which can be captured in multiple channels as cell phenotype. Therefore, this image set provides a basis for testing cell phenotype classification methods with respect to their ability to discriminate a wide range of complex cell phenotypes. More details about the cell profiling assay and compound induced cell phenotypes can be found here [80]. As will be explained in Subsection 4.3.3 below,

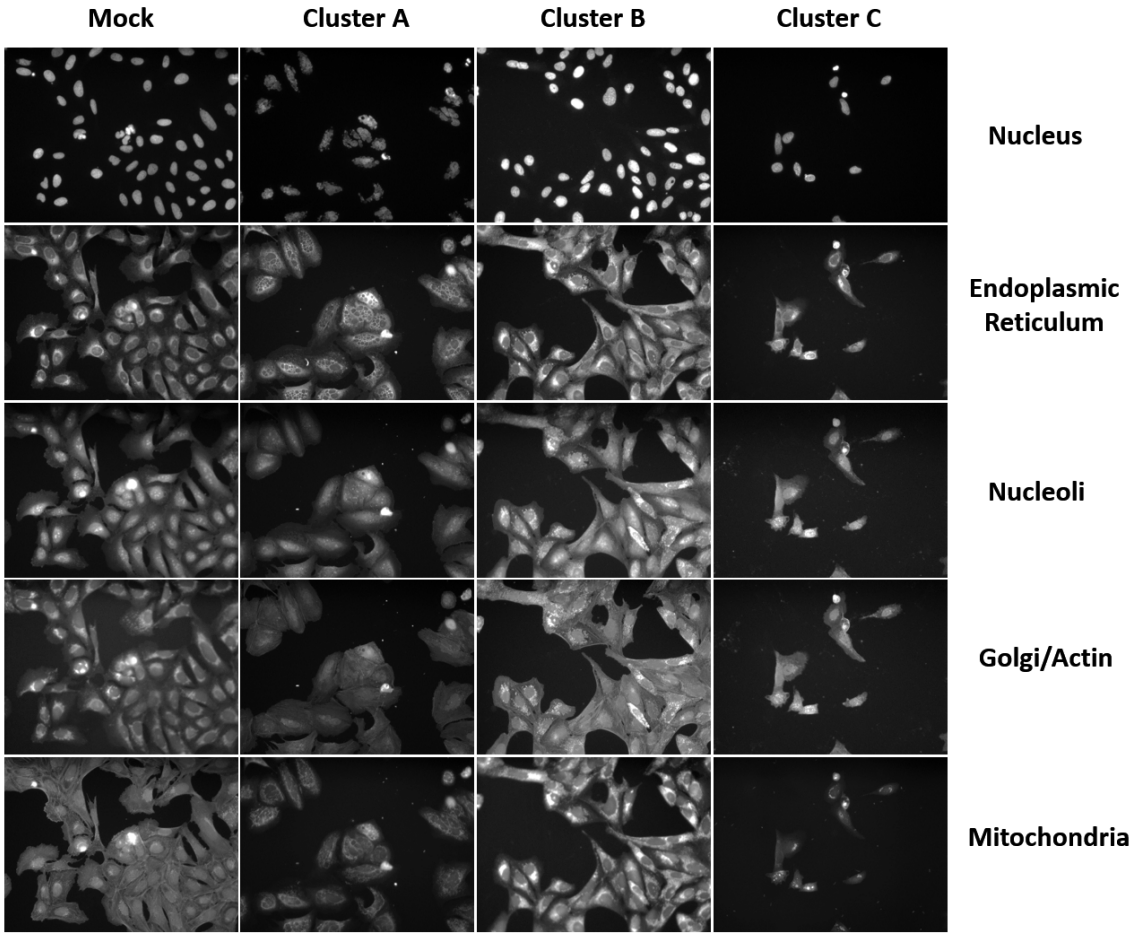


Figure 4.2: Typical five-channel image samples split in separate channels for each cluster. Each channel represents one or two cell components respectively: Hoechst 33342 (Nucleus), Concanavalin A (Endoplasmic Reticulum), SYTO 14 (Nucleoli), WGA + Phalloidin (Golgi and Actin) and MitoTracker Deep Red (Mitochondria).

only a part of the dataset is used for this chapter. Figure 4.2 shows typical five-channel image samples for different compound clusters that will be described in Subsection 4.3.2.

Data label

Compounds can be mainly clustered into three classes based on their mechanism-of-action and previous study on image-based cell profile similarity using hierarchical clustering [80]. We use the same clustering strategy for our data labels in this chapter. More specifically, Cluster A contains three compounds: fenbendazole, oxibendazole and paclitaxel, and it represents tubulin modulators; Cluster B contains five neuronal receptor modulators, including fluphenazine, fluphenazine dihydrochloride, metoclopramide, metoclopramide monohydrochloride and procaine hydrochloride; and Cluster C contains five compounds that are structurally related cardenolide glycosides, which are

Table 4.1: Chemical compound clusters.

Cluster	Compounds	Sample size
A	Fenbendazole, Oxibendazole, Paclitaxel	108
B	Fluphenazine, Fluphenazine Dihydrochloride, Metoclopramide, Metoclopramide Monohydrochloride, Procaine Hydrochloride	180
C	Lanatoside C, Peruvoside, Digitoxin, Neriifolin, Digoxin	216
Mock	No compound treatment	1215

lanatoside C, peruvoside, digitoxin, neriifolin and digoxin. Table 4.1 is the summary of compound clusters.

4.3.3 Data preprocessing and augmentation

To select cell images that can be used for this chapter, we search through the whole BBBC022v1 dataset. For each five-channel image sample, we first check its associated compound name in the metadata file that is provided together with the dataset. If and only if the compound is found to be included in Cluster A, Cluster B or Cluster C, we retrieve the corresponding five channel-separated images from the dataset. We then resize the images from original resolution 520×696 to 100×100. After this data cleaning step, we are able to have 2520 cell images in total, corresponding to 504 five-channel image samples, with 108 in Cluster A, 180 in Cluster B and 216 in Cluster C. The retrieved dataset is randomly split into five folds for further cross-validation experiments. During each run of cross-validation, one fold is chosen for testing (20%), and the rest of them (80%) are further split into two parts: training (60%) and validation (20%) to train and select the model for evaluation on the testing set. For the four classes classification problem, we include the mock samples in addition to the other three clusters during the data retrieval, and repeat the same above data preprocessing steps to obtain the image data for training, validation and testing, while the images are resized to a slightly higher resolution (200×200) to keep more subtle details, as the trained model based on lower resolution (100×100) images does not give good performance during our initial trial. There are in total 1215 five-channel image samples selected for the mock class.

Unlike the approach used by Dürr and Sick [81], we do not perform single cell segmentation in our data preprocessing, rather, we consider the whole raw images, without segmentation, as our samples directly. This improvement saves a great amount of time and work for both training and evaluation.

It is well known that deep networks typically require a large amount of training samples to achieve satisfactory performance. Due to the limited number of image samples we have for each

of the three compound clusters, real-time augmentation is also performed on our training dataset during the training phase. We augment our training dataset by 1) random rotation from $-90°$ to $90°$, 2) random translation within $\pm 10\%$ of image pixels both horizontally and vertically, 3) random flip both horizontally and vertically. The randomly augmented image samples are used to train our convolutional neural networks without overfitting the models.

4.3.4 AlexNet

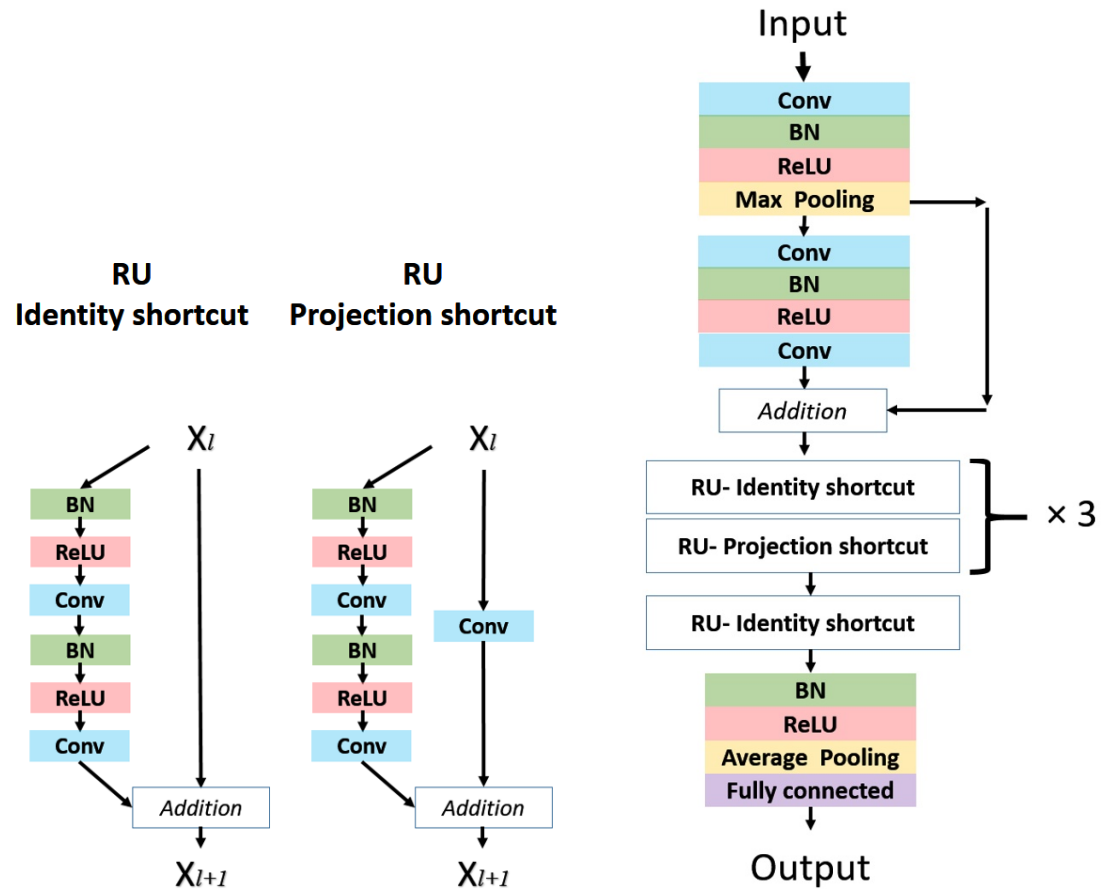
In order to classify the image samples, we first start with a simple AlexNet model, which consists of five convolutional layers and three fully connected layers. Each convolutional layer is followed by a ReLU activation layer, and a 2×2 max-pooling layer is respectively added at the end of the first, second and fifth activation layers. Two batch normalization layers are present in the network, one after the first pooling layer and the other after the second pooling layer. We change the kernel depths for the five convolutional layers to 32, 64, 128, 128, and 64 respectively, while the kernel sizes are kept the same as the original architecture [1]. The convolutional layers are then followed by three fully connected layers with 256, 256 and 3 nodes and finally activated by softmax as the output. For the four classes classification problem, 4 nodes are used instead for the last fully connected layer.

4.3.5 ResNet

The introduction of residual connections into convolutional neural network makes ResNet currently the basis of many state-of-the-art CNN models. Similar to our previous work [85], we also train an 18-layer ResNet model (ResNet-18) integrated with the latest design of residual unit, which is proposed to make the model easier to train and also has better performance [25]. The basic residual unit consists of six sequential components: Batch Normalization, ReLU, Convolution, Batch Normalization, ReLU and Convolution. The identity shortcut is used when the input and output dimensions are the same, otherwise, we consider the projection shortcut to match the dimensions by using a convolutional layer. Figure 4.3(a) shows the two types of residual units used in this chapter, one with identity shortcut (left) and the other with projection shortcut (right). The overall architecture is shown in Figure 4.3(b).

4.3.6 ResNet variants

For a fair comparison of ResNet variants with the basic ResNet, we fix the number of layers to 18 for all models, therefore resulting in WRN-18, ResNeXt-18 and PyramidNet-18 respectively. By default, we use 16, 32, 32 and 64 as the number of filters for the four stages of all networks (each stage has two residual blocks, one with identity shortcut and the other with projection shortcut) in the comparison. Due to the limitation of computation resources, the widening factor k for WRN-18



(a) (b)

Figure 4.3: (a) Residual unit with identity shortcut (left) and projection shortcut (right); (b) Overall architecture of the ResNet used in the chapter.

is only set to 2 or 4. For example, if $k = 2$, the number of filters for the four stages will be 32, 64, 64 and 128 respectively. For ResNeXt-18, the cardinality, which denotes the size of aggregated transformations, is set to 4. Finally, for PyramidNet-18, although the authors [28] propose two ways to increase the number of feature maps (additive-based and multiplicative-based), we only consider the simple additive-based version in this chapter as the following:

$$D_n = \begin{cases} \text{starting filter number} + \alpha, & \text{if } n = 1 \\ D_{n-1} + \alpha, & \text{if } n > 1 \end{cases} \tag{4.1}$$

where D_n is the number of filters for the nth residual block, and α is the step factor, which is 6 in our case, determined by the number of residual blocks, starting and ending filter numbers.

4.4 Experiments and Results

All models are implemented using Keras, a deep learning library written in python with either TensorFlow or Theano as a backend. Our AlexNet has about 10.7 million trainable parameters while ResNet-18 has about 11.1 million. To optimize the weights, we use stochastic gradient descent for all models, with a batch size of 100 to compute the gradients using back propagation. The initial learning rate is set to 0.001, decay by 1e-6 over each update, and Nestrov momentum is set to 0.9. To speed up the training, we also monitor on validation accuracies and use the early stopping strategy. All experiments are done on 12 Intel Core i7-6850K processors with a NVIDIA GeForce GTX 1080/PCIe/SSE2 GPU with CUDA 8.0 installed in a Ubuntu 16.04 LTS, except for the comparison experiments, we use NVIDIA GeForce GTX TITAN X GPU to satisfy the memory requirements.

4.4.1 Three classes classification problem

We first start with the three classes classification problem (three compound clusters only) to get an initial idea about the performance of our proposed segmentation-free method. In order to compare the performance of AlexNet and ResNet, five-fold cross-validation is performed in the experiments. The overall accuracy is computed by averaging the accuracies evaluated on the testing sets in five runs. Table 4.2 shows the accuracies from five-fold cross-validation experiment on both AlexNet and ResNet-18. ResNet-18 has an overall accuracy of 98.2%, which outperforms AlexNet at 91.3%. In addition, our proposed segmentation-free method using ResNet gives a comparable result on whole-image cell phenotype classification with three compound clusters, compared to what has been reported by Dürr and Sick [81], where an AlexNet based CNN model achieves a performance of 97.3% for single-cell phenotype classification, also with three compound clusters.

We also investigate the misclassified samples by our trained ResNet-18, and summarize them as confusion matrices in Figure 4.4. In total, 9 out of 504 samples are misclassified. It is noticed that almost all of the mis-classifications are related to Cluster A, which is within our expectation since we have much fewer samples in Cluster A than Cluster B and Cluster C. It's possible that our trained model may require more samples to learn discriminative features representing phenotypes that are induced by Cluster A compounds.

4.4.2 Four classes classification problem

Next, we further evaluate our segmentation-free method on the four classes classification problem (three compound clusters plus the mock class), where we also compare the performance, efficiency and applicability of our method with previous method of Dürr and Sick [81] in detail. Table 4.3 shows the performance comparison and Table 4.4 shows the efficiency and applicability comparison.

Table 4.2: Comparison of performance among different methods based on five-fold cross-validation (three classes classification).

	#1 (%)	#2 (%)	#3 (%)	#4 (%)	#5 (%)	Average (%)
Dürr and Sick [81][a]	-	-	-	-	-	97.2%
Ours with AlexNet	92.1%	94.1%	89.1%	91.1%	90.0%	91.3%
Ours with ResNet-18	99.0%	99.0%	100%	98.0%	95.0%	98.2%

[a]AlexNet based model.

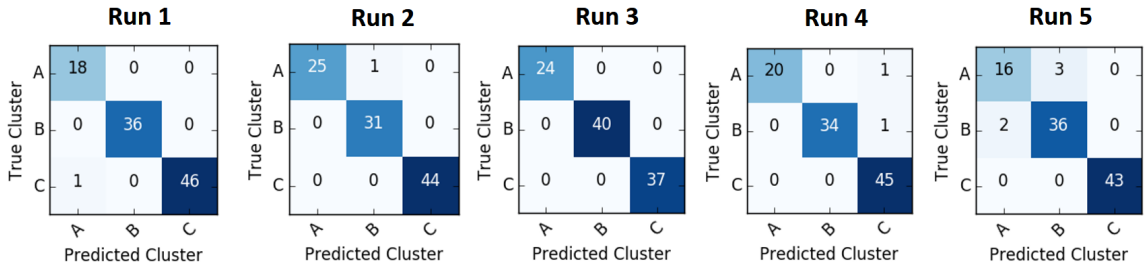


Figure 4.4: The confusion matrices of ResNet-18 model from five-fold cross-validation (three classes classification).

Overall, our method with ResNet-18 gives a comparable performance to that of Dürr and Sick [81] while it significantly improves the efficiency and applicability. The total modeling time required for the previous method [81] is the time for data preprocessing (mostly for segmentation) plus 3.75 hours for the training, assuming the epoch number is set to 100 (although Dürr and Sick [81] used 500 epoches.). As is shown in Table 4.4, not only our method saves time in not doing segmentation, but also in the training, as we have much fewer whole-image training samples as compared to single-cell images. Our method with ResNet-18 is able to give an accuracy of 93.8% with 0.83 hours training time.

In terms of applicability, our method does not require an overhead step in segmentation in order for the image data to be fit with the trained classifier, and no further voting step or other similar algorithm is needed during the prediction, which is required in previous method [81], *i.e.*, to merge the results from multiple single-cell phenotype classifications.

Table 4.3: Comparison of performance among different methods based on five-fold cross-validation (four classes classification).

	#1 (%)	#2 (%)	#3 (%)	#4 (%)	#5 (%)	Average (%)
Dürr and Sick [81][a]	-	-	-	-	-	93.4%
Ours with AlexNet	86.6%	82.3%	87.2%	93.3%	82.2%	86.3%
Ours with ResNet-18	95.6%	94.8%	92.2%	94.5%	92.1%	93.8%

[a]AlexNet based model.

Table 4.4: Comparison of performance, efficiency and applicability among different methods (four classes classification).

Methods	Performance	
	Parameter number	*Accuracy* (%)
Dürr and Sick [81][a]	∼1.3 million	93.4%
Ours with AlexNet	∼10.7 million	86.3%
Ours with ResNet-18	∼11.1 million	93.8%
	Efficiency: *Preprocessing and training time*[b]	
Dürr and Sick [81][a]	segmentation time + 3.75h	
Ours with AlexNet	2.83h	
Ours with ResNet-18	0.83h	
	Applicability	
	Segmentation required	*Voting algorithm required*
Dürr and Sick [81][a]	Yes	Yes
Ours with AlexNet	No	No
Ours with ResNet-18	No	No

[a]AlexNet based model.
[b]Time calculated assuming 100 epoches in the training phase.

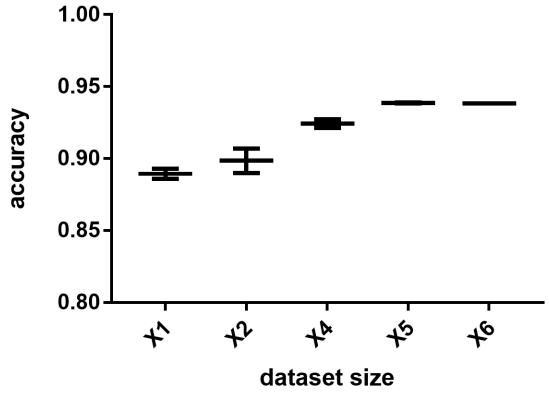


Figure 4.5: Model performance in terms of accuracy versus dataset size used for training. $\times 1$ stands for the original size.

4.4.3 Performance comparison of ResNet with its variants

With the compelling performance achieved by ResNet, we finally explore the possibility of using its variants to further improve the performance on the four classes classification problem. Before the comparison experiments, we also investigate the importance of data augmentation on the final performance and to what extent the model performance can still benefit from data augmentation. Given a dataset of size M, we can artificially increase the size to $M \times t$ by increasing the training steps per epoch t times, therefore we train our ResNet-18 for different step numbers for each epoch as shown in Figure 4.5 for a total number of 100 epoches. We can see that the accuracy increases with the growing of dataset size, and saturates at the stage where the dataset size is five times of the original size. We therefore use $t = 5$ for all the networks in the following comparison experiments.

In addition to WRN-18, ResNeXt-18 and PyramidNet-18, we also include wide PyramidNet-18, which is a wide version of PyramidNet-18 for the comparison. All training details are kept the same as described above, and again five-fold cross-validation is used. Table 4.5 shows the performance comparison of all residual networks for the four classes classification problem. Unexpectedly, although all networks give relatively better performance on the four classes classification problem compared to AlexNet, none of them outperforms the basic ResNet-18. One possible explanation could be the overfitting problem due to the limited training data size and the ResNet variants are tend to be more complex models than the basic ResNet (the use of data augmentation is helpful but still very limited). Moreover, it should be noted that both WRN-18 (k=2) and wide PyramidNet-18 (k=2) give similar performance as the ResNet-18, suggesting the potential of residual learning for the cell phenotype classification task.

Table 4.5: Comparison of performance among different residual networks (four classes classification). The networks include regular ResNet, WRN, ResNeXt, PyramidNet and Wide PyramidNet (a combination of WRN and PyramidNet). All networks are experimented with 18 layers.

	#1 (%)	#2 (%)	#3 (%)	#4 (%)	#5 (%)	Average (%)
Dürr and Sick [81][a]	-	-	-	-	-	93.4%
AlexNet	86.6%	82.3%	87.2%	93.3%	82.2%	86.3%
ResNet-18	95.6%	94.8%	92.2%	94.5%	92.1%	93.8%
WRN-18 (k=2)	95.3%	93.6%	93.0%	94.8%	91.8%	93.7%
WRN-18 (k=4)	94.5%	93.3%	91.0%	92.4%	93.6%	93.0%
ResNeXt-18	92.2%	92.4%	94.5%	93.3%	91.5%	92.8%
PyramidNet-18	93.6%	94.2%	92.7%	92.4%	93.6%	93.3%
Wide PyramidNet-18 (k=2)	93.6%	91.3%	94.5%	93.6%	94.5%	93.5%

[a] AlexNet based model.

4.5 Conclusion

In this chapter, we present a segmentation-free method for whole-image cell phenotype classification using both AlexNet and ResNet. Our results show that the latter has better performance. Compared to previous work [81] that uses single cell segmentation before training and evaluation, our pipeline is much more efficient since less data preprocessing is required. Moreover, our implemented ResNet-18 model also gives a comparable performance with accuracy rate of 93.8% on the four classes classification problem (three compound clusters plus the mock class) and 98.2% on the three classes classification problem (three compound clusters only) on BBBC022v1 dataset. Finally, we also explore the possibility of using several recent ResNet variants to further improve the performance.

The three compound clusters used in this work and previous work [80, 81] are known clusters that can lead to different phenotypes. However, BBBC022v1 dataset still contains a large number of chemical compounds that are not part of the above three clusters. Future work can be focus on the identification of new compound clusters in an unsupervised learning manner and then the revalidation with supervised learning methods. Moreover, to further increase the performance, more complex learning models can be explored.

Chapter 5

Case-Based Histopathological Malignancy Diagnosis

In practice, histopathological diagnosis of tumor malignancy often requires a human expert to scan through histopathological images at multiple magnification levels, after which a final diagnosis can be accurately determined. However, previous research on such classification tasks using convolutional neural networks primarily determine a diagnosis for a single magnification level. In this chapter, we propose a case-based approach for histopathological malignancy diagnosis, where a case is defined as a sequence of histopathological images at multiple magnification levels. Each magnification level represents a set of features that is complementary to each other, therefore by using multi-view representation fusion technique, a better representation of histopathological images could be learned through the joint learning of image sequences from multiple magnification levels. Effectively, through mimicking what a human expert would actually do, our approach makes a diagnosis decision based on features learned in combination at multiple magnification levels. Our results show that the case-based approach achieves better performance than the state-of-the-art methods when evaluated on BreaKHis, a histopathological image dataset for breast tumors.

5.1 Introduction

Histopathology is regarded as the gold standard method for cancer diagnosis, including almost all types of cancers, such as breast, lung, colon and prostate cancer [86, 87, 88]. Suspicious tissues are biopsied and the biopsy undergoes fixation, sectioning, and finally mounting on a slide. The biopsy section then is subjected to haematoxylin and eosin (H&E) staining which is a routinely used staining procedure that enhances tissue structure and cell morphology. A pathologist would then thoroughly examine the H&E stained slides under a microscope at multiple magnification levels,

searching for morphological signatures indicating the onset or progression of cancerous tissues whose presence determines whether the tumor should be diagnosed as benign or malignant. The whole process, however, can be very time-consuming, since it is often required that the pathologist switch between magnification levels and jump among different image locations [89]. In addition, the diagnosis from a pathologist can sometimes be subjective and heavily dependent on the experience of the pathologist [88].

In order to address the above problems, computer aided diagnosis (CAD) systems have been proposed to facilitate cancer diagnosis, not only to reduce labor work for the pathologist, but also to improve objectivity and consistency. Despite the work that has been done in the last few decades [87, 88, 90, 91], tumor malignancy classification remains still a challenge for most automatic cancer diagnosis applications due to the tremendous complexity of histopathological images, which can be due to various reasons including the staining variations in specimen treatment process [92] and the diversity of tissue characteristics in different cancers. Therefore, a robust and reliable CAD system for cancer diagnosis has to be designed to capture all discriminative features in histopathological images effectively. However, as has been pointed out by many researchers [87, 88, 90, 93], when using traditional classification approaches, the feature engineering step can be very difficult that requires a fair amount of expert domain knowledge.

Recently, a wide variety of new deep learning technologies [1, 94], such as the convolutional neural network (CNN), first developed by LeCun *et al*. [95], have achieved great success on various computer vision and pattern recognition tasks. Indeed, CNN has become the state-of-the-art method for image based classification problems, consistently outperforming traditional machine learning methods. More importantly, CNN can automatically extract discriminative features from images by itself. As a result, no hand-crafted feature engineering step is required anymore, which saves considerable efforts in most applications including histopathological image classification. In this work, we present our proposed case-based approach for histopathological malignancy diagnosis based on deep residual neural networks.

5.2 Literature Review

Due to its superior performance compared to traditional machine learning methods, CNN has been widely applied to histopathological cancer diagnosis problems. Cireşan *et al*. [96] use CNN to detect mitosis in breast cancer histological images and won the ICPR 2012 mitosis detection competition. Sirinukunwattana *et al*. [97] propose a spatially constrained CNN for nucleus detection and then a Neighboring Ensemble Predictor (NEP) coupled with CNN for nucleus classification in colon caner histological images, and achieve the highest average F1 score for this problem compared to other methods. Although both of the above two papers are not directly working on tumor

malignancy classification, their results could undoubtedly benefit cancer diagnosis, since both mitosis and nuclear characteristics are important indicators for cancerous tissue detection. Direct work on malignancy classification have also been published. For example, Cruz-Roa *et al*. [98] show that a CNN classifier achieves a balanced accuracy of 84.23% for the detection of invasive ductal carcinoma, where the best performance of methods using handcrafted features and classifiers is 78.74%. Similarly, Litjens *et al*. [99] also demonstrate that CNN improves the efficacy of prostate cancer diagnosis.

We note that the previous work mentioned above on histopathological image classification using convolutional neural networks are done on whole slide images (WSI), and the patches used for training are extracted from the original images at a certain fixed magnification level. However, an experienced pathologist would not choose to determine a diagnosis decision based on a single magnification level. In practice, it is often required that the pathologists evaluate the histopathological slides at multiple magnification levels [89, 100], as different magnifications give different features. For instance, lower magnification gives global texture information and tissue structure while higher magnification resolve more on cellular morphology and sub-cellular details [87]. Sometimes it is difficult to determine a diagnosis merely based on a single magnification level. Only by integrating all the features at multiple magnification levels, a confident diagnosis can be determined.

Recently, an image dataset BreaKHis is released [101], which provides histopathological images of breast tumor at multiple magnification levels (40×, 100×, 200× and 400×). Both traditional methods using handcrafted features [101] and CNN method [93] have been applied on this dataset for malignancy classification, and it has been shown that by combining different CNNs using fusion rules, the CNN performance has an improvement of 6% in classification accuracy, compared to traditional methods. However, one disadvantage of this paper [93], is that four CNN classifiers have to be trained, with one classifier specialized for each of the four magnifications. Seeking to find a better solution to this problem, Bayramoglu *et al*. [102] propose a magnification independent approach with both single-task (malignancy) and multi-task (malignancy and magnification) classification, where they ignore magnification information of the image and train a unique CNN classifier for all magnifications. Although the performance is slightly impaired, it indeed improves the efficiency. Recently, Jiang *et al*. [103] use a variation of ResNet, SE-ResNet (a Squeeze and Excitation network), to achieve accuracies between 90.66% and 93.81% over the full eight classes of the BreaKHis dataset. Nevertheless, when evaluated on the testing sets, all previous works [93, 102, 103] on BreaKHis dataset using CNN fail to determine a diagnosis for a patient based on features from multiple magnification levels at the same time. Instead, they give separate classification accuracy for each individual magnification, independent to other available magnifications.

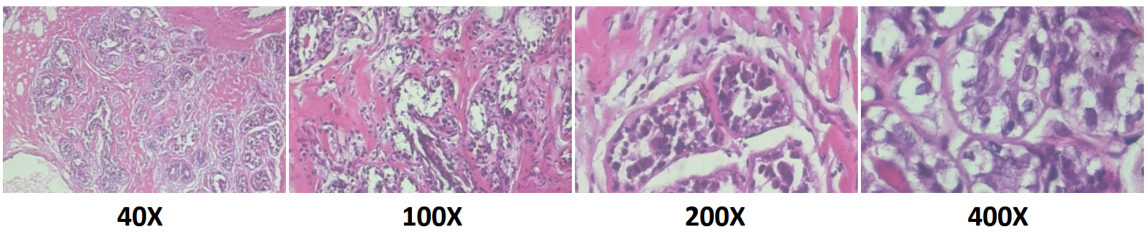


Figure 5.1: A typical histopathological case of breast tumor with different magnifications: 40×, 100×, 200× and 400×.

5.3 Methods

In order to build a more reasonable and reliable computer aided diagnosis system, we propose a case-based approach for histopathological malignancy classification, where a case is defined as a sequence of images including one or more images from each of all the available levels of magnification for a certain dataset. For example, for the BreaKHis dataset, a typical case could consist of one or more images at each of the following magnifications in order: 40×, 100×, 200× and 400× (Figure 5.1). A trained classifier should be able to learn all the features from different magnification images, and give a unique and more accurate result based on all information given (*e.g.*, tissue structure at lower magnification, cell phenotype at higher magnification), equivalent to how an histopathological expert would choose to perform analysis at multiple magnification levels.

In this section, we first present our algorithm that constructs a case-based image set from any given histopathological image dataset with multiple magnifications and malignancies (Subsection 5.3.1). We then introduce a CNN model to classify our histopathological cases (Subsection 5.3.2). Finally, we describe the three performance metrics that will be used for the evaluation of histopathological case-based classification (Subsection 5.3.3).

5.3.1 Case-based image set initialization

Histopathological image datasets are often given as images in multiple separated magnifications, but not as cases. Therefore, the first step is to build an appropriate number of histopathological cases based on the given dataset. To limit the size of the input set, the cases will include exactly one image from each magnification level. Algorithm 5.1 describes the initialization of a case-based image set from the original dataset with multiple magnifications and malignancies. Put simply, for each case build, the algorithm randomly chooses one image from each subset of images that belong to different magnifications, with the restriction that all images in the same case must have the same malignancy label, which will also be the final class label for the resulting case. For simplicity, we illustrate in Algorithm 5.1, assuming that two types of malignancy (benign and malignant) and four levels of magnification (40×, 100×, 200× and 400×) are available, which is the case for

Algorithm 5.1: Case-based image set initialization

Input : image sets $\mathrm{I}_{Malignancy \times Magnification}$, where *Malignancy* is the set of malignancy types, *e.g.*, {benign, malignant}, and *Magnification* is the set of magnifications, *e.g.*, {40, 100, 200, 400}

Output : $X \leftarrow$ data, $y \leftarrow$ label

Parameters: k = expected number of output cases

Initialize i = 0;
foreach $mal \in$ *Malignancy* **do**
    Initialize $counter_{mal}$ *= 0* ;
    Initialize new current combination X_i;
    repeat
        foreach $mag \in$ *Magnification* **do**
            randomly pick an image from image set $\mathrm{I}_{(mal,\ mag)}$ and add to X_i;
        end
        if *current combination* X_i *not in* X **then**
            add X_i to X;
            $y_i = mal$;
            i += 1;
            $counter_{mal}$ += 1;
        end
    until $counter_{mal} \geq k/|Malignancy|$;
end

BreaKHis dataset. However, this algorithm can be applied to any number of malignancy types and magnification levels.

The only parameter passed to the algorithm is the expected number of output cases k (which we assume to be a multiple of the number of types of malignancy). In training set initialization, we want this set size, which we will denote as k_{train}, to be relatively large in order to avoid over-fitting our model later on, but also not too large due to limited computational resources and running time. Therefore, this parameter needs to be fine-tuned for different problem settings as we will show in more detail in Section 5.4.

Algorithm 5.1 can be applied to both training and testing sets, depending on the inputs of image sets. Note in the training phase, a case consists of one single image from each magnification level, but not necessary from the same particular patient. The images can be randomly selected from different patients as long as they share the same malignancy. This is why the patient information does not come in Algorithm 5.1. However, in the testing phase, we may want the cases to be patient specific, which can be achieved by setting patient specific images as the input to Algorithm 5.1. After the whole process, the initialized case-based image sets are ready for training or evaluation.

5.3.2 ResNet-based classifier

We choose to use deep residual neural networks (ResNets) to classify the histopathological cases. ResNets are a special kind of convolutional neural networks that have residual units in parallel to regular convolutional layers. The design of residual units are quite flexible such that they can also be further engineered in order to get better performance [3, 25]. We start with a simple 18-layer ResNet model (ResNet-18), as this model can be easily adapted to even deeper models (*e.g.*, 152 layers) if required. The overall architecture of ResNet-18 is shown in Figure 4.3 (Chapter 4).

The model contains two types of residual units: the residual unit with an identity shortcut and the residual unit with a projection shortcut. The only difference between these two types is that in the projection shortcut, an additional convolutional layer is required due to the change of dimension from input to output. Each residual unit contains six sequential components: Batch Normalization, Rectified Linear Unit (ReLU), Convolution, Batch Normalization, ReLU and Convolution. An average pooling layer is used before the final fully connected layer.

5.3.3 Metrics

Spanhol *et al*. [93] have introduced two ways to report method performances for medical image classification: image recognition rate and patient recognition rate. Here, to accommodate for our case-based approach, however, we use a case level metric instead of an image level metric. The case recognition rate is defined as follows:

$$\text{Case Recognition Rate} = \frac{N_{rec}}{N_{all}}, \tag{5.1}$$

where N_{all} is the total number of all cases constructed for the testing set, and N_{rec} is the number of correctly classified cases.

Unlike the case recognition rate, the patient recognition rate takes patient information into account. For each patient p in the testing set, let $N_{p_{all}}$ be the total number of cases that belong to patient p, and $N_{p_{rec}}$ be the number of correctly classified cases for patient p, then the patient recognition rate can be defined as [93]:

$$\text{Patient Recognition Rate} = \frac{\sum_p (N_{p_{rec}}/N_{p_{all}})}{\text{Total Number of Patients}}. \tag{5.2}$$

In addition to the above two recognition rates, we also give a new metric defined at the diagnosis level. First, we give a final diagnosis to each patient in the testing set based on a simple voting strategy where we assume that the diagnosis is benign if the ratio of benign to malignant cases for

the patient p is above a threshold, *malignancy_threshold*:

$$\text{Patient Diagnosis }_p = \begin{cases} benign, & \text{if } \frac{N_{p_{benign}}}{N_{p_{all}}} > \textit{malignancy\_threshold} \\ malignant, & \text{otherwise,} \end{cases} \tag{5.3}$$

where $N_{p_{benign}}$ is the number of cases that are diagnosed as benign for patient p. For example, if *malignancy_threshold* is set to 0.5, the patient p is assigned a diagnosis of benign if more than half of the cases for patient p are classified as benign. Based on the diagnoses assigned to the patients, diagnosis accuracy for the classification is defined as the follows:

$$\text{Diagnosis Accuracy} = \frac{\text{Number of Correctly Diagnosed Patients}}{\text{Total Number of Patients}}. \tag{5.4}$$

We believe the diagnosis accuracy metric should be emphasized more for future research on histopathological diagnosis problems, as it is of utmost clinical importance that a computer-aided diagnosis system be able to give a final diagnosis for a patient, and based on the accuracy at which the diagnosis is correct or not, we can judge its performance.

5.4 Experiments and Results

This section evaluates our case-based approach for histopathological diagnosis that is proposed in Section 5.3.

5.4.1 Dataset

To test the proposed case-based approach for histopathological diagnosis, we use the BreaKHis database [101], a recently released dataset of breast tumor histopathological images. BreaKHis contains both benign and malignant breast tumor images, which were collected from 82 patients at multiple magnification levels (40×, 100×, 200× and 400×). Each patient may have a different number of images for each magnification. In total, there are 2480 benign and 5429 malignant images, with each image acquired in three channels (RGB).

Besides the histopathological images, BreaKHis also provides a five-fold protocol for testing. We use the same testing protocol as previous work [93, 102], where the whole dataset is split into training (70%) and testing (30%) set for five trials, such that none of the images associated with the patients in the training set are used in the testing set. In the end, 54 out of the total of 82 patients are grouped into the training set, and the rest of the 28 patients are used as evaluation samples for all the five folds.

The BreaKHis images are originally of size 700×460×3. To speed up the processing times and

lessen the memory requirements, the images are resized to $100\times100\times3$ for both the training and testing sets.

5.4.2 Implementation

With all images from BreaKHis, we implement Algorithm 5.1 to build histopathological cases for both the training and testing sets. To find the best parameter k_{train} for Algorithm 5.1 when initializing the training sets, we utilize fold 1 for a series of experiments by setting the number of output cases over a range of values from 100 to 40,000 as shown in Figure 5.2. After comparing the case-level accuracies for the different sizes of training sets, we choose the smallest size of the training set that gives the best accuracy as our final k_{train}. Note that for some of the smaller sizes of the training sets, we repeat some of the experiments independently three or five times since the performance of trained model can vary a lot for these sizes. The final chosen parameter k_{train} for training set initialization achieves a balance between computational resource requirement and performance. On the other hand, for testing set initialization, we simply use k_{test} = 30,000 for the size of the testing sets as evaluation on thirty thousands cases gives a quite stable estimation of model performance according to our trial experiments.

For all experiments, we implement our classifier ResNet using Keras, a deep learning library written in python with either TensorFlow or Theano as a backend [104]. We use Theano as the backend in this chapter. To optimize the weights, we use stochastic gradient descent, with a batch size of 100 to compute the gradients using back propagation. The initial learning rate is set to 0.001, decay by 1e-6 over each update, and Nestrov momentum is set to 0.9. We train our neural network for 100 epoches. All experiments are done on 4 Intel Xeon(R) E3-1271 v3 processors with a NVIDIA Quadro K2000/PCIe/SSE2 GPU with CUDA 7.5 installed in a Ubuntu 16.04 LTS.

5.4.3 Results

First, regarding the choice of k_{train}, as is shown in Figure 5.2, with the increase of the total number of cases used for training, the testing accuracy also increases and finally reaches the plateau. When the model is trained on only 100 cases, there is a large variation in model performance based on five independently conducted experiments. In the worst case, for 100 cases, the performance is not much better than a random guess. However, the model performance is significantly improved when trained on a large number of cases. In addition, the variation becomes smaller as well. From the bottom plot in Figure 5.2, we can see that the curve starts to converge when the number of cases is increased to $5,000$, and reaches a maximum at around $10,000$. To understand the effect of the choice of k_{train} on the running time, when setting k_{train} equal to $40,000$, the total training time required for a single fold is around 900 seconds per epoch, while it is 2 seconds per epoch for k_{train}

No. of cases / Experiments	100	1k	5k	10k	15k	20k	25k	30k	35k	40k
Exp. 1	0.8565	0.9156	0.9243	0.9317	0.9199	0.9124	0.9192	0.9276	0.9208	0.9294
Exp. 2	0.8133	0.9028	0.9382	0.9316						
Exp. 3	0.7758	0.9291	0.9222	0.9106						
Exp. 4	0.5008	0.9119								
Exp. 5	0.8501	0.9325								

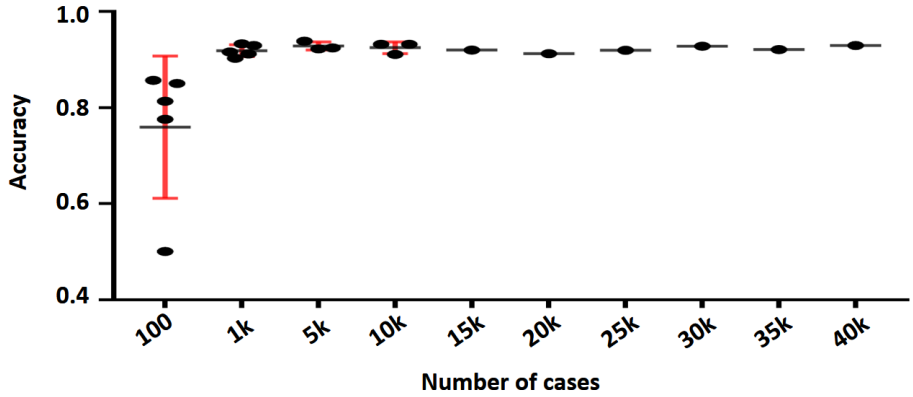


Figure 5.2: Performance in terms of case-level accuracy versus number of histopathological cases k_{train} used for training. Top table shows the testing accuracies in each experiment; The bottom plot is the visualization of the table.

set to 100. By setting our parameter k_{train} equal to $10,000$, we significantly reduce the running time by around four times, from 900 seconds to 226 seconds, when compared to k_{train} equal to $40,000$, without sacrificing accuracy. Therefore, we choose to set our parameter k_{train} equal to $10,000$ in training set initialization algorithm for all the following experiments.

With the parameters k_{train} and k_{test} set, we can then thoroughly evaluate our case-based approach based on five-fold testing protocol, using the metrics that we described in Subsection 5.3.3. For each fold, we first build the case-based training and testing sets using Algorithm 5.1, by setting k_{train} = 10,000 for the training set and k_{test} = 30,000 for the testing set. Note that both the training and testing sets are balanced in terms of the different malignancy types. After the models are trained, we then evaluate the model performance using the following three metrics: case recognition rate, patient recognition rate, and diagnosis accuracy. For the diagnosis of benign or malignant, we set *malignancy_threshold* to 0.5. Table 5.1 shows the final results.

Our case-based approach gives average accuracies of 91.48% (case-level), 86.36% (patient-level) and 88.57% (diagnosis-level) on the testing sets. As we are the first to use case-level and diagnosis-level accuracies, we can't compare the results for these metrics to previous results. However, based on patient-level accuracy, our case-based approach (86.36%) outperforms the multi-task CNN method (82.13%, average of four magnifications) [102] and the magnification independent

Table 5.1: Performance of case-based approach for histopathological malignancy diagnosis based on three metrics: case-level accuracy (Equation 5.1), patient-level accuracy (Equation 5.2) and diagnosis-level accuracy (Equation 5.4).

Accuracy Type	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Case Recogn. Rate	0.9246	0.8596	0.9355	0.9220	0.9323	0.9148
Patient Recogn. Rate	0.8731	0.8424	0.8753	0.8090	0.9182	0.8636
Diagnosis Accuracy	25/28	23/28	26/28	23/28	27/28	0.8857

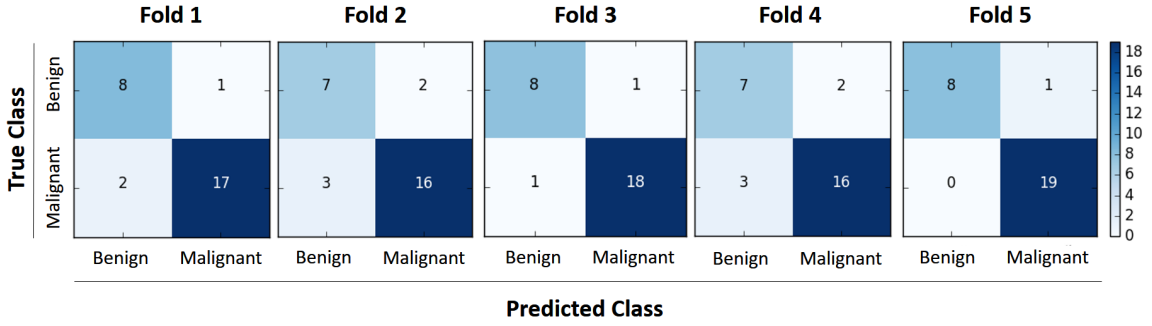


Figure 5.3: The confusion matrices of case-based approach for histopathological malignancy diagnosis in five folds.

single-task CNN method (83.25%, average of four magnifications) [102], and achieves a comparable performance to the best results obtained from the combination of four patch image extraction strategies and three fusion rules using a patch-based method for specific magnifications (40×: 90.0%; 100×: 88.4%; 200×: 84.6%; 400×: 86.1%) [93].

We further investigate the misclassified patients in terms of malignancy diagnosis for all five folds, and summarize the results as confusion matrices in Figure 5.3. In total, 16 out of 140 patient samples over the five folds are misclassified, with a false positive rate of 5.0% and a false negative rate of 6.43%.

5.5 Conclusion

In this chapter, we propose a case-based approach for histopathological malignancy diagnosis using deep residual neural networks. We first introduce an algorithm for case-based image set initialization for both training and testing based on histopathological images at multiple magnification levels, and then present a ResNet-based classifier and three metrics to report method performances for medical image classification. Finally, we evaluate our proposed approach using the breast tumor histopathological image dataset BreaKHis. Our results show that the case-based approach achieves better performance than the state-of-the-art methods. Moreover, we believe our case-based approach

is a more reasonable way for histopathological malignancy classification since it makes diagnosis decision based on features learned at multiple magnifications. Another principle advantage of our work over the previous work [93, 102] is that our method gives a single diagnosis for the patient, whereas in the previous work four potentially differing diagnoses are given for the same patient, one for each of four magnification levels. To be clinically applicable, these latter approaches would then require a final voting step or similar diagnosis selection step which are not discussed in their papers [93, 102]. For future work, more complex deep CNN architectures will be investigated.

Chapter 6

Disease Progression Learning

Medical images often have intrinsic characteristics that can be leveraged for neural network learning. For example, images that belong to different stages of a disease may continuously follow a certain progression pattern. In this chapter, we propose a novel method that leverages disease progression learning for medical image recognition. The disease progression learning emphasizes on the learning of the underlying connections among multiple stages of a disease, with each stage being a sequential view of the disease. In our method, sequences of images ordered by disease stages are learned by a neural network that consists of a shared vision model for feature extraction and a long short-term memory network for the learning of stage sequences. Auxiliary vision outputs are also included to capture stage features that tend to be discrete along the disease progression. Our proposed method is evaluated on a public diabetic retinopathy dataset, and achieves about 3.3% improvement in disease staging accuracy, compared to the baseline method that does not use disease progression learning. This proposed approach could be generally applied to any medical image recognition problems that involve disease progression.

6.1 Introduction

Many medical image recognition problems are associated with disease progression, for example, to predict survival time based on brain tumor images (*e.g.*, short-term survival, medium-term survival and long-term survival) [105], or a disease staging problem. As illustrated in Figure 6.1(a, b, c), the disease staging problem commonly exists across multiple modalities, including but not limited to classifying breast histopathological images into normal, benign, in-situ or invasive [5]; MRI images of white matter into mild, moderate or severe based on age-related changes [6]; and retinal fundus images into no-diabetic-retinopathy (NDR), simple-diabetic-retinopathy (SDR), pre-proliferative-diabetic-retinopathy (PPDR) or proliferative-diabetic-retinopathy (PDR) [7]. Although there is a

stage difference memory driven by the disease progression, most previous research works only consider each different stage as an independent class, and the memory information from the stage sequence is not explicitly represented in the neural network.

In this paper, we propose a novel method that leverages disease progression learning for medical image recognition. Concretely, given any medical recognition problem that is associated with a disease progression, we use long short-term memory (LSTM) to model the disease progression. To the best of our knowledge, we are the first to use LSTM for the learning of stage sequence along the disease progression for medical image recognition, with each stage represented by feature vectors that are extracted from a well-established vision model (e.g. GoogleNet or ResNet). Auxiliary outputs from the vision model are also adopted in order to capture stage features that may not be continuous along the disease progression. Our proposed method is evaluated on a diabetic retinopathy dataset, where it shows a performance increase of around 3.3% in disease staging accuracy, compared to the baseline method that is similar but without disease progression learning.

6.2 Literature Review

Deep learning has been widely applied to medical image recognition since its great success in natural image recognition, and has achieved state-of-the-art performance in various areas such as anatomical structure identification, lesion detection and classification [106, 107]. Unlike general object classification, medical images often have intrinsic characteristics that can be exploited to facilitate neural network learning for improved results. For example, medical recognition tasks on images acquired from computed tomography (CT) or magnetic resonance imaging (MRI) generally favor a 3D convolutional neural network (CNN) over a 2D CNN, due to the additional spatial information in three dimensions [108, 109]. Another example of neural networks that exploit medical intrinsic information is the BrainNetCNN, where special edge-to-edge, edge-to-node and node-to-graph convolutional filters are designed to leverage topological locality of brain networks for the prediction of neurodevelopmental outcomes [110].

LSTM is a widely-used network that is powerful for sequential data learning, as it has memory units that efficiently remember previous steps [32]. There are previous publications using LSTM for medical data, but mostly based on diagnostic text reports [111], 3D image stacks [112, 113], or clinical measurements/admissions [114, 115, 116], among which, some also use the concept of disease progression modeling [115, 116]. However, all previous work either do not clearly define stage classes for a disease progression, or still consider each stage as an independent class and the sequence learning only occurs within each individual stage class. *I.e.*, there is no explicit learning of the stage sequence itself, other than learning of temporal sequence (*e.g.*, a series of clinical events), or spatial sequence (*e.g.*, 3D MRI/CT images).

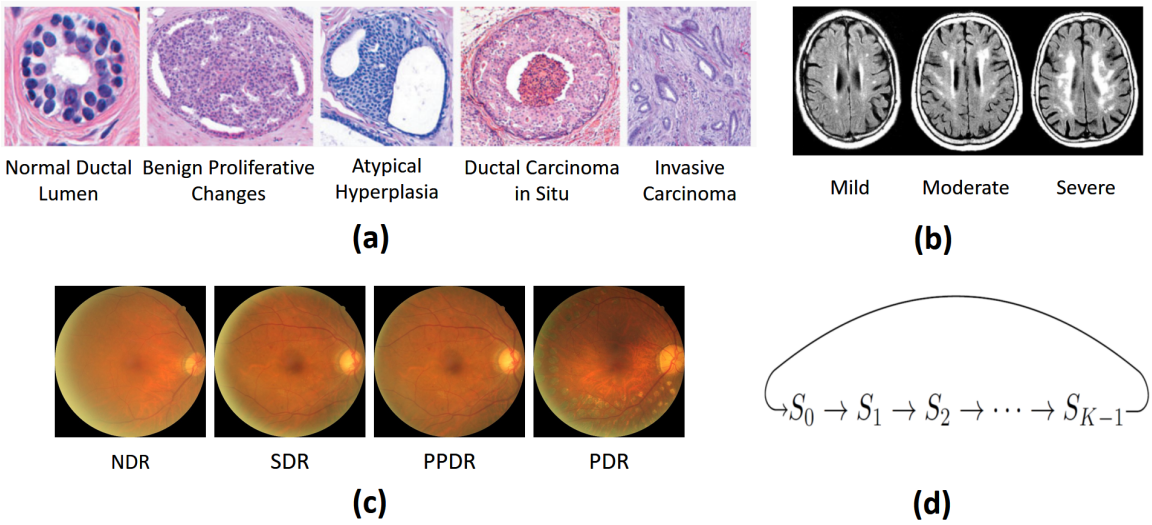


Figure 6.1: Examples of disease progression with different stages: (a) Breast cancer with five stages: normal, benign proliferation, atypical hyperplasia, in-situ and invasive [5]; (b) Aging related changes in white matter with three severity grades: mild, moderate and severe[6]; (c) Diabetic retinopathy with four stages: no-diabetic-retinopathy (NDR), simple-diabetic-retinopathy (SDR), pre-proliferative-diabetic-retinopathy (PPDR) and proliferative-diabetic-retinopathy (PDR) [7]; (d) Cyclic form of the stage sequence.

6.3 Methods

Here, we present our method for disease progression learning.

6.3.1 Disease progression learning

Given a disease progression with K sequential stages $S = \{S_k, k = 0, 1, ..., K-1\}$, with $S_k > S_{k-1}$ for each $k \in (0, K-1]$, where the greater-than sign indicates a disease progression, meaning stage S_k is a subsequent stage of S_{k-1}, *e.g.*, {*NDR, SDR, PPDR, PDR*} for diabetic retinopathy and {*normal, benign, in-situ, malignant*} for epithelial cancers, we want the neural network to learn disease stage progression by presenting the network with a sequence of images ordered by disease stage as the input $\boldsymbol{x} = [I^{S_0}, I^{S_1}, ..., I^{S_{K-1}}]$, where I^{S_k} is a randomly selected image that belongs to stage S_k, and the corresponding output is simply the ordered full sequence of all disease stages: $\boldsymbol{y} = [S_0, S_1, ..., S_{K-1}]$. However, with the above design, all the input samples would share the same output $\boldsymbol{y}$ that is fixed by the disease stage progression, making the network training meaningless. To overcome this problem, we artificially define $S_0 > S_{K-1}$ to make the stage sequence cyclic (Figure 6.1(d)), so that multiple ordered full sequences of all disease stages can be generated. Therefore, $[S_1, S_2, ..., S_{K-1}, S_0]$, for example, is also considered as a valid ordered full sequence containing all stages for a certain disease. As a result, the notation of stage sequence for a training sample can

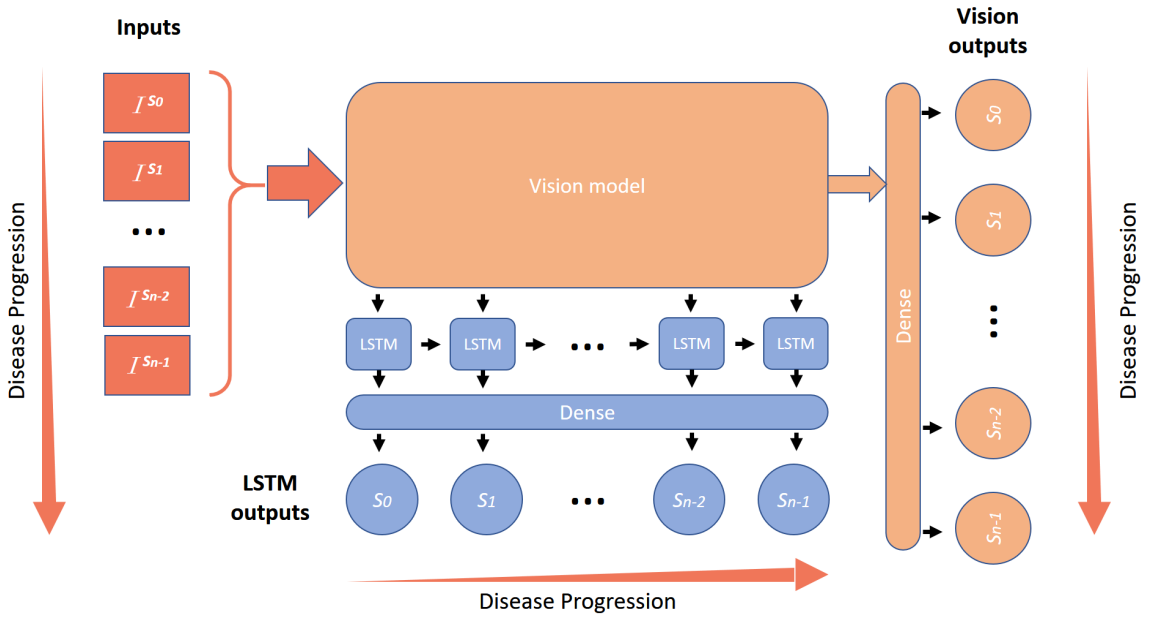


Figure 6.2: The architecture of our proposed network. Given the input of an image sequence $\boldsymbol{x} = [I^{S_0}, I^{S_1}, ..., I^{S_{n-1}}]$, the proposed method contains a vision model (*e.g.*, GoogleNet or ResNet) for the feature extraction, followed by a LSTM model for the purpose of disease progression learning. Auxiliary vision outputs are also included to capture stage features that tend to be discrete along the disease progression.

be updated more formally by the introduction of a modulo operation as the following:

$$\begin{aligned}(\boldsymbol{x}, \boldsymbol{y}) = (\ &[I^{S_{i+0 \pmod K}}, I^{S_{i+1 \pmod K}}, \ldots, I^{S_{i+K-1 \pmod K}}], \\ &[S_{i+0 \pmod K}, S_{i+1 \pmod K}, \ldots, S_{i+K-1 \pmod K}]\ ),\end{aligned} \tag{6.1}$$

where $i \in [0, K-1]$ can be considered as the step shift size, indicating the starting stage of a sequence, and K is the total number of disease stages. For simplicity of notation and explanation, we use the stage sequence of $i = 0$ as an illustration in the following.

As shown in Figure 6.2, the proposed method contains a vision model (*e.g.*, GoogleNet or ResNet) for the feature extraction, followed by a LSTM model for the purpose of disease progression learning. The vision model extracts a feature vector $\boldsymbol{z}^{S_k} \in \mathbb{R}^C$ from its corresponding input image I^{S_k} for each disease stage S_k, and then the concatenated feature vector sequence of all stages $\boldsymbol{z} = [\boldsymbol{z}^{S_0}, \boldsymbol{z}^{S_1}, ..., \boldsymbol{z}^{S_{K-1}}] \in \mathbb{R}^{K \times C}$ is given as the input to LSTM. The LSTM maintains a hidden state $\boldsymbol{h}^{S_k} \in \mathbb{R}^G$ and a cell state $\boldsymbol{c}^{S_k} \in \mathbb{R}^G$, which are updated at each stage step $k \in (0, K-1]$:

$$\boldsymbol{h}^{S_k}, \boldsymbol{c}^{S_k} = \text{LSTM}(\boldsymbol{z}^{S_k}, \boldsymbol{h}^{S_{k-1}}, \boldsymbol{c}^{S_{k-1}}). \tag{6.2}$$

The hidden state sequence of all stages $\boldsymbol{h} = [\boldsymbol{h}^{S_0}, \boldsymbol{h}^{S_1}, ..., \boldsymbol{h}^{S_{K-1}}] \in \mathbb{R}^{K \times G}$ is collected from the

multi-output LSTM to learn a softmax classifier $\mathcal{F}$, which finally outputs the hypotheses $\hat{\boldsymbol{y}}$ of the true stage labels $\boldsymbol{y}$:

$$\hat{\boldsymbol{y}} = \mathcal{F}(\boldsymbol{h}; \boldsymbol{\theta}), \tag{6.3}$$

where $\boldsymbol{\theta}$ is the parameter for classifier $\mathcal{F}$. Note that $\hat{\boldsymbol{y}} = [\hat{\boldsymbol{y}}^{S_0}, \hat{\boldsymbol{y}}^{S_1}, ..., \hat{\boldsymbol{y}}^{S_{K-1}}] \in \mathbb{R}^{K \times K}$, where each sequence element represents a probability distribution over K stage labels for its corresponding stage. Also, it is worth noting that although for the simplicity of notation, we denote both $\boldsymbol{h}$ and $\hat{\boldsymbol{y}}$ as vector sequences of all stages, the softmax classifier $\mathcal{F}$ is actually applied to each individual hidden state $\boldsymbol{h}^{S_k}$ independently to obtain its corresponding hypothesis $\hat{\boldsymbol{y}}^{S_k}$.

Our loss function is a weighted summation of cross entropy losses at all stages:

$$Loss(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{k=0}^{K-1} \alpha_k \cdot l(\hat{\boldsymbol{y}}^{S_k}, S_k), \tag{6.4}$$

where α_k is the loss weight for each stage output and

$$l(\hat{\boldsymbol{y}}^{S_k}, S_k) = \sum_{j=0}^{K-1} -\log \hat{y}_j^{S_k} \cdot \delta(S_k = S_j) \tag{6.5}$$

is the cross entropy loss function, with $\hat{y}_j^{S_k}$ denoting for the probability value of image I^{S_k} belonging to the stage label S_j.

6.3.2 Auxiliary vision outputs

In addition to disease progression learning, the network should also be able to capture stage-wise discriminative features that tend to be discrete among different stages along the disease progression, *e.g.*, features that only appear in a certain disease stage. Given this thought, we also design auxiliary outputs directly from the vision model extracted features $\boldsymbol{z}$ with another softmax classifier $\mathcal{F}_v$ on top of the vision model:

$$\hat{\boldsymbol{y}}_v = \mathcal{F}_v(\boldsymbol{z}; \boldsymbol{\theta}_v). \tag{6.6}$$

To distinguish the two classifiers on top of LSTM model ($\mathcal{F}_l$) and vision model ($\mathcal{F}_v$), Equation 6.3 is updated as:

$$\hat{\boldsymbol{y}}_l = \mathcal{F}_l(\boldsymbol{h}; \boldsymbol{\theta}_l). \tag{6.7}$$

Similarly to LSTM outputs, the loss for auxiliary vision outputs is also defined as the weighted summation of cross entropy losses at all disease stages. Taking both losses into account, the final

loss function for our proposed network is:

$$Loss(\hat{\boldsymbol{y}}_l, \hat{\boldsymbol{y}}_v, \boldsymbol{y}) = \sum_{k=0}^{K-1} \big(\alpha_k \cdot l(\hat{\boldsymbol{y}}_l^{S_k}, S_k) + \beta_k \cdot l(\hat{\boldsymbol{y}}_v^{S_k}, S_k)\big). \tag{6.8}$$

The loss weights α_k and β_k for each stage should be chosen depending on each individual disease progression. A heuristic choice is that more weight should be given to β_k if the stage features tend to be more discrete.

6.3.3 Non-regression disease stage sequence

In the context of disease progression learning (Section 6.3.1), we artificially define $S_0 > S_{K-1}$ to make the stage sequence cyclic, so that more variations of stage sequences can be generated for a particular disease to facilitate the neural network training. However, it is a bit counter-intuitive at the first thought to define $S_0 > S_{K-1}$, which violates the concept of monotonic disease progression. Alternatively, we could also use other approaches that do not require any similar assumptions, such as non-regression disease stage sequence, where the sequence can still start with any arbitrary disease stages and the disease simply stops progression when it reaches its final stage. In this way, the order of disease stages is still maintained without the assumption defined in the cyclic strategy. For example, $[S_1, S_2, ..., S_{K-2}, S_{K-1}, S_{K-1}]$ is also a valid non-regression stage sequence. Therefore, a training sample of non-regression disease stage sequence can be formulated as the following:

$$\begin{aligned}(\boldsymbol{x}, \boldsymbol{y}) = (\ &[I^{S_{\min(i+0,K-1)}}, I^{S_{\min(i+1,K-1)}}, ..., I^{S_{\min(i+K-1,K-1)}}], \\ &[S_{\min(i+0,K-1)}, S_{\min(i+1,K-1)}, ..., S_{\min(i+K-1,K-1)}]\ ),\end{aligned} \tag{6.9}$$

where $i \in [0, K-1]$ is the step shift size, indicating the starting stage of a sequence, and K is the total number of disease stages.

6.3.4 Testing phase

In the testing phase, given an image I_{test}, an artificial image sequence is generated by repeating I_{test} for K iterations, and then the sequence is fed into the trained network. Due to the design of our network, we have two options to predict its stage label based on either LSTM output $\hat{\boldsymbol{y}}_l$ or vision output $\hat{\boldsymbol{y}}_v$ (see Figure 6.2). In both cases, only the first stage output in the sequence is reported as the final predicted label. Note that the input image sequence can be started with any arbitrary stage as we described earlier in Section 6.3.1 and Section 6.3.3. Therefore a faked sequence starting with I_{test} can still result in a reasonable prediction of stage label for I_{test} based on its corresponding first stage output, although the rest stage outputs are invalid since there is no disease progression in the

input sequence.

6.3.5 Baseline network

The baseline network for this study is the same vision model followed by the same softmax classifier $\mathcal{F}_v$ as used in our proposed network, except that the input is a single image, similar to most previous research work on medical image classification.

For a fair comparison, we also make sure that both networks are trained on exactly the same amount of augmented data, which we will describe in detail in Section 6.4.2, to rule out the possibility that a superior performance could be gained simply due to larger training samples, given the fact that the input size of our proposed network is K times bigger than that of the baseline network.

6.4 Experiments and Results

6.4.1 Dataset

To evaluate our proposed network, we choose to use a recently published dataset on diabetic retinopathy [7], which contains fundus photographs of four stages. The dataset has two sets of class labels based on whether to grade with wider retinal area: Davis grading of one figure and Davis grading of concatenated figures, and we use the former in our experiments. There are in total 9939 images, with 6561 of NDR, 2113 of SDR, 460 of PPDR and 805 of PDR.

Unlike previous work [7], we address the data imbalance problem by undersampling by a random selection of 460 images from each stage class for each independent experiment, and moreover, all the images are resized to 200×200 instead of 1272×1272 to speed up training, since it is not our purpose here to compete with the result in [7], but rather to investigate the advantage of disease progression learning in stage classification.

Out of the final resulting dataset, 10% of the data is randomly reserved for testing, and the rest is further split into two parts: 90% for training and 10% for validation. Data augmentation is performed on the training set by only using random rotations from -5° to 5° without shifts or flips, as all the images share the same position.

6.4.2 Implementation details

We examine two types of vision models in our experiments: GoogleNet and ResNet-50, both of which were pretrained on ImageNet. The freeze layer is set to 64 for GoogleNet and 36 for ResNet-50. We use $C = 256$ for the dimension of extracted feature vectors, $G = 256$ for the LSTM model and $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{1} \in \mathbb{R}^K$. All models are implemented in Keras with Theano backend, and trained

Table 6.1: Performance comparisons of the baseline method and our proposed method trained on cyclic stage sequences (with both vision outputs and LSTM outputs).

Vision model	Steps [a]	Method	*Accuracy*
GoogleNet Inception v3	100	BASELINE	$57.0 \pm 3.2\%$
		OURS(VISION OUTPUT)	**59.4** $\pm 3.1\%$
		OURS(LSTM OUTPUT)	$59.2 \pm 3.3\%$
	500	BASELINE	$59.5 \pm 3.3\%$
		OURS(VISION OUTPUT)	**63.1** $\pm 2.7\%$
		OURS(LSTM OUTPUT)	$62.9 \pm 3.0\%$
ResNet-50	100	BASELINE	$57.3 \pm 6.4\%$
		OURS(VISION OUTPUT)	**60.7** $\pm 5.4\%$
		OURS(LSTM OUTPUT)	$60.7 \pm 5.7\%$
	500	BASELINE	$58.1 \pm 4.8\%$
		OURS(VISION OUTPUT)	$61.4 \pm 3.8\%$
		OURS(LSTM OUTPUT)	**61.7** $\pm 3.9\%$

[a]Training steps per epoch.

using stochastic gradient descent, with the initial learning rate set to 0.001, decay by 1e-6 over each update, and Nestrov momentum is set to 0.9.

For each independent experiment, we evaluate the proposed network and baseline network on the same random split of the dataset. We repeat the above comparison experiment for 20 times, each with a different split. To make sure both networks are trained on the same amount of augmented data, the same number of iteration steps per epoch (*i.e.*, 100 steps and 500 steps) is used in the training, until the validation loss does not drop for ten epochs (patience = 10, with the maximum number of epochs set to 100). In order to compensate the input size difference (*i.e.*, one-image input against four-image input), the batch size for the baseline network training is set to 64 and decreased to 16 for our proposed network. Note that this is the only difference in hyper-parameter settings for the training of the two networks.

6.4.3 Results

Table 6.1 shows the performance comparison of our proposed method and the baseline method, with both results of GoogleNet Inception v3 and ResNet-50 as the vision model. As shown in the table, our proposed method outperforms the baseline across all experimental settings that are used in this chapter (choices of vision model and training steps per epoch), and on average, there is about 3.3% accuracy gain (2.4%, 3.6%, 3.4% and 3.6% for each setting respectively). However, we do not observe a significant performance difference between vision outputs and LSTM outputs (59.4% vs 59.2%, 63.1% vs 62.9%, 60.7% vs 60.7% and 61.4% vs 61.7%) for our proposed method, probably

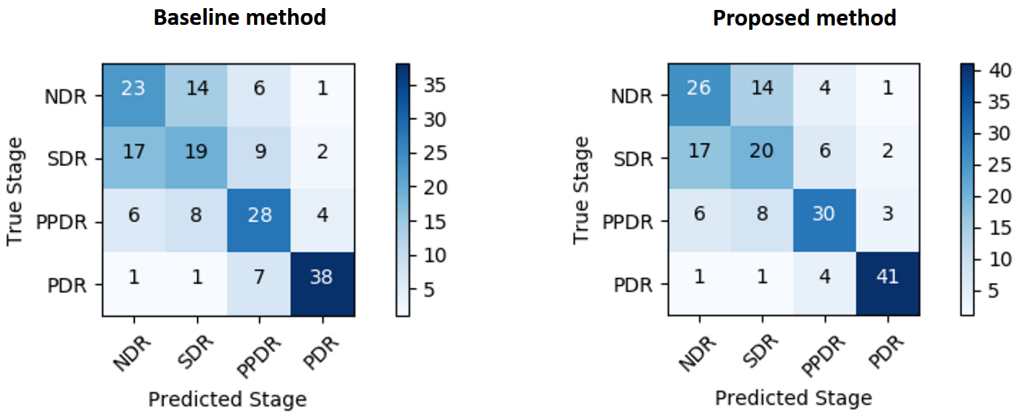


Figure 6.3: Confusion matrices of the baseline method (left) and our proposed method (right) using GoogleNet as the vision model with 500 training steps per epoch.

due to the joint training of our model, so that the LSTM backpropagates the gradients to CNN to force it to learn features that are more stage-relevant, thus improving the vision outputs.

For both methods, accuracy performance can be improved with an increased number of training steps per epoch, which is within our expectation since more steps means more data augmentation. The best performance in our performed experiments is 63.1% for our proposed method and 59.5% for the baseline method, both of which are using GoogleNet Inception v3 pretrained on ImageNet with 500 training steps per epoch, and the confusion matrices are given in Figure 6.3. It is also noted that most of the misclassified samples are located in NDR and SDR, which is reasonable since the two stages are often quite indistinguishable from a clinical perspective.

We believe that there is still room for performance improvement on this particular diabetic retinopathy problem by doing more data augmentation, using oversampling instead of undersampling (*i.e.*, to make full use of images in the dataset), and using the original high resolution images. However, the purpose of this chapter is to present and validate the idea of disease progression learning, and we leave the optimization for our next study.

To further test the alternative design of disease progression learning using non-regression stage sequences, we perform the comparison experiments using the same above settings, except that the model is trained on non-regression stage sequences (described in Section 6.3.3), instead of cyclic stage sequences (described in Section 6.3.1). As shown in Table 6.2, the non-regression stage sequence fails to outperform cyclic stage sequence in all experiment settings, and in some case (GoogleNet and 100 training steps per epoch), it is even worse compared to the baseline method. The best result achieved by non-regression stage sequence is 61.8%, which is still worse than that of cyclic sequence 63.1%.

Based on the above results, we argue that despite being counter-intuitive at first thought, the

Table 6.2: Performance comparisons of the baseline method and our proposed method trained on non-regression stage sequences (with both vision outputs and LSTM outputs).

Vision model	Steps [a]	Method	*Accuracy*
GoogleNet Inception v3	100	BASELINE	**57.5** $\pm$ 2.9%
		OURS(VISION OUTPUT)	56.2 $\pm$ 3.8%
		OURS(LSTM OUTPUT)	55.7 $\pm$ 4.4%
	500	BASELINE	59.1 $\pm$ 3.5%
		OURS(VISION OUTPUT)	**61.8** $\pm$ 2.7%
		OURS(LSTM OUTPUT)	60.9 $\pm$ 2.5%
ResNet-50	100	BASELINE	56.5 $\pm$ 5.8%
		OURS(VISION OUTPUT)	58.1 $\pm$ 5.0%
		OURS(LSTM OUTPUT)	**59.6** $\pm$ 4.3%
	500	BASELINE	58.2 $\pm$ 3.5%
		OURS(VISION OUTPUT)	**60.5** $\pm$ 2.5%
		OURS(LSTM OUTPUT)	60.1 $\pm$ 2.8%

[a]Training steps per epoch.

cyclic strategy is merely to facilitate the training, which allows for cyclic variations of the training data that then de-emphasizes the first class as the first node, etc. This in turn allows the same disease progression sequence to be used for more training data. Since the change will be very abrupt from the last node to the first node, the learning will still retain the last node's characteristics as being substantially distinct from the first node of the sequence.

6.5 Conclusion

In this chapter, we present a novel method for medical image recognition by leveraging disease progression learning, where stage sequences are learned by LSTM after feature extraction with a shared vision model for the images from each stage. Compared to the baseline method that is a pure vision model, our proposed method has an average of 3.3% accuracy increase based on our performed experiments, when evaluated for the problem of disease staging on a diabetic retinopathy dataset.

Chapter 7

Conclusions

Inspired by the fact that human typically integrate all sources of available information in multiple views for an efficient learning or an optimal decision, multi-view learning has been a popular research topic in machine learning. In this thesis, we address several problems in both medical and non-medical domains from the perspective of multi-view learning, with the emphasis on learning the underlying connections among different views. The problems addressed are text-to-image synthesis, cell phenotype classification, histopathological malignancy diagnosis and disease progression learning. The given multi-view data associated with the above problems can be present within a single modality or across multiple modalities, *e.g.*, image and text. Therefore, the thesis can be divided into two parts based the number of modalities that the data is presented in: (1) learning through text-image pairs in a multi-modal setting (*i.e.*, views in both text and image modalities), and (2) learning through image sequences in a unimodal setting (*i.e.*, all views in the image modality).

In the first part of the thesis (learning through text-image pairs), we address the problem of text-to-image synthesis, where we propose a dual adversarial inference mechanism to learn representations that are aligned between the two views (text and image), given the paired text-image data. The learned representations include not only the content information that is present in both views, but also the style information that is present in the image view while missing in the text view. We demonstrate the disentanglement of content and style with our proposed method, and show the importance of incorporating an inference mechanism in the model performance; In the second part of the thesis (learning through image sequences), it is often the case that the original data must be reshaped into associated sequences in order for the multi-view learning approaches to be applicable, where each element in the sequence represents a distinct view. The feature sets representing the given multi-view data could be either independent or complementary to each other, depending on the problem setting. We address the cell phenotype classification problem by considering cell

component channels as different views of cell phenotype, the histopathological malignancy diagnosis problem by constructing image sequences from different magnification levels, and the disease progression learning problem by considering each disease stage as a sequential view of the disease.

From the multi-view representation learning point of view, we cover examples in both categories of multi-view representation alignment and multi-view representation fusion. More specifically, the text-to-image synthesis requires the representations to be aligned as matched or mismatched text-image pairs, while the disease progression learning aligns the representations from different disease stages in a sequential order. The cell phenotype classification and histopathological malignancy diagnosis problems on the other hand, focus on the fusion of the learned representations from multiple views for a better generalized representation of the data. The distinct feature sets from multiple channels in cell phenotype classification are independent to each other, since they give information on different cell components, and certain cell component may be sufficient to determine a specific cell phenotype. In histopathological malignancy diagnosis, the feature sets from multiple magnification levels are complementary to each other, since all of them must be taken into consideration before a final diagnosis decision can be accurately determined.

Future perspectives

Modalities other than text-image pairs We propose a dual adversarial inference mechanism in Chapter 3 and validate the idea in the framework of text-to-image synthesis, where only two modalities are involved, *i.e.*, text and image. However, this method could also be applied to modalities other than text and image. Therefore, as a future work, we would like to extend our work in the areas such as text-to-speech or speech-to-text tasks. In addition, the number of modalities can grow beyond two, depending on real-life use cases.

More views in image sequences We show the potential of leveraging disease progression learning for medical image recognition problems in Chapter 6. However, being constrained by the available datasets, we construct image sequences consisting only four disease stages, thus limiting the power of disease progression learning. One of the future work could be focused on validating our approach in a long sequence, where a continuous progression pattern is more evident.

Bibliography

[1] 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.

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

[3] 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.

[4] Yingming Li, Ming Yang, and Zhongfei Mark Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[5] Harold J Burstein, Kornelia Polyak, Julia S Wong, Susan C Lester, and Carolyn M Kaelin. Ductal carcinoma in situ of the breast. *NEJM*, 350(14):1430–1441, 2004.

[6] Domenico Inzitari, Giovanni Pracucci, Anna Poggesi, et al. Changes in white matter as determinant of global functional decline in older independent outpatients: three year follow-up of ladis (leukoaraiosis and disability) study cohort. *Bmj*, 339:b2477, 2009.

[7] Hidenori Takahashi, Hironobu Tampo, Yusuke Arai, Yuji Inoue, and Hidetoshi Kawashima. Applying artificial intelligence to disease staging: Deep learning for improved staging of diabetic retinopathy. *PloS one*, 12(6):e0179790, 2017.

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

[9] Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.

[10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[11] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[12] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[13] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[14] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[15] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[16] Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*, 2013.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[18] Haohan Wang, Bhiksha Raj, and Eric P Xing. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[19] 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.

[20] 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.

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

[22] 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.

[23] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[26] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5987–5995. IEEE, 2017.

[28] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6307–6315. IEEE, 2017.

[29] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[30] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[31] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[34] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2017.

[35] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[36] 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.

[37] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *ICLR*, 2017.

[38] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[39] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223, 2017.

[40] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[41] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3-4):321–377, 12 1936.

[42] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255, 2013.

[43] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[44] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[45] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016.

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

[47] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[48] 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. In *ICCV*, 2017.

[49] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008.

[50] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

[51] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[52] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[53] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *CVPR*, 2018.

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

[55] Ayushman Dash, John Cristian Borges Gamboa, Sheraz Ahmed, Marcus Liwicki, and Muhammad Zeshan Afzal. Tac-gan-text conditioned auxiliary classifier generative adversarial network. In *arXiv preprint arXiv:1703.06412*, 2017.

[56] Miriam Cha, Youngjune L Gown, and HT Kung. Adversarial learning of semantic relevance in text to image synthesis. In *AAAI*, 2019.

[57] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[58] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *CVPR*, 2019.

[59] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*, 2018.

[60] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *ICLR*, 2017.

[61] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[62] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Alice: Towards understanding adversarial learning for joint distribution matching. In *NIPS*, 2017.

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

[64] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *ICCV*, 2017.

[65] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, 2017.

[66] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

[67] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *ICML*, 2018.

[68] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016.

[69] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[70] Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *NIPS*, 2018.

[71] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[72] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *CVPR*, 2016.

[73] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[74] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *NIPS*, 2018.

[75] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. In *arXiv preprint arXiv:1710.10916*, 2017.

[76] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3):287–314, 1994.

[77] Ilyes Khemakhem, Diederik P Kingma, and Aapo Hyvärinen. Variational autoencoders and nonlinear ica: A unifying framework. 2019.

[78] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[79] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *CVPR*, 2018.

[80] Sigrun M Gustafsdottir, Vebjorn Ljosa, Katherine L Sokolnicki, J Anthony Wilson, Deepika Walpita, Melissa M Kemp, Kathleen Petri Seiler, Hyman A Carrel, Todd R Golub, Stuart L Schreiber, et al. Multiplex cytological profiling assay to measure diverse cellular states. *PloS one*, 8(12):e80999, 2013.

[81] Oliver Dürr and Beate Sick. Single-cell phenotype classification using deep convolutional neural networks. *Journal of biomolecular screening*, 21(9):998–1003, 2016.

[82] Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. *Molecular systems biology*, 13(4):924, 2017.

[83] Philipp Eulenberg, Niklas Köhler, Thomas Blasi, Andrew Filby, Anne E Carpenter, Paul Rees, Fabian J Theis, and F Alexander Wolf. Reconstructing cell cycle and disease progression using deep learning. *Nature communications*, 8(1):463, 2017.

[84] Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nat Methods*, 9(7):637, 2012.

[85] Qicheng Lao and Thomas Fevens. Case-based histopathological malignancy diagnosis using convolutional neural networks. In *British Machine Vision Conference 2017, BMVC*, 2017.

[86] Raphael Rubin, David S Strayer, Emanuel Rubin, et al. *Rubin's pathology: clinicopathologic foundations of medicine*. Lippincott Williams & Wilkins, 2008.

[87] Metin N Gurcan, Laura E Boucheron, Ali Can, Anant Madabhushi, Nasir M Rajpoot, and Bulent Yener. Histopathological image analysis: A review. *IEEE reviews in biomedical engineering*, 2:147–171, 2009.

[88] Lei He, L Rodney Long, Sameer Antani, and George R Thoma. Histology image analysis for carcinoma detection and grading. *Computer methods and programs in biomedicine*, 107(3):538–556, 2012.

[89] Lucia Roa-Peña, Francisco Gómez, and Eduardo Romero. An experimental study of pathologist's navigation patterns in virtual microscopy. *Diagnostic pathology*, 5(1):71, 2010.

[90] Cigdem Demir and Bülent Yener. Automated cancer diagnosis based on histopathological images: a systematic survey. *Rensselaer Polytechnic Institute, Tech. Rep*, 2005.

[91] Mitko Veta, Josien PW Pluim, Paul J Van Diest, and Max A Viergever. Breast cancer histopathology image analysis: A review. *IEEE Transactions on Biomedical Engineering*, 61(5):1400–1411, 2014.

[92] Michael T McCann, John A Ozolek, Carlos A Castro, Bahram Parvin, and Jelena Kovacevic. Automated histology analysis: Opportunities for signal processing. *IEEE Signal Processing Magazine*, 32(1):78–87, 2015.

[93] Fabio Alexandre Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. Breast cancer histopathological image classification using convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2560–2567. IEEE, 2016.

[94] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[95] 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.

[96] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer, 2013.

[97] Korsuk Sirinukunwattana, Shan E Ahmed Raza, Yee-Wah Tsang, David RJ Snead, Ian A Cree, and Nasir M Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE transactions on medical imaging*, 35(5):1196–1206, 2016.

[98] Angel Cruz-Roa, Ajay Basavanhally, Fabio González, Hannah Gilmore, Michael Feldman, Shridar Ganesan, Natalie Shih, John Tomaszewski, and Anant Madabhushi. Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In *SPIE medical imaging*, pages 904103–904103. International Society for Optics and Photonics, 2014.

[99] Geert Litjens, Clara I Sánchez, Nadya Timofeeva, Meyke Hermsen, Iris Nagtegaal, Iringo Kovacs, Christina Hulsbergen-Van De Kaa, Peter Bult, Bram Van Ginneken, and Jeroen Van Der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, 6, 2016.

[100] David Romo, Juan D García-Arteaga, Pablo Arbeláez, and Eduardo Romero. A discriminant multi-scale histopathology descriptor using dictionary learning. In *SPIE Medical Imaging*, pages 90410Q–90410Q. International Society for Optics and Photonics, 2014.

[101] Fabio A Spanhol, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. A dataset for breast cancer histopathological image classification. *IEEE Transactions on Biomedical Engineering*, 63(7):1455–1462, 2016.

[102] Neslihan Bayramoglu, Juho Kannala, and Janne Heikkilä. Deep learning for magnification independent breast cancer histopathology image classification. In *23th International Conference on Pattern Recognition, ICPR*, 2016.

[103] Yun Jiang, Li Chen, Hai Zhang, and Xiao Xiao. Breast cancer histopathological image classification using convolutional neural networks with small se-resnet module. In *PloS one*, 2019.

[104] François Chollet. Keras, 2015. `https://github.com/fchollet/keras`.

[105] Dong Nie, Han Zhang, Ehsan Adeli, Luyan Liu, and Dinggang Shen. 3d deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. In *MICCAI*, pages 212–220. Springer, 2016.

[106] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, et al. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[107] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[108] Adrien Payan and Giovanni Montana. Predicting alzheimer's disease: a neuroimaging study with 3d convolutional neural networks. *arXiv preprint arXiv:1502.02506*, 2015.

[109] Xiaojie Huang, Junjie Shan, and Vivek Vaidya. Lung nodule detection in ct using 3d convolutional neural networks. In *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*, pages 379–383. IEEE, 2017.

[110] Jeremy Kawahara, Colin J Brown, Steven P Miller, et al. Brainnetcnn: convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage*, 146:1038–1049, 2017.

[111] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *CVPR*, pages 6428–6436, 2017.

[112] Jianxu Chen, Lin Yang, Yizhe Zhang, Mark Alber, and Danny Z Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. In *NIPS*, pages 3036–3044, 2016.

[113] Wufeng Xue, Andrea Lum, Ashley Mercado, Mark Landis, James Warrington, and Shuo Li. Full quantification of left ventricle via deep multitask learning network respecting intra-and inter-task relatedness. In *MICCAI*, pages 276–284. Springer, 2017.

[114] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[115] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Deepcare: A deep dynamic memory model for predictive medicine. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 30–41. Springer, 2016.

[116] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In *Machine Learning for Healthcare Conference*, pages 301–318, 2016.