Universiteit
Leiden
The Netherlands

Exploring images with deep learning for classification, retrieval and synthesis

Liu, Y.

Citation
Liu, Y. (2018, October 24). *Exploring images with deep learning for classification, retrieval and synthesis*. *ASCI dissertation series*. Retrieved from https://hdl.handle.net/1887/66480

Version:	Not Applicable (or Unknown)
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/66480

Note: To cite this publication please use the final published version (if applicable).

Cover Page

Universiteit Leiden

The handle http://hdl.handle.net/1887/66480 holds various files of this Leiden University dissertation.

Author: Liu, Y.
Title: Exploring images with deep learning for classification, retrieval and synthesis
Issue Date: 2018-10-24

Exploring Images with Deep Learning for Classification, Retrieval and Synthesis

Yu Liu

Copyright © 2018 Yu Liu, All Rights Reserved

ISBN 978-94-6375-139-1

Printed by Ridderprint BV, The Netherlands

An electronic version of this dissertation is available at
Link https://openaccess.leidenuniv.nl/handle/1887/9744

Cover design: Wei Liu, Yu Liu

Exploring Images with Deep Learning for Classification, Retrieval and Synthesis

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof.mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op woensdag 24 oktober 2018
klokke 11.15 uur

door

Yu Liu

geboren te Heilongjiang, China
in 1988

Promotiecommissie

Promotors:	Prof. dr. J.N. Kok	
	Dr. M.S. Lew	
Overige leden:	Prof. dr. A. Plaat	
	Prof. dr. T.H.W. Bäck	
	Prof. dr. W. Kraaij	
	Prof. dr. H. Trautmann	(University of Münster)
	Prof. dr. A. Hanjalic	(Delft University of Technology)
	Prof. dr. ir. B.P.F. Lelieveldt	
	Dr. ir. R. Poppe	(Utrecht University)

Yu Liu was financially supported through the China Scholarship Council (CSC) to participate in the PhD programme of Leiden University. Grant number 201406060010.

This work was carried out in the ASCI graduate school. ASCI dissertation series number: 387

The research in this thesis was performed at the LIACS Media Lab, Leiden University, The Netherlands, and we would like to thank the NVIDIA Corporation for the donation of GPU cards.

Contents

Chapter 1

Introduction

1.1 Motivation

In 2018, the number of mobile phone users will reach about 4.9 billion. Assuming an average of 5 photos taken per day using the built-in cameras would result in about 9 trillion photos annually. In addition, these photos are frequently uploaded, shared and retrieved in social networks and thus have become an important part of our daily lives. However, it is challenging to mine semantically meaningful visual information from such a huge amount of data. Thanks to the major advances of deep neural networks since 2012, they have been a powerful tool to help analyze visual content for a variety of tasks and have triggered a massive amount of research in content based multimedia analysis and computer vision. This thesis aims towards *developing new paradigms and architectures in deep learning* to address three common and important research themes: classification, retrieval and synthesis. As shown in Figure 1.1, we visibly depicts the three themes.

- **Classification** is the most fundamental task in the field of computer vision. It aims to correctly predict the class label for a given image, for example, we can use a classification model to classify the input horse image. In addition to image-level classification, we also study the tasks of pixel-level classification, including semantic segmentation and edge detection. (Chapters 2 and 3)
- **Retrieval** aims to efficiently search for similar samples from the database to the query. For instance, we develop a retrieval model to retrieve similar horse images. Besides, we also consider the cross-modal retrieval problem between images and texts, and do some work to bridge the modality gap between vision and language. (Chapters 4, 5, 6 and 7)
- **Synthesis** is able to generate new image samples that never existed in the image database. For example, by training a synthesis model, we can translate a horse image to a zebra image. In addition, we can synthesize diverse zebra images based on different branches of the network. In this thesis, we mainly focus on two synthesis applications: image-to-image translation and fashion style transfer. (Chapter 8)

In the next sections, we first introduce the background and developments related to the three themes in recent years. Then we present the thesis outline, our research questions and main contributions.

1.2 Background and Related Work

Deep learning [1, 2] has been one of the pillars of numerous artificial intelligence research fields, such as computer vision, machine learning and natural language

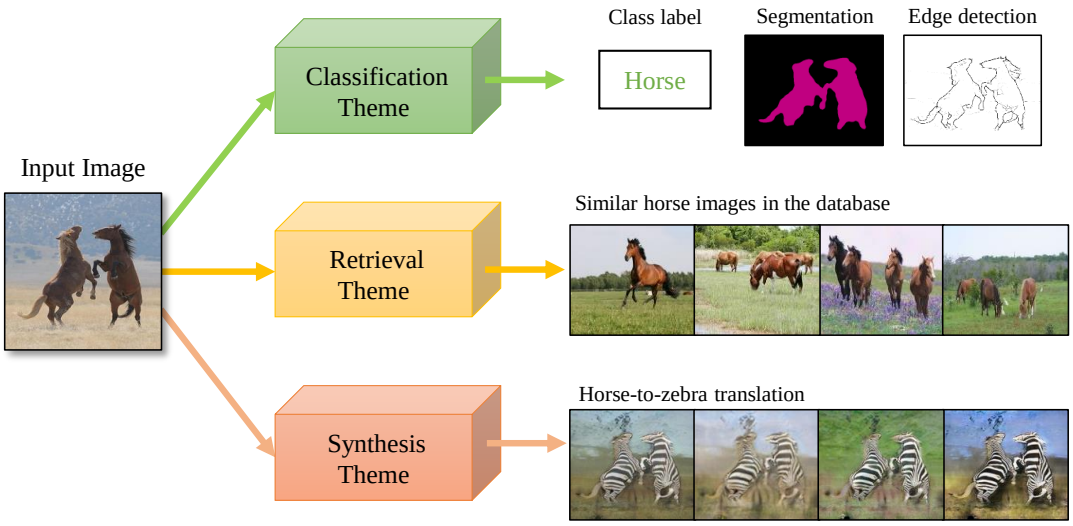


Figure 1.1: Conceptual illustration of the three research themes in this thesis, including classification, retrieval and synthesis.

processing. By distilling high-level semantic information with deep network architectures, deep learning approaches can help narrow the gap between visual representations and human-level vision. In recent years, deep learning has been extensively studied in the field of computer vision to help tackle many challenging tasks, such as image classification, image retrieval and image synthesis.

1.2.1 Classification

In recent decades, exploiting and developing convolutional neural networks (CNNs) [3] has been a leading and promising trend in computer vision community. CNNs can explore high-level visual concepts in images by employing deep architectures composed of multiple neural layers. In 2012, Krizhevsky *et al.* [4] proposed a new CNN model named AlexNet for generic image classification, which has been a milestone in the developments of CNNs. Its success in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competitions [5] motivates a huge amount of work leveraging CNNs to solve various vision tasks. According to the empirical observations in [6], CNNs based approaches can achieve new state-of-the-art performance for different recognition tasks by fine-tuning the ImageNet models on the target datasets. We summarize the related work on classification from the following four aspects.

Increasing the depth

A dominant line of research on CNNs is to increase the network depth to learn more discriminative representations (Figure 1.2(a)). For instance, the depth has increased

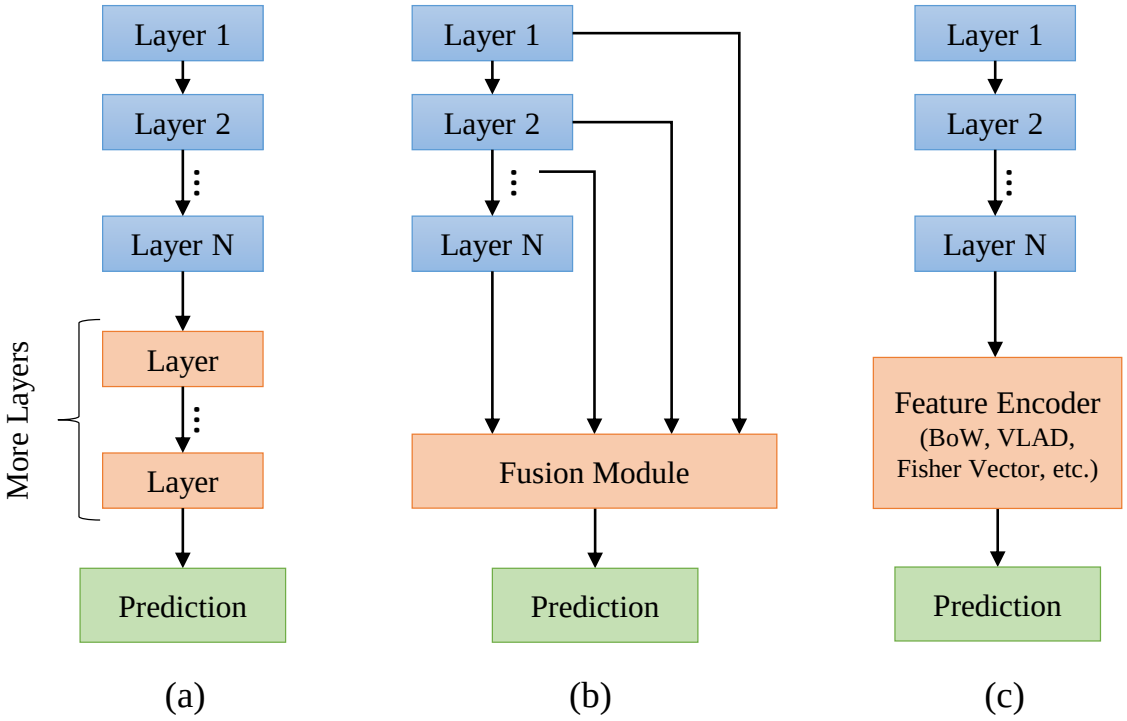


Figure 1.2: Illustration of three classification pipelines. (a) increasing the depth with more new layers. (b) fusing intermediate layers to produce an aggregation feature. (c) encoding deep features with sophisticated feature encoders.

from several layers (*e.g.* LeNet [3] and Alexnet [4]) to several tens of layers (*e.g.* VGGnet [7] and GoogLeNet [8]). However, training deeper networks becomes more difficult because of vanishing gradients and degradation. To overcome this challenge, Highway networks [9] and ResNet [10] proposed to add shortcut connections between neighboring layers, which can help alleviate the vanishing gradient issue and ease the training convergence. Their approaches have promoted the study on constructing deeper neural networks (*e.g.* hundreds of layers) and breaking the potential bottleneck that may limit the learning capabilities. Furthermore, extended studies [11, 12, 13, 14] based on ResNet provided additional insights by delving into the residual learning mechanism. Nevertheless, it is non-tractable to optimize much deeper neural networks due to the large amount of network parameters and the expensive cost of physical memory.

Fusing multiple layers

An alternative to creating shortcuts between adjacent layers is to integrate existing intermediate layers in a deep neural network to generate a fused feature (Figure 1.2(b)), rather than deepening the network with additional new layers. Commonly, the topmost activations in deep networks (*i.e.* fully-connected layers) can act as the most important features to describe the image content. However, it

is important to note that intermediate activations (*i.e.* convolutional layers) can also provide informative and complementary clues about images, including low-level boundaries, textures and spatial contexts. Therefore, researchers [15, 16] began to transfer their attention to intermediate layers, and explored their influence on the classification performance. In contrast to using pre-trained models, extensive research efforts [17, 18] turned to training deep fusion networks where multi-level intermediate layers are fused together by adding new side branches. It is worth noting that the fused information occurs not just from adjacent layers but from the earliest layers as well. In the literature, deep fused representations have been shown to generate better predictions due to integrating the strengths of different intermediate layers within deep neural network.

Encoding deep features

Although CNNs are able to express more powerful visual features, they have weak robustness to severe geometrical deformations and spatial contexts. Fortunately, sophisticated encoding techniques including BoW [19], VLAD [20] and Fisher Vector [21] have been adopted to address these issues. Motivated by the strengths of encoding techniques, it is natural to encode deep features to further improve their discriminatory power (Figure 1.2(c)). To obtain local features from CNN models, most approaches [22, 23, 24, 25] have examined local patches or region proposals in one image. The local CNN features are used to construct a visual codebook, based on which an encoder technique can be used to aggregate them to a deep image representation. For example, Gong *et al.* [22] employed image patches at multiple scales, and then aggregated local patch responses at the finer scales with the VLAD method. Yoo *et al.* [25] utilized multi-scale dense local CNN features to compute the Fisher Vector kernels.

Pixel-level classification

In addition to image-level classification, CNNs also show strong generalization power for diverse tasks of pixel-level classification, such as semantic segmentation [26, 27, 28], edge detection [29, 30, 31], depth estimation [32, 33, 34] and saliency detection [35, 36, 37]. In particular, fully convolutional networks (FCNs) [26] have become a fundamental architecture to perform pixel-level predictions. Specifically, FCNs are recast from the pre-trained CNN counterparts, by replacing the fully-connected layers with extra convolutional layers while retaining the parameters. In this way, the size of the input images can be arbitrary and the output can be viewed as two-dimensional feature maps. In addition, it is beneficial to extract richer region features from FCNs, compared to a global representation from CNNs.

1.2.2 Retrieval

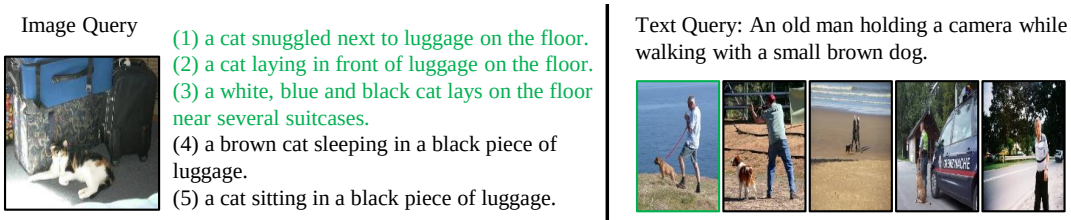
One of the primary aims of image retrieval is to search for similar images (usually based on pictorial content) to the query from the database. It has become important to numerous practical scenarios (*e.g.* Google image search, face recognition, *etc.*) and therefore has triggered a massive amount of research activities in both multimedia and computer vision fields [19, 38, 39]. Bag-of-Words (BoW) is one of the most widely-used models in image retrieval systems, where local features, such as SIFT [40] and color clues [41], are quantized to visual words based on a pre-trained codebook. Then, similar to document retrieval [19, 39], an inverted index structure is built with the visual words towards making the retrieval system scalable and efficient. However, image retrieval remains challenging in bridging low-level image representations and high-level semantic concepts.

Image retrieval

To alleviate the above challenge, recent works in the literature have paid attention on utilizing deep visual features for image retrieval [42, 43, 44]. The work of Wan *et al.* [42] suggested that a deep CNN model pre-trained on a large dataset can be transferred to new content-based image retrieval (CBIR) tasks and fine-tuning the model with a similarity metric could further boost the retrieval performance. Babenko *et al.* [45] focused on holistic descriptors where the whole image was mapped to a single deep feature vector. They further designed a simple global image descriptor based on sum-pooled convolutional features for image retrieval. Zheng *et al.* [46] proposed a deep embedding method using deep features as global and regional signatures instead of a Hamming embedding[47]. It is an incorporation of the SIFT descriptor and CNN features and could achieve promising improvements. Moreover, Zheng *et al.* [48] presented a comprehensive review on SIFT and CNN-based methods and discussed the benefits of integrating SIFT and CNN features.

Cross-modal retrieval

Nowadays, multimedia data in various media types (*e.g.* image, video, text, and audio) is growing exponentially due to the increasing popularity of the Internet and social networks. This trend motivates a massive amount of research activities in multi-modal understanding and reasoning. For example, we can recognize a picture of a panda after hearing the description "black and white bears" without ever having seen one. This demonstrates the cross-modal interaction between vision and language. These heterogeneous data offers us the opportunity to understand the world better, while giving rise to the challenges of bridging different modalities. Specifically, the matching problem between images and texts [49, 50, 51, 52, 53, 54] is one of the most important tasks in multi-modal research. In practice, image-text matching



(a) Image-to-text retrieval

(b) Text-to-image retrieval

Figure 1.3: Example of cross-modal retrieval. (a) Given an image query, related text samples are retrieved to describe the image. (b) For a text query, it can search for several image samples from the database. The matched samples are highlighted with green color.

approaches are usually developed for cross-modal retrieval (Figure 1.3). This task remains challenging due to the heterogenous representations and the cross-modal gap between vision and language, which is also a core issue for other multi-modal applications such as image captioning [55, 56] and visual question answering [57, 58], zero-shot recognition [59, 60].

With the increasing progress of deep learning, research efforts have been made to incorporate Canonical Correlation Analysis (CCA) [61] into deep neural networks [49, 50, 51, 62, 63]. However, existing deep CCA models rely on expensive decorrelation computations, which limit their generalization abilities at large-scale data. Alternatively, a number of recent approaches [52, 55, 64, 65, 66] address the task by designing two-branch networks to embed visual and textual features into a common latent space, and then learn latent embeddings by optimizing a ranking loss to discriminate matched and unmatched image-text pairs. For instance, Wang *et al.* [53] built a simple and efficient matching network to preserve the structure relations between images and texts in the latent space. To associate image regions with words, the attention mechanism was integrated into visual-textual embedding models [67, 68]. In addition to the pairwise ranking loss, recent approaches [69, 70] leveraged extra loss functions (*e.g.* instance loss and classification loss) to enhance the discrimination of the learned embedding features.

Another line of research [71, 72, 73, 74, 75] focused on learning dual embeddings between two modalities, *e.g.* projecting visual features into the textual feature space and vice versa. For instance, Feng *et al.* [71] proposed a correspondence cross-modal autoencoder model. 2WayNet [76] built the projections between two modalities and regularized them with Euclidean loss. Recently, Gu *et al.* [77] utilized two generative models to synthesize grounded visual and textual representations. Essentially, these dual embedding models are motivated by autoencoders.

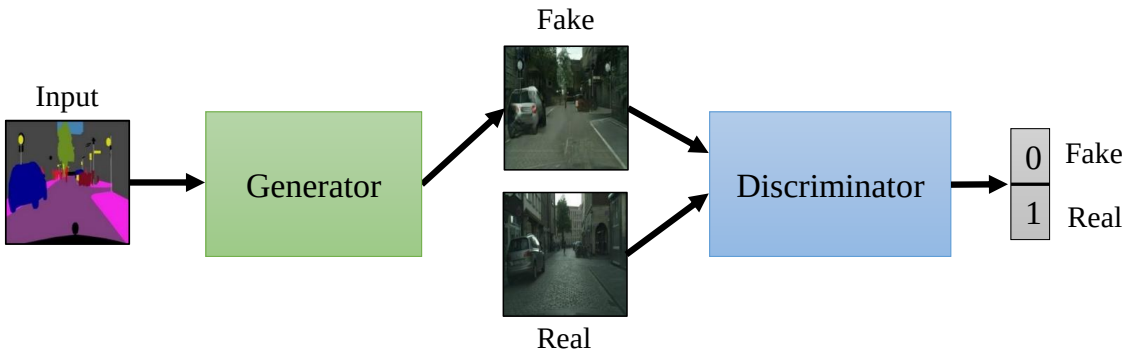


Figure 1.4: Illustration of the GAN framework. In this example, given a labelled map, the generator can synthesize a fake photo image similar to the real one, but The discriminator learns to correctly classify real and fake images.

1.2.3 Synthesis

Together with the increasing progress of deep neural networks, numerous approaches based on supervised learning have been developed to address diverse image translation tasks, such as contour detection [31], semantic segmentation [26] and face conversion [78]. However, these supervised models highly depend on a large amount of fully labelled image pairs which are time consuming to create manually and sometimes biased when collecting annotated data. In addition, some ground-truth data are not available in some cases, for example, the painting stylization transfer between Monet to Van Gogh. Driven by these limitations, researchers have turned to examine unsupervised learning approaches to break the bottleneck of limited data.

Generative adversarial networks

Generative models have attracted increasing attention with the emergence of generative adversarial networks (GANs)[79]. Informally, the GAN framework can be viewed as a game between two players: the generator and the discriminator (Figure 1.4). To be specific, the generator aims to synthesize fake images and tries to trick the discriminator into thinking that the synthesized images are real. In contrast, the discriminator needs to distinguish the real images from the fake images. By continuing this game iteratively, both players learn to become better until the generator can generate realistic-looking images and the discriminator can not tell real and fake samples. Typically, a simple analogy is that: an art forger (the generator) attempts to forge artistic paintings, but an art investigator (the discriminator) is able to detect imitations. In recent years, GANs have been widely adopted for addressing a wide range of image synthesis applications, such as style transfer [80, 81], texture synthesis [82, 83] and text-to-image synthesis [84, 85]. To improve the quality and diversity of generative models, conditional GANs (cGAN) have been designed to

guide image generation conditioned on class labels [86], attributes [87], images [88] and texts [84].

Image-to-image translation

GANs have shown great success on the task of general-purpose image-to-image translation [88], which learns to model mapping functions between different image domains. Many recent approaches [89, 90, 91, 92] were focused on using unpaired images to tackle the problem of unsupervised image translation. In addition to the adversarial constraint, they further exploited extra constraints to enhance relations of two different domains. On the one hand, some of them [89, 90, 93] fed image samples into a unified encoder to discover their common representations. Then another generator was used to translate common representations to samples in the target domain. On the other hand, some work [92, 94] attempted to relate two different domains by using additional self-constraints within one domain. Representatively, CycleGAN [94] proposed a cycle-consistency constraint that can reconstruct the input image itself.

Fashion style transfer

Online shopping has driven a range of fashion oriented applications recently, for example, fashion clothing retrieval [95], fashion recommendation [96], fashion parsing [97] and fashion style transfer [81]. Specifically, fashion clothing swapping, which is a common application belonging to fashion style transfer, aims to visualize what the person would look like with the target clothes. This application allows consumers to see what they would look like by wearing different clothes, without the effort of dressing them physically. In the past, this problem has been studied in the fields of multimedia and computer graphics [98, 99, 100, 101]. For example, the work in [102] used an image-based visual hull rendering approach to transfer the appearance of a target garment to another person image. The ClothCap approach [103] captured the 3D deformations of the clothing and estimated the minimally clothed body shape and pose under the clothing. These non-parametric solutions [100, 104] involve using extra information to model the deformations, such as from motion capture, 3D measurements and depth sensors. During the test stage, they still require online image warping or registration algorithms which are time-consuming for real-time applications. Recent research turned to address this problem using deep generative approaches (*e.g.* GANs), without requiring complicated 2D image warping and 3D graphic algorithms. For example, FashionGAN [85] employed a textual description as condition to perform the clothing swapping. The methods in [105, 106] took a stand-alone and flat clothing image to re-dress the person in the reference image.

1.3 Thesis Outline and Research Questions

In Section 1.2, we have introduced recent advances on the three research themes. Although deep learning is leading state-of-the-art performance for numerous tasks, we should notice its limitations and challenges, such as theoretical interpretability, model complexity, training with limited data, *etc.* There is still considerable space for promoting the developments of deep learning. In the next research chapters (Chapters 2-8), we propose new approaches to address the research questions (**RQ**) and challenges in terms of the three research themes. In Chapter 9, we discuss our main findings, limitations & possible solutions and future research directions.

- **Chapter 2** aims to address the first research question **RQ 1: How can we develop a simple and efficient deep fusion network upon a plain CNN?** As discussed in Section 1.2.1, some works [17, 18, 26, 31] attempt to create new side branches upon a plain CNN and integrate multi-level intermediate layers to generate a fused representation. However, they still have two main limitations. First, some of them spend a large number of new parameters creating the side branches. For example, DAG-CNNs [18] add several fully-connected layers on top of intermediate convolutional layers, which will largely increase the total number of parameters. Second, the fusion modules for integrating different side branches are inferior. DAG-CNNs [18] and FCN-8s [26] use a simple sum pooling to fuse the side branches, which fails to consider the weights of different side branches. Although HED [31] employs a 1×1 convolution to learn the fused weights, they are shared over spatial dimensions, failing to discover the spatial properties. In this chapter, we propose a novel convolutional fusion network (CFN) built on top of plain CNNs, which can aggregate intermediate layers with adaptive weights and generate a discriminatively fused representation. This chapter is based on the published papers [107, 108]:

  > **Liu, Y.**, Guo, Y., and Lew, M.S., "On the Exploration of Convolutional Fusion Networks for Visual Recognition." Proceedings of the 23rd International Conference on MultiMedia Modeling (MMM), 2017. **(Best Paper Award)**
  >
  > **Liu, Y.**, Guo, Y., Georgiou, T., and Lew, M.S., "Fusion that matters: convolutional fusion networks for visual recognition." Multimedia Tools and Applications, 2018.

- The work in **Chapter 3** aims to tackle the second question **RQ 2: How can we explore diverse supervision that can adapt to different intermediate layers in deep neural networks for robust edge detection?** Edge detection that aims to distinguish important edges from image pixels,

can generally act as a fundamental task for other high-level vision applications, like object detection and segmentation. Recently, the developments in the design of edge features have moved from carefully-engineered descriptors [109, 110, 111, 112] to hierarchical deep features [29, 30, 31, 113]. Nevertheless, we should still realize one difficult issue in edge detection that is caused by *false positives*: many non-edge pixels are incorrectly predicted as edges when comparing with the human annotated ground-truth. To correct the false positives earlier, HED [31] imposes the ground-truth supervision on the intermediate layers while training the deep model. However, using only a general supervision (*i.e.* the ground-truth annotation) for all the layers is inconsistent with the diverse representations of hierarchical layers. In addition, the general supervision can not be well-suited to all intermediate layers. In contrast to using the general supervision, we propose and develop relaxed deep supervision (RDS) within convolutional neural networks for robust edge detection. This chapter is based on the published paper [114]:

> **Liu, Y.** and Lew, M.S., "Learning Relaxed Deep Supervision for Better Edge Detection." Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

- In **Chapter 4**, we move our attention to the retrieval theme and tackle the third question **RQ 3: How can we incorporate deep visual representations into the inverted index structure for accurate and efficient image retrieval?** A robust image retrieval system should be typically optimized regarding two factors: accuracy and efficiency. To increase the retrieval accuracy, some works [22, 42, 45] begin to utilize deep visual features to discover the similarities among images. However, they are inefficient due to relying on the nearest neighbouring search. To maintain the efficiency, traditional methods [19, 39], take advantage of the inverted index structure that is able to reduce computational time and memory cost for scalable image search. Regarding both the accuracy and efficiency, we exploit a DeepIndex framework for accurate and efficient image retrieval, by incorporating deep visual features into the inverted index scheme. This chapter is based on the published paper [115]:

  > **Liu, Y.**, Guo, Y., Wu, S., and Lew, M.S., "DeepIndex for Accurate and Efficient Image Retrieval." Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (ICMR), 2015.

- In addition to image retrieval, in **Chapter 5** we further address the problem of cross-modal retrieval with **RQ 4: How can we build a deep matching network to unify images and texts into a more discriminative space without increasing the number of network parameters?** The image-text matching problem remains challenging due to the heterogenous representations and the cross-modal gap between two modalities. In recent

years, a variety of multi-modal deep neural networks have been proposed to model the matching task [52, 53, 76]. However, the multi-modal matching performance is still far from competitive with the intra-modal tasks, for example, image retrieval. In this chapter, we introduce an efficient approach to couple visual and textual features based on a new recurrent residual fusion (RRF) building block. This chapter is based on the published paper [64]:

> **Liu, Y.**, Guo, Y., Bakker, E.M., and Lew, M.S., "Learning a Recurrent Residual Fusion Network for Multimodal Matching." Proceedings of the 16th IEEE International Conference on Computer Vision (ICCV), 2017.

- In terms of cross-modal retrieval, in **Chapter 6** we further pose the fifth research question **RQ 5: How can we preserve both inter-modal correlations and intra-modal consistency for learning robust visual and textual embeddings?** Currently, there are two main paradigms to perform visual-textual embeddings. The first one is to learn a common space where related images and texts can be unified into similar latent embeddings [52, 53, 76]. Second, it exploits dual embeddings by reconstructing an input feature in the source space to be the one in the target space [72, 76, 77]. Both the latent and dual embeddings can capture inter-modal semantic correlations between visual and textual data. In addition, they can be combined together to integrate individual advantages. However, they fail to preserve the intra-modal semantic consistency, *i.e.* image-to-image and text-to-text. Importantly, a robust embedding method should be able to reconstruct representations of both the source and target modalities. To achieve this, we propose cycle-consistent embeddings in a deep neural network for matching visual and textual representations. This chapter is based on the submitted journal paper [116]:

  > **Liu, Y.**, Guo, Y., Liu, L., Bakker, E.M., and Lew, M.S., "CycleMatch: A Cycle-consistent Embedding Network for Image-Text Matching." Submitted to IEEE Transactions on Multimedia (In Revision).

- In **Chapter 7**, we aim to integrate both matching and classification by answering the sixth question **RQ 6: How can we design a unified network for joint multi-modal matching and classification?** We note that, learning visual-textual embeddings is influenced by the notable variance in images or texts. For example, in the MSCOCO dataset [117], each image is described with five sentences from human labelers. Although the sentences can consistently mention some primary objects in the image, they have some biased differences that may make it difficult to perform a robust matching. However, object labels can generally provide more consistent and less biased information than sentences. Classification with the object labels is beneficial

to correct the biased sentences and improve the image-text matching. Additionally, the matching component can help the classification component to generate a discriminative multi-modal representation. Unlike many current approaches which only focus on either multi-modal matching or classification, we propose a unified network to jointly learn Multi-modal Matching and Classification (MMC-Net) between images and texts. This chapter is based on the submitted paper [118]:

> **Liu, Y.**, Liu, L., Guo, Y., and Lew, M.S., "Learning Visual and Textual Representations for Multimodal Matching and Classification." Pattern Recognition, vol 84: 51-67, 2018.

- **Chapter 8** presents two applications about image synthesis: image-to-image translation and fashion style transfer.

  For image-to-image translation, we pose the seventh research question **RQ 7: What factors will affect the performance of generative models on the translation tasks?** Image-to-image translation between different domains is a common image synthesis task, with the aim of arbitrarily manipulating the source image content given a target one. To tackle the challenging case of unpaired image-to-image translation, CycleGAN [94] presents a cycle-consistency loss by reconstructing the generated image back to the source domain. In conjunction with the original adversarial loss, the cycle-consistency loss is beneficial to constrain the unsupervised domain mappings. CycleGAN has become a fundamental approach for general-purpose image-to-image translation, while few work investigate the important factors within it. To address the problem, we present an extensive and empirical study on cycle-consistent generative networks. This work is based on the published paper [119]:

  > **Liu, Y.**, Guo, Y., Chen, W., and Lew, M.S., "An Extensive Study of Cycle-Consistent Generative Networks for Image-to-Image Translation." Proceedings of the 24th International Conference on Pattern Recognition (ICPR), 2018.

  In terms of fashion style transfer, we need to tackle the last research question **RQ 8: How can we exploit a generative model to directly transfer the fashion style between two person images?** Currently, fashion style transfer based on image synthesis has become a popular application for online shopping. Specifically, fashion clothing swapping aims to visualize what the person would look like with the target clothes. It can be viewed as a specific task belonging to fashion style transfer. Recently, FashionGAN [85] specifies a textual description and uses it to re-dress the person in the reference image. other works in CAGAN [105] and VITON [106] employ a stand-alone and flat clothing image to condition the image synthesis, which may provide richer visual content than the textual description. However, the stand-alone and flat

clothing images are not always available to users or consumers. Therefore, we pose a more practical task, that is, person-to-person clothing swapping, where the input condition is also a person image like the reference image. In this case, the goal becomes transferring the wearing clothes between two person images. It is more challenging as the desired clothes worn on the condition person image have varying deformations due to different human poses. To tackle this challenge, we propose a novel multi-stage generative network (SwapGAN) that integrates three generators to perform a multi-stage synthesis process. This work is based on the submitted journal paper [120]:

> **Liu, Y.**, Chen, W., Liu, L., and Lew, M.S., "SwapGAN: A Multi-stage Generative Approach for Person-to-Person Fashion Style Transfer." Submitted to IEEE Transactions on Multimedia (In Review).

- Finally, **Chapter 9** summaries the main findings from the research of this thesis. Also, we discuss the limitations and potential solutions, and point out directions for future research.

Additionally, this thesis draws on insights and experiences from the related work in other publications during my PhD studies:

- **Liu, Y.** and Lew, M.S., "Improving the Discrimination between Foreground and Background for Semantic Segmentation." Proceedings of the 24th IEEE International Conference on Image Processing (ICIP), 2017.
- **Liu, Y.**, Guo, Y., and Lew, M.S., "What Convnets Make for Image Captioning." Proceedings of the 23rd International Conference on MultiMedia Modeling (MMM), 2017.
- Guo, Y., **Liu, Y.**, Lao, S., Bakker, E.M., Bai, L., and Lew, M.S., "Bag of Surrogate Parts Feature for Visual Recognition." IEEE Transactions on Multimedia, vol 20: 1525-1536, 2018.
- Shan, H., **Liu, Y.**, and Stefanov, T., "A Simple Convolutional Neural Network for Accurate P300 Detection and Character Spelling in Brain Computer Interface." International Joint Conference on Artificial Intelligence (IJCAI), 2018.
- Guo, Y., **Liu, Y.**, de Boer, M.H.T., Liu, L., and Lew, M.S., "A Dual Prediction Network for Image Captioning." Proceedings of the 19-th IEEE International Conference on Multimedia and Expo (ICME), 2018.
- Georgiou, T., Schmitt, S., Olhofer, M., **Liu, Y.**, Back, T., and Lew, M.S., "Learning Fluid Flows." Proceedings of International Joint Conference on Neural Networks (IJCNN), 2018.

- Guo, Y., **Liu, Y.**, Bakker, E.M., Guo, Y., and Lew, M.S., "CNN-RNN: a large-scale hierarchical image classification framework." Multimedia Tools and Applications, vol 77: 10251-10271, 2018.
- Guo, Y., **Liu, Y.**, Georgiou, T., and Lew, M.S., "A review of semantic segmentation using deep neural networks." International Journal of Multimedia Information Retrieval, vol 7: 87-93, 2018.
- Jia, Q., Fan, X., **Liu, Y.**, Luo, Z., and Guo, H., "Hierarchical projective invariant contexts for shape recognition." Pattern Recognition, vol 52: 358-374, 2016.
- Guo, Y., **Liu, Y.**, Oerlemans, A., Lao, S., Wu, S., and Lew, M.S., "Deep learning for visual understanding: A review." Neurocomputing, vol 187: 27-48, 2016.
- Guo, Y., Lao, S., **Liu, Y.**, Bai, L., Liu, S., and Lew, M.S., "Convolutional Neural Networks Features: Principal Pyramidal Convolution." Proceedings of the 16th Pacific-Rim Conference on Multimedia (PCM), 2015.

1.4 Main Contributions

The research of this thesis contributes at three levels: models and algorithms, practical scenarios and empirical analysis.

1.4.1 Models and algorithms

From Chapter 2 to Chapter 8, we develop new approaches based on deep learning to address the research questions regarding the three themes. The key contributions in these approaches are listed below.

An efficient deep fusion model for image classification. We propose a novel deep fusion network, the convolutional fusion network (CFN), where we can efficiently integrate multiple intermediate layers in CNNs with adaptive weights. In addition, our CFN is adaptive to not only image-level classification, but also pixel-level classification.

A diverse deep supervision algorithm for edge detection. In contrast to prior work using a general supervision, we develop relaxed deep supervision (RDS) with additional relaxed labels. Consequently, more discriminative layers can process more false positives in edge detection. RDS can incorporate the diversities into the supervisory signals to improve the performance of edge detection.

An accurate and efficient model for image retrieval. We propose a novel image retrieval approach (DeepIndex) which can integrate deep visual representations

with the inverted index scheme. Our approach takes advantage of the discriminatory capabilities of deep features and the efficient search of the inverted index.

An efficient image-text matching model for cross-modal retrieval. We develop a novel recurrent residual fusion network (RRF-Net) to couple visual and textual features. Since RRF-Net connects the residual learning with the recurrent mechanism, it can recursively improve visual-textual embeddings while sharing the network parameters. In addition, we develop a fusion module to efficiently integrate intermediate recurrent outputs.

A cycle-consistent embedding algorithm for cross-modal retrieval. To preserve both inter-modal correlations and intra-modal consistency, we propose cycle-consistent embeddings by cascading dual mappings and reconstructed mappings in a cyclic fashion. Our embedding method can effectively promote the performance of cross-modal retrieval, compared to traditional embedding methods.

A unified model for multi-modal matching and classification. We propose a unified network (MMC-Net) to jointly model multi-modal matching and classification. Our approach can suggest that combining the matching and classification components can help boost each other. In addition, we employ a multi-stage training algorithm to make the two components compatible.

Two extended deep generative models for image-to-image translation. As few work investigate the important factors within cycle-consistent generative networks (CycleGAN), we present two extended models, namely Long CycleGAN and Nest CycleGAN, and then conduct an extensive and empirical study on the models. Our work examines the benefits of using more generators and cycles on the generation quality.

A multi-stage generative model for fashion style transfer. In contrast to traditional non-parametric approaches, we propose a novel multi-stage generative network (SwapGAN) to transfer the clothing style in one person image to another one. The SwapGAN model can be trained end-to-end with three different generators and one discriminator.

1.4.2 Practical scenarios

In addition to improve the performance of diverse tasks, our research also aims towards adapting to practical scenarios in real world.

Accurate and efficient image retrieval. Image retrieval is a widely used application in ours lives. In some cases, the retrieval speed is the same important as the accuracy. Our DeepIndex framework (in Chapter 4) can take into account both accuracy and efficiency in image retrieval. Specifically, deep visual features can help

improve the retrieval accuracy, and the inverted index scheme is more efficient than the nearest neighboring search.

Joint multi-modal matching and classification. In practice, we may need to search for similar samples as the query sample, and know its class label as well. Motivated by this need, we develop MMC-Net (in Chapter 7) to unify both multi-modal matching and classification in one model, unlike prior approaches which focus on either matching or classification. Our work shows a simple and efficient way to fulfill the two tasks simultaneously.

Person-to-person fashion style transfer. Prior work performs the clothes-to-person style transfer, however, in practice stand-alone and flat clothing images are not always available. Instead, our work (in Chapter 8) aims to address a more practical case, in which the goal is to swap the clothes between two person images directly. This task becomes more challenging due to varying human poses. Our SwapGAN is proposed to solve this practical problem by cascading three generators in a multi-stage manner.

1.4.3 Empirical analysis

Furthermore, this thesis provides numerous experiments and in-depth analysis, which can help motivate further research on the three research themes.

Fusion that matters deep neural networks. Instead of deepening neural networks with more layers, our CFN model (in Chapter 2) is an efficient alternative to improving the capabilities of CNNs while maintaining the model complexity. In the experiments, we provide a detailed analysis to verify the effectiveness of CFN. In addition, we compare CFN with other deep models and offer an extensive discussion about them. Moreover, our CFN can be adaptive to different computer vision tasks like image classification, semantic segmentation, edge detection, *etc.* In a nutshell, our work suggests that deep fusion networks can efficiently promote the feature representational abilities of plain CNNs.

Cycle-consistency that matters visual-textual embeddings. Our proposed cycle-consistent embedding approach (in Chapter 6) is an integration of three embeddings, namely dual embedding, reconstructed embedding and latent embedding. Our approach can model both inter-modal correlations and intra-modal consistency while matching visual and textual representations. In the experiments, we conduct a comparable analysis between our approach and existing embedding approaches. The superiority of our approach over others can increase the awareness of using cycle-consistency for multi-modal research tasks.

Two factors that matter cycle-consistent adversarial networks. To provide deep insights into CycleGAN, we developed two extended models (in Chapter 8) for examining two factors: the number of generators and the number of cycles.

The qualitative and quantitative results for a range of translation tasks verify the benefits of using more generators and cycles, compared to the vanilla CycleGAN. The results in our study can help ease future research based on cycle-consistent generative networks.

Chapter 2

Convolutional Fusion Networks for Image Classification

In the previous chapter we have introduced the background and research questions for this thesis. Starting with this chapter, we begin to answer the research questions with our proposed approaches. In this chapter, we address how we can develop a simple and efficient deep fusion network upon a plain CNN (RQ 1).

Despite recent advances in deep fusion networks, they still have limitations due to expensive parameters and weak fusion modules. To address this issue, we propose a novel convolutional fusion network (CFN) to integrate multi-level deep features and fuse a richer visual representation. Specifically, CFN uses 1×1 convolutional layers and global average pooling to generate side branches with adding only a few parameters, and employs a locally-connected fusion module, which can learn adaptive weights for different side branches and form a better fused feature. Moreover, we propose fully convolutional fusion networks (FCFNs) that are an extension of CFNs for pixel-level classification, including semantic segmentation and edge detection. Our experiments demonstrate that our approach can achieve consistent performance improvements for diverse tasks.

Keywords
Image classification, Convolutional neural networks, Fully convolutional networks, Adaptive fusion

2.1 Introduction

A significant progress on convolutional neural networks is increasing their depth to learn more powerful visual representations. In particular, the depth has increased from several layers (*e.g.* LeNet [3] and Alexnet [4]) to several tens of layers (*e.g.* VGGnet [7] and GoogLeNet [8]). Nevertheless, training a very deep network is extremely difficult because of vanishing gradients and degradation. To overcome this challenge, recent work in both Highway networks [9] and ResNet [10] proposes to add shortcut connections between neighboring convolutional layers, which are able to alleviate the vanishing gradient issue and ease the training stage. Nevertheless, it is non-tractable to optimize very deep neural networks due to their large amount of parameters and the expensive cost of physical memory.

An alternative is to explore integration with the existing intermediate layers in a deep neural network, rather than deepening the network with new layers. Commonly, the topmost activations in deep networks (*i.e.* fully-connected layers) serve as discriminative visual representations to describe the image content. However, it is important to note that intermediate activations (*i.e.* convolutional layers) can also provide informative and complementary clues about images, including low-level textures, boundaries, and local parts. Therefore, researchers [15, 16, 25] have given greater attention to intermediate layers, and evaluated their contributions regarding image recognition performance. In addition, a large number of approaches [16, 23, 24, 121] have leveraged sophisticated encoding schemes (*e.g.* BoW, VLAD and Fisher Vector) to further encode intermediate feature activations. These approaches extract deep features from off-the-shelf CNNs without training new networks.

Moreover, extensive research efforts [17, 18, 26, 31] have turned to explicitly training deep fusion networks where multi-level intermediate layers are fused together by adding new side branches. As a result, the deep fused representation allows us to integrate the strengths of individual layers and generate superior prediction. Although these deep fusion networks have achieved promising performance, they may spend a large number of additional parameters required for generating the side branches [18]. In addition, their fusion modules (*e.g.* sum pooling) do not fully consider the importance of different side branches. Motivated by this problem, this chapter focuses on the research question **RQ 1: How can we develop a simple and efficient deep fusion network upon a plain CNN?**

To address this question, we propose a convolutional fusion networks (CFN), which is a new fusion architecture to integrate intermediate layers with adaptive weights. To be specific, CFN mainly consists of three key components: (1) *Efficient side outputs*: we use efficient 1×1 convolution and global average pooling [122] to generate side branches from intermediate layers and as a result it has a small number of additional parameters. (2) *Early fusion and late prediction*: it can not only provide a richer representation, but also reduce the number of parameters, compared to the

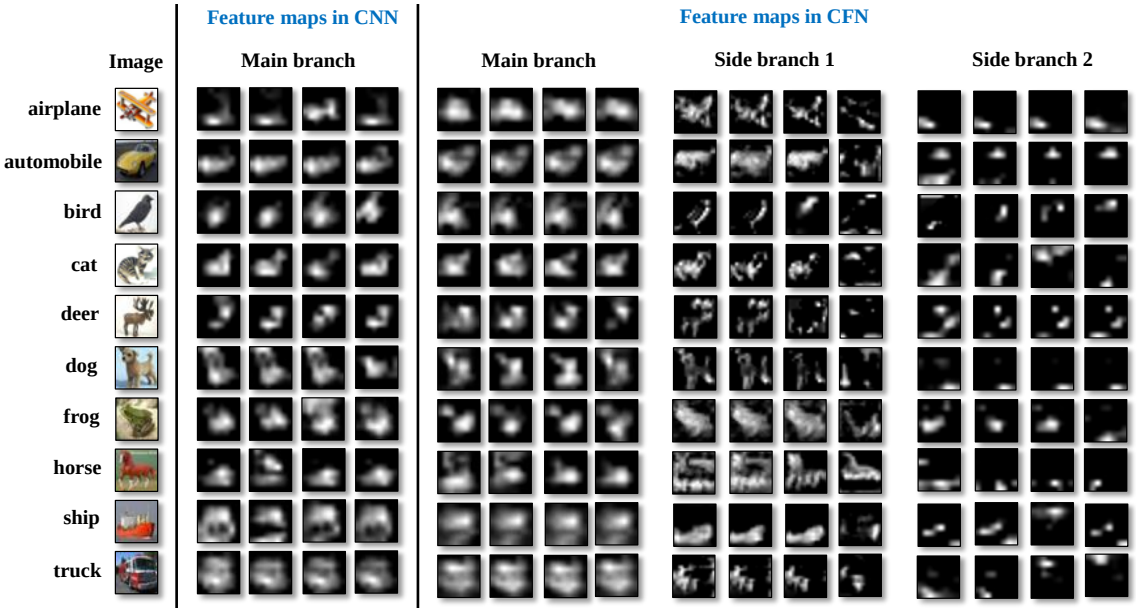


Figure 2.1: Illustration of features activations of the last convolutional layer in (a) CNN and (b) CFN. The CIFAR-10 images are used here. Compared with CNN, CFN can learn complementary clues in the side branches to the full depth main branch. For example, the side branch 1 mainly learns the boundaries or shapes around objects, and the side branch 2 focuses on some semantic "parts" that fire strong near the objects.

"early prediction and late fusion" strategy [18]. (3) *Locally-connected fusion*: we propose to adapt a locally-connected layer to act as a fusion module. It allows us to learn adaptive weights for different side outputs and generate a better fused representation. Figure 2.1 visually compares the feature activations learned in CNN and CFN, respectively. It can be seen that aggregating multi-level intermediate layers is essential to integrate their individual information.

The contributions of this chapter are as follows:

- We propose a new fusion architecture (CFN) which can provide promising insights towards how to efficiently exploit and fuse multi-level features in deep neural networks. In particular, to the best of our knowledge, this is the first attempt to use a locally-connected layer as a fusion module.

- We introduce CFN models to address the image-level classification task. The results on the CIFAR and ImageNet datasets demonstrate that CFN can achieve promising improvements over the plain CNN. In addition, we transfer the trained CFN model to three new tasks, including scene recognition, fine-grained recognition and image retrieval. By using the transferred model, we can achieve consistent performance improvements on these tasks.

- We further develop fully convolutional fusion networks (FCFN), which are able to perform pixel-level classification tasks including semantic segmentation and

edge detection. As a result, FCFN, as a fully convolutional extension, reveals the strong generalization capabilities of CFN for diverse tasks.

The rest of this chapter is organized as follows. Section 2.2 introduces the details of constructing the proposed CFN for image-level classification problem. In addition, we compare and highlight the differences of CFN from other deep models. The FCFN counterpart for pixel-level classification is described in Section 2.3. Section 2.4 presents experimental results that demonstrate the performance of CFN and FCFN on various visual recognition tasks. Finally, Section 2.5 concludes this work and point out two future directions.

2.2 Convolutional Fusion Networks

In this section, we introduce the details of building CFN on top of a plain CNN model, and formulate its training procedure. In addition, we compare its differences from other deep models.

2.2.1 Network architecture

First, we show a general architecture of a plain CNN model. As illustrated in Figure 2.2, it mainly comprises of successive convolutional layers and pooling layers. In addition, a 1×1 convolutional layer followed by global average pooling is used because of its high efficiency [8, 10, 122]. Based on this plain CNN, we can develop the proposed CFN by adding new side branches from intermediate layers and aggregating them in a locally-connected fusion module. Figure 2.3 illustrates the architecture of the proposed CFN. Our CFN is built on top of a plain CNN. To be specific, CFN mainly consists of the following three key components.

(1) Efficient side outputs

Prior work often added new fully-connected (FC) layers in the side branch [18], but this strategy may severely increase the number of parameters. Instead, CFN is able to efficiently create the side branches from the intermediate layers by adding only a few parameters. First, the side branches are built from the pooling layers (Figure 2.3). Each side branch has a 1×1 convolutional and global average pooling as well. All 1×1 convolutional layers must have the same number of channels so that they can be integrated together. Then, global average pooling (GAP) is performed over the 1×1 convolutional maps so as to obtain a one-dimensional feature vector, called the GAP feature. As a result, the side branches have the similar top layers (1×1 Conv and GAP) to the full-depth main branch. One difference is that the 1×1 Conv in the main branch follows a convolutional layer but not a pooling layer. For

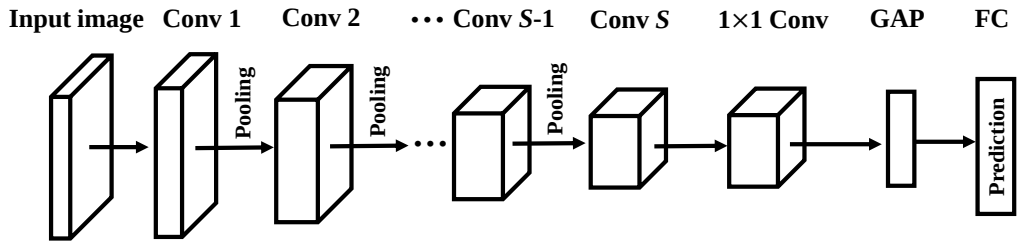


Figure 2.2: The general pipeline of a plain CNN model. Note that one 1×1 convolutional layer and global average pooling are used on the top layers.

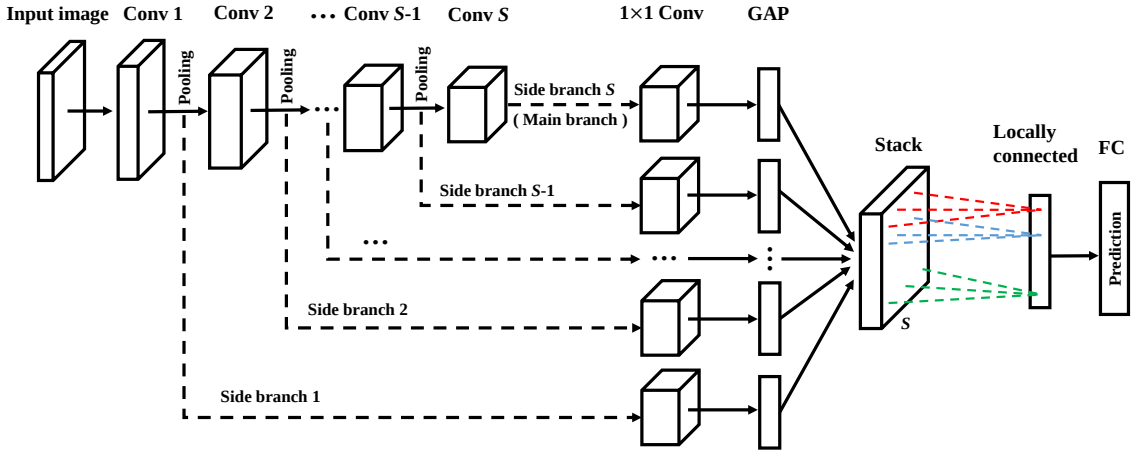


Figure 2.3: The general pipeline of the proposed CFN. First, the side branches start from the pooling layers and consist of a 1×1 convolution layer and global average pooling. Then, all side outputs are stacked together. A locally-connected layer is used to learn adaptive weights for the side outputs (drawn in different color). Finally, the fused feature is fed to the FC layer to make a better prediction.

concise formulation, we consider the full-depth main branch as another side branch.

Assume that there are S side branches in total and the last side branch (*i.e.* S-th) indicates the main branch. We notate $h_{i,j}^{(s)}$ as the input of the 1×1 convolution in the s-th side branch, where $s = 1, 2, \ldots, S$ and (i, j) is the spatial location in the feature maps. As the 1×1 convolution has K channels, its output associated with the k-th kernel is denoted as $f_{i,j,k}^{(s)}$, where $k = 1, \ldots, K$. Let $H^{(s)}$ and $W^{(s)}$ be the height and width of features maps derived from the s-th 1×1 convolution. Then, the global average pooling performed over the feature map $f_k^{(s)}$ is calculated by

$$g_k^{(s)} = \frac{1}{H^{(s)} W^{(s)}} \sum_{i=1}^{H^{(s)}} \sum_{j=1}^{W^{(s)}} f_{i,j,k}^{(s)}, \tag{2.1}$$

Where $g_k^{(s)}$ is the k-th element in the s-th GAP feature vector. We notate $g^{(s)} = [g_1^{(s)}, \ldots, g_K^{(s)}]$, a $1 \times K$ dimensional vector, as the GAP feature from the s-th side branch. $g^{(S)}$ represents the GAP feature from the full-depth main branch.

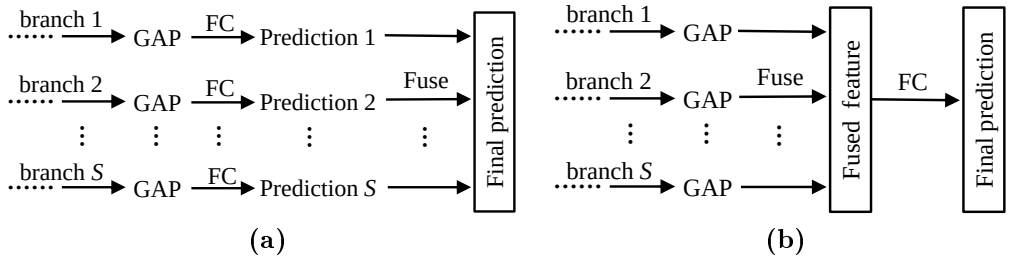


Figure 2.4: Comparison between EPLF and EFLP. (a) The schematic pipeline of EPLF strategy; (b) The schematic pipeline of EFLP strategy.

(2) Early fusion and late prediction

Considering how to incorporate the side branches, some work [18, 26, 31] used an "early prediction and late fusion" (EPLF) strategy. In Figure 2.4a, EPLF computes a prediction from the GAP feature using a fully-connected layer and then fuses side predictions together to make the final prediction. In contrast to EPLF [18], in which a couple of FC layers are added, we present another strategy called "early fusion and late prediction" (EFLP). EFLP first fuses the GAP features from the side branches and obtains a fused feature. Then, a fully-connected layer following the fused feature is used to estimate the final prediction. As seen in Figure 2.4b, EFLP has fewer parameters due to using only one fully-connected layer. Assume that each fully-connected layer has C units that correspond to the number of object categories. The fusion module has W_{fuse} parameters. Quantitatively, we can compare the number of parameters between EFLP and EPLF by

$$W_{EFLP} = K(C+1) + W_{fuse} < W_{EPLF} = SK(C+1) + W_{fuse}. \tag{2.2}$$

Hence, we make use of EFLP to fuse intermediate features earlier due to its efficiency. We observe that EFLP can achieve the same accuracy as EPLF, though EPLF contains more parameters. More importantly, the fused feature in EFLP is able to act as a richer image representation, however, EPLF cannot generate such a rich fused representation. In the experiments, we transfer the fused feature in EFLP to diverse vision tasks and show its promising generalization ability.

(3) Locally-connected fusion

Another significant component in CFN is that it employs a locally-connected (LC) layer to fuse the side branches. Owing to its no-sharing filters over spatial dimensions, LC layer can learn different weights in each local field [123]. For example, DeepFace [124] used the LC filters to learn more discriminative face representations instead of spatially-sharing convolutional filters. Differently, our aim is to adapt a LC layer to learn adaptive weights for different side branches, and generate a better

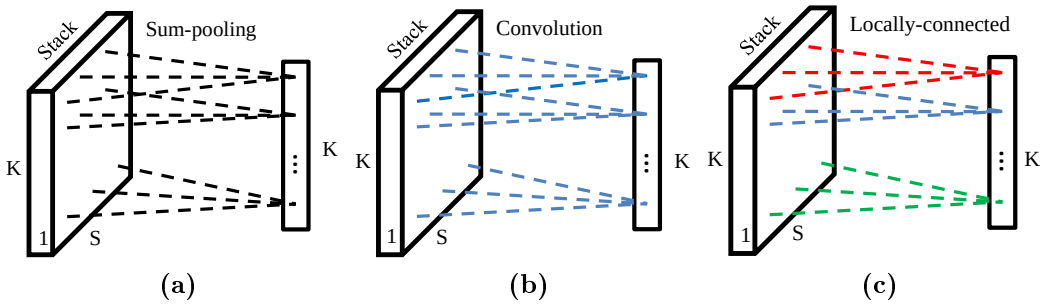


Figure 2.5: Comparison of three fusion modules. (a) No weights: Sum-pooling fusion has no weights; (b) Sharing weights: Convolution fusion learns sharing weights over spatial dimensions, as drawn in the same color; (c) No-sharing weights: Locally-connected fusion learns no-sharing weights over spatial dimensions, as drawn in different colors. To learn element-wise weights, we use the 1×1 kernel size.

fused feature. To the best of our knowledge, this is the first attempt to apply a locally-connected layer to a fusion module.

At first, we stack all GAP features together from $g^{(1)}$ to $g^{(S)}$, and form a layer G with size of $1\times K \times S$, see Figure 2.3. For example, the s-th feature map of G is $g^{(s)}$. Then, one LC layer which has K of no-sharing filters is convolved over G. Each filter has $1\times1\times S$ kernel size. Since LC is able to learn adaptive weights for different elements in the GAP features which measure the importance of the side branches, it is able to produce a better fused feature. Finally, the fused feature convolved by LC also has a $1\times K$ shape, denoted as $g^{(f)}$. The i-th element in $g^{(f)}$ is expressed:

$$g_i^{(f)} = \sigma\left(\sum_{j=1}^{S} W_{i,j}^{(f)} \cdot g_i^{(j)} + b_i^{(f)}\right), \tag{2.3}$$

where $i = 1, 2, \ldots, K$; σ indicates the ReLU activation function. $W_{i,j}^{(f)}$ and $b_i^{(f)}$ represent the weights and bias for fusing the i-th elements of GAP features from different side branches. The number of parameters in the LC fusion is $K \times (S+1)$. Using these additional parameters gives the benefit of adaptive fusion while it does not require any manual tuning.

To clearly demonstrate the advantage of the LC fusion module, Figure 2.5 compares LC fusion with other fusion methods. In Figure 2.5a, the sum-pooling fusion simply sums up the side outputs together without learning any weights, whereas this way treats each side branch equally and fails to consider their different importance. In Figure 2.5b, the convolutional fusion can learn only one sharing filter over all spatial dimensions (as drawn with the same blue color). In contrast, LC enables the fusion module to learn independent weights over each local field (*i.e.* size $1\times1\times S$) (drawn in different colors in Figure 2.5c). Although LC fusion consumes more parameters than the sum-pooling fusion (no weights) and the convolutional fusion ($S+1$),

these parameters are a negligible proportion of the total number of the network parameters.

2.2.2 Training procedure

CFN has a similar training procedure as a standard CNN, including forward pass and backward propagation. Assume a training dataset which contains N images: $\{x^{(i)}, y^{(i)}\}$, where $x^{(i)}$ is the i-th input image and $y^{(i)}$ is its ground-truth class label. W indicates the set of all parameters learned in the CFN (including the LC fusion weights). The full objective is to minimize the total loss cost

$$\text{argmin}_W \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(x^{(i)}; W), y^{(i)}), \tag{2.4}$$

where $f(x^{(i)}; W)$ indicates the predicted class of $x^{(i)}$. We use the softmax loss function to compute the cost $\mathcal{L}$. To minimize the loss cost, the partial derivatives of the loss cost with respect to any weight are recursively computed by the chain rule during the backward propagation [3]. Since the main parts in the CFN model are the side branches, we will induce the computations of their partial derivatives. We consider each image independently for notational simplicity.

First, we compute the gradient of the loss cost with respect to the outputs of the side branches. Taking the s-th side branch as an example, we compute the gradient of $\mathcal{L}$ with respect to the side output $g^{(s)}$

$$\frac{\partial \mathcal{L}}{\partial g^{(s)}} = \frac{\partial \mathcal{L}}{\partial g^{(f)}} \cdot \frac{\partial g^{(f)}}{\partial g^{(s)}}, s = 1, 2, \ldots, S. \tag{2.5}$$

Second, we formulate the gradient of $\mathcal{L}$ with respect to the inputs of the side branches. Let $a^{(s)}$ be the input of the s-th side branch. As depicted in Figure 2.3, $a^{(s)}$ corresponds to the pooling layer. However, the input of the main branch, denoted as $a^{(S)}$, refers to the last convolutional layer (*i.e.* conv S). It is important to note that the gradient of $a^{(s)}$ depends on several side branches. To be more specific, the gradient of $a^{(1)}$ is influenced by S branches; the gradient of $a^{(2)}$ needs to consider the gradient from the 2-nd to S-th branch; but the gradient of $a^{(S)}$ is updated by only the main branch. Then, the gradient of $\mathcal{L}$ with respect to the side input $a^{(s)}$ can be computed via

$$\frac{\partial \mathcal{L}}{\partial a^{(s)}} = \sum_{i=s}^{S} \frac{\partial \mathcal{L}}{\partial g^{(i)}} \cdot \frac{\partial g^{(i)}}{\partial a^{(i)}}, \tag{2.6}$$

where i indexes the related branch that contributes to the gradient of $a^{(s)}$. It needs to sum up the gradients from several side branches. As is common practice, we employ a standard stochastic gradient descent (SGD) algorithm with mini-batch [4] to train the entire CFN model.

2.2.3 Comparisons with other models

To get more insights into CFN, we compare it with other deep models.

Comparison with CNN. Typically, a plain CNN only estimates a final prediction based on the topmost layer. As a result, the effects of intermediate layers towards the prediction are implicit and indirect. In contrast, CFN connects the intermediate layers using additional side branches, and fuses them to jointly make the final predictions. In this way, CFN allows us to take advantage of intermediate layers explicitly and directly. This advantage explains why CFN is able to achieve more accurate prediction than a plain CNN.

Comparison with DSN. Deeply supervised nets (DSNs) [125] are the first model to add extra supervision to intermediate layers for earlier guidance. As a result, it can improve the directness and transparency of learning a deep neural network. Therefore, we can view DSN as a "**loss fusion**" model. Instead, CFN still uses one supervision towards the final prediction derived from the fused representation, however it is able to increase the effects of the loss cost on the intermediate layers without adding more supervision signals. In a word, we clarify that CFN is a "**feature fusion**" model. It is important to note that there is no technical conflict between CFN and DSN, so combining these two models together is a promising research direction.

Comparison with ResNet. ResNet [10] addresses the vanishing gradient problem by adding densely shortcut connections. CFN has three main differences with ResNet: (1) The side branches in CFN are not shortcut connections. They start from pooling layers and merge into a fusion module together. (2) In contrast to adding a "linear" connection in a residual block, we still use the non-linear ReLU in the side branches of CFN. (3) CFN employs a sophisticated fusion module to generate a richer feature, rather than using the simple summation employed in ResNet. As mentioned in the ResNet work, when the network is not overly deep, for example having 11 or 18 layers, ResNet may show few improvements over a plain CNN. However, CFN can obtain some considerable gains over CNN. Hence, CFN can serve as an alternative for improving the discriminative capabilities of not-very-deep models, instead of purely increasing the depth. ResNet tells us that "**depth that matters**", but CFN concludes to "**fusion that matters**".

2.3 Fully Convolutional Fusion Networks

Deep neural networks allow to bridge the gap between different vision tasks. For instance, CNN models for image-level classification can be well-adapted to other pixel-level classification tasks which aim to generate a per-pixel prediction in images. As a common practice, it is essential to cast traditional convolutional neural

networks to their corresponding fully convolutional networks (FCNs) by replacing the fully-connected layers with more convolutional layers. FCNs are able to infer any size of images without requiring specific input dimensionality. In this section, we introduce fully convolutional fusion networks (FCFN), which are used for two representative pixel-level classification tasks: semantic segmentation and edge detection. Similar to CFN, FCFN models are able to learn better pixel predictions based on the locally-connected fusion module.

2.3.1 Semantic segmentation

Semantic segmentation intends to predict a category label for spatial pixels in an image. FCN-8s [26] is a milestone model in the development of employing CNNs for semantic segmentation, and yields significant improvements in comparison with non-deep-learning approaches. First, FCN-8s is fine-tuned from the VGG-16 model [7] pre-trained on the ImageNet dataset [5]. Then, it adds two side branches to the full-depth main branch, which allow to integrate both coarse-level and fine-level pixel predictions to improve the semantic segmentation performance. Particularly, FCN-8s uses a simple sum-pooling to fuse the multi-level predictions. In contrast to FCN-8s, we extend the proposed CFN model and build the FCFN counterpart for generating fused pixel features. Moreover, we use two locally-connected layers in a two-stage fusion manner.

Recall that the locally-connected (LC) fusion module is able to learn independent weights for each spatial pixel in an image. We need to extend its formulations to be suitable for the LC fusion module in FCFN. In the first fusion module, two branches involving K channels of feature maps are taken as input. Note that the top layers are upsampled 2 times to retain the same spatial dimensions as the bottom layers. We consider the adaptive weights of each channel separately and reshape each two-dimensional feature map to a one-dimensional feature vector. For example, $g_{k,i}^{(1)}$ indicates the feature activation of the i-th pixel of the k-th channel in the first branch, and $g_{k,i}^{(2)}$ is the corresponding activation in the second branch, where $i = 1, \ldots, H \times W$ and $k = 1, \ldots, K$. Therefore, the fused pixel feature is given by

$$g_{k,i}^{(f)} = \sigma\left(\sum_{j=1}^{2} W_{k,i,j}^{(f)} \cdot g_{k,i}^{(j)} + b_{k,i}^{(f)}\right), \tag{2.7}$$

The number of parameters in this fusion module is $H \times W \times C \times 2$, where C is the number of object categories. Moreover, the second fusion module integrates coarser feature maps with the output of the first fusion module. Let $g_{k,i}^{'(1)}$ be the activation in the coarser layer. For notational simplicity, the activation $g_{k,i}^{(f)}$ from the output of the first fusion module, is renamed to $g_{k,i}^{'(2)}$. The computation in the second LC

fusion is

$$g_{k,i}^{'(f)} = \sigma\left(\sum_{j=1}^{2} W_{k,i,j}^{'(f)} \cdot g_{k,i}^{'(j)} + b_{k,i}^{'(f)}\right), \tag{2.8}$$

where $g_{k,i}^{'(f)}$ represents the final fused feature by using the two-stage fusion. Considering the computation of the loss cost with respect to the ground-truth, we still employ the softmax loss function and accumulate the loss of all pixels together.

$$\mathcal{L} = -\sum_{i=1}^{H\times W}\sum_{k=1}^{K} \mathbf{h}(y_i = k)\log p_{k,i}, \tag{2.9}$$

where y_i is the ground-truth pixel label. $\mathbf{h}(y_i = k)$ is equal to 1 when $y_i = k$, and 0 otherwise. The predicted pixel probability is normalized with the softmax function, where $p_{k,i} = \frac{\exp(g_{k,i}^{'(f)})}{\sum_{k=1}^{K}\exp(g_{k,i}^{'(f)})}$. As above, we give the loss computation for one image, but it is straightforward to extend it to a mini-batch size of images. Likewise, we use the SGD with mini-batch to train the entire FCFN model.

2.3.2 Edge detection

The problem of edge detection is to extract semantically meaningful edges in images, Typically, edge detection acts as a low-level task, but has significant contributions to other high-level visual tasks, such as object detection and image segmentation. Driven by the increasing developments of deep learning, edge features have moved from carefully-engineered descriptors such as Canny [109], gPb [126] and Structured Edges (SE) [127]), to discriminative deep features [29, 30, 31]. In particular, HED [31] is the first work to use FCNs for end-to-end edge detection, and leads to state-of-the-art performance on well-known benchmarks. HED integrates the strengths of efficient end-to-end FCNs [26] and additional deep supervision [125].

In contrast to HED that uses a convolutional fusion module, our FCFN fuses five intermediate side branches with a locally-connected layer. To be specific, one side branch generates a feature map where the activations measure the probabilities of pixels being edges. Five feature maps from side branches stack together, and are reshaped from $(H, W, 5)$ to $(H \times W, 5)$. We compute the fused prediction $g_i^{(f)}$

$$g_i^{(f)} = \sigma\left(\sum_{j=1}^{5} W_{i,j}^{(f)} \cdot g_i^{(j)} + b_i^{(f)}\right), \tag{2.10}$$

where $i = 1, \ldots, H \times W$. The sigmoid cross-entropy loss function is

$$\mathcal{L}_{fuse} = -\sum_{i=1}^{H\times W}\left[\beta_i \log g_i^{(f)} + (1-\beta_i)\log(1-g_i^{(f)})\right], \tag{2.11}$$

where the parameter β_i regulates the importance of edge and non-edge pixels, as mentioned in [31]. It is important to note that we also impose the intermediate supervision on the side branches similar to [31, 125], to discard the negative edges in the earlier intermediate layers. The loss cost in the k-th side branch (*i.e.* $k = 1, \ldots, 5$) is represented as follows

$$\mathcal{L}_{side}^{(k)} = -\sum_{i=1}^{H\times W} \left[\beta_i \log g_i^{(k)} + (1-\beta_i)\log(1-g_i^{(k)})\right], \tag{2.12}$$

where $g_i^{(k)}$ accounts for the predicted probability of the i-th pixel being an edge point. Finally, the overall loss cost in FCFN integrates a fused loss term and five intermediate loss terms together:

$$\mathcal{L} = \mathcal{L}_{fuse} + \sum_{k=1}^{5} \mathcal{L}_{side}^{(k)}. \tag{2.13}$$

This edge detection network is also fine-tuned end-to-end from the VGG-16 model and updated with the SGD algorithm with mini-batch.

2.4 Experiments

This experimental section evaluates the performance of the proposed CFN for image-level classification and FCFN for pixel-level classification. First, we train the CFN models on the datasets: CIFAR-10/100 [128] and ImageNet 2012 [5]. Then, we transfer the trained CFN model to three new tasks, including scene recognition, fine-grained recognition and image retrieval. Moreover, we train the specific FCFN models for semantic segmentation on the PASCAL dataset [129], and edge detection on the BSDS dataset [126], respectively. All experiments were conducted using the Caffe library [130] on a NVIDIA TITAN X card with 12 GB memory.

2.4.1 Image classification on CIFAR

Both CIFAR-10 [128] and CIFAR-100 [128] consist of 50,000 training images and 10,000 testing images. They define 10 and 100 object categories, respectively. We preprocessed their RGB images by global contrast normalization [131], and randomly shuffled the training set. We measure the classification performance by computing the error rates.

Table 2.1: Two plain CNN models built for the classification experiments on the CIFAR-10/100 dataset.

CNN-A	CNN-B
Input 32 × 32 RGB image	
5 × 5 × 64 conv, ReLU	3 × 3 × 96 conv, ReLU 3 × 3 × 96 conv, ReLU
3 × 3 max-pooling, stride 2. Dropout ratio 0.5	
5 × 5 × 64 conv, ReLU	3 × 3 × 192 conv, ReLU 3 × 3 × 192 conv, ReLU
3 × 3 average-pooling, stride 2. Dropout ratio 0.5	
5 × 5 × 64 conv, ReLU	3 × 3 × 192 conv, ReLU 3 × 3 × 192 conv, ReLU
1 × 1 × 192 conv, ReLU	
8 × 8 global average pooling. Dropout ratio 0.5	
10 or 100-way fully-connected layer	
Softmax classifier	

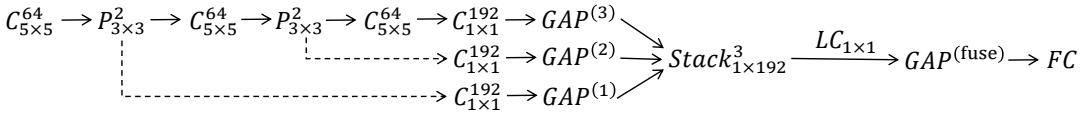
Network architecture and training details

We employ two plain CNNs models to build their CFN counterparts. Table 2.1 describes the two CNNs used for CIFAR-10/100, called CNN-A and CNN-B. (1) CNN-A is a shallow network similar to the Caffe-Quick model [130]. It has three 5×5 convolutions and a 1×1 convolution. The global average pooling is performed over the 1×1 convolutional maps. Finally, a fully-connected layer with 10 or 100 units is used to predict object categories; (2) CNN-B replaces each 5×5 convolutional layer in CNN-A with two 3×3 layers, as suggested in VGGnet [7]. In addition, CNN-B utilizes more feature channels than CNN-A. Note that, when training the CNN-B model on the CIFAR-100 dataset, the first and second convolutional layer use 192 channels instead of 96 channels. Correspondingly, the CFN-A and CFN-B models are built upon CNN-A and CNN-B respectively, by constructing two additional side branches after the pooling layers, as depicted in Figure 2.6(a) and (b).

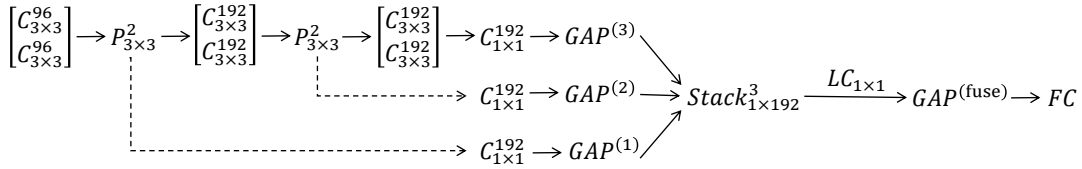
We use the same hyper-parameters to train CNN and CFN, for example, a weight decay of 0.0001, a momentum of 0.9, and a mini-batch size of 100. The learning rate is initialized with 0.1 and is divided by 10 after 10×10^4 iterations. The whole training will be terminated after 12×10^4 iterations. As for CFN, the initialized weights in the LC fusion module are set to 0.333, as there are three side branches in total (including the full-depth main branch).

Results and discussion

Table 2.2 shows the results on CIFAR-10/100 test sets. We can analyze the results considering the following three aspects:



(a) CFN-A



(b) CFN-B

Figure 2.6: Illustration of the proposed CFN models built for the CIFAR dataset. For the convolutional layers (denoted as C), the right lower number indicates the kernel size; the right upper numbers indicate the number of channels. For the pooling layers (denoted as P), the right lower numbers indicate the window size; the right upper numbers equal the size of strides.

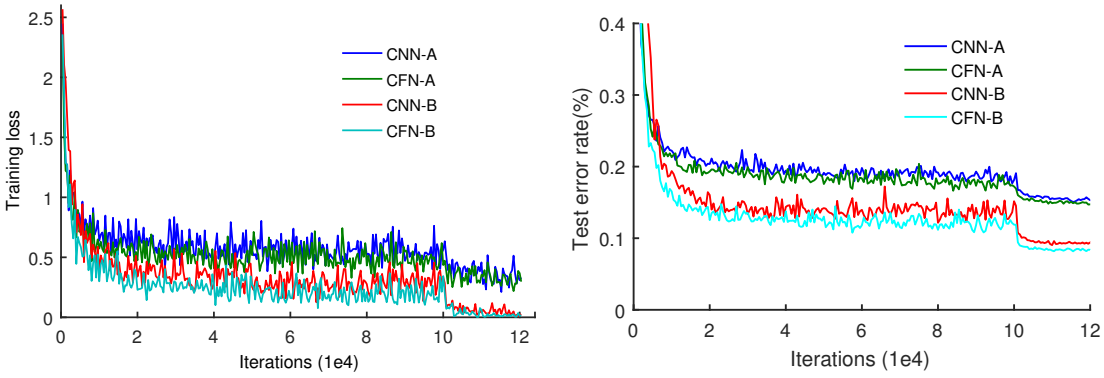
Table 2.2: Error rates (%) of image classification on the CIFAR-10/100 test set (without data augmentation). Better results are in bold face. CFNs can outperform the baseline CNNs by adding only a few parameters.

Model	#Parameters	CIFAR-10	CIFAR-100
CNN-A	0.224M (basic)	15.57	40.62
CNN-Sum-A	0.224M (basic) + 0.025M (side) + 0 (fusion)	15.33	40.32
CNN-Conv-A	0.224M (basic) + 0.025M (side) + 4 (fusion)	15.19	40.15
CFN-A	0.224M (basic) + 0.025M (side) + 768 (fusion)	**14.73**	**39.54**
CNN-B	1.287M (basic)	9.28	31.89
CNN-Sum-B	1.287M + 0.074M (side) + 0 (fusion)	8.84	31.42
CNN-Conv-B	1.287M + 0.074M (side) + 4 (fusion)	8.68	31.16
CFN-B	1.287M + 0.074M (side) + 768 (fusion)	**8.27**	**30.68**

(1) CFN achieves ∼1% improvements on the classification performance compared to the plain CNNs (both CNN-A and CNN-B). For example, on the CIFAR-10 dataset, CFN-A and CFN-B obtain 14.73 and 8.27 error rates that are ∼1% lower than the results of CNN-A and CNN-B, that are 15.57 and 9.28, respectively. The comparison between CFN and CNN demonstrates the effectiveness of fusing multi-level intermediate layers. Additionally, CFN is able to improve the expressive capabilities of deep neural networks for learning superior visual representations.

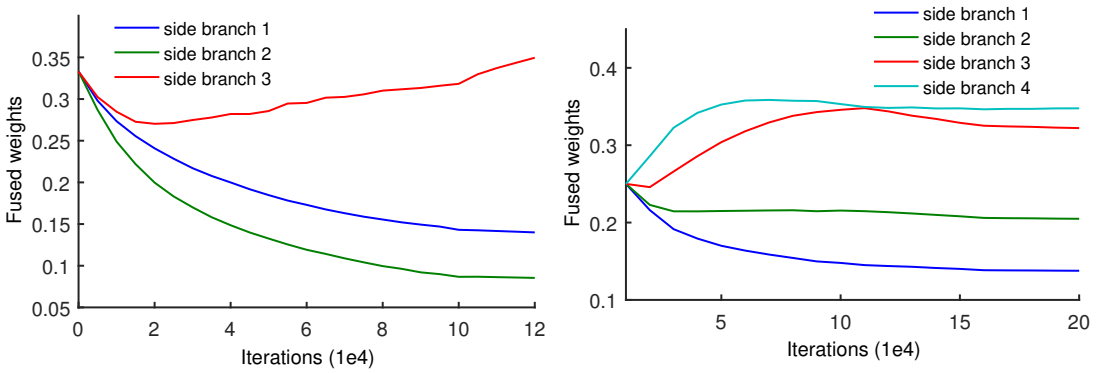
(2) In order to analyze the advantage of using the LC fusion, we also implement the existing sum-pooling fusion and convolutional fusion methods, denoted as CNN-Sum and CNN-Conv. By comparing CFN with CNN-Sum and CNN-Conv, we can observe that the LC fusion outperforms the other two fusion methods by a considerable margin. Hence, learning adaptive weights is essential to generate a better fused feature.

(3) Moreover, we compute the number of parameters in the models to estimate their efficiency. In the second column of Table 2.2, the additional number of parameters



(a) **(b)**

Figure 2.7: Comparison between CFN and CNN on the CIFAR-10 dataset. (a) The training loss when training CFN and CNN. (b) The test error rates along with the increasing iterations.



(a) CIFAR-10 **(b)** ImageNet

Figure 2.8: Illustration of adaptive weights of the side branches learned in the LC fusion. All side branches are initialized with the same weights before training. During the training stage, we can observe that the top branches have larger weights than the bottom branches.

for extra side branches and LC fusion are significantly smaller than the number of basic parameters in the models. Although the LC fusion consumes more parameters than the sum-pooling fusion and convolutional fusion, these parameters result in a minimal increase of the network complexity. In addition, we compare the training time between CNN and CFN. For example on the CIFAR-10 dataset, CNN-B and CFN-B train for approximately 1.67 and 2.08 hours, respectively.

Figure 2.7 shows the training loss and the test accuracy while training CFN and CNN. It can be seen that, both CFN-A and CFN-B models have less training loss and lower test error rates than the corresponding CNN models. In addition, Figure 2.8a presents the adaptive weights learned in the LC fusion of CFN-B. Recall that LC learns 192 filters (each filter is of size 1×3) and each filter has 1×3 weights. We compute the average weight in each branch, and estimate its fluctuation. By

Table 2.3: Test error rates on CIFAR-10/100 to compare CFN-B with other deep models. A superscripted * indicates the use of the standard data augmentation [125].

Method	Layers	CIFAR-10	CIFAR-10*	CIFAR-100
Maxout Networks [131]	5	11.68%	9.38%	38.57%
NIN [122]	9	10.41%	8.81%	35.68%
DSN [125]	9	9.69%	7.97%	34.54%
ALL-CNN [132]	9	9.08%	7.25%	33.71%
RCNN-160 [133]	6	8.69%	7.09%	31.75%
NIN + SReLU [134]	9	8.41%	6.98%	31.10%
CNN (baseline)	8	9.28%	7.34%	31.89%
CFN (ours)	8	**8.27%**	**6.77%**	**30.68%**

comparison, the side branch 3 (*a.k.a.* the full-depth main branch) plays a core role, while the other two side branches are complementary to the main branch. After a large amount of training iterations, the adaptive weights tend to be stable. Moreover, in Figure 2.1, we visualize and compare the learned feature maps in CNN-B and CFN-B. We select ten images from the CIFAR-10 dataset. The feature maps in the 1×1 convolutional layer of three side branches are extracted. We rank the feature maps by averaging spatial activations and select the top-4 maps to visualize. We can observe that CFN can learn complementary clues in the side branches, while retaining the necessary information in the main branch.

Comparison with other approaches.

Table 2.3 reports recent results on CIFAR datasets. For fair comparisons, we compare CFN-B with other not-very-deep models. Notably, "not-very-deep" is a relative concept. We use it to emphasize the differences between the models in Table 2.3 and other ResNet-like models [10]. Our method (CFN) and the compared methods develop less than 10-layer models to evaluate their effectiveness. These models certainly belong to deep neural networks, however, they are not very deep, compared to the ResNets that have more than hundreds of layers built on datasets like CIFAR-10/100. In addition, we report the depth of these models for a clear comparison and analysis. In summary, CFN obtains comparative results and outperforms these compared methods. In this work, we aim to investigate the potential of integrating multiple intermediate layers, and these results verify the effectiveness of CFN. Building CFN on top of a much deeper model (*e.g.* ResNet) is beyond the focus of our work, but it is suggestive for future research.

2.4.2 Image classification on ImageNet

The ImageNet 2012 dataset [5] consists of about 1.2 million training images, 50,000 validation images and 100,000 test images.

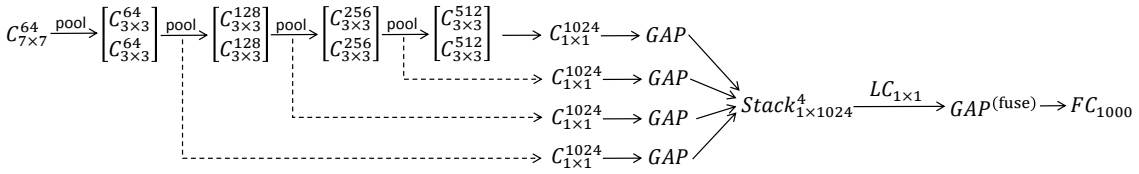


Figure 2.9: Overview of the CFN-11 architecture built on top of CNN-11. Three additional side branches are generated from the pooling layers, and fused together with the full-depth main branch (*i.e.* the last side branch).

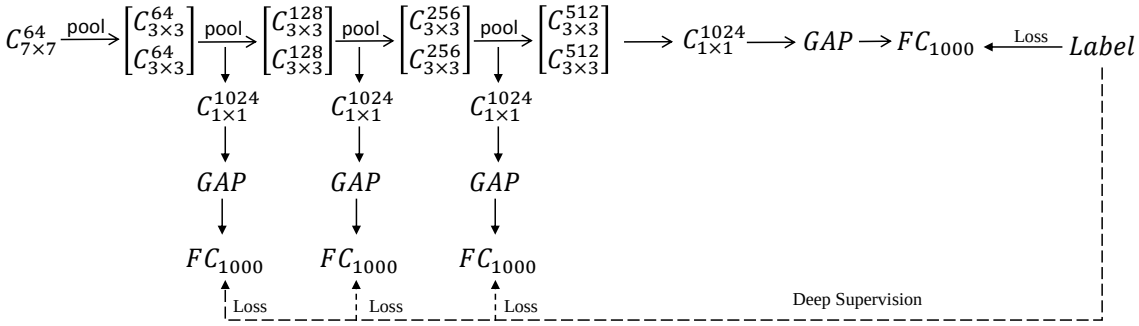


Figure 2.10: Overview of the DSN-11 architecture built on top of CNN-11. DSN-11 creates three side branches that can provide intermediate predictions for the input image. The ground-truth label is also used to guide these intermediate predictions, to enhance the discriminative abilities of hidden layers.

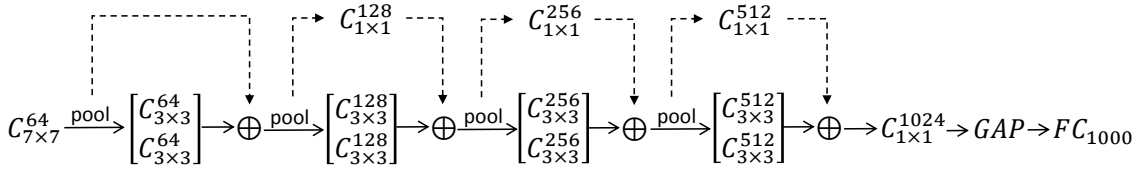


Figure 2.11: Overview of the ResNet-11 architecture built on top of CNN-11. There are four residual connections in total. Due to inconsistent numbers of channels, 1x1 convolution layers are needed in the residual connections, but they are not followed by ReLU to make sure linear transformation.

Network architecture and training details

We developed a basic 11-layer plain CNN (called CNN-11) where the channels of convolutional layers range from 64 to 1024. This baseline model is inspired by prior widely-used deep models [7, 8, 10]. Based on this CNN, we built its CFN counterpart (called CFN-11) as illustrated in Figure 2.9. Notably, we can create three extra side branches from the intermediate pooling layers (excluding the first pooling layer).

The training setup in our implementation follows the empirical practice in existing literature [4, 7, 8, 10]. The original image is resized to 256×256. In training phase, a 224×224 crop is randomly sampled from the resized image or its flipped one. The cropped input image is subtracted with per-pixel mean. We initialize the weights

and biases following GoogLeNet [8], for example a weight decay of 0.0001, and a momentum of 0.9. Batch normalization (BN) [135] is added after every convolutional layer. The learning rate starts from 0.01 and decreases to 0.001 and to 0.0001 at 10×10^4 iterations and 15×10^4 iterations respectively. The whole training will be terminated after 20×10^4 iterations. The LC weights in the fusion module are initialized with 0.25, as there are four side branches in total. We use SGD to optimize the models in a mini batch of size 64.

Results and discussion

Table 2.4 compares the results on the validation set. The following gives an analysis of the results from several aspects.

(1) CNN-11 is able to achieve competitive results when compared to AlexNet [4], however, it consumes much fewer parameters (∼6.3 millions) than Alexnet (∼60 millions). This is due to replacing several fully-connected layers with simple global average pooling.

(2) CFN-11 obtains an improvement of ∼1% over CNN-11 with adding only a few parameters (∼0.5 millions). It shows a consistent performance improvement by CFN for a large-scale dataset. Moreover, for fair comparison with other deep models, we implement the DSN-11 and ResNet-11 based on the plain CNN-11, which are shown in Figure 2.10 and Figure 2.11, respectively. It can be seen that, CFN-11 can still achieve better accuracy than DSN-11 and ResNet-11. Therefore, we can view CFN as an alternative to improve the feature representational abalities of such a not-overly deep CNN model, rather than increasing the depth as in ResNet. Notably, CFN-11 can improve CNN-11, but ResNet-11 cannot. But this does not show that CFN may be better than ResNet, as the networks are not very deep. Our primary purpose is to evaluate the superiority of CFN over CNN.

(3) To test the generalization of CFN to deeper networks, we build a 19-layer model following a similar principle as for the 11-layer model. Likewise, CFN-19 outperforms CNN-19 with ∼1% gains for both the top-1 and top-5 performance. For simplicity, we did not use the same hyperparameters as in ResNet [10], such as scale augmentation, large mini-batch size, multi-scale test. Therefore, our results of CNN-19 and CFN-19 are not as high as CNN-18 and ResNet-18 in [10]. We believe that our results can raise awareness of the potential of building deep multi-layer fusion networks. It is promising to develop much deeper networks to test the effectiveness of CFN, such as 50 or 100 layers.

Similar to CIFAR-10, Figure 2.8b illustrates the adaptive weights learned in the LC fusion of CFN-11. It is important to note that, the top branches (*i.e.* side 3 and side 4) have larger weights than the bottom branches (*i.e.* side 1 and side 2). Additionally, we extract the feature activations of the 1×1 convolutional layer in

Table 2.4: Error rates (%) regarding image classification on the ImageNet 2012 validation set.

Method	AlexNet	CNN-11	DSN-11	ResNet-11	CFN-11	CNN-19	CFN-19
Top-1	42.90	43.11	42.24	43.02	41.96	36.99	35.47
Top-5	19.80	19.91	19.24	19.85	19.09	14.74	13.93

Table 2.5: Configurations of six datasets for scene recognition, fine-grained recognition and image retrieval.

	Scene 15	Indoor 67	Flower	Bird	Holidays	UKB
#categories	15	67	102	200	–	–
#train images	1,500	5,360	2,040	5,994	991	10,200
#test images	2,985	1,340	6,149	5,794	500	10,200

one side branch. Figure 2.12 shows and compares the feature maps learned from different side branches.

2.4.3 Transferring deep fused features

To evaluate the generalization of CFN, we transfer the trained ImageNet model (*e.g.* CFN-11) to three new tasks: scene recognition, fine-grained recognition and image retrieval. Each task is evaluated on two widelly-used datasets: Scene-15 [136] and Indoor-67 [137], Flower [138] and Bird [139], and Holidays [140] and UKB [141]. The configurations of these six datasets are summarized in Table 2.5. Also, image examples are shown in Figure 2.13.

Specifically, AlexNet [4] acts as a baseline that uses the fc7 layer (4096-Dim) to provide an image representation. For CNN-11, we use the output of the global average pooling (1024-Dim) as image feature. Notably, CFN-11 allows us to utilize a fused feature (1024-Dim) that integrates multiple intermediate layers. For scene and fine-grained recognition, a linear SVM [142] is trained to compute the classification accuracy. For image retrieval, we compute the mean average precision (mAP) on Holidays and the N-S score on UKB. In terms of mAP, we take a ranked list of retrieved candidates and calculate performance based on the rank of the positive instances in the list. Given a query image, N-S is used to measure how many of the matched images are in the top-4 rank.

Results and discussion

Table 2.6 reports the transfer learning results on the six datasets. Although it is challenging to generalize a deep model to diverse visual tasks, he following summarizes the capability of CFNs in this respect.

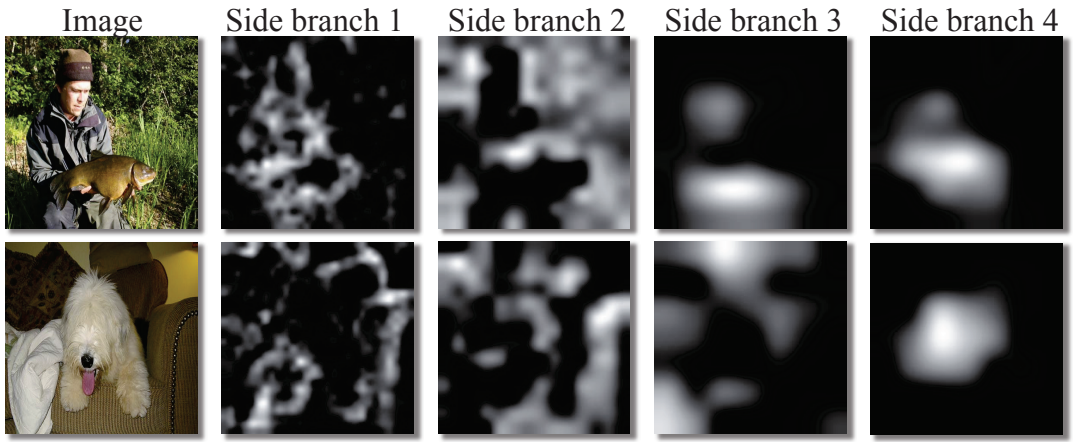


Figure 2.12: Illustration of feature maps in the four side branches. On one hand, the side branch 1 and 2 can capture some low-level clues about images, such as boundaries and textures. On the other hand, side branch 3 and 4 aim to obtain more abstract features that fire strong around objects. Therefore, CFN can incorporate multi-layer intermediate features explicitly and adaptively so as to improve visual representation.

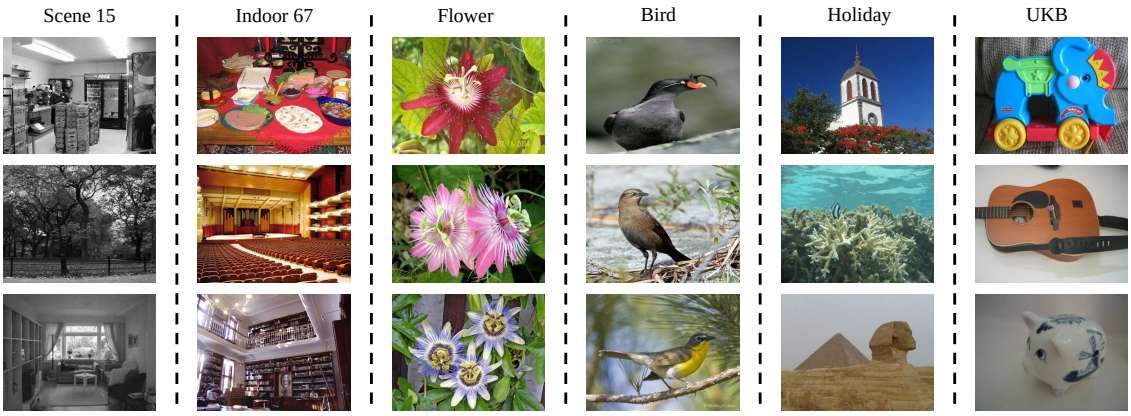


Figure 2.13: Image examples from six datasets about scene recognition, fine-grained recognition and image retrieval. We can see their significant differences with respect to the image content.

(1) Overall, CFN-11 obtains consistent improvements for the three tasks on all datasets, compared with the baseline CNN-11. In addition, CFN-11 outperforms Alexnet while using a much lower dimensional feature vector. These results reveal that learning fused deep representations is beneficial for not only image classification, but also a variety of visual tasks, even though the images in these tasks have large differences.

(2) Notably, the improvements on these three tasks are more significant than those on the ImageNet itself. In particular, CFN-11 yields a gain of $\sim$6% on the Flower dataset for fine-grained recognition. On other datasets, an accuracy improvement of $\sim$2% is obtained as well (Note that the UKB uses the N-S score which is different from precision accuracy). We believe that fine-tuning the models on the target datasets will further improve the results.

Table 2.6: Results on transferring the ImageNet model to three target tasks.

Method	#Dim	Scene recognition		Fine-grained recognition		Image retrieval	
		Scene 15	Indoor 67	Flower	Bird	Holidays	UKB
AlexNet [4]	4096	83.99	58.28	78.68	45.79	76.77	3.45
CNN-11	1024	84.32	60.45	76.79	45.98	78.33	3.47
CFN-11	1024	**86.83**	**62.24**	**82.57**	**48.12**	**80.32**	**3.54**

2.4.4 Semantic segmentation on PASCAL VOC

We conduct the semantic segmentation experiment on the PASCAL VOC 2012 segmentation dataset [129] that consists of 20 foreground object classes and a background class. The original dataset contains 1,464 training images, 1,449 validation images and 1,456 test images. When evaluating the validation set, we use a merged training dataset with the original training images and the augmented training images as in [143]. As there are validation images included in the merged training set, we need to pick the non-intersecting set of 904 images [26] as a new validation set.

We used the same hyper-parameters to train both the baseline FCN-8s [26] and the proposed FCFN, including a fixed learning rate of 10^{-4}, a weight decay of 0.0001, a momentum of 0.9, and a mini-batch size of 1. The training stage will be terminated after 100K iterations. It is worth mentioning that, we fine-tune FCN-8s directly from the VGG-16 model, without pre-training FCN-32s and FCN-16s. FCFN undergoes the same training procedure. The segmentation performance is measured with the pixel intersection-over-union (IoU):

$$IoU = \frac{TP}{TP + FP + FN}, \tag{2.14}$$

where TP, FP and FN denote the true positive, false positive and false negative counts, respectively.

Results and discussion

Table 2.7 reports the mean IoU accuracy and the detailed results of 20 object classes. The proposed FCFN achieves 1.6% gains on the mean IoU performance compared to the baseline FCN-8s. In addition, FCFN achieves superior results for more object classes, compared to FCN-8s. Figure 2.14 shows a visual example to highlight the segmentation details between the two models. We clarify that FCFN is a general architecture that can be integrated with other sophisticated techniques such as CRF [27] and Recurrent Neural Networks (RNN) [144], in order to further recover the segmentation details.

Table 2.7: Semantic segmentation results (IoU accuracy) on the PASCAL VOC 2012 validation set. For the 20 object classes, better results are in bold face.

	FCN-8s [26]	FCFN
aero	**75.5**	75.2
bike	**34.5**	33.8
bird	69.5	**72.0**
boat	**56.7**	53.3
bottle	59.7	**63.8**
bus	68.7	**71.2**
car	**70.3**	69.2
cat	73.4	**75.0**
chair	23.8	**24.0**
cow	53.0	**63.4**
table	39.7	**40.7**
dog	63.3	**65.6**
horse	46.3	**57.6**
mbike	**75.2**	74.5
person	73.9	**75.4**
plant	**42.2**	40.2
sheep	59.7	**62.3**
sofa	27.0	**30.3**
train	73.4	**74.0**
tv	**58.7**	55.7
mean	**57.1**	**60.3**

2.4.5 Edge detection on BSDS500

We evaluate the edge detection performance on the BSDS500 dataset [126] that consists of 200 training, 100 validation and 200 testing images. One image is manually annotated by five human annotators on average. The validation set is used to fine-tune the hyper-parameters, similar to HED [31]. For example, we use a momentum of 0.9 and a weight decay of 0.0002. In addition, the weights of the side-output convolutional filters are initialized with 0, and the initialization of the LC fusion filter is set to 0.2 due to fusing five side branches. The training images are resized to 400×400 and the batch size is 8. The learning rate is initialized with 10^{-6}, and the training is terminated after 25 epoches. The performance measurements for edge detection include the fixed contour threshold (ODS), the per-image best threshold (OIS) and the average precision (AP). Both ODS and OIS compute the F-score

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}. \tag{2.15}$$

Notably, ODS uses a fixed threshold to binarize all edge detection images in the test set, while OIS computes the best threshold for each image separately.

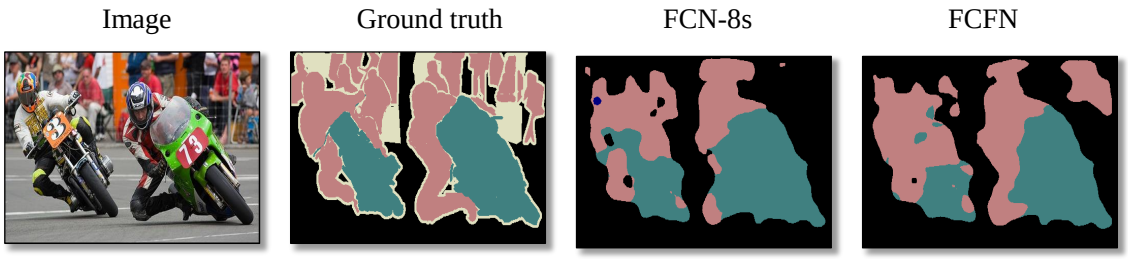


Figure 2.14: Comparison of a semantic segmentation example between the baseline FCN-8s and the proposed FCFN.

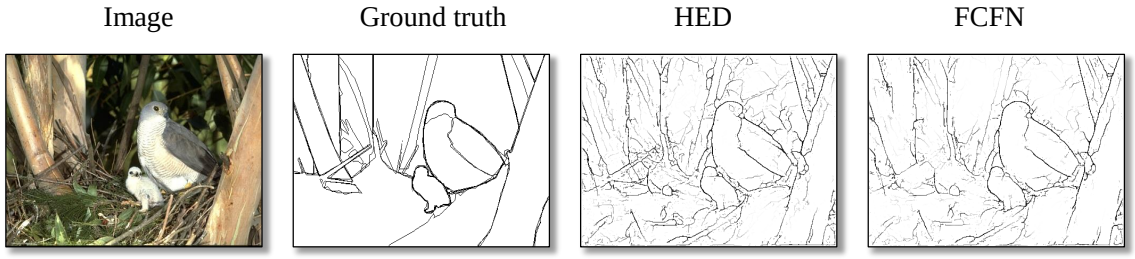


Figure 2.15: Comparison of an edge detection example between the baseline HED and the proposed FCFN. The FCFN results look more similar with the ground-truth annotations than the HED results.

Table 2.8: Edge detection results on the BSDS dataset. The upper group lists some representative approaches without using deep learning. The lower group gives the deep learning based approaches.

Method	ODS	OIS	AP
Canny [109]	.600	.630	.580
gPb-owt-ucm [126]	.726	.757	.696
SE-Var [127]	.746	.767	.803
DeepEdge [29]	.753	.772	.807
DeepContour [30]	.757	.776	.790
HED [31]	0.780	0.802	0.786
FCFN	**0.784**	**0.806**	**0.788**

Results and discussion

Table 2.8 provides a comparison of edge detection results on the BSDS dataset. First, we can see that deep learning approaches (in the lower group) largely promote the state-of-the-art performance compared to the hand-crafted edge detection approaches (in the upper group). In addition, the proposed FCFN outperforms the baseline HED with considerable improvements. This shows the advantage of learning adaptive weights in the locally-connected fusion module. Figure 2.15 shows an edge detection example and compares the visual details between FCFN and HED. FCFN can detect more satisfactory edges than HED.

2.5 Chapter Conclusions

In this chapter, we proposed a deep fusion architecture (CFN) built on top of plain CNNs. It allowed to aggregate intermediate layers with adaptive weights, and generated a discriminative feature representation. We conducted comprehensive experiments to evaluate its effectiveness for both image-level and pixel-level classification tasks. We can summarize several remarks and insights based on the experiments:

(1) On the CIFAR and ImageNet datasets, the CFN models have achieved considerable improvements while adding few parameters, even though these models are not very deep. CFN is a simple yet efficient architecture that has potential to be adapted to both deep (e.g. 10 layers) and much deeper (e.g. 100 layers) networks. In future work, we aim to build CFN on top of other deeper networks.

(2) CFN shows promising results when it is transferred to three different tasks, since CFN inherits the generalization capabilities of CNN. Additionally, CFN yields remarkable gains over CNN in the Flower dataset for fine-grained recognition. We find that it is quite important and necessary to make use of intermediate features to describe fine-grained attributes of objects.

(3) Although the FCFN models need to learn more adaptive weights in the fusion module, it can bring considerable performance improvements for semantic segmentation and edge detection. We find that many complementary details related to objects (*e.g.* boundary) are obtained from the intermediate layers.

Future work. Recall that the proposed CFN (and FCFN) model is a general extension of a plain CNN, and can be applied to a variety of visual recognition tasks. We can further improve CFN from the following two promising directions.
(1) While computing adaptive weights in the LC fusion module, we use a 1×1 kernel filter to independently consider each spatial location in the feature maps. A potential improvement would be to utilize larger kernel sizes such as 1×2 and 1×3, which can incorporate the contextual information in the feature maps.
(2) The adaptive weights are learned with the training images and are then directly applied to the test images for inference. It may be beneficial to learn input-specific weights to decrease the variance between images. Jaderberg, et al. [145] proposed a new learnable module, called the Spatial Transformer, that can perform explicit spatial transformations of features within CNNs. Similarly, Brabandere, et al. [146] proposed a Dynamic Filter Network (DFN), where filters are dynamically generated conditioned on an input image. Driven by these works, CFN can also learn dynamical filters conditioned on an input image.

Chapter 3

Recognizing Image Edges

In the previous chapter, we have shown the generalization power of deep neural networks for pixel-level classification. In this chapter, we focus on how we can develop diverse supervision in CNNs for edge detection (RQ 2).

To improve the robustness of edge detection, we build hierarchical supervisory signals with additional relaxed labels and adapt the signals to consider the diversities in hierarchial layers. Specifically, we begin by capturing the relaxed labels from simple detectors (*e.g.* Canny). These relaxed labels can be seen as some false positives that are difficult to be classified. Then we merge them with the general ground-truth to generate the relaxed deep supervision (RDS). We can employ the RDS to supervise the edge detection network in a coarse-to-fine paradigm. Moreover, we compensate for the lack of training images by capturing coarse edge annotations from a segmentation dataset. We pre-train the model with coarse annotations and then fine-tune it with fine annotations. Extensive experiments demonstrate that our approach achieves superior performance on the BSDS500 dataset (ODS F-score of .792) and promising cross-dataset results on the NYUD dataset.

Keywords
Edge detection, Fully convolutional networks, Deep supervision, Pre-training

3.1 Introduction

Edge detection, which aims to extract the important edges from images, has served as a fundamental task in the computer vision community for several decades. Typically, edge detection is considered as a low-level problem, and it is frequently used for other high-level vision applications, for example, object detection [147] and segmentation [126]. Most of the traditional edge detection approaches [109, 110, 111, 112, 148, 149, 150, 151] extract discriminative local features with color and gradient clues, such as gPb [126], Sketch tokens [152] and Structured Edges (SE) [127].

Recently, edge detection has achieved significant advances due to the developments of deep features. Figure 3.1 displays the basic pipeline of current edge detection systems based on deep learning. Based on different levels in predicting edges, we broadly divide them into three categories.

(1) *Pixel-level prediction*: extract deep feature per pixel and classify it to edge or non-edge class. Early work such as [113] developed a convolutional RBM to learn pixel-level features. Hwang and Liu [153] stacked pixel features in a multi-scale CNN model and then fed them to an SVM classifier. Bertasius *et al.* [29] built four CNN models to learn multi-scale features to detect edge points. Then they improved their network structure with less computational cost [154].

(2) *Patch-level prediction*: estimate edge maps for the input patches and then integrate them for the whole edge map. For example, the N4-Fields [155] extracted patch features from a pre-trained CNN model, and then mapped them to the nearest neighbor annotation from a pre-built dictionary. Shen *et al.* [30] clustered contour patches for mid-level shape classes and solved the model using a positive-sharing loss function.

(3) *Image-level prediction*: predict the whole edge map end-to-end given one input image. Considering the inefficiency of the above two categories, Xie and Tu [31] proposed a holistically-nested edge detection (HED) approach that was the first attempt to perform holistic image training and prediction for edge detection. Their work took advantage of the high efficiency of end-to-end fully convolutional networks (FCNs) [26], and additional deep supervision of deeply supervised nets (DSN) [125].

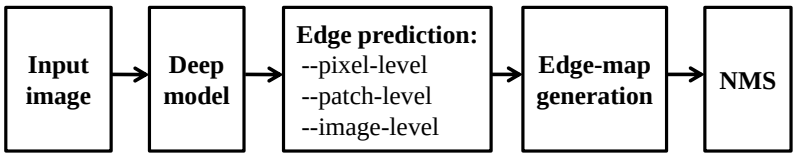


Figure 3.1: Pipeline of deep learning based edge detection. NMS is short for non-maximal suppression.

One difficulty in edge detection is attributed to *false positives*: many non-edge pixels are incorrectly predicted as edges compared with the human annotated ground-truth. To alleviate this issue, HED [31] imposed additional supervision (*i.e.* the annotated ground-truth) on the intermediate layers while training the deep model, and therefore the false positives could be corrected earlier. However, using only a general supervision for all the layers is inconsistent with the diverse representations of hierarchical layers. In addition, the general supervision can not be well-suited to all intermediate layers. Driven by this issue, in this chapter we pose a new research question **RQ 2: How can we explore diverse supervision that can adapt to different intermediate layers in deep neural networks for robust edge detection?**

To this end, we propose diverse deep supervision that can vary from coarse level to fine level as deep features become more discriminative. Our diverse supervision is called relaxed deep supervision (RDS), having additional relaxed labels, in addition to the positive labels (*i.e.* edge points) and negative labels (*i.e.* non-edge points). The relaxed labels are used to adapt to the diversities of intermediate layers. To be specific, we capture the relaxed labels from simple and efficient off-the-shelf detectors, for instance, Canny [109] or SE [127]. Then, we insert the extracted relaxed labels into the original ground-truth to generate RDS. In contrast to using a fixed general supervision, RDS can guide intermediate layers in a coarse-to-fine paradigm and process the false positives using a "delayed strategy". In this way, the loss cost of the relaxed labels are ignored in current supervision, and will be reconsidered in the next supervision. Therefore, more discriminative layers are assigned to process more false positives (difficult points). RDS can incorporate network diversities to improve the performance of edge detection.

Another problem about edge detection is that it requires more expensive human annotations, than other vision tasks like image classification and object detection. In addition, the frequently benchmarked BSDS500 dataset [126] has only 200 training images that limits the learning ability of various edge detectors based on deep learning. To alleviate this deficiency, we propose to generate coarse edge annotations (CEA) from a large collection of segmentation annotations such as the PASCAL Context dataset [156]. We pre-train the model with CEA and then fine-tune it with the target dataset, BSDS500.

The contributions of this work are as follows:

- We propose relaxed deep supervision to guide the intermediate predictions. Compared with traditional deep supervision, RDS can adapt to the hierarchical diversities with minimal manual efforts.
- We show that pre-training the model with a large collection of CEA is an efficient way to enhance the learning ability of CNNs and thus can achieve considerable improvements.

- Despite the apparent simplicity of RDS, our approach achieves competitive accuracy (ODS=.792) on the well-known benchmark BSDS500. In addition, our approach shows promising generalization between different datasets.

The rest of this chapter is structured as follows. Section 3.2 presents the proposed relaxed deep supervision for edge detection. The pre-training procedure with CEA is introduced in Section 3.3. In 3.4, we describe the implementation details in Section and report the experimental results. Finally, Section 3.5 summarizes the conclusions and future work.

3.2 Relaxed Deep Supervision

In this section, we present the proposed network with relaxed deep supervision for edge detection and formulate the algorithm.

3.2.1 Network details

Model Architecture

Our edge detection architecture is built on top of HED network [31], which is trimmed from the VGG-16 net [7] (Figure 3.2). The network architecture contains five convolutional nets connected with the max-pooling layers. Each convolutional net has several convolutional layers. In order to add deep supervision to guide the intermediate layers, five side-output layers (from side-output 1 to side-output 5) are inserted behind the intermediate layers. Due to the deconvolutional operation, the side-output predictions keep the same spatial size as the input image. In order to integrate multi-scale predictions, one weighted-fusion layer followed by fusion-output prediction is concatenated with five side-output predictions. Notably, HED utilizes the original ground-truth G as a general supervisory signal to guide the whole network, including five side-output predictions and the last fusion-output prediction.

Although the fusion-output prediction in HED is integrated with multi-scale predictions, their general supervision fails to present hierarchical diversities. Instead, our main aim is to explicitly make use of diverse supervision associated with different intermediate layers. To this end, we propose to integrate additional *relaxed labels* into the general supervision, and generate hierarchical and specific supervision, called relaxed deep supervision (RDS). Our approach stems from the fact that hierarchical layers can represent specific abstracts of the input image [157, 158]. In Figure 3.2, the bottom side-output predictions (*e.g.* side-output 1, 2) easily detect a large number of small edges and noise. In contrast, the top predictions(*e.g.* side-output 4, 5)

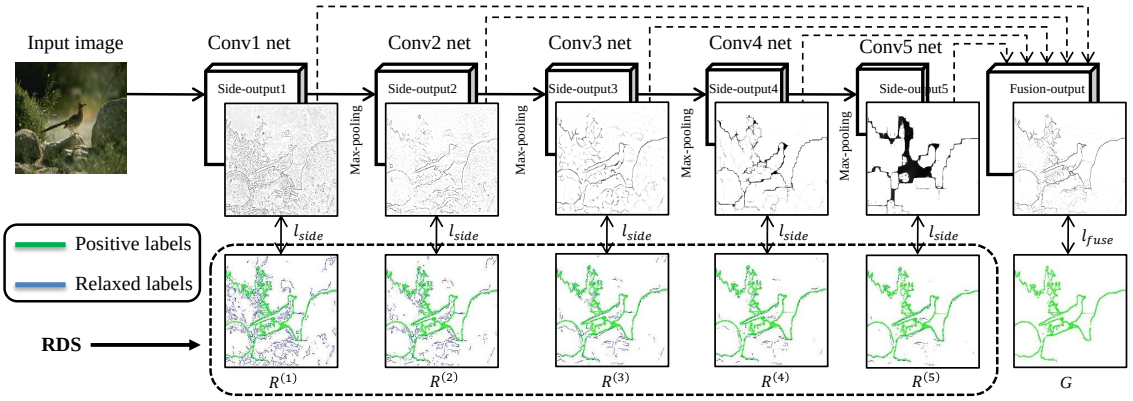


Figure 3.2: The network architecture with RDS (best viewed in color and zoom-in). The proposed RDS, including positive labels (green color), negative labels (white color for clear visualization), relaxed labels (blue color), is used to supervise the corresponding side-output prediction. The last fusion-output is still supervised by the original ground-truth G. The total loss cost in the network is the sum of all l_{side} and l_{fuse}.

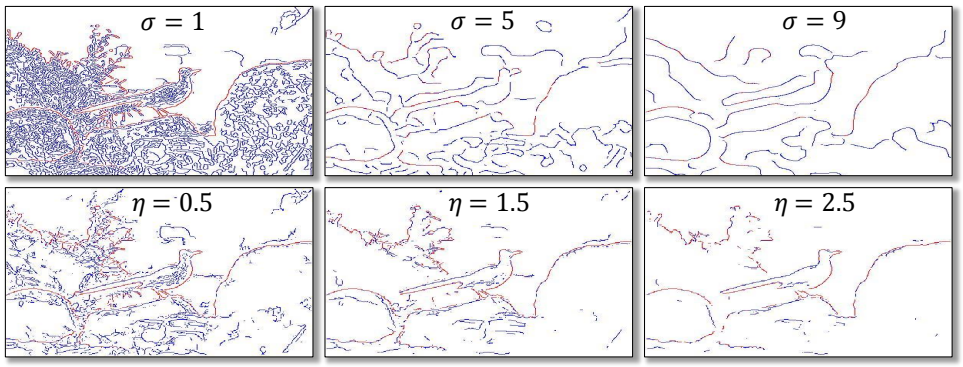


Figure 3.3: Illustration of extracting relaxed labels (blue color). The first and second rows display three edge responses from Canny [109] and SE [127], respectively.

can fire stronger responses around the positive labels. However, the general supervision can not be well-suited to all side-output predictions. In contrast, our RDS can not only preserve the strong supervision from the ground-truth, but also allow specific diversities by introducing the relaxed labels. In the following, we present two simple and efficient ways to capture the relaxed labels based on off-the-shelf edge detectors, including Canny [109] and SE [127].

Relaxed labels based on Canny detector

The Canny algorithm [109] can detect different scales of edge responses based on the parameter σ, which is the standard deviation of the Gaussian filter. The aforementioned relaxed labels can be extracted from Canny edge responses. First, we adjust different scales ($\sigma \in \{1, 3, 5, 7, 9\}$) to obtain various edge responses for five side-output predictions. We denote these binary edge responses with $\{C^{(k)}\}_{k=1}^{5}$. For example, $C^{(3)}$ is the edge response when $\sigma = 5$. Second, for the k-th side-output

prediction, we define its *relaxed labels*: "*belong to the positive labels of* $C^{(k)}$*, but are not included in the positive labels of the original ground-truth* G." The relaxed labels can present the complementary clues that are not in the ground-truth. Therefore, the set of relaxed labels can be computed as follows:

$$D^{(k)} = \mathrm{H}(C^{(k)} - C^{(k)} \cap G), \tag{3.1}$$

where the function H is used to collect the set of positive labels from the input binary map. As shown in Figure 3.3, the first row gives three scales of Canny edge responses (both red and blue color) when $\sigma = 1, 5, 9$. We highlight the relaxed labels in blue color, and the red points indicate the overlap edges between $C^{(k)}$ and G. The ground-truth G can be seen in Figure 3.2.

Relaxed labels based on SE detector

To demonstrate the generalization of our method, we also employ another edge detector: Structured Edges (SE) [127]. SE outputs one edge map with pixel-wise probabilities ranging from 0 to 1. Similarly, we need to create five binary edge responses from the SE edge map. We begin by computing the mean value of edge probabilities in the SE edge map, denoted as v. Then we adjust a threshold t to binarize the SE edge map by $t = \eta \cdot v$, where $\eta \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. As a result, we can have five binary edge responses, denoted as $\{S^{(k)}\}_{k=1}^{5}$. Similar to the above definition of relaxed labels, we compute the set of relaxed labels based on SE

$$D^{(k)} = \mathrm{H}(S^{(k)} - S^{(k)} \cap G). \tag{3.2}$$

The second row in Figure 3.3 displays the edge responses from SE and their relaxed labels (blue color). We can observe that the relaxed labels from SE detector are visually sparser than those from Canny detector.

RDS generation

It can be observed that various relaxed labels are well-suited to our needs of highlighting hierarchical diversities within the supervision. In the next stage, we need to insert the set of relaxed labels into the original ground-truth. This merging operation is used to generate the desirable RDS, which is an union of positive, negative, and relaxed labels. We denote five different RDS by $\{R^{(k)}\}_{k=1}^{5}$. The construction step can be seen in Figure 3.4, where the set $D^{(3)}$ is extracted based on $S^{(3)}$. The generated $R^{(3)}$ can not only preserve the positive labels in the ground-truth G, but also contain specific relaxed labels. Notably, the relaxed labels correspond to the non-edge points in G. These non-edge points can be viewed as some false positives that are difficult to classify.

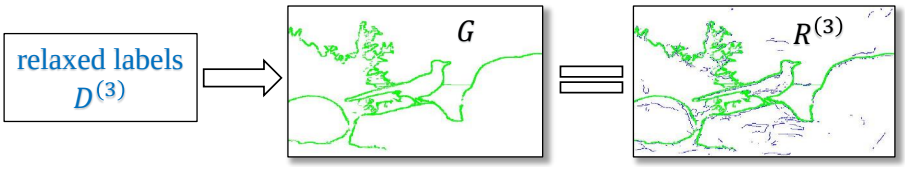


Figure 3.4: Illustration of generating the RDS (best viewed in zoom-in). $R^{(3)}$ is merged by the set $D^{(3)}$ and G.

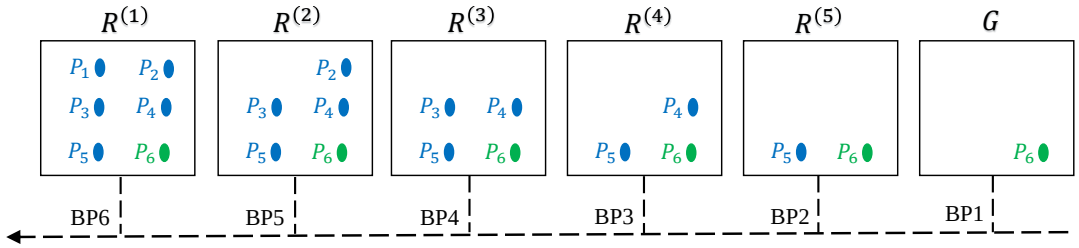


Figure 3.5: RDS employs a coarse-to-fine supervision strategy. The blue points indicate the relaxed labels, and the green point is one positive label.

3.2.2 Loss formulation

For a training dataset containing N images: $\{I_i, G_i\}_{i=1}^{N}$, I_i is the i-th input image and G_i is its edge ground-truth. $I_{i,j}$ denotes the j-th raw pixel over the spatial dimensions of I_i. Assume that we use the relaxed labels derived from SE detector $\left(\{D^{(k)}\}_{k=1}^{K}\right)$. The corresponding RDS are denoted as $\{R_i^{(k)}\}_{k=1}^{K}$, and $K = 5$ in the network. Five different side-output predictions are separately supervised with the corresponding RDS, and the fusion-output prediction is still supervised with the original ground-truth (Figure 3.2). In addition, early supervision (*e.g.* $R^{(1)}$ and $R^{(2)}$) has more relaxed labels than late supervision (*e.g.* $R^{(4)}$ and $R^{(5)}$). This is consistent with the hierarchical characteristics of CNN models. Finally, the total loss function L_{RDS} is expressed with

$$\sum_{i=1}^{N}\sum_{j=1}^{|I_i|}\left(\sum_{k=1}^{K} l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) + l_{fuse}(\widehat{G}_{i,j}^{fuse}, G_{i,j})\right), \tag{3.3}$$

where $|I_i|$ is the total number of pixels in I_i. l_{side} and l_{fuse} represent the loss cost per pixel, from the side-output and fusion-output, respectively. $\widehat{G}_{i,j}^{(k)}$ and $\widehat{G}_{i,j}^{fuse}$ indicates the j-th pixel prediction from the k-th side-output and the fusion-output, respectively. For notational simplicity, the network parameters, such as weights and bias, are not included in the equation. In $R_i^{(k)}$, the relaxed labels are set to 2, different from the positive labels (set to 1) and negative labels (set to 0). Therefore, we compute l_{side} based on the types of pixel labels

$$l_{side}(\widehat{G}_{i,j}^{(k)}, R_{i,j}^{(k)}) = \begin{cases} \alpha \cdot \log \mathrm{P}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \log(1 - \mathrm{P}(\widehat{G}_{i,j}^{(k)})), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \tag{3.4}$$

where $\mathrm{P}(\widehat{G}_{i,j}^{(k)})$, using sigmoid function, indicates the probability of current pixel being an edge point; α and β are used to balance the biased distribution between edge and non-edge pixels. Since about 90% pixels belong to non-edge class, we set $\alpha = 9\beta$ to enhance the edge class, for instance, $\alpha = 9$ and $\beta = 1$. Notice that, we compute l_{side} when the pixel has positive or negative label. However, when the pixel has a relaxed label ($R_{i,j}^{(k)} = 2$), we do not compute its loss cost and set $l_{side} = 0$. On the other hand, the computation of l_{fuse} excludes the third term in Eq. (3.4), because there are no relaxed labels in G_i. Next, we consider the back propagation (BP). We can deduce the partial derivatives of l_{side} *w.r.t.* $\widehat{G}_{i,j}^{(k)}$ by

$$\bigtriangledown \frac{l_{side}}{\widehat{G}_{i,j}^{(k)}} = \begin{cases} \alpha \cdot (\mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}) - 1), & R_{i,j}^{(k)} = 1 \\ \beta \cdot \mathrm{sigmoid}(\widehat{G}_{i,j}^{(k)}), & R_{i,j}^{(k)} = 0 \\ 0, & R_{i,j}^{(k)} = 2 \end{cases} \tag{3.5}$$

We follow the chain rule [159] to update the network parameters using stochastic gradient descent (SGD) with a mini-batch size [157].

Discussion. Training with RDS can maintain the strong supervision from the ground-truth, and incorporate hierarchical diversities. Here we will discuss how RDS improves edge detection. As mentioned before, one difficult issue in edge detection is attributed to the false positives. The relaxed labels based on Canny/SE actually correspond to some false positives that are difficult to classify. RDS processes these false positives using a *coarse-to-fine* paradigm: the false positives (with relaxed labels) in current supervision are ignored without computing their loss cost, but can be reconsidered in the next supervision. In this way, top layers are responsible for classifying the ambiguous false positives due to their high discriminative power. This paradigm is similar to hierarchical object classification [160], in which difficult classes are classified from coarse-category prediction to fine-category prediction.

We further demonstrate the paradigm in Figure 3.5. In $R^{(1)}$, P_1 serves as a relaxed label that is difficult to be predicted in the side-output 1. Thus we do not compute the loss cost of P_1 and delay its prediction until in $R^{(2)}$. In $R^{(2)}$, P_1 is converted to be a negative label (no-edge), so this provides evidence that the side-output 2 associated with stronger discrimination is able to predict P_1. Similarly, $R^{(5)}$ is able to recognize most relaxed labels except for P_5. Therefore, RDS can incrementally improve the strength of the supervision and assign more false positives to more high-level layers. Moreover, the network can run in a *coarse-to-fine* BP procedure. First, the whole network can be updated with *coarse* supervision G; Then, *fine* supervision $R^{(k)}$ (with specific relaxed labels) is used to fine-tune their local nets. For example, P_6 can be updated by all BPs (six times), and P_4 will be updated twice (by G and $R^{(5)}$). In a nutshell, RDS can benefit the whole training for edge detection. It can help reduce the total loss in the forward pass stage and facilitate efficient updates in the back propagation stage.

(a) Fine edge annotations

(b) Coarse edge annotations

Figure 3.6: Comparison between fine and coarse edge annotations. (a) displays three images and their ground-truth from BSDS500 [126]. (b) shows the images, segmentations from Pascal Context [156] in the first and second row, and coarse edge annotations (CEA) in the third row.

3.3 Pre-training Procedure

Generally, collecting more training data can develop the learning ability of CNNs. For many visual recognition tasks such as image classification and object detection, large-scale datasets are often available, *e.g.* ImageNet [5], MSCOCO [117] and PASCAL VOC [129]. However, the BSDS500 dataset [126] contains only 200 training images for learning edge detectors. This small training set limits current edge detection algorithms in improving the performance. In addition, fine edge annotations (FEA) require more expensive human effort than image classification.

To alleviate this issue, we attempt to extract coarse edge annotations (CEA) from a large collection of segmentation annotations. Here, we utilize the Pascal Context

Algorithm 1: RDS: training and testing procedure

Input: Training dataset; VGG-16 net; training iterations T_1,T_2
Initializing: network parameters $\mathbf{W}$ using VGG model
Preparation: for one image I_i, extract the set of relaxed labels $\{D_i^{(k)}\}_{k=1}^5$ and generate RDS $\{R_i^{(k)}\}_{k=1}^5$.
Pre-training: use Pascal Context data and its CEA, $t = 0$
  while $t < T_1$ **do**
    $t \leftarrow t + 1$
    Forward propagate to compute L_{CEA} in Eq. (3.6);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (3.5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \Delta\mathbf{W}$ with SGD;
  end while
Training: use the target training data set (*e.g.* BSDS500), $t = 0$
  while $t < T_2$ **do**
    $t \leftarrow t + 1$
    Forward propagate to compute L_{RDS} in Eq. (3.3);
    Backward propagate to get gradients $\Delta\mathbf{W}$, like Eq. (3.5);
    Update $\mathbf{W}^t = \mathbf{W}^{t-1} - \lambda_t \triangle \mathbf{W}$ with SGD;
  end while
Testing: feed one image into the learned network with parameters $\mathbf{W}$ and output edge map E_i
Post-processing: non-max suppression on E_i
Output: final edge map $E_i^{'}$

dataset [156], which provides full-scene segmentations for more than 400 classes, and has 10,103 train and validation images in total. Thus we extract the edges alongside the segmentations. In contrast to FEA, CEA only provides the outside boundaries of objects (See the car, people and building in Figure 3.6), but it can facilitate the network learning due to a large number of images. Notably, there are no overlap images between Pascal Context and BSDS500, which are from Flickr and Corel, respectively. During training with CEA, we simply compute the fusion-output loss function and exclude the intermediate supervision by

$$L_{CEA} = \sum_{i=1}^{N} \sum_{j=1}^{|I_i|} \left(l_{fuse}(\widehat{G}_{i,j}^{fuse}, G_{i,j}) \right). \tag{3.6}$$

In summary, we pre-train the model with the Pascal Context dataset and its CEA according to Eq. (3.6), and then fine-tune the model with the BSDS500 dataset as Eq. (3.3). We show the whole algorithm procedure in Algorithm 1, including the training and testing stages.

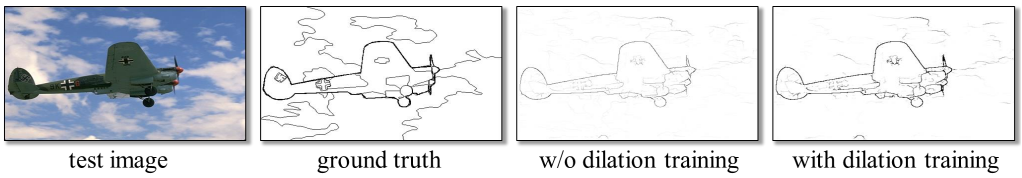


Figure 3.7: Qualitative comparison of edge detection results between without and with ground-truth dilation.

3.4 Experiments

3.4.1 Implementation details

Training details. We implemented our approach using the publicly available Caffe framework [130] and HED implementation [31]. We refer to some basic parameters as HED net, including momentum (0.9), weight decay (0.0002), initialization of the side-output filters (0), and initialization of fusion-output filter (0.2). The training images are resized to 400×400 and the batch size is 8. More importantly, we present some different parameters in our experiments. For example, the learning rate is fixed with 1e-9. This learning rate is quite efficient and reducing it during training iterations has no remarkable improvement. The training will be terminated after 25 epoches. Another difference is the class-balanced parameters α and β in Eq. (3.4). We utilize the fixed class-balanced parameters ($\alpha = 9, \beta = 1$) for all images.

Ground-truth dilation. Frequently, human subjects annotate the ground-truth edges with thin boundaries (*e.g.* one pixel width). However, the predicted edges from deep models have rather thick boundaries. To tackle this inconsistency, we dilate the positive labels in the ground-truth of a train set using a traditional morphologic dilation operator. Figure 3.7 compares the detection results between with and without dilation training. It can be seen that training with the dilated ground-truth contributes to predicting stronger edge maps. Quantitatively, the dilation process can increase the ODS accuracy about .02 on the BSDS500 test set. Hence, the ground-truth dilation is a simple and efficient step for improving the performance on edge detection. Note that we do not dilate the test set. In addition, the postprocessing non-maximal suppression (NMS) [109] can be used to thin the predicted edges.

3.4.2 Ablation study on BSDS500

Dataset. The BSDS500 dataset [126] consists of 200 training, 100 validation, and 200 testing images. The validation set is used to fine-tune the hyper-parameters. Each image is manually annotated by five human annotators on average. For training images, we just preserve their positive labels annotated by at least three human annotators. In testing stage, we extract the fusion-output prediction to evaluate

Table 3.1: Results on BSDS500 testing set. RDS(Canny) and RDS(SE) derive the relaxed labels from Canny and SE. CEA uses the extra data from Pascal Context.

	ODS	OIS	AP
Baseline 1	.762	.782	.766
Baseline 2	.780	.802	.786
RDS(Canny)	.785	.803	.813
RDS(SE)	.787	.804	.817
RDS(gPb)	.786	.803	.814
CEA	.765	.785	.724
RDS(Canny) + CEA	.790	.809	**.819**
RDS(SE) + CEA	**.792**	**.810**	.818

the performance. As mentioned in Section 2.4.5, we use the fixed contour threshold (ODS), the per-image best threshold (OIS) and the average precision (AP).

Baseline methods. To experimentally evaluate the effectiveness of RDS, we implemented two baseline methods. (1) *Baseline 1*: only supervises the fusion-output prediction with the general supervision (*i.e.* original ground-truth). (2) *Baseline 2*: imposes the general supervision to not only the fusion-output prediction, but also five side-output predictions. In Table 3.1, the Baseline 1 achieves ODS=.762 on BSDS500. Relatively, the Baseline 2 improves the accuracy to ODS=.780. This verifies the benefit of using additional intermediate supervision. The performance gap with/without intermediate supervision in HED [31] is less than that of our Baseline1 and Baseline2. The reason is that we do not perform data augmentation (*e.g.* rotation and flip) that has been employed in HED. Although the data augmentation may decrease the improvement of intermediate supervision, we believe that it should not remove our awareness of its importance.

Component analysis

Table 3.1 reports the results of our approach. To give more insights, we discuss them from three aspects.

(1) RDS yields considerable improvements over the general supervision approach (Baseline 2). This verifies the advantage of RDS for incorporating hierarchical diversities. The result of RDS with relaxed labels from Canny, denoted as RDS(Canny), achieves ODS=.785. The RDS(SE) result reaches ODS=.787.

(2) RDS is relatively insensitive to different choices of relaxed labels. First, we can see that RDS can obtain similar results with Canny and SE. In addition, we use another detector, gPb [126], to capture the relaxed labels. Similarly, its result (ODS=.786) is consistent with RDS(Canny) and RDS(SE). Thus we have not invested too much effort in optimizing various relaxed labels now.

Table 3.2: Comparing the importance of early and late supervision on BSDS500 testing dataset.

$R^{(1)}$	$R^{(2)}$	$R^{(3)}$	$R^{(4)}$	$R^{(5)}$	G	ODS	OIS	AP
					√	.762	.782	.766
√	√	√			√	.770	.795	.778
			√	√	√	.780	.801	.785
√	√	√	√	√	√	.787	.804	.817

(3) Pre-training with CEA demonstrate further gains for both RDS(Canny) and RDS(SE), reaching ODS=.790 and .792, respectively. In addition, we also evaluate the model pre-trained with CEA (without fine-tuning on the BSDS500 training set), which achieves ODS=.765. These results show the necessity and advantage of using a large-scale dataset.

Early supervision and late supervision

We have known the advantage of additional deep supervision. This experiment aims to examine whether all the intermediate supervision has the same importance or not. We employ the RDS(SE) method in an attempt to resolve this question. As shown in Table 3.2, we briefly divide two groups: early supervision and late supervision. The early supervision consists of $R^{(1)}$, $R^{(2)}$, and $R^{(3)}$, and the late supervision includes $R^{(4)}$ and $R^{(5)}$. In addition, the fuse-output supervision with G is necessary all the time. We train the model with early and late supervision separately and compare their effects. We can see that (1) compared with no intermediate supervision, using the late supervision achieves more boosts than the early supervision; (2) training with both early and late supervision outperforms any single way. These results show that all intermediate supervision provides useful and complementary information.

Comparisons with other approaches

Here we compare our RDS(SE)+CEA result against other leading methods on BSDS500. These methods can be categorized into non deep-learning and deep learning approaches, as seen in the upper part and lower part in Table 3.3). Precision/recall curves are illustrated in Figure 3.8. As far as we know, the recent work, MES [112], shows superior results for the non deep-learning approaches. On the other hand, HED [31], as an edge detector based on deep learning, leads the other methods, meanwhile retaining high efficiency. Our method, RDS, improves the ODS by 1 point and OIS by 0.6 point as compared with HED-latemerge. It is worth mentioning that HED has better average precision (AP), due to its late-merging step. However, we do not perform this optional late-merging step. Besides, HED further presents better results using multi-scale augmentation. Nevertheless, our results are

Table 3.3: Edge detection results on the BSDS500 dataset. Our approach is competitive with other state-of-the-art approaches. Note that, HED-multiscale augments the training images with three scales.

	ODS	OIS	AP
gPb-owt-ucm [126]	.726	.757	.696
Sketch Tokens [152]	.727	.746	.780
SCG [110]	.739	.758	.773
MS [150]	.74	.77	.78
SE-Var [127]	.746	.767	.803
OEF [151]	.749	.772	.817
MES [112]	.756	.776	.756
DeepNet [113]	.738	.759	.758
N4-Fields [155]	.753	.769	.784
DeepEdge [29]	.753	.772	.807
MSC [161]	.756	.776	.787
CSCNN [153]	.756	775	.798
DeepContour [30]	.757	.776	.790
HFL [154]	.767	.788	.795
HED-latemerge [31]	.782	.804	**.833**
HED-multiscale [31]	.790	.808	.811
RDS (ours)	**.792**	**.810**	.818

still competitive. In addition to the above quantitative results, we further show some qualitative image examples. In Figure 3.9, we illustrate some examples of our results.

3.4.3 Cross-dataset generalization

To investigate the generalization of one edge detector, it is necessary to conduct experiments on another dataset. Following the experimental setup in [30, 127], the NYUD dataset (v2) [162] is used as the cross dataset. With the model trained on BSDS500 training set, we evaluate the BSDS500 models on the NYUD dataset with its 654 testing images. Since these models are trained with color images, we only test the color images in the NYUD dataset. In Table 3.4, we compare our ODS results with SE [127, 163] and DeepContour [30]. To compensate for the relatively inexact ground truth in NYUD dataset, we increase the maximum tolerance (maxDist) allowed for correct matches of edge predictions to ground truth from .0075 to .011 [127]. We can see that, RDS achieves better cross-dataset generalization results, no matter what the maximum tolerance is.

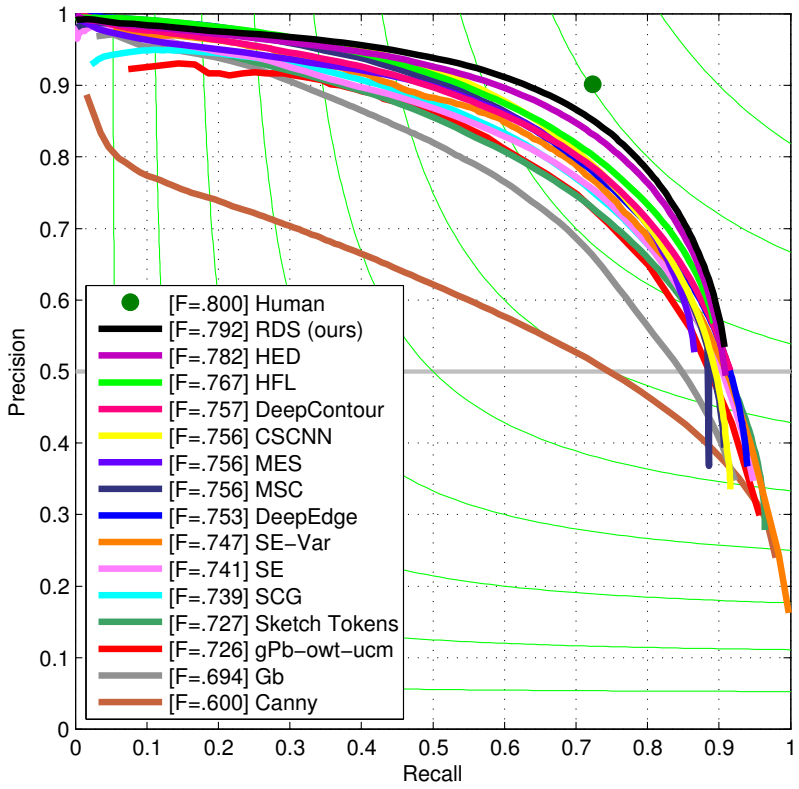


Figure 3.8: Precision and recall curves on BSDS500 test dataset. These methods are ranked according to their best F-score (ODS). Our method achieves superior result as compared with other top-tier performance.

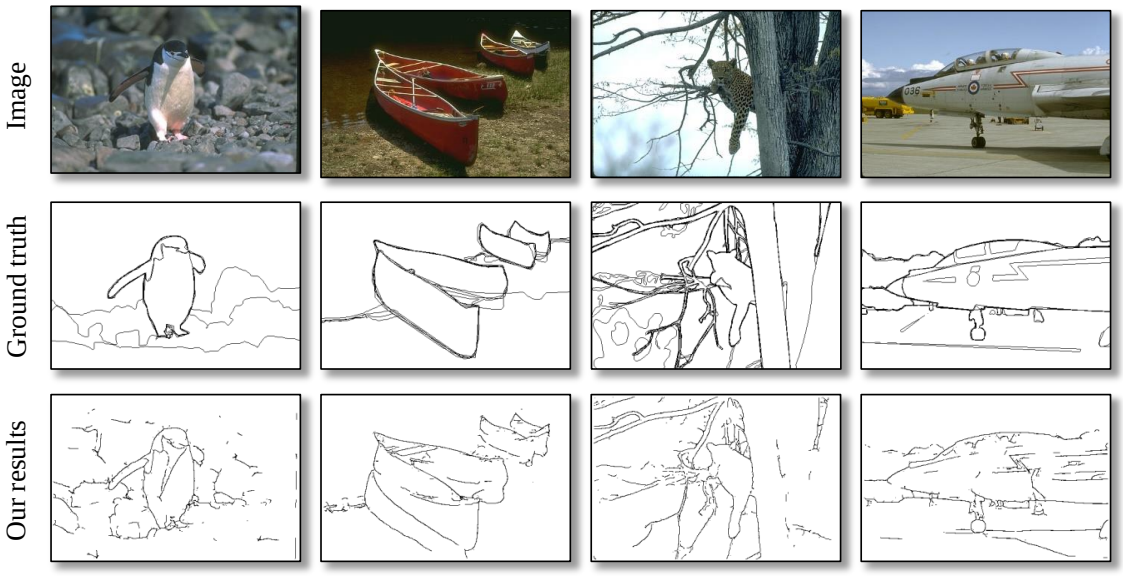


Figure 3.9: Illustration of five edge detection results. Our method can detect meaningful edges, even though they still have some differences from the ground-truth annotations.

Table 3.4: Cross-dataset generalization results (ODS F-score). The model trained on BSDS500 is used to evaluate the NYUD test set.

	maxDist=.0075	maxDist=.011
DeepContour [30]	.550	-
SE [127, 163]	.550	.64
RDS(SE)	.611	.627
RDS(SE) + CEA	**.655**	**.674**

3.4.4 Computational cost

Moreover, we report the computational cost of the proposed RDS method, including the training and testing stages. The experimental environment is Intel i7 CPU with 64GB RAM and NVIDIA K40 GPU. (1) Training stage: we need to extract the relaxed labels using off-the-shelf Canny or SE. They are both quite efficient detectors with about 15 and 2.5 FPS (frames per second), respectively. Next, we use the CEA data to pre-train the network with for 10,000 iterations, which takes about 10 hours on one K40 GPU. Finally, it spends less than one hour to train the model on the BSDS500 training set (200 images) for 25 epoches. (2) Testing stage: apart from computing the relaxed labels, our method takes about 500ms to predict the fusion-output edge map. Similar to HED, RDS has the similar order of magnitude in terms of computational speed.

3.5 Chapter Conclusions

In this chapter, we developed an edge detection method influenced by relaxed deep supervision (RDS) to guide the training of deep neural networks. Compared with the general deep supervision, RDS generated diverse supervisory signals to guide different intermediate layers. It can make the network have more focus on the false positives. Consequently, our method achieved considerable improvements, meanwhile retaining high efficiency. In addition, we proposed to pre-trained the model with coarse edge annotations (CEA) extracted from a large collection of segmentation annotations. This pre-training step can alleviate the lack of expensive edge annotations. Our results on the BSDS500 dataset demonstarted competitive performance (ODS=.792) with the state-of-the-art approaches. Another cross-dataset test indicated the promising generalization power of our method.

Future work. The work in this chapter has provided promising insights into efficiently exploiting diverse deep supervision to guide the network. Therefore, it is feasible to apply this relaxation strategy to other visual recognition tasks, such as object recognition and image segmentation. In addition, we will study theoretical analysis to provide more insights into diverse deep supervision.

Chapter 4

DeepIndex for Image Retrieval

The previous two research chapters focused on research about image-level and pixel-level classification. In this chapter, we turn our focus on the second research theme: *retrieval*, and answer how we can utilize deep visual representations for accurate and efficient image retrieval (RQ 3).

In a conventional image retrieval system, a number of local features are designed to describe key points in images. Then the well-known Bag-of-Words model is used to quantize the local features into visual words. In addition, an inverted index scheme is created to reduce the computational burden and query time. However, the local features are weak to distill high-level semantic concepts from the images. In the past few years, deep visual representations have shown powerful capabilities of bridging the semantic gap between low-level and high-level features. Inspired by this, in this chapter we exploit a DeepIndex framework for accurate and efficient image retrieval, by incorporating deep visual features into the inverted index scheme. DeepIndex can take advantage of the powerful discrimination of deep features and the fast search of the inverted index. To integrate more deep features, we further extend our framework to be a multiple DeepIndex. We find that the multiple DeepIndex can be viewed as a good attempt to couple different deep features. Extensive experiments on three benchmarks demonstrate the effectiveness of the proposed method. Our method is efficient in terms of memory cost and query time.

Keywords

Image retrieval, Convolutional neural networks, Bag of words, Inverted index

4.1 Introduction

Image retrieval is a practical and common application in the real world and therefore has triggered a massive amount of research activities in both multimedia and computer vision fields [19, 38, 39]. Bag-of-Words (BoW) is a traditional and efficient method in existing image retrieval systems, where local features, such as the SIFT [40] and color clues [41], are quantized to visual words with a pre-trained codebook. Then, similar to document retrieval [19, 39], an inverted index is built with the visual words to reduce computational and memory cost for scalable image search. Recently, Zheng *et al.* [164] performed low-level feature fusion with the SIFT and color features using a coupled inverted index framework. However, image retrieval remains challenging due to the well-known semantic gap between low-level image representations and high-level semantic concepts.

To bridge the semantic gap, recent works are dedicated to using more discriminative visual features learned in deep neural networks. The work of Wan *et al.* [42] finds that a deep CNN model pre-trained on a large dataset can be transferred for new content-based image retrieval (CBIR) tasks and that similarity learning can further boost the retrieval performance. Babenko *et al.* [45] focus on holistic descriptors where the whole image is mapped to a single deep feature vector. To extract richer regional features, Gong *et al.* [22] employed image patches at multiple scales, and then aggregated local patch responses at the finer scales via the VLAD [20] encoding. Yoo *et al.* [25] utilized multi-scale dense local CNN features to compute the Fisher Vector kernels. Zhang *et al.* [46] proposed a deep embedding method by incorporating the SIFT descriptor and CNN features. However, these prior works mainly focus on the accuracy and omit the importance of the retrieval efficiency, including memory cost and query time. To ensure both the accuracy and efficiency for image retrieval, we need to answer the question **RQ 3: How can we incorporate deep visual representations into the inverted index structure for accurate and efficient image retrieval?**

In this chapter, we propose a novel DeepIndex framework for accurate and efficient image retrieval, which can incorporate deep features into the inverted index scheme. We present a Bag-of-Deep-Features(BDF) model to cluster deep features into a number of visual words. In contrast to prior works that use resource-consuming algorithms for matching deep features, our DeepIndex(DPI) employs an efficient inverted index for fast image search, which can achieve competitive performance while reducing the computational time and memory cost. Furthermore, we extend DeepIndex by integrating different deep features and build a 2-D DeepIndex structure that consists of two kinds of variants: intra-CNN and inter-CNN. Intra-CNN uses two deep features from the same CNN architecture (*e.g.* AlexNet [157]), while inter-CNN selects the features from two different CNN architectures (*e.g.* AlexNet [157] and VGG [7]). The performance of inter-CNN is better than that of intra-CNN,

because the former one can fuse mutual deep features learned by different CNN models. However, intra-CNN is simpler and faster than inter-CNN. Notably, both intra-CNN and inter-CNN can serve as a solution to couple different deep features at an indexing level. Last but not least, we introduce a global image signature (GIS) into DeepIndex in order to enhance the query accuracy. In the experiments, we evaluate the proposed method on three datasets, where the results demonstrate its effectiveness and efficiency. Also, our results can compete with the state-of-the-art performance in terms of retrieval accuracy and computational cost.

The contributions of this work are as follows:

- We present a good attempt to incorporate deep features into the inverted index scheme and exploit a novel DeepIndex framework for accurate and efficient image retrieval.
- We present a 2-D DeepIndex variant that can be an alternative to effectively integrate different deep features at an indexing level.
- Our experiments show the promising advantages of leveraging deep visual representations to improve traditional image retrieval methods.

The rest of this chapter is structured as follows. Section 4.2 describes the bag-of-deep-features method. The DeepIndex framework is introduced in Section 4.3. The experimental results are reported in Section 4.4. Finally, Section 4.5 summarizes the conclusions.

4.2 Bag of Deep Features

Traditional image retrieval methods extract low-level features from images, such as SIFT and color descriptors, and employ them to construct the Bag-of-Features (BoF) or Bag-of-Words (BoW) model. However, few works have shown the utility of deep features into BoF. In this section, we present a Bag-of-Deep-Features (BDF) model, where visual words can be clustered based on CNN features.

4.2.1 Spatial patches

Generally, extracting only the global image feature is not discriminative enough for image retrieval, and may miss some local clues, such as spatial locations and contexts. Thus it is encouraged to extract rich regional features within finer scales. There are three common approaches to search for local regions in an image, including sliding window, region proposal and spatial pyramid.

Firstly, the sliding window approach is a quite common approach in object recognition and object detection that scans an image using windows of different scales,

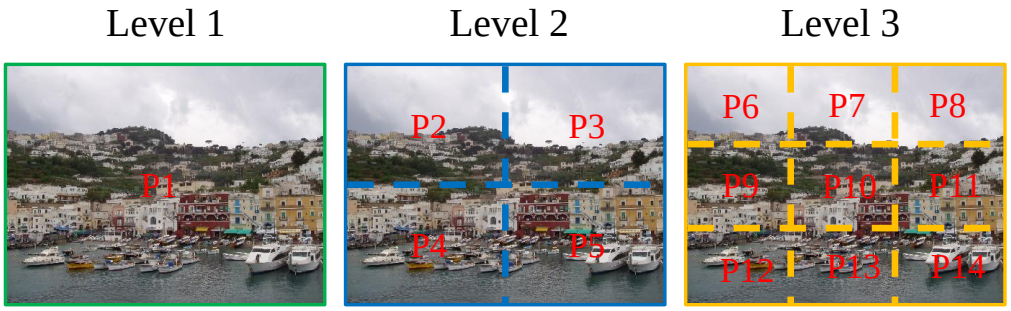


Figure 4.1: A three-level spatial pyramid. There are 14 image patches P_i in total, $i = 1, \ldots, 14$. we can describe each patch with a CNN feature.

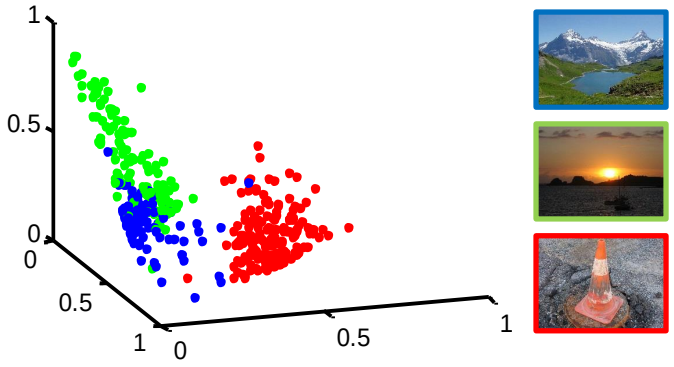


Figure 4.2: Visualizing the CNN features for three groups of images from the Holidays dataset [47]. Note that, each group has a few images and we show one of them in the right side. Each point in the 3D space represents an image patch, and its color corresponds to one of the three groups. We can see the clear separations of different groups.

locations, and aspect ratios. For example, Gong *et al.* [22] scan the whole image with two levels of overlapping windows that generates numerous local patches. Secondly, the region proposal approach can detect the objects of interest in images using fewer candidates than sliding windows. For object detection, RCNN [165] adopts the selective search into CNNs replacing sliding windows. Sun *et al.* [166] extract CNN features for object-like image patches with a region proposal detector. Thirdly, in contrast to the above two methods, the spatial pyramid approach [167] is an efficient way to preserve the spatial information in the images. Razavian *et al.* [6], augment the datasets by cropping and rotating images in several directions, and then use spatial search to divide the whole image into different levels of patches whose union covers the whole image.

Considering the above three approaches, we employ the spatial pyramid one to enrich the image representation because of its simplicity and efficiency. As seen in Figure 4.1, we partition one image into three levels: Level 1 contains only one patch P_1 that is the whole image; Level 2 divides the image into four patches (P_2, P_3, P_4, P_5) whose union covers the whole image; Level 3 consists of nine non-overlapping patches, denoted with P_6 to P_{14}. In total, there are 14 patches for one image, and their CNN features can be computed independently. In contrast to [6] which uses larger levels for training images than query images, we apply the same spatial levels for all images in the training and testing sets.

4.2.2 Feature extraction and quantization

The success of convolutional neural networks in image classification[157] has shown the strong efficiency and discriminative ability of learning deep visual features. It is common to extract the image representation from the fully connected (fc) layers [6, 168], because they are closer to class posteriors. However, it is still questionable about which fc layer is favorable for image retrieval. Different from prior works using either the first or the second fc feature, we aim to study the benefit of aggregating multiple fc features.

Specifically, we employ two common CNN architectures pre-trained on ImageNet [5]. The first one is AlexNet proposed by Krizhevsky *et al.* [157] in 2012. The second one is the VGG-19 model from Simonyan *et al.* [169] proposed in 2014. AlexNet has eight successive layers (5 convolutional layers, 3 fully connected layers), and the first and second fc layers are named by fc6 and fc7 respectively. VGG-19 consists of 16 convolutional layers and 3 fully connected layers and we name the first and second fc layers fc17 and fc18. To visually demonstrate deep features, we select three groups of images from the Holidays dataset[47]. The fc18 features of the image patches are extracted (4096-dimension). Then we map the features into a 3D space by the classical Multi-Dimensional Scaling (MDS). As seen in Figure 4.2, the separability of three groups is clear in the 3D space.

After feature extraction, we perform feature quantization based on the BoW model. Given an image I, x_i represents the feature vector of the i-th patch. After extracting the features of all image patches, we can learn a codebook with the k-means algorithm. Then the quantization function $q(\cdot)$ is used to map a patch feature x_i to its nearest visual word v_k in the codebook, *i.e.* $q(x_i) \mapsto v_k$. Note that, the codebooks associated with different fc features (*i.e.* fc6, fc7, fc17, fc18) are constructed independently. L_2 normalization is used to normalize the features.

4.3 DeepIndex

To reduce the retrieval time and memory cost, we propose the DeepIndex framework in which the inverted index is created based on the visual words. In addition, we can integrate multiple deep features with a multiple DeepIndex variant. Finally, the global image signature is utilized to increase the matching accuracy.

4.3.1 Single DeepIndex

We create an inverted index structure in which each entry corresponds to a visual word defined by the pre-computed codebook $\{v_i\}_{i=1}^{K}$. We represent the inverted index as $\mathcal{W} = \{W_1, W_2, \ldots, W_K\}$, where each entry W_i consists of a list

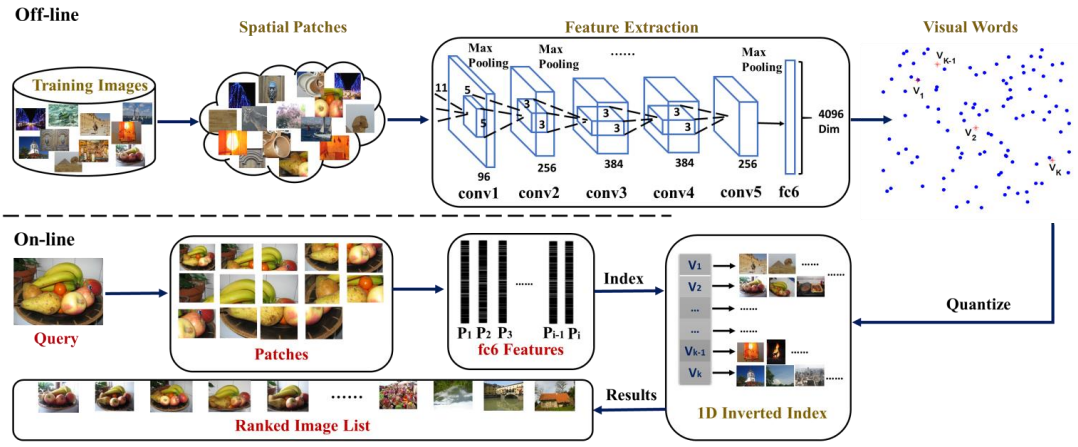


Figure 4.3: The flowchart of Single DeepIndex framework, including off-line and on-line stages. Here, the fc6 features from AlexNet are extracted to cluster the visual words, which are used to construct the inverted index structure. More details are in Section 4.3.

of indexed items, such as image ID, term-frequency (TF) score and other meta-data [39, 164, 170]. The indexed items following each entry W_i are counted as the retrieved candidates of the query feature. Therefore, the matching function $h_q(\cdot)$ for two deep features x and y can be expressed with

$$h_q(x, y) = \delta_{q(x),q(y)}, \tag{4.1}$$

where δ is the Kronecker delta response and $q(\cdot) \in [1, K]$. However, this matching function cannot weight the visual words according to their frequency. Generally, rare visual words are assumed to be more discriminative and should be assigned higher weights. Driven by the *tf-idf* scheme[19], we update the matching function

$$h(x, y) = \delta_{q(x),q(y)} \cdot idf(q(y))^2, \tag{4.2}$$

where $idf(i) = N/n_i$ and n_i is the number of images containing v_i.

For simplicity, we call the proposed indexing scheme single DeepIndex (1-D DPI) because it uses one kind of deep features. Specifically, we present four variants of 1-D DPI, including DPI_6,DPI_7,DPI_{17} and DPI_{18}, depending on which fully-connected feature of AlexNet and VGG-19 is used. The entire procedure of DeepIndex for image retrieval is illustrated in Figure 4.3. It takes the DPI_6 method as an example, but it is suitable for other deep features as well. There are two stages: off-line stage and on-line stage. During the off-line stage, we mainly cluster the codebook with the training image patches, and construct the inverted index structure. The on-line stage will query an image with its patch features and obtain similar images by searching the inverted index structure.

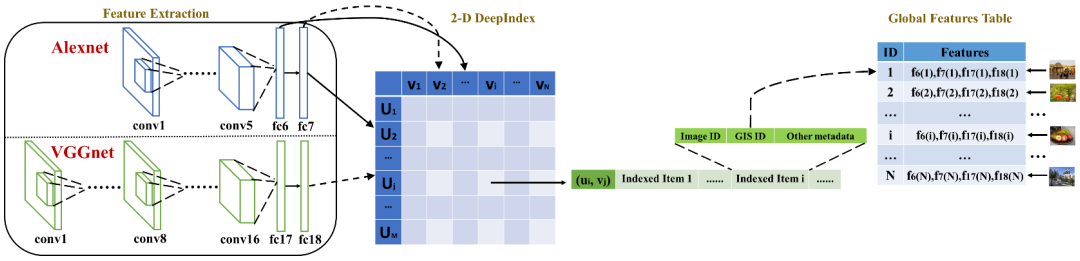


Figure 4.4: The framework of 2-D DeepIndex with deep features, including intra-CNN and inter-CNN. For intra-CNN, it uses the fc6 and fc7 jointly. For inter-CNN, the fc7 and fc18 are incorporated for indexing. Besides, the global image signature, being an additional clue in the indexed items, is stored in a table.

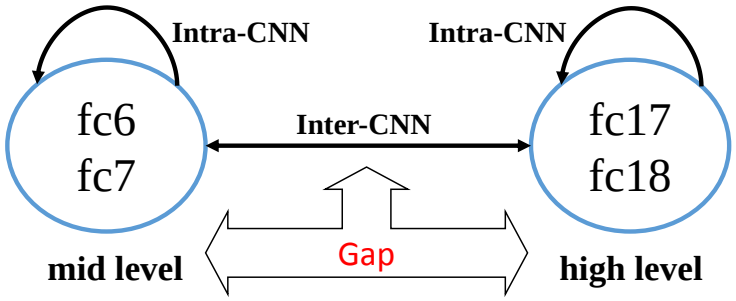


Figure 4.5: A conceptual comparison between Intra-CNN and Inter-CNN.

4.3.2 Multiple DeepIndex

Currently, most works mainly focus on comparing performance of different fully connected layers and choose a superior one. However, different neural layers imply different levels of abstraction of the image. Thus we utilize different deep features to compensate each other and to improve the retrieval accuracy. Based on this idea, we present an extended framework, called multiple DeepIndex (multi-DPI). The multi-index structure was first proposed in Babenko *et al.* [171]. It decomposes the SIFT descriptor into several blocks by product quantization. The multi-index structure is then organized around the codebooks of corresponding blocks. Similarly, Zheng *et al.* [164] built the coupled multi-index with traditional SIFT features and additional discriminative color names. Their results demonstrate that the feature fusion at the indexing level is better than the single indexing. Motivated by these works, we exploit a multiple DeepIndex that can incorporate multiple deep features into a multi-index structure. In this work, we take the two dimensional DeepIndex (2-D DPI) as an example.

To be specific, we denote $\mathcal{X} = [x^r, x^c]$ as a coupled deep features for a patch P_i, where x^r is extracted from one fc layer as the row indexing, and x^c comes from another fc layer as the column indexing. Then, two codebooks are pre-computed with different fc features separately, *i.e.* $\mathcal{U} = u_1, u_2, \ldots, u_M$ and $\mathcal{V} = v_1, v_2, \ldots, v_N$, where M and N are codebook sizes. The 2-D DPI structure contains $M \times N$

entries, where $\mathcal{W} = W_{11}, W_{12}, \ldots, W_{ij}, \ldots, W_{MN}, i = 1, 2, \ldots, M, j = 1, 2, \ldots, N$. After building the 2-D DPI, each feature tuple like $\mathcal{X} = [x^r, x^c]$ can be quantized into a visual word pair (u_i, v_j) based on the codebooks $\mathcal{U}$ and $\mathcal{V}$, where u_i and v_j are the nearest centroids to x^r and x^c, respectively. Similar to the 1-D DPI, additional clues (*e.g.* image ID and other meta-data) related to the feature tuple $\mathcal{X}$ are saved in the corresponding entry W_{ij}.

Given two feature tuples $\mathcal{X} = [x^r, x^c]$ and $\mathcal{Y} = [y^r, y^c]$, the matching function for 2-D indexing can be rewritten by

$$h(\mathcal{X}, \mathcal{Y}) = \delta_{q^r(x^r), q^r(y^r)} \cdot \delta_{q^c(x^c), q^c(y^c)} \cdot idf^2, \tag{4.3}$$

where $q^r(\cdot)$ and $q^c(\cdot)$ present two different quantization functions. Notice that, a right match is valid only if the two features tuples are similar in both row and column indexing. In this way, the 2-D DeepIndex can enhance the matching strength so as to improve the retrieval accuracy.

Moreover, we define two methods for selecting fc features, named **intra-CNN** and **inter-CNN**. (1) The Intra-CNN method uses two fc layers from the same CNN architecture. As the two black solid lines seen in Figure 4.4, fc6 activation is taken as column indexing, and the fc7 activation serves as row indexing. We can construct two Intra-CNN members: $\text{DPI}_{6,7}$ and $\text{DPI}_{17,18}$. (2) The Inter-CNN method chooses two fc layers coming from two different CNN architectures. For example, the two black dash lines in Figure 4.4, fc7 in Alexnet and fc18 in VGG-19 can serve as column and row indexing respectively. In total, we can have four Inter-CNN members, including $\text{DPI}_{6,17}$, $\text{DPI}_{7,17}$, $\text{DPI}_{6,18}$ and $\text{DPI}_{7,18}$.

We provide more insights into Intra-CNN and Inter-CNN in Figure 4.5. By comparing the depth of AlexNet and VGG-19, we categorize fc6 and fc7 as mid-level features, and fc17 and fc18 as high-level features. Intra-CNN is simpler to build than Inter-CNN. However, Inter-CNN can be viewed as a solution to bridge the gap between mid-level and high-level deep networks. More comparison and analysis about intra-CNN and inter-CNN is reported in the experiments.

4.3.3 Global image signature

To further improve the matching accuracy in the inverted index structure, we employ an additional discriminative feature to constrain the matching condition and filter out false matches, which is called the 'signature'. The most popular one is the hamming embedding signature [47] that uses a 64-D binary signature for each SIFT descriptor, and stores it in the meta-data of the inverted items. Also, Zheng *et al.* [164] use the hamming embedding for SIFT features and generate another signature for color names. The discrimination of deep image features has been demonstrated

in existing works [6, 42, 172], for example, only one fc feature vector extracted from the whole image can achieve desirable results for many tasks.

In this work, we propose to use this global deep feature as an additional signature for DeepIndex, called global image signature(GIS). Although the spatial patches already contains global feature representation at Level 1, they are used to enrich the representations of images and exploit more features at local regions. In addition, all the patch features are clustered into visual words and quantized to another space that is different from the original feature space. Thus, it is not a redundant process to use the global feature again. Since GIS is quite efficient, all the patches in one image can share the same GIS. We store all GIS features in a Global Features Table and search them by the GIS ID stored in the indexed items, as seen in Figure 4.4.

We compute the similarity of two GIS features with the root feature process [166, 173]. Specifically, we obtain the root feature by first L_1 normalizing the feature vector and then computing the square root per dimension. The distance $d(x, y)$ is computed using the Hellinger kernel $S(x, y) = \sum_{i=1}^{m} \sqrt[2]{x_i y_i}$:

$$d(x, y) = 2 - 2 \cdot S(x, y).$$

Then we take GIS into DeepIndex, and add this distance to update the matching score. For 1-D DeepIndex, given two patch features x and y, we can update Eq. 4.2

$$h(x, y) = \delta_{q(x),q(y)} \cdot idf^2 \cdot c(x, y), \tag{4.4}$$

where $c(x, y) = exp(\alpha \cdot d(gis(x), gis(y)))$; $gis(\cdot)$ returns the corresponding global image feature of the current image patch; α adjusts the GIS matching strength. For 2-D DeepIndex, there are two feature tuples, its matching function becomes

$$h(\mathcal{X}, \mathcal{Y}) = \delta_{q^r(x^r),q^r(y^r)} \cdot \delta_{q^c(x^c),q^c(y^c)} \cdot idf^2 \cdot c(\mathcal{X}, \mathcal{Y}), \tag{4.5}$$

where $c(\mathcal{X}, \mathcal{Y}) = c(gis(x^r), gis(y^r)) \cdot c(gis(x^c), gis(y^c))$. We find that GIS is an efficient global constraint and can compensate for patch matching. Finally, two patches can be matched only when their visual words are identical and their GIS features are similar.

4.4 Experiments

We evaluate the proposed method on three datasets and conduct component analysis to verify its effectiveness. In addition, we present its computational complexity in terms of memory cost and query time.

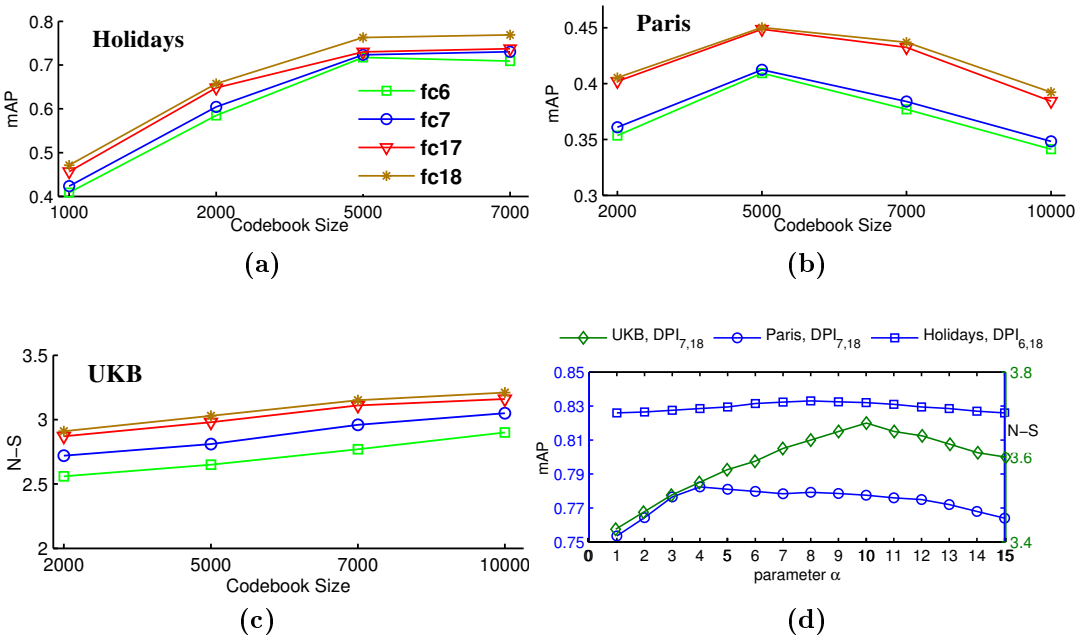


Figure 4.6: (a)-(c) Effect of codebook sizes on three dataset. The selected sizes are 5000, 5000 and 10000 for Holidays, Paris and UKB, respectively. (d) Influence of parameter α.

4.4.1 Datasets and metrics

Holidays [47] contains 1,491 vacation photographs corresponding to 500 groups. There are 500 queries, most of which have 1-2 ground truth images which have been rectified to a natural orientation. The performance is measured by mean average precision(mAP) over the provided queries (also seen in Section 2.4.3).

Paris [174] has 6,412 images obtained from Flickr. 55 images serve as queries. For each image and landmark, one of four possible labels is generated: good, ok, bad, and junk. The mAP is again used as the accuracy measurement.

UKB [175] includes 10,200 indoor photos of 2,550 objects(4 images per object). Each image is used to query the rest of the dataset in turn. The performance is reported by the average recall of the top four results, referred to as N-S score that is a number between 0 and 4 (also seen in Section 2.4.3) . But some works still use mAP to measure the performance on this dataset.

4.4.2 Results and discussion

Codebook size.

The visual words are clustered using features from the training images. To elevate the efficiency of k-means, we use the algorithm from Fast Library for Approximate

Table 4.1: Quantitative results on the 1-D DeepIndex and 2-D DeepIndex. Multiple assignment (MA) is used to increase the retrieval recall. We compare the performance of four 1-D DPI methods, two Intra-CNN methods and four Inter-CNN methods. The best results on the datasets are in boldface.

Method	Holidays (mAP)			Paris (mAP)			UKB (N-S)		
	MA=1	MA=3	MA=5	MA=1	MA=5	MA=10	MA=1	MA=5	MA=10
DPI_6	71.73	73.54	72.01	40.94	56.89	65.21	2.90	3.03	3.02
DPI_7	72.34	74.90	73.58	41.24	57.45	65.78	3.05	3.12	3.04
DPI_{17}	73.02	73.22	72.62	44.87	61.01	70.24	3.16	3.19	3.15
DPI_{18}	76.31	76.72	75.63	45.03	61.23	71.33	3.21	3.25	3.19
DPI_{6+7}	72.00	78.88	77.17	29.35	62.89	71.20	3.02	3.13	3.05
DPI_{17+18}	75.75	79.96	79.34	32.28	63.29	71.69	3.16	3.25	3.26
DPI_{7+17}	74.01	80.53	80.20	33.45	64.12	73.24	3.21	3.25	3.19
DPI_{6+17}	73.32	81.62	81.15	33.95	65.08	74.35	3.22	3.26	3.22
DPI_{7+18}	74.66	81.23	81.74	36.56	66.18	**75.35**	3.26	**3.37**	3.32
DPI_{6+18}	73.82	81.64	**82.38**	34.12	65.40	74.52	3.19	3.23	3.29

Nearest Neighbors(FLANN) [176]. We test four kinds of 1-D DeepIndex (*i.e.* DPI_6, DPI_7, DPI_{17} and DPI_{18}) to find proper codebook sizes. The results are shown in Figure 4.6. To balance the accuracy and efficiency, we set the codebook size K =5000, 5000 and 10000 for Holidays, Paris and UKB, respectively. It is noteworthy that the codebook sizes of deep features are much smaller than traditional BoW with local features, because the number of image patches is much smaller than the number of key points.

Ablation study of DeepIndex

We report ablation results of 1-D DPI and 2-D DPI on the three datasets in Table 4.1. First, we analyze the effect of multiple assignment (MA) [170] on the performance, which is a common technique when retrieving the inverted index items. When MA=1, it means that only the nearest inverted index item can be retrieved. In this case, we can see that the 2-D method is not better than the 1-D method because of the low recall. To further improve the recall, we can increase the multiple assignment (MA) [170]. In this way, the 2-D DPI can perform better than the 1-D DPI, which demonstrates the benefit of integrating different features.

Next, we can observe that the inter-CNN methods are better than the intra-CNN ones. The reason is that two deep features in intra-CNN are from the same CNN architecture, such as fc6 and fc7 in AlexNet, and their implicit relationships (*i.e.* fc6 is the input of fc7) may limit the learning of the 2-D inverted index. For simplicity, we call the fc6 and fc7 features as 'mid-level' descriptions and the fc17 and fc18 features as 'high-level' descriptions. As a result of the mutual compensation of mid-level and high-level features in inter-CNN, it can bridge the gap between different CNNs at the 2-D inverted index level and achieve superior retrieval accuracy. In details, $DPI_{6,18}$ obtains 82.38% mAP on Holidays; $DPI_{7,18}$ has 75.35% mAP on Paris; $DPI_{7,18}$ achieves 3.37 N-S score.

Figure 4.7: Retrieval results on the Holidays and UKB datasets. The 2-D DPI method can have more relevant retrieved candidates than the 1-D DPI.

Table 4.2: Effect of PCA Compression on the performance of DeepIndex.

Dimensions	Holidays (mAP)	Paris (mAP)	UKB (N-S)
4096	83.30	78.24	3.68
2048	84.11	79.45	3.72
1024	84.63	80.65	3.74
512	**85.65**	**81.24**	**3.76**
256	83.67	78.75	3.71
128	82.72	77.24	3.65

Moreover, we study the influence of the global image signature on 2-D DPI. We choose to test the superior methods on each dataset, as listed in Table 4.1. The parameter α in GIS ranges from 1 to 15 and the results are shown in Figure 4.6d. For Holidays, the GIS increases $\text{DPI}_{6,18}$ to 83.3% mAP when α is 8. Similarly, the result of $\text{DPI}_{7,18}$ for Paris reaches 78.24% mAP with $\alpha = 4$. Also, the $\text{DPI}_{7,18}$ method gets 3.68 N-S score on UKB with $\alpha = 10$. All these results show that GIS can help in providing a global constraint to enhance the matching accuracy. All the following results contain the GIS process. In addition to the quantitative evaluation, we show two queries from Holidays and UKB in Figure 4.7. It can be seen that the 2-D DPI method can retrieve more relevant images than the 1-D DPI.

Dimensionality reduction

The deep visual features we use have 4096 dimensions. To reduce the feature dimensionality, we further study the influence of feature compression for deep features. Specifically, we conduct PCA compression for the 4096-Dimension deep features. Also, the GIS is compressed by PCA. We report the results in Table 4.2. Interestingly, when the dimension decreases to 512, we achieve the best results on all the

Table 4.3: Comparison results with other methods on three datasets.

Groups	Methods	Holidays (mAP)	Paris (mAP)	UKB (N-S and mAP)
Non-CNN	[41]	78.90	-	3.50
Non-CNN	[177]	80.86	-	3.60
Non-CNN	[170]	81.30	-	3.42(87.8)
Non-CNN	[178]	82.20	78.20	-
Non-CNN	[179]	83.90	-	3.54(90.7)
Non-CNN	[164]	84.02	-	3.71(94.7)
Non-CNN	[178]	88.00	80.50	-
CNN	[45]	74.70	-	3.43
CNN	[166]	79.00	-	3.61
CNN	[22]	80.20	-	-
CNN	[6]	84.30	79.50	-(91.1)
CNN	[42]	-	**86.83**	-
CNN	Ours	85.65	81.24	3.76
SIFT-CNN	[46]	85.30	-	3.79
SIFT-CNN	[46]	**88.08**	-	**3.85**

three datasets. Even in the extreme case where the dimensionality is down to 128, it can still obtain desirable results compared with many of SIFT-based methods. This implies that the original deep feature is discriminative while containing some redundant information for the retrieval tasks. Feature compression can help refine the feature representation and maintain the high performance. A similar observation is also suggested in other related works [42, 45].

4.4.3 Comparison with other methods

We compare our results with other state-of-the-art methods. We simply divide them into three groups: CNN methods, Non-CNN methods and SIFT-CNN methods. We do not consider and perform various post-processing algorithms, such as query expansion, spatial verification and graph fusion. For CNN methods, we do not consider fine-tuning for specific tasks. For fairness, we compare the results with other methods that exclude the post-processing and fine-tuning steps.

The whole comparison is listed in Table 4.3. For Holidays, our proposed method (85.56%) exceeds other CNN-based methods, and is in competition with the best results [178] and [46]. In the work by Tolias *et al.* [178], their representation takes several millions of features per image which is not scalable to large datasets. In Zhang *et al.* [46], they use both the SIFT descriptor and CNN features to increase the accuracy. On the Paris dataset, our result (81.24%) outperforms most methods, except [42] that introduces the similarity learning algorithm into deep learning. In UKB, our method (3.76) is better than the coupled multi-index method [164], and

Table 4.4: Memory cost (bytes) and query time (seconds) for one image on Holidays.

Complexity	[46]	1-D DPI	2-D DPI
ImageID	4×500	4×14	4×14
Signature	10.18KB	512×4	$512 \times 4 \times 2$
Total Memory	12.13KB	2.06KB	4.06KB
Query Time	2.32	0.25	0.45

is also competitive with [46].

Complexity analysis

Although our results are inferior to those of [46], our method is more efficient in terms of memory cost and query time. As seen in Table 4.4, we compare the computing complexity of DeepIndex with [46] on Holidays. Our experimental environment is Intel i7 CPU at 2.67Ghz with 12GB RAM and NVIDIA GTX 660 with 2GB GRAM. Zheng *et al.* [46] extracts 500 SIFT keypoints for each image. Considering the memory cost per image, both the 1-D DPI (2.06KB) and 2-D DPI (4.06KB) are more efficient than [46] that requires significantly more memory for the SIFT descriptors. Also, our average query time is shorter, *i.e.* less than 0.5 seconds compared to 2.3 seconds for [46]. These results are consistent with our motivation of exploiting an accurate and efficient image retrieval method.

4.5 Chapter Conclusions

In this chapter, we exploited the DeepIndex framework for accurate and efficient image retrieval that could incorporate deep features into the inverted index scheme. In addition, we integrated multiple deep features with the multiple DeepIndex which was able to bridge different deep representations at an indexing level. Experimental results showed that our method achieved competitive performance on the Holidays, Paris and UKB datasets, while retaining the retrieval efficiency in terms of memory cost and query time.

Future work. One the one hand, a straightforward improvement is to further extend multiple DeepIndex by using more deep features, *e.g.* 3-D DeepIndex and so on. But we should note that it will increase the computational cost. On the other hand, it is encouraged to integrate some traditional retrieval techniques with DeepIndex, such as query expansion and late fusion. We believe that deep learning approaches would be compatible with other traditional algorithms.

Chapter 5

Image-Text Matching for Cross-modal Retrieval

In the previous chapter, we have started the research theme on image retrieval. Nowadays, cross-modal retrieval using vision and language has drawn increasing attention due to the availability of large-scale multimedia data. This observation motivates our research on how we can develop an efficient deep matching network for cross-modal retrieval (RQ 4).

A major challenge in matching visual and textual representations is that they typically have different modality-specific features based on individual feature encoders. Existing approaches take advantage of the power of deep models to learn a discriminative embedding space where related images and texts can be gathered, however, few of them consider maintaining the model complexity. In this chapter, we introduce an efficient approach to couple visual and textual features based on a recurrent residual fusion (RRF) block. RRF adapts the residual learning to the recurrent mechanism, so that it can recursively improve feature embeddings while retaining the shared parameters. In addition, a fusion module is used to integrate the intermediate recurrent outputs and generate a more powerful representation. In the matching network, RRF can be viewed as a feature enhancement component that gathers visual and textual representations into a more discriminative embedding space. Moreover, we present a bi-rank loss function to enforce separability of the two modalities in the embedding space. In the experiments, we verify the effectiveness of the proposed approach on two multi-modal datasets where it can achieve competitive performance with the state-of-the-art approaches.

Keywords

Cross-modal retrieval, Image-text matching, Deep neural networks, Ranking loss

5.1 Introduction

The matching problem between images and texts [49, 50, 51, 52, 53, 54] is one of the most important tasks in the area of multi-modal information retrieval. This task remains challenging due to the heterogenous representations and the cross-modal gap between vision and language, which is also a core issue for other multi-modal applications such as image captioning [55, 56], visual question answering [57, 58] and zero-shot recognition [59, 60].

A main line of research for multi-modal matching is to learn a latent embedding space where related images and texts can be unified into similar representations [63, 180, 181]. Previously, Canonical Correlation Analysis (CCA) [61] has been a well-known and representative embedding technique for decades. CCA can learn a linear transformation to project two modalities into a common space where their correlations are maximized. Also, some extensive techniques are applied to the classical CCA, including randomized CCA [182], nonparametric CCA [183], and kernel CCA [184].

Driven by the successful developments of deep learning, more and more works extract powerful visual and textual features from deep neural networks. For example, recent works [50, 51, 52, 53, 55, 185] employ convolutional neural networks (CNNs) [4] to extract deep image features, and learn descriptive text features based on recurrent neural networks (RNNs) [186]. Then they can incorporate deep learning features with traditional embedding techniques (*e.g.* CCA and its variants). In addition, extensive research efforts [49, 62] have been dedicated to directly learning a deep CCA model that can be end-to-end trainable. Instead of using CCA, recent works developed a variety of multi-modal deep neural networks to model the matching task [52, 53, 55, 76, 181]. Nevertheless, the performance of multi-modal matching is still far from competitive with that of an intra-modal task like image retrieval. In addition, most of prior works are inefficient with respect to the model complexity. Regarding this task, we aim to address **RQ 4: How can we build a deep matching network to unify images and texts into a more discriminative space without increasing the number of network parameters?**

In this chapter, we propose a deep matching network using recurrent residual fusion (RRF) as building blocks for improving feature embeddings. Our new matching network (RRF-Net) has two branches for representing images and texts, respectively. Each branch consists of four fully-connected layers that are used to project a source representation into a common latent space. The proposed RRF building block is introduced in the third fully-connected layer of the two branches. Specifically, RRF integrates three main components to improve the feature embedding procedure in the network.

The first component in RRF is inspired by the residual learning in ResNet [10]. We add an identity connection to sum the input of a fully-connected layer with its output. This component enables the fully-connected layer to learn residual embedding features and provides high performance. Secondly, RRF employs a recurrent mechanism with the residual learning by adding a recurrent connection whose direction is inverse to the identity connection. As the parameters of the fully-connected layer are shared during the recurrent procedure, RRF is able to recurrently improve feature embeddings while retaining the parameters. The third component is the use of a fusion module, which aims to integrate intermediate recurrent outputs to generate a more powerful fused output. The fusion module facilitates making use of more complementary information in the intermediate layers and explicitly transferring their effects to the final output. We provide two efficient fusion modules: sum-pooling fusion and convolutional fusion.

Moreover, we present a bi-directional rank loss function (called bi-rank loss), including image-to-text rank loss and text-to-image rank loss, to train the proposed RRF-Net. The original bi-directional loss function only considers the cross-modal relationships between images and texts. Instead, the bi-rank loss can preserve not only cross-modal relationships, but also intra-modal relationships (*e.g.* image-image and text-text). As a result, it is able to enforce separability of the two modalities in a unified embedding space. Extensive experiments show remarkable improvements of the bi-rank loss over the original bi-directional loss.

The contributions of this work are as follows:

- We introduce a new RRF building block and adapt it to a deep matching network. RRF provides promising insights into efficiently improving the co-embedding between images and texts.
- We present a bi-rank loss function to train the RRF-Net for better ensuring the cross-modal and intra-modal constraints in the unified space.
- The experimental results demonstrate that our approach achieves competitive performance on public benchmarks for image-to-text and text-to-image retrieval.

The rest of this chapter is structured as follows. Section 5.2 describes the proposed recurrent residual fusion method. The image-text matching network is presented in Section 5.3. The experimental results are reported in Section 5.4. Section 5.5 summarizes the conclusions.

5.2 Recurrent Residual Fusion

We describe the details of the RRF block (Figure 5.2) with three components: an identity connection, a recurrent connection and a fusion module.

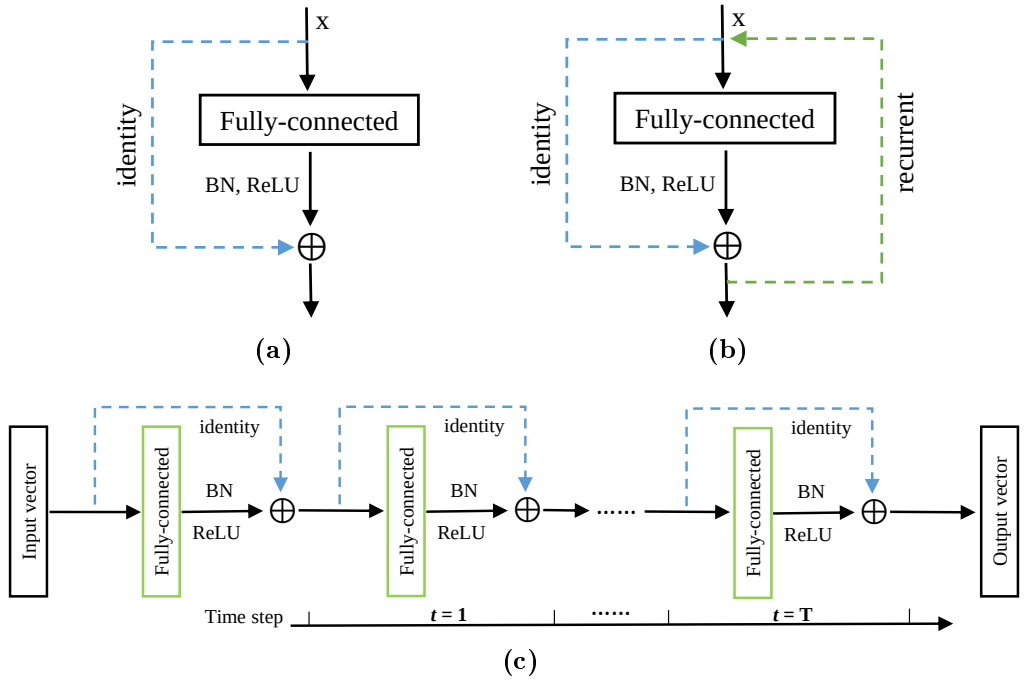


Figure 5.1: Illustration of basic building blocks. (a) An identity mapping (blue) is added to a fully-connected layer. (b) A recurrent connection (green) is introduced that uses the current output state to update the next input state. (c) We unfold the building block in (b) over recurrent steps, resulting in a very deep network. All fully-connected layers (in green) share the same parameters. t represents the recurrent step, ranging from 1 to T.

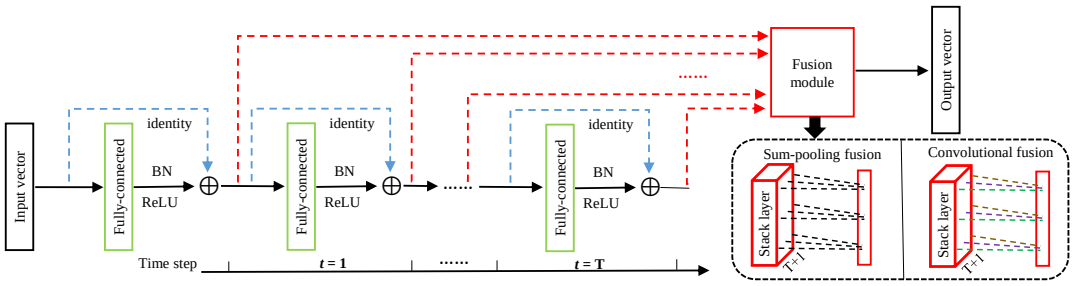


Figure 5.2: The RRF building block. Built upon recurrent residual learning, we develop a fusion module (in red) to integrate the intermediate output vectors from each recurrent step. The final output vector learns more information than the original output in Figure 5.1c. Specifically, there are two types of fusion modules: the sum-pooling fusion simply fixes equal weights, but the convolutional fusion can learn adaptive weights (drawn in different colors).

Identity connection

The basic building block in ResNet [10] adds an extra identity mapping with the traditional non-linear transformations based on convolutional layers. Instead of using a convolutional layer, we develop an identity connection on top of a fully-connected layer. As can be seen in Figure 5.1a, our residual block consists of a

fully-connected layer (FC), a batch normalization layer (BN) [135] and a Rectified Linear Unit (ReLU) layer [4]. The input and output channels of the FC layer should have the same size. The computation can be presented by

$$h(x) = \sigma(f(x)) + x, \tag{5.1}$$

where x and $h(x)$ represent the input and output of the building block, respectively. $f(\cdot)$ indicates the FC layer, and $\sigma(\cdot)$ is the ReLU activation function.

Recurrent connection

RNNs [186, 187] are proposed for modeling sequential contexts in tasks like machine translation and image captioning. We seek to introduce the recurrent mechanism to the residual learning block. As can be seen in Figure 5.1b, we add a recurrent connection whose direction is inverse to the identity connection. As a result, the current output can be used as the next input, and then the next input continues adding an identity mapping to the residual mapping to compute the next output. As the fully-connected parameters are shared during the recurrent procedure, the whole structure is able to become much deeper without consuming more parameters. We unfold the structure across recurrent steps in Figure 5.1c. Assume that there are T recurrent steps in total, so the structure has $T+1$ layers inside, and each layer uses the same parameters as drawn in green. Mathematically, the recurrent residual procedure is formulated via

$$x_t = h(x_{t-1}) \tag{5.2}$$

$$f(x_t) = w \cdot x_t + b \tag{5.3}$$

$$h(x_t) = \sigma(f(x_t)) + x_t \tag{5.4}$$

where $t = 1, \ldots, T$ and $x_0 = x$ is the original input vector. x_t is updated by the previous output $h(x_{t-1})$ which adds the residual mapping $f(x_t)$ with the identity mapping x_t. The parameters w, b indicate the shared weights and bias in the fully-connected layer. Note that the parameters used in the BN layer are not shared during recurrence, however, the number of these parameters is much lower than that of the total parameters in the model. The input vector can be refined over recurrence while maintaining the efficiency due to tying the shared parameters. Finally, the output vector learns to be a more discriminative representation.

Fusion module

Typically, a plain network can learn multiple representations from bottom layers to top layers, however, the final output only connects with the topmost layer. For example in Figure 5.1c, the output vector is directly affected by the result at the last recurrent step. Although the recurrent procedure can transfer the effects of

intermediate layers to the final output, their effects are implicit and indirect compared with the topmost layer. Therefore, we develop a fusion module to explicitly aggregate the intermediate layers involved in the recurrent procedure. Figure 5.2 highlights the fusion module in red. Specifically, several new side branches (dot lines in red) are generated from intermediate layers and then merged into a fusion module. As the intermediate layers have the same dimension, the fusion module is able to integrate them without adding extra new transition layers. In a fusion module, $T+1$ side outputs are stacked as a layer S. S is of size $1 \times N \times (T+1)$, where N is the dimension of each side output. Based on S, we employ two fusion methods to compute a fused output vector: sum-pooling fusion and convolutional fusion.

(1) *Sum-pooling fusion.* As can be seen in the right bottom of Figure 5.2, it computes a summation across the feature channels of the stack layer S. The fused output vector S_{sum} is represented by

$$S_{sum} = \sum_{i=0}^{T} h(x_i) = \sum_{i=0}^{T} \sigma(f(x_i)) + x_i. \tag{5.5}$$

The sum-pooling fusion supposes that each side branch has the same importance without learning any weights.

(2) *Convolutional fusion.* Normally, each side branch (or intermediate layer) may influence the output vector with different importance. Therefore, we use a convolutional layer in the fusion module to learn adaptive weights (or importance) for better fusing side branches. The filter f in the convolutional layer has $1 \times 1 \times (T+1)$ dimensions. S is convolved by f to generate the fused vector S_{conv}:

$$S_{conv} = w_f * S + b_f \tag{5.6}$$

where w_f and b_f represent the weights and bias, respectively. It is worth noting that these additional parameters (*i.e.* $T+1$) are a minimal increase to the total number of parameters used in a deep network.

In summary, the RRF block incorporate the above three components and inherits their individual advantages. It acts as a feature enhancement to the power of the input vector and aims to generate a more informative output vector. Unlike other deep fusion networks in which different layers are aggregated, RRF delves into improving the discrimination of one layer over recurrence. Also, RRF is a general structure that can be potentially applied to many existing layers in a deep network.

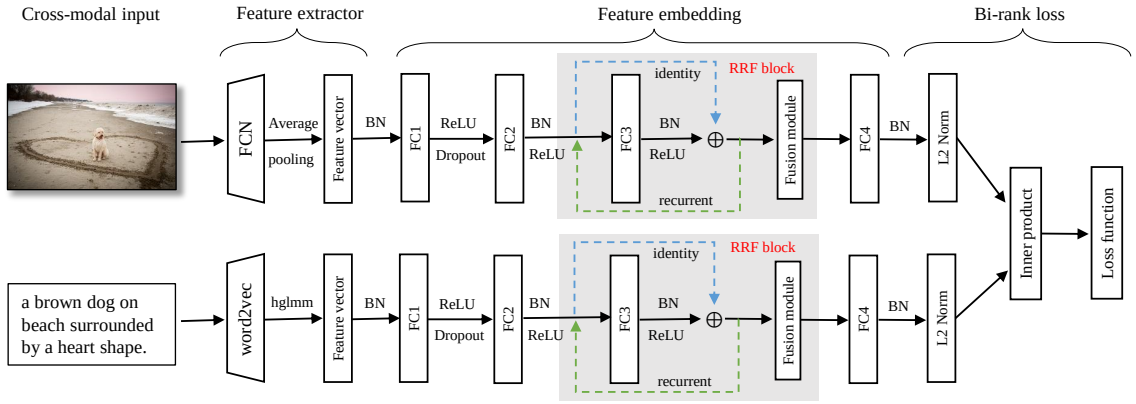


Figure 5.3: The overview architecture of the proposed RRF-Net for image and text matching. This two-branch network comprises three key steps: (1) feature extractors are used for capturing visual and textual representations. (2) Four fully-connected layers (from FC1 to FC4) in two branches are used for learning feature embeddings. Importantly, a RRF block is built upon the FC3 layer to improve its embedding capability. The details inside the RRF block are described in Figure 5.2. (3) After normalizing the two output vectors and computing their inner product, we employ a bi-rank loss to train the entire network.

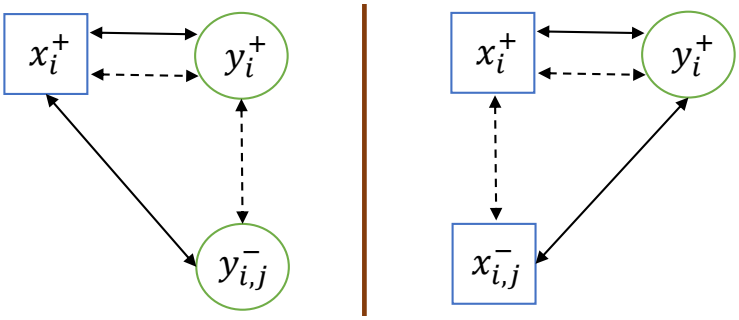


Figure 5.4: Illustration of computing the bi-rank loss that are used to train the RRF-Net. Left: image-to-text rank loss; Right: text-to-image rank loss. x and y indicate the image and text, respectively.

5.3 Matching Network

In this section, we present a new deep matching network called RRF-Net, where the RRF blocks are introduced to improve latent embeddings between images and texts. Figure 5.3 illustrates the architecture of the network, and we will describe its three key steps as below.

5.3.1 Feature extractor

As a common practice, we capture visual and textual features using off-the-shelf feature extractors. Taking these features as input instead of the raw data can ease the training procedure and lead to fast convergence.

Image feature extractor: we choose the powerful ResNet-152 [10] pre-trained on ImageNet [5]. To efficiently extract dense region representations, CNN models are first recast to fully convolutional networks (FCNs) [26]. Given one input image, we

set its smaller side to 512 and isotropically resize the other side. The last max-pooling layer in the ResNet-152 model is averaged to generate a 2048-dimensional visual feature vector.

Text feature extractor: we employ the Hybrid Gaussian-Laplacian mixture model (HGLMM) [51] which is built based on word2vec model [188]. For each sentence, HGLMM computes one 18000-dimensional vector with 30 centers (*i.e.* 300*30*2). To decrease the memory cost [53], we also use PCA to reduce the dimension from 18000 to 6000. Finally, the 6000-dimensional vector acts as a powerful feature.

5.3.2 Feature embedding

To learn a discriminative embedding space, we develop four fully-connected layers on top of the two feature extractors. Their channels are {2048, 512, 512, 512} in both branches. Note that the parameters in each branch are unshared as they are responsible for different modalities. Specifically, ReLU is used for FC1, FC2 and FC3, but not for FC4. A dropout layer with 0.5 probability is added after FC1, and other FC layers are regularized with batch normalization (BN) [135].

The core component in each branch is the FC3 layer as it introduces the RRF building block. RRF increases the FC3 layer to depth $T + 1$ while retaining the parameters. Consequently, it facilitates deeper learning of latent embeddings and further unifies the visual and textual representations. Notably, the BN layer after FC3 learns unshared parameters during recurrent steps, however, these few extra parameters raise little cost to the entire network. Moreover, a RRF block can be imposed on any fully-connected layer. But in the current architecture, FC3 is more suitable than other layers. Also, we observe that using only a RRF block seems sufficient for enhancing feature embeddings.

5.3.3 Bi-rank loss

After unifying images and texts into a joint embedding space, the next step is to compare their similarities. Given an image x and a text y, their FC4 embedding features are denoted as $f(x)$ and $f(y)$. We compute the similarity $s(x, y)$ with the cosine distance

$$s(x, y) = 1 - \frac{f(x) \cdot f(y)}{||f(x)|| \cdot ||f(y)||}. \tag{5.7}$$

Smaller distances indicate larger similarities. To train the network, we define a bi-rank loss function, including image-to-text and text-to-image rank loss.

Image-to-text rank loss

For an input image x_i^+, its matching text is represented by y_i^+. To obtain more representative non-matching pairs, we collect the top N most dissimilar texts in each mini-batch as a negative text set Y_i^-. Then, we compute the triplet rank loss for $\{x_i^+, y_i^+, y_{i,j}^-\}$, where $y_{i,j}^- \in Y_i^-$ and $j = 1, 2, \ldots, N$. First, the matching cross-modal similarity $s(x_i^+, y_i^+)$ should be larger than any of the non-matching cross-modal similarities $s(x_i^+, y_{i,j}^-)$. Second, we further constrain the intra-modal similarity $s(y_i^+, y_{i,j}^-)$ from exceeding $s(x_i^+, y_i^+)$. This loss can ensure both the cross-modal (*i.e.* image-text) and the intra-modal (*i.e.* text-text) relations. An example is shown in the left of Figure 5.4. Finally, this loss function is expressed with

$$\begin{aligned} l_{i2t} = \sum_{j=1}^{N} \Big(& \alpha_1 \max[0, s(x_i^+, y_i^+) - s(x_i^+, y_{i,j}^-) + m] \\ & + \alpha_2 \max[0, s(x_i^+, y_i^+) - s(y_i^+, y_{i,j}^-) + m] \Big), \end{aligned} \tag{5.8}$$

where α_1 and α_2 measure the importance of the two terms. m is a parameter to adjust the margin between the two distances.

Text-to-image rank loss

Given one text y_i^+, we collect its top N most dissimilar images in each mini-batch as a negative image set X_i^-. Similarly, we compare the similarities within each triplet $\{y_i^+, x_i^+, x_{i,j}^-\}$, where $x_{i,j}^- \in X_i^-$. Their relations can be seen in the right of Figure 5.4. The text-to-image rank loss is as follows

$$\begin{aligned} l_{t2i} = \sum_{j=1}^{N} \Big(& \alpha_1 \max[0, s(y_i^+, x_i^+) - s(y_i^+, x_{i,j}^-) + m] \\ & + \alpha_2 \max[0, s(y_i^+, x_i^+) - s(x_i^+, x_{i,j}^-) + m] \Big), \end{aligned} \tag{5.9}$$

Full objective

The objective is to minimize the total loss by adding the two rank loss functions

$$l(x_i^+, y_i^+, X_i^-, Y_i^-) = \frac{\beta_1 l_{i2t} + \beta_2 l_{t2i}}{N}, \tag{5.10}$$

where the weights β_1 and β_2 control the importance of the two terms of one-directional rank loss. Compared with [53] which searches for extra positive intra-modal pairs, our bi-rank loss directly uses the negative intra-modal pairs and needs a minimal amount of additional computations.

Table 5.1: Evaluation results on the proposed RRF-Net on the Flickr30K test set. Higher R@K is better. All of the four RRF-Net models outperform the baseline. When $T = 3$, it obtains better performance (in bold).

Method	Image to Text		Text to Image	
	R@1	R@5	R@1	R@5
Baseline	45.0	75.5	33.6	66.5
RRF-Net, T=1	46.4	76.1	34.3	67.3
RRF-Net, T=2	46.9	76.8	34.8	67.7
RRF-Net, T=3	**47.6**	**77.4**	**35.4**	**68.3**
RRF-Net, T=4	46.2	76.6	35.1	67.6

5.4 Experiments

In this section, we evaluate our approach and report its results on two widely-used multi-modal datasets for bi-directional image-text retrieval.

Datasets. (1) Flickr30K [189]: following the dataset splits in [190], we use 29,783 training images, 1,000 validation images and 1,000 test images. Each image is annotated by five sentence-level texts. It has $29,783 * 5 = 148,915$ training pairs. (2) MSCOCO [117]: it consists of 82,783 training images and 40,504 validation images. 1,000 test images are selected from the validation set [190]. We choose five sentences for each image and generate $82,783 * 5 = 413,915$ training pairs.

Implementation details. The hyper-parameters are evaluated on the validation set of each dataset. To be more specific, the parameters $\{\alpha_1, \alpha_2, \beta_1, \beta_2\}$ are set with $\{1, 0.5, 2, 1\}$, and $m = 0.1$. Following [53], the number of non-matching pairs is $N = 50$. We trained the model with a weight decay of 0.0005, a momentum of 0.9, and a mini-batch size of 1500. The learning rate was initialized with 0.1 and is divided by 10 when the decrease in loss stabilizes. It is necessary to shuffle the training samples randomly.

Baseline method. It uses the same 4-layer plain network in Figure 5.3 but excludes the RRF block from the FC3 layer. We employed the same hyper-parameters for training the RRF-Net model and the baseline model.

5.4.1 Results and discussion

To measure the performance of image-text retrieval, we adopt the evaluation metric R@K which is the recall rate of a correctly retrieved ground-truth at top K candidates (*e.g.* $K = 1, 5, 10$) [55].

Figure 5.5: Qualitative results on Flickr30K and MSCOCO. First column: the baseline model; Second column: RRF-Net model with $T = 3$; Third column: the ensemble model with $M = \{1, 2, 3, 4\}$. For image-to-text retrieval, the ground-truth matching texts are in green. For text-to-image retrieval, the red number in the upper left corner of one image is the ranking order, and the green frame corresponds to the ground-truth matching image.

Evaluation for the RRF-Net

In Table 5.1, we show the results of four RRF-Net models with $T = 1, 2, 3, 4$ (here we use the convolutional fusion). Compared with the baseline model, all four RRF-Net models achieved considerable improvements. This verifies the effectiveness of imposing RRF blocks in a deep matching network. We can observe that, the results when $T = 3$ are superior to other time steps. The drop of performance from T=3 and T=4 may be due to the potential overfitting in the model. It shows a trade-off between the number of recurrent steps and the test performance. The following experiments are performed with $T = 3$. We believe that evaluating more recurrent steps is still promising in future research. The first and second columns in Figure 5.5 compare the examples between the baseline and the RRF-Net.

Evaluation for fusion modules

Recall that we define two types of fusion modules. Table 5.2 compares their quantitative results. First, we trained a RRF-Net model without using any fusion module, which is actually a recurrent residual model in Figure 5.1c. By comparison, we can

Table 5.2: Evaluation for fusion modules on the Flickr30K test set. The convolutional fusion shows better results by learning adaptive weights.

Method	Image to Text		Text to Image	
	R@1	R@5	R@1	R@5
RRF-Net w/o fusion module	45.8	75.9	34.2	67.1
RRF-Net with sum fusion	47.1	76.8	35.0	67.6
RRF-Net with conv fusion	47.6	77.4	35.4	68.3

Table 5.3: Compared results (R@K) between the bi-rank loss and the original bi-directional loss on the Flickr30K test set.

Method	Image to Text		Text to Image	
	R@1	R@5	R@1	R@5
Baseline, bi-directional	43.4	73.8	32.5	65.4
Baseline, bi-rank	45.0	75.5	33.6	66.5
RRF-Net, bi-directional	46.4	76.5	34.1	67.4
RRF-Net, bi-rank	47.6	77.4	35.4	68.3

see that using fusion modules can achieve remarkable improvements. This evaluation reveals the benefit of integrating the intermediate recurrent layers. Moreover, the advantage of the sum-pooling fusion is that it is parameter-free, however, the convolution fusion yields better results than the sum-pooling fusion due to learning adaptive weights. In the following, we implemented the RRF-Net model with the convolutional fusion.

Evaluation for the bi-rank loss

Table 5.3 presents the quantitative comparison between the bi-rank loss and the original bi-directional loss. Actually, the original bi-directional loss is a specific case of the bi-rank loss. We implemented the bi-directional loss by setting $\{\alpha_1, \alpha_2, \beta_1, \beta_2\}$ with $\{1, 0, 2, 0\}$. The baseline and RRF-Net models are both evaluated in this test. In summary, it can be seen that the bi-rank loss brings $\sim$1% performance improvements compared with the bi-directional loss.

5.4.2 Comparison with other approaches

We compared our results with the state-of-the-art approaches in Table 5.4. Overall, RRF-Net achieves competitive (and often better) performance on both Flickr30K and MSCOCO datasets. On the FLICKR30K dataset, DSPE [53] and 2WayNet [76] lead recent state-of-the-art results. Although 2WayNet has the best R@1 results on Flickr30K, the proposed RRF-Net outperforms it on the R@5 accuracy. Additionally, our approach on MSCOCO outperforms the top state-of-the-art approaches.

Table 5.4: Comparison with the state-of-the-art approaches on Flickr30K and MSCOCO for cross-modal retrieval. Our RRF-Net can compete with 2WayNet [76] on the Flickr30K dataset and achieve superior results on the MSCOCO dataset.

Method	Flickr30K dataset						MSCOCO dataset					
	Image to Text			Text to Image			Image to Text			Text to Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
DVSA [55]	22.2	48.2	61.4	15.2	37.7	50.5	38.4	69.9	80.5	27.4	60.2	74.8
UVSE [66]	23.0	50.7	62.9	16.8	42.0	56.5	-	-	-	-	-	-
Mean vector [51]	24.8	52.5	64.3	20.5	46.3	59.3	33.2	61.8	75.1	24.2	56.4	72.4
Deep CCA [49]	27.9	56.9	68.2	26.8	52.9	66.9	-	-	-	-	-	-
VQA-aware [191]	33.9	62.5	74.5	24.9	52.6	64.8	50.5	80.1	89.7	37.0	70.9	82.9
GMM+HGLMM [51]	35.0	62.0	73.8	25.0	52.7	66.0	39.4	67.9	80.9	25.1	59.8	76.6
m-RNN [190]	35.4	63.8	73.7	22.8	50.7	63.1	41.0	73.0	83.5	29.0	42.2	77.0
RNN-FV [185]	35.6	62.5	74.2	27.4	55.9	70.0	41.5	72.0	82.9	29.2	64.7	80.4
mCNN(ensemble) [52]	33.6	64.1	74.9	26.2	56.3	69.6	42.8	73.1	84.1	32.6	68.6	82.8
HM-LSTM [65]	38.1	-	76.5	27.7	-	68.8	43.9	-	87.8	36.1	-	86.7
DSPE [53]	40.3	68.9	79.9	29.7	60.1	72.1	50.1	79.7	89.2	39.6	75.2	86.9
sm-LSTM [68]	42.5	71.9	81.5	30.2	60.4	72.3	53.2	83.1	91.5	40.7	75.8	87.4
2WayNet [76]	**49.8**	67.5	-	**36.0**	55.6	-	55.8	75.2	-	39.7	63.3	-
RRF-Net	47.6	**77.4**	**87.1**	35.4	**68.3**	**79.9**	**56.4**	**85.3**	**91.5**	**43.9**	**78.1**	**88.6**

Table 5.5: Model ensemble results (R@K, $K = 1, 5$) on the Flickr30K test set. Merging more models is significant to obtain better results.

Method	Image to Text		Text to Image	
	R@1	R@5	R@1	R@5
RRF-Net, $M = \{3\}$	47.6	77.4	35.4	68.3
RRF-Net, $M = \{1, 3\}$	49.1	78.4	36.8	69.8
RRF-Net, $M = \{1, 2, 3\}$	50.3	79.2	37.4	70.4
RRF-Net, $M = \{1, 2, 3, 4\}$	**50.8**	**79.5**	**37.6**	**70.9**

Recall that we used the ResNet-152 model to extract visual features. To provide more comparison, we were also curious about the performance when using another well-known CNN: VGG-19 [7]. For Flickr30K, RRF-Net yields R@1=42.1 and 31.2 for image-to-text and text-to-image retrieval, respectively. This was not as high as the proposed RRF-Net performance, but still higher than DSPE [53]. Therefore, RRF-Net presents consistently high performance for diverse feature extractors.

5.4.3 Model ensemble

Although the performance of different RRF-Net models varies, it is beneficial to integrate the retrieved results from multiple models at the test stage. To integrate the strengths of individual RRF-Net models, we employ a simple yet efficient ensemble approach by computing the averaged similarity $s^{'}(x, y)$ given a test pair (x, y):

$$s^{'}(x, y) = \frac{\sum_{m \in M} s^{m}(x, y)}{|M|}, \tag{5.11}$$

where M is the index set, and $s^m(x, y)$ is the similarity computed by the RRF-Net model with $T = m$. For example when $M = \{1, 3\}$, the model ensemble merges the RRF-Net models with $T = 1$ and $T = 3$. As reported in Table 5.5, merging the four models (i.e. $M = \{1, 2, 3, 4\}$) together can significantly improve the performance compared with the single RRF-Net model (i.e. $M = \{3\}$). This ensemble approach can refine the retrieved candidates without increasing the training complexity. In Figure 5.5, the third column shows its retrieval results.

5.5 Chapter Conclusions

In this chapter, we have exploited the RRF block and RRF-Net which can bridge the gap between image and text features in a deep matching network. RRF can be viewed as a feature enhancement component to gather visual and textual representations into a more discriminative embedding space. In addition, we have presented a bi-rank loss function to enhancing the matching constraints in the embedding space. Experiments showed that RRF-Net can achieve competitive performance on the datasets, Flickr30K and MSCOCO.

Future work. This work can provide promising insights towards how to efficiently narrowing the semantic gap between vision and language. Image-text matching is a fundamental technique for many multi-modal research tasks. Therefore, it is promising that the RRF building block could be seamlessly integrated into other multi-modal systems like image captioning and visual question answering.

Chapter 6

Cycle-consistent Embeddings for Cross-modal Retrieval

In the previous chapter, we have exploited an image-text matching network to correlate visual-textual features in a latent embedding space. In this chapter, we further address how we can preserve inter-modal correlations and intra-modal consistency while matching visual and textual representations (RQ5).

To narrow the modality gap between vision and language, prior approaches attempt to discover their correlated semantics in a common feature space. However, these approaches omit the intra-modal semantic consistency when learning the inter-modal correlations. To address this problem, we propose cycle-consistent embeddings in a deep neural network for matching visual and textual representations. Our approach named as CycleMatch can maintain both inter-modal correlations and intra-modal consistency by cascading dual mappings and reconstructed mappings in a cyclic fashion. Moreover, in order to achieve a robust inference, we propose to employ two late-fusion approaches: average fusion and adaptive fusion. Both of them can effectively integrate the matching scores of different embedding features, without increasing the network complexity and training time. In the experiments on cross-modal retrieval, we demonstrate comprehensive results to verify the effectiveness of the proposed approach. Our approach achieves state-of-the-art performance on two well-known multi-modal datasets, Flickr30K and MSCOCO.

Keywords
Cross-modal retrieval, Embedding, Deep neural networks, Late fusion

6.1 Introduction

Nowadays, the explosive growth of multimedia data in social networks (*e.g.* image, video, text and audio) have triggered a massive amount of research activities in multi-modal understanding and reasoning. In this chapter, we focus on the task of image-text matching, which aims to incorporate heterogeneous representations from visual and textual modalities. In practice, this task plays an essential role in a wide variety of vision-and-language applications, for examples, cross-modal retrieval [192, 193], visual question answering [58, 194], zero-shot recognition [195, 196] and visual grounding [197, 198].

The core issue with image-text matching is searching for an appropriate embedding space where related images and texts can be matched correctly. Driven by the great strides made by deep learning [4, 7, 10], recent research has been dedicated to exploring deep neural networks for learning powerful embedding features, in order to narrow the modality gap between visual and textual domains. These networks are typically composed of two branches for generating visual and textual embedding features in a common latent space, respectively [53, 64, 65, 67, 68]. Then, a similarity-based ranking loss is used to measure the latent embedding features. Latent embeddings can distill common semantic information about both the visual content and textual description. To directly match the similarities between vision and language, researchers further exploit dual embeddings by translating an input feature in the source space to be the feature in the target space [71, 72, 76, 77]. Both the latent and dual embeddings can capture inter-modal semantic correlations, however, they are limited in preserving intra-modal semantic consistency. Our motivation for this work is that: *A robust embedding method should be able to learn representations of both the source and target modalities.* Inspired by this motivation, in this chapter we focus on solving the fifth research question **RQ 5: How can we preserve both inter-modal correlations and intra-modal consistency for learning robust visual and textual embeddings?**

Inspired by the idea of cycle-consistent learning [94, 199], we propose cycle-consistent embeddings in an image-text matching network, which can incorporate both *inter-modal correlations* and *intra-modal consistency* for learning robust visual and textual embeddings. Figure 6.1 illustrates our embedding method by integrating three feature embeddings, including dual, reconstructed and latent embeddings. Specifically, it has two cycle branches, one starting from an image feature in the visual space and the other from a text feature in the textual space. For each branch, it first accomplishes a dual mapping by translating an input feature in the source space to be a dual embedding in the target space. Inverse to the dual mapping, we then exploit a reconstructed mapping, with the aim of translating the dual embedding back to the source space. Moreover, we learn a latent space during the dual and reconstructed mappings and correlate the latent embeddings. In the three feature

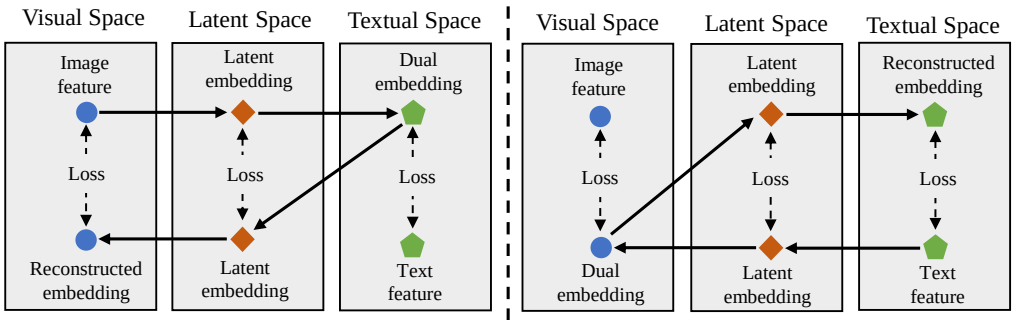


Figure 6.1: Schematic pipeline of our proposed cycle-consistent embedding method. It is composed of two cycle branches starting from (Left) visual space and (Right) textual space, respectively. We first perform a dual mapping by transforming the input feature into the target feature space. Then the dual embedding is used to generate a reconstructed embedding in a reconstructed mapping. In addition, we construct a latent space to correlate latent embeddings of the two mappings. The two branches share the mapping functions for transformations between three feature spaces, and can be trained jointly by optimizing the matching losses in the three feature spaces.

spaces, we compute their ranking losses to jointly optimize the whole embedding learning. Consequently, our visual-textual embedding method can learn not only *inter-modal mappings* (*i.e.* image-to-text and text-to-image), but also *intra-modal mappings* (*i.e.* image-to-image and text-to-text).

The contributions of this work are as follows:

- We propose a novel deep cycle-consistent embedding network for image-text matching. Our approach called CycleMatch can cascade dual and reconstructed mappings together to maintain inter-modal correlations and intra-modal consistency. To our best knowledge, this is the first work to explore the usage of cycle consistency for solving the task of image-text matching.

- To improve the inference at the test stage, we present two late-fusion approaches to efficiently integrate the matching scores of multiple embedding features without increasing the training complexity.

- In the experiments, our cycle-consistency embedding outperforms traditional embeddings with considerable improvements for cross-modal retrieval on two multi-modal datasets, *i.e.* Flickr30K and MSCOCO. In addition, our results are competitive with the state-of-the-art approaches.

The rest of this chapter is structured as follows. Related works are introduced in Section 6.2. Section 6.3 details the proposed CycleMatch. The experimental results are reported in Section 6.4. Finally, Section 6.5 summarizes the conclusions.

6.2 Related Work

Our work is related to the image-text matching methods based on deep neural networks, and other works about cycle-consistent learning.

Deep visual-textual embeddings

With the increasing progress of deep learning, research efforts have been made to CCA into deep neural networks [49, 50, 51, 62]. However, most deep CCA models rely on expensive decorrelation computations, which limit their generalization abilities at large-scale data. Alternatively, a number of recent approaches [52, 55, 64, 65, 66] address the task by designing two-branch networks to embed visual and textual features into a common latent space, and then learn latent embeddings by optimizing a ranking loss between matched and unmatched image-text pairs. For instance, Wang *et al.* [53] built a simple and efficient matching network to preserve the structure relations between images and texts in the latent space. To associate image regions with words, the attention mechanism was integrated into visual-textual embedding models [67, 68]. In addition to the pairwise ranking loss, recent approaches [69, 70] leveraged extra loss functions to enhance the discrimination of the learned embedding features.

Another line of research [71, 72, 73, 74, 75] focuses on learning dual embeddings between two modalities, *e.g.* projecting visual features into the textual feature space and vice versa. Essentially, the dual embedding models are motivated by autoencoders. For instance, Feng *et al.* [71] proposed a correspondence cross-modal autoencoder model. 2WayNet [76] built the projections between two modalities and regularized them with Euclidean loss. Recently, the work of Gu *et al.* [77] utilized two generative models to synthesize grounded visual and textual representations. Also, Huang *et al.* [200] jointly modeled image-sentence matching and sentence generation. Note that, latent embeddings can be additionally used in the dual embedding models to enhance cross-modal relations.

In contrast to the above studies, our approach builds a reconstructed mapping upon the dual mapping, and generates cycle-consistent embeddings that are beneficial to the process of matching visual-textual representations. In Figure 6.2, we show the differences of our model from previous works.

Cycle-consistent learning

There are a few papers exploring cycle consistency for diverse applications [94, 199, 201, 202, 203]. They are mainly motivated by the fact that, cycle-consistent learning is encouraged to produce additional feedback signals to improve the bi-directional translations. Specifically, He *et al.* [199] proposed a dual-learning mechanism based

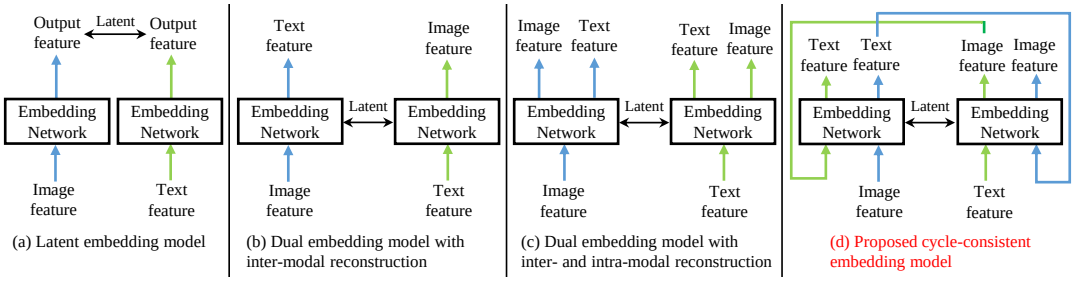


Figure 6.2: Conceptual illustration of variants of image-text matching models. (a) Latent embedding model. (b) Dual embedding model with inter-modal reconstruction. (c) Dual embedding model with inter-modal and intra-modal reconstruction. (d) Our cycle-consistent embedding model. Notice that the models in (b)(c)(d) also impose latent embeddings on hidden layers. Our model cascades the two embedding networks in a cyclic fashion, which can enhance interactions between two embedding networks.

on deep reinforcement learning, where one agent was used to learn the primal task, *e.g.* English-to-French translation, and the other agent for the dual task, *e.g.* French-to-English translation. More recently, Zhu *et al.* [94] exploited cycle-consistent adversarial networks (CycleGAN), which combined a cycle-consistency loss with an adversarial loss [79] to perform unpaired image-to-image translations between two different visual domains. A similar idea was also presented in [204, 205].

Although prior works have shown the effectiveness of using cycle-consistent constraints for intra-modal domain mappings, yet in the context of cross-modal representation learning, its effectiveness has not been well investigated. In contrast to prior approaches that utilize cycle-consistent constraints within one modality (*e.g.* neural machine translation and image-to-image translation), our work is the first to extend the usage of cycle consistency for learning visual-textual embeddings. The work of Chen and Zitnick [206] is relevant to ours, as their model can both generate textual captions and reconstruct visual features given an image representation. However, their model lacks the inverse cycle mapping, *i.e.* text-to-image-to-text, which can be jointly learned in our model. Last but not least, these existing works did not consider matching latent embeddings during the cycle-consistent scheme.

6.3 Cycle-consistent Embeddings

In this section, we present the proposed network (CycleMatch) with cycle-consistent embeddings for matching visual and textual representations. For a robust inference, we exploit two late-fusion approaches by taking advantage of multiple embedding features learned in the network.

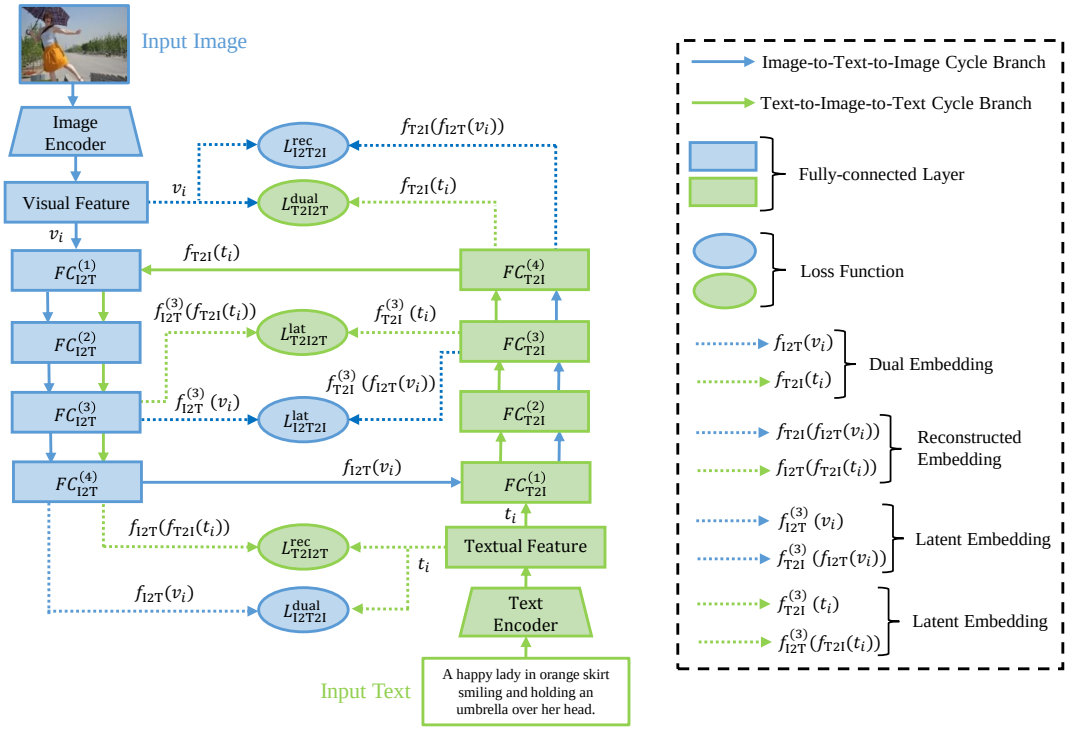


Figure 6.3: The proposed CycleMatch exploits two cycle branches for image-text matching. For each branch, it is divided into two sub-branches from the *fourth* FC layer (*i.e.* $FC^{(4)}_{\text{IT}}$ and $FC^{(4)}_{\text{TI}}$). One sub-branch continues accomplishing the dual mapping to the target feature space, while the other sub-branch is used to perform the reconstructed mapping back to the source feature space. Consequently, the cycle branches allow to jointly learn dual, reconstructed and latent embedding features. We can train the network end-to-end by optimizing several loss functions simultaneously.

6.3.1 System architecture

Figure 6.3 depicts an overview of the CycleMatch architecture. The entire network consists of three components: feature encoder, feature embedding and feature matching. First of all, given an input image I_i and text T_i, we employ individual feature encoders to extract the visual feature $\boldsymbol{v}_i = En_{\text{img}}(I_i)$ and textual feature $\boldsymbol{t}_i = En_{\text{text}}(T_i)$. Then, we develop several fully-connected (FC) layers (*i.e.* $FC^{(j)}_{\text{I2T}}$) to perform the *Image-to-Text* (I2T) mapping and several other FC layers (*i.e.* $FC^{(j)}_{\text{T2I}}$) for the *Text-to-Image* (T2I) mapping. Let $f_{\text{I2T}}(\cdot)$ and $f_{\text{T2I}}(\cdot)$ represent the mapping functions for I2T and T2I, respectively. In addition, connecting FC_{I2T} and FC_{T2I} can form two cycle mappings between the visual and textual feature spaces. Specifically, given $\boldsymbol{v}_i$, we first transform it to be $f_{\text{I2T}}(\boldsymbol{v}_i)$ in the textual feature space and then learn its reconstructed feature $f_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))$ in the visual feature space. Moreover, we also correlate intermediate features derived from $FC^{(3)}_{\text{I2T}}$ and $FC^{(3)}_{\text{T2I}}$, so as to learn a latent feature space. Similarly, $\boldsymbol{t}_i$ is used to start another cycle mapping. In a nutshell, each cycle mapping can learn dual, reconstructed and latent embeddings in a cyclic fashion.

6.3.2 Formulation

Next, we will detail the above three embeddings and formulate their loss functions separately. The entire network contains two cycle-consistent embedding branches: one for *image-to-text-to-image* (I2T2I) mapping and the other for *text-to-image-to-text* (T2I2T) mapping. Here, we take the I2T2I mapping for an example.

Dual embedding

In a dataset collection with N image-text pairs, we take as input $\boldsymbol{v}_i$ into $FC^{(j)}_{\text{I2T}}$, where $i = 1, \ldots, N$ and $j = 1, \ldots, 4$, and generate the dual embedding $f_{\text{I2T}}(\boldsymbol{v}_i)$ in the textual space, which should have the same dimension as the ground-truth textual feature $\boldsymbol{t_i}$. Then, we need to normalize the two features and compute their similarity using the cosine distance

$$s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}_i) = \frac{f_{\text{I2T}}(\boldsymbol{v}_i) \cdot \boldsymbol{t}_i}{||f_{\text{I2T}}(\boldsymbol{v}_i)|| \cdot ||\boldsymbol{t}_i||}. \tag{6.1}$$

During training, it is important to construct a number of negative pairs, in addition to the positive pair. Thereby, we search for the top K negative samples in a mini-batch for both $f_{\text{I2T}}(\boldsymbol{v}_i)$ and $\boldsymbol{t}_i$, which are denoted with $f_{\text{I2T}}(\boldsymbol{v}^-_{i,k})$ and $\boldsymbol{t}^-_{i,k}$, respectively, where $k = 1, \ldots, K$. To learn dual mappings, we need to employ a pairwise ranking loss function with respect to positive and negative pairs:

$$\begin{aligned}
&\mathcal{L}^{\text{dual}}_{\text{I2T2I}} = \\
&\sum_{i=1}^{N}\sum_{k=1}^{K}\Big\{\max\left[0, m - s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}_i) + s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}^-_{i,k})\right] \\
&+ \alpha \max\left[0, m - s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}_i) + s(f_{\text{I2T}}(\boldsymbol{v}^-_{i,k}), \boldsymbol{t}_i)\right]\Big\},
\end{aligned} \tag{6.2}$$

where m is a margin parameter and α adjusts the weights of the two loss terms. Ideally, the matched distance $s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}_i)$ should be smaller than any of the unmatched distances $s(f_{\text{I2T}}(\boldsymbol{v}_i), \boldsymbol{t}^-_{i,k})$ and $s(f_{\text{I2T}}(\boldsymbol{v}^-_{i,k}), \boldsymbol{t}_i)$.

Reconstructed embedding

In addition to learning inter-modal correlations from dual mappings, we further explore reconstructed mappings to maintain the intra-modal semantic consistency. We cascade the dual and reconstructed mappings to form an intra-modal autoencoder and minimize the reconstruction error based on the ranking loss instead of the traditional Euclidean loss. Specifically, we feed $f_{\text{I2T}}(\boldsymbol{v}_i)$ into $FC^{(j)}_{\text{T2I}}$, to produce

a reconstructed embedding feature $\tilde{\boldsymbol{v}}_i$ in the visual feature space with

$$\tilde{\boldsymbol{v}}_i = f_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i)) = f_{\text{T2I}} \circ f_{\text{I2T}}(\boldsymbol{v}_i). \tag{6.3}$$

The ranking loss for making the reconstructed embedding feature $\tilde{\boldsymbol{v}}_i$ match with the original visual feature $\boldsymbol{v}_i$ can be written as follows

$$\begin{aligned}\mathcal{L}^{\text{rec}}_{\text{I2T2I}} = \sum_{i=1}^{N}\sum_{k=1}^{K}\Big\{ & \max\left[0, m - s(\tilde{\boldsymbol{v}}_i, \boldsymbol{v}_i) + s(\tilde{\boldsymbol{v}}_i, \boldsymbol{v}^-_{i,k})\right] \\ & + \alpha \max\left[0, m - s(\tilde{\boldsymbol{v}}_i, \boldsymbol{v}_i) + s(\tilde{\boldsymbol{v}}^-_{i,k}, \boldsymbol{v}_i)\right]\Big\}.\end{aligned} \tag{6.4}$$

Since $\mathcal{L}^{\text{rec}}_{\text{I2T2I}}$ also has an effect on the parameters of $FC^{(j)}_{\text{I2T}}$, the reconstructed mappings can help to improve the learning of dual mappings as well.

Latent embedding

Furthermore, we exploit a latent feature space to enhance the correlations between the dual and reconstructed mappings. Latent embeddings are able to distill common semantic information from visual and textual representations. Specifically, we make use of the intermediate representations from the third FC layers, *i.e.* $FC^{(3)}_{\text{I2T}}$ and $FC^{(3)}_{\text{T2I}}$. When $\boldsymbol{v_i}$ passes through $FC^{(3)}_{\text{I2T}}$, we can extract an intermediate feature $f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i)$. Also, the dual embedding $f_{\text{I2T}}(\boldsymbol{v}_i)$ passes through $FC^{(3)}_{\text{T2I}}$ to generate another intermediate feature $f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))$. The ranking loss for matching latent embeddings thereby becomes

$$\begin{aligned}\mathcal{L}^{\text{lat}}_{\text{I2T2I}} = \sum_{i=1}^{N}\sum_{k=1}^{K}\Big\{ & \max\Big[0, m - s\big(f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i), f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))\big) \\ & + s\big(f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i), f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}^-_{i,k}))\big)\Big] \\ & + \alpha \max\Big[0, m - s\big(f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i), f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))\big) \\ & + s\big(f^{(3)}_{\text{I2T}}(\boldsymbol{v}^-_{i,k}), f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))\big)\Big]\Big\}.\end{aligned} \tag{6.5}$$

6.3.3 Full objective

Similar to the above I2T2I branch, it is straightforward to express the matching losses in the T2I2T branch, including $\mathcal{L}^{\text{dual}}_{\text{T2I2T}}$, $\mathcal{L}^{\text{rec}}_{\text{T2I2T}}$ and $\mathcal{L}^{\text{lat}}_{\text{T2I2T}}$. During training, we need to incorporate all the loss functions jointly. Finally, the full objective is to

minimize the total loss:

$$\begin{aligned}&\underset{W_{\text{I2T}},W_{\text{T2I}}}{\arg\min}\ \mathcal{L}_{\text{total}} = \\ &\mathcal{L}^{\text{dual}}_{\text{I2T2I}} + \mathcal{L}^{\text{rec}}_{\text{I2T2I}} + \mathcal{L}^{\text{lat}}_{\text{I2T2I}} + \mathcal{L}^{\text{dual}}_{\text{T2I2T}} + \mathcal{L}^{\text{rec}}_{\text{T2I2T}} + \mathcal{L}^{\text{lat}}_{\text{T2I2T}},\end{aligned} \tag{6.6}$$

where W_{I2T} and W_{T2I} indicate the parameters in $FC^{(j)}_{\text{I2T}}$ and $FC^{(j)}_{\text{T2I}}$, respectively. They are unshared due to the specialization of two different modalities. To demonstrate the effectiveness of our CycleMatch, we utilize the t-SNE [207] algorithm to visualize the embedding features learned in the visual, textual and latent feature spaces, separately. As shown in Figure 6.4, we randomly select 100 image-text pairs from the Flickr30K dataset [189]. From all the feature maps, we can visibly observe high similarities between two matched samples.

6.3.4 Late-fusion inference

By performing cycle-consistent embeddings, we can represent one sample with a set of three different features, for instance, $\{\boldsymbol{v}_i, f_{\text{I2T}}(\boldsymbol{v}_i), f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i)\}$ for an image. Since the reconstructed embedding $\tilde{\boldsymbol{v}}_i$ and the other latent embedding $f^{(3)}_{\text{T2I}}(f_{\text{I2T}}(\boldsymbol{v}_i))$ are related to $\boldsymbol{v}_i$ and $f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i)$, we do not consider them for simplicity. Each of the three features can be used to measure an image-text matching score. Instead of using only one score, it is encouraged to leverage different scores together to achieve a more robust inference. This is driven by the late-fusion technique [208] in multimedia retrieval, which is a simple and efficient approach to combine the prediction scores of individual features. In this work, we present two effective late-fusion approaches, namely average fusion and adaptive fusion.

Average fusion

Given a query image I_q, we extract three features $\{\boldsymbol{v}_q, f_{\text{I2T}}(\boldsymbol{v}_q), f^{(3)}_{\text{I2T}}(\boldsymbol{v}_q)\}$. Similarly, an arbitrary text T_i in the dataset can be described with $\{\boldsymbol{t}_i, f_{\text{T2I}}(\boldsymbol{t}_i), f^{(3)}_{\text{T2I}}(\boldsymbol{t}_i)\}$. We can compute three similarity scores between I_q and T_i:

$$\begin{cases}\text{visual score}: s^{(1)}(\boldsymbol{v}_q, \boldsymbol{t}_i) = s(\boldsymbol{v}_q, f_{\text{T2I}}(\boldsymbol{t}_i)), \\ \text{textual score}: s^{(2)}(\boldsymbol{v}_q, \boldsymbol{t}_i) = s(f_{\text{I2T}}(\boldsymbol{v}_q), \boldsymbol{t}_i), \\ \text{latent score}: s^{(3)}(\boldsymbol{v}_q, \boldsymbol{t}_i) = s(f^{(3)}_{\text{I2T}}(\boldsymbol{v}_q), f^{(3)}_{\text{T2I}}(\boldsymbol{t}_i)).\end{cases} \tag{6.7}$$

Then we combine the three scores to obtain an average fusion score as follows

$$s^{avg}(\boldsymbol{v}_q, \boldsymbol{t}_i) = \frac{\sum_{j=1}^{3} s^{(j)}(\boldsymbol{v}_q, \boldsymbol{t}_i)}{3}. \tag{6.8}$$

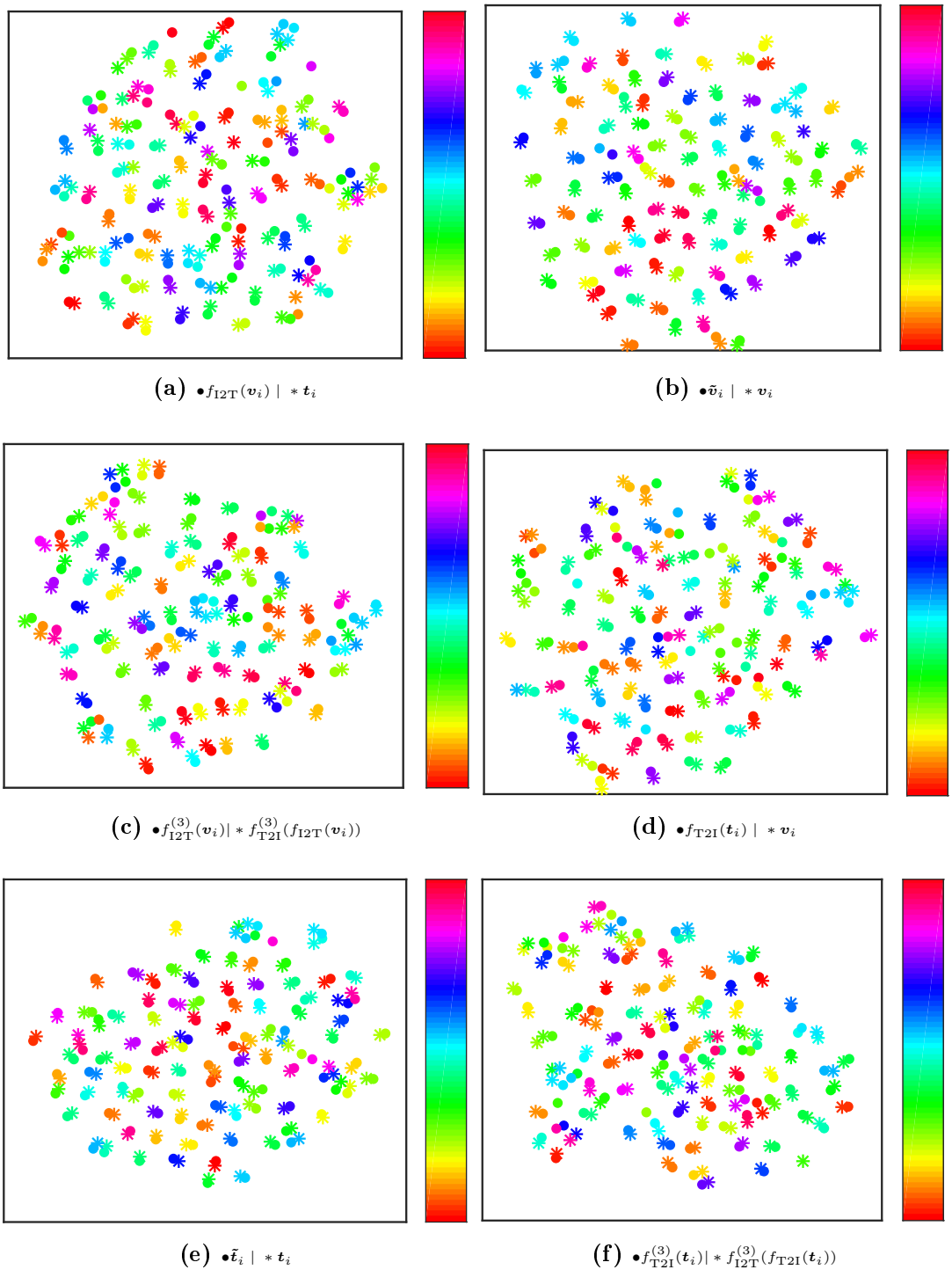


Figure 6.4: Visualization of our embedding features by using 100 image-text pairs in Flickr30K [189]. The first and second rows represent the embedding features learned in the I2T2I and T2I2T branches respectively. In each feature map, matched samples are shown with the same color. In (a)(d), the dual embedding features ('•') can match with the corresponding target features ('∗'); In (b)(e), the reconstructed embedding features ('•') look closely similar to the source features ('∗'). In (c)(f), the two latent embedding features ('•' and '∗') can learn to correlate with each other as well.

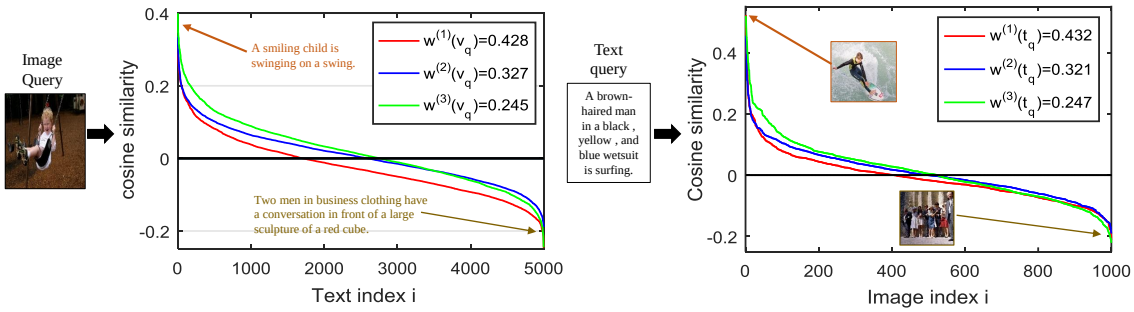


Figure 6.5: Illustration of the sorted score curves based on three different features. For the query image in left, the first curve (in red) forms the smallest area above the X axis, so the corresponding feature (*i.e.* visual embedding feature) can have the largest weight (0.428). We show a matched text at the beginning of the curves and an unmatched text at the end of the curves. Similarly, we demonstrate a text query example in right.

It is similar to compute the fusion score $s^{avg}(\boldsymbol{t}_q, \boldsymbol{v}_i)$ in terms of a query text T_q.

Adaptive fusion

To study the importance of different features, we further learn adaptive weights when combining the three scores. As suggested in [209], the score curve by using a superior feature can be sorted in an "L" shape, while the curve by using an inferior feature tends to gradually descend. In addition, the area under the curve can be used as an indicator to measure the weight of the corresponding feature. Driven by this observation, we can use the sorted score curves of the above three features to decide their weights. Specifically, we utilize each of the three features to compute the score curve of a query image I_q to all the text samples. Then, we sort the score curves and compute their areas with respect to the horizonal axis. In Figure 6.5, we show three sorted score curves for either a query image (Left) or text (Right). In contrast to [209] where the scores are in $[0, 1]$, our scores are based on the cosine distance, ranging from -1 to 1. Accordingly, we can obtain both a positive area and a negative area, locating on the two sides of the $X = 0$ axis. To alleviate the effects of long tails in the curves, we utilize only the positive area to compute the weight and omit the negative one. The positive area associated with the j-th feature can be approximated by

$$area_{+}^{(j)}(\boldsymbol{v}_q) = \sum_{i=1}^{N} \max\left[0, s^{(j)}(\boldsymbol{v}_q, \boldsymbol{t}_i)\right]. \tag{6.9}$$

Smaller positive area means that the corresponding feature should have greater weights. Hence, the adaptive weights of I_q *w.r.t.* the three features can be expressed

with

$$w^{(j)}(\boldsymbol{v}_q) = \frac{1}{area_+^{(j)}(\boldsymbol{v}_q)}. \tag{6.10}$$

In addition, we normalize the three weights to make sure $\sum_{j=1}^{3} w^{(j)}(\boldsymbol{v}_q) = 1$. Finally, the adaptive fusion score for matching I_q and T_i becomes

$$s^{adt}(\boldsymbol{v}_q, \boldsymbol{t}_i) = \sum_j w^{(j)}(\boldsymbol{v}_q) \cdot s^{(j)}(\boldsymbol{v}_q, \boldsymbol{t}_i). \tag{6.11}$$

Likewise, we demonstrate a text query T_q in the right of Figure 6.5, and show its adaptive weights, $w^{(j)}(\boldsymbol{t}_q)$. Notice that our adaptive fusion approach can achieve specific weights for different query samples. It is an unsupervised and efficient manner without adding extra parameters and manual tuning. In the experiments, we analyze the effects of these two late-fusion approaches on the inference of cross-modal retrieval.

6.4 Experiments

First, we compare CycleMatch with various baseline models to verify its effectiveness. In addition, we present in-depth analysis on the two late-fusion approaches. Moreover, our results can be competitive with the state-of-the-art performance for cross-modal retrieval on two well-known datasets. Finally, we present additional ablation study on the effect of feature encoders and variance of test splits.

6.4.1 Experimental setup

We present the Dataset protocols, evaluation metrics, Network details, training details and training time, used in our experimental setup.

Dataset protocols

The experiments are performed on two well-known datasets: Flickr30K [189] and MSCOCO [117]. 1) Flickr30K [189] consists of 31,783 images and each image is associated with five different sentences. We use the dataset split of [190], namely 29,783 training images, 1,000 validation images and 1,000 test images. 2) MSCOCO [117] is one of the largest multi-modal datasets, which includes 82,783 training images and 40,504 validation images. We pick five ground-truth sentences for each image. 1,000 test images are selected from the validation set [190]. Notice that some works [53, 69, 77] merge the remaining validation images into the training set, to further increase the performance. However, we keep only using the original training set for fairness.

Evaluation metrics

For evaluating the performance of cross-modal retrieval, we adopt the common metric R@K, which measures the recall rate of a correctly retrieved ground-truth at top K retrieved candidates. Generally, K is set to 1, 5 and 10 for both image-to-text and text-to-image retrieval.

Network details

In terms of the image encoder, we employed the powerful ResNet-152 [10] pre-trained on the ImageNet dataset [5]. Besides, we recast the CNN model to its fully convolutional network (FCN) counterpart, which can capture rich region representations. The last layer of the FCN model is spatially averaged to generate a 2,048 dimensional visual representation. To extract the textual representation, we utilized the pre-trained RNN encoder proposed in [210]. It can represent one sentence with a 4,096 dimensional feature vector. Currently, we did not fine-tune the feature encoders during the training.

As for the two groups of four FC layers in CycleMatch (*i.e.* $FC^{(j)}_{\text{I2T}}$ and $FC^{(j)}_{\text{T2I}}$), the channels of the first three layers are fixed as [2048,512,512]. Note that, $FC^{(4)}_{\text{I2T}}$ should have the same dimension as the textual feature and $FC^{(4)}_{\text{T2I}}$ should be equal to the size of the visual feature.

Training details

We implemented the proposed approach based on the Caffe library [130]. It is important to shuffle the training samples randomly during the data preparation stage. The hyper-parameters are evaluated on the validation set of each dataset. We trained the model using SGD with a mini-batch size of 500, a weight decay of 0.0005, a momentum of 0.9 and an initialized learning rate of 0.1. The learning rate is divided by 10 when the decrease in loss stabilizes. We set $\alpha = 2$ and $m = 0.1$ in all the experiments. The number of negative samples in each min-batch is 50. The whole training procedure terminates after 60 epochs for both datasets.

Time complexity

We use the total loss in Eq. (6.6) to perform the training procedure. Each loss term is a simple and efficient ranking loss that is widely used in retrieval tasks. We used a Titan X card with 12 GB to train all models in the experiments. For the full CycleMatch model, training required about 19 hours on the Flickr30K dataset and 47 hours on the MSCOCO dataset, respectively.

Table 6.1: Summary of various embedding methods for image-text matching.

Embedding methods	Main description
LatentMatch	It is a latent embedding model by matching $f^{(3)}_{\text{I2T}}(\boldsymbol{v}_i)$ and $f^{(3)}_{\text{T2I}}(\boldsymbol{t}_i)$.
DualMatch	It is a dual embedding model with two dual mappings: I→T and T→I.
CycleMatch(w/o latent)	It is an ablation model without latent embeddings.
CycleMatch(I2T2I)	It consists of an I2T2I cycle branch and an I→T dual mapping.
CycleMatch(T2I2T)	It is composed of a T2I2T cycle branch and a T→I dual mapping.
CycleMatch	It is the fully implemented model by integrating two cycle branches.

Table 6.2: Comparison of the cross-modal retrieval results on Flickr30k and MSCOCO. Higher R@K numbers are better, where $K = 1, 5, 10$. The full CycleMatch model outperforms other baseline models on both datasets.

Method	Flickr30K dataset						MSCOCO dataset					
	Image to Text			Text to Image			Image to Text			Text to Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
LatentMatch	49.7	77.4	85.0	37.8	69.8	80.6	53.9	82.9	90.8	43.0	75.8	85.9
DualMatch	53.4	80.5	87.1	40.1	70.9	81.0	56.3	83.5	91.5	45.5	76.7	87.5
CycleMatch(w/o latent)	56.8	81.7	90.3	41.1	72.5	81.3	58.5	84.0	92.4	46.9	78.3	88.7
CycleMatch(I2T2I)	57.0	82.4	**91.0**	42.4	73.6	82.0	**61.1**	85.5	93.1	46.3	79.3	89.0
CycleMatch(T2I2T)	56.4	81.9	90.6	**43.2**	74.3	82.6	59.7	84.7	92.6	**47.6**	79.7	89.6
CycleMatch	**57.8**	**83.3**	90.9	**43.2**	**74.8**	**83.8**	60.5	**86.3**	**93.7**	47.2	**80.3**	**90.4**

6.4.2 Comparisons with baseline methods

To demonstrate the superiority of our approach, we implemented several other embedding approaches based on the same network settings and training hyper-parameters as CycleMatch. Table 6.1 describes the details regarding these methods. In terms of inference, LatentMatch is evaluated with only the latent score. However, all the other models have both visual and textual scores, therefore we utilize the average fusion approach to accomplish their inference for a fair comparison. Table 6.2 reports results of these models on both Flickr30K and MSCOCO for both image-to-text retrieval and text-to-image retrieval. It can be seen that, CycleMatch surpasses LatentMatch and DualMatch with significant improvements, and achieves overall superior performance over other variants of CycleMatch. Furthermore, we can observe the following findings:

Impact of reconstructed embeddings. The main difference between DualMatch and CycleMatch(w/o latent) is that the latter model uses a reconstructed mapping upon the traditional dual mapping. The performance improvement from CycleMatch(w/o latent) shows the benefit of learning reconstructed embeddings in a cyclic fashion.

Impact of latent embeddings. By comparing the results of CycleMatch and CycleMatch(w/o latent), we find that integrating the latent embeddings into CycleMatch brings further improvements over all R@K measurements. For example, R@5 shows about 2% gains for both I→T and T→I. Although using only latent

embeddings (*i.e.* LatentMatch) is inferior to other models, it is beneficial to adopt them to improve other embedding methods like CycleMatch.

Impact of cycle branches. Both CycleMatch(I2T2I) and CycleMatch(T2I2T) can outperform LatentMatch and DualMatch, even though only one cycle-consistent embedding branch is used. By comparing these two models, CycleMatch(I2T2I) performs better for I→T retrieval, while CycleMatch(T2I2T) yields better results for T→I retrieval. When we incorporate the two cycle branches jointly to construct a full CycleMatch, it achieves overall superior performance over any single cycle branch on both datasets. It is consistent with our motivation that it is beneficial to model image-text co-translation simultaneously.

In addition to the R@K performance, we further analyze the matching scores by using our embedding features. To be specific, we randomly select 100 image-text pairs from the test set, and compute the similarity between an image and a text. As shown in Figure 6.6, matched image-text pairs (with the same index) have greater similarity scores than unmatched ones. This means that our embedding features are able to learn the correlations between visual and textual representations.

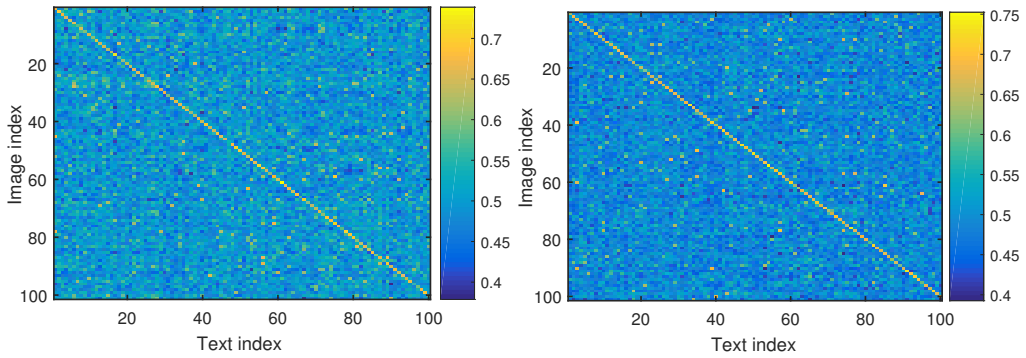
6.4.3 Analysis of late-fusion inference

Recall that CycleMatch contains visual, textual and latent scores for inference (Section 6.3.4). In this experiment, we compare three strategies to study the effect of two late-fusion inference approaches on the retrieval performance of CycleMatch. Specifically, the one-score strategy uses only a single visual score; the two-score strategy integrates visual and textual scores together; the three-score strategy combines all three scores by further adding the latent score. Table 6.3 reports the results of the three strategies. For the two-score and three-score strategies, we present the results of using the average and adaptive fusion, respectively. From the results, we can make the following observations:

1) The two-score strategy improves the one-score counterpart with 1%-3% gains. As the visual and textual scores match the samples in two different feature spaces, their complementary scores are able to improve the inference quality.

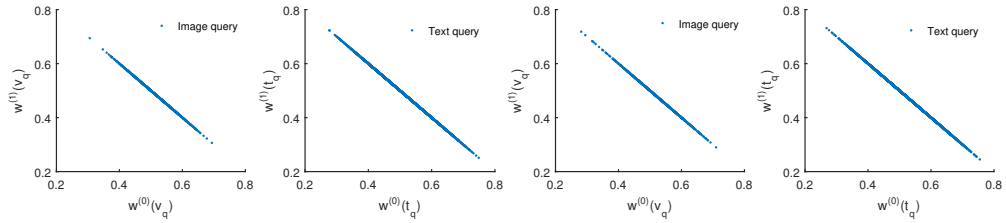
2) The adaptive fusion outperforms the average one in terms of both two-score and three-score strategies. Although their performance gap over the R@K measurements is not significant, the adaptive fusion is an efficient method without imposing extra parameters and manual tuning. In addition, the inference time of the adaptive fusion is close to that of the average fusion.

3) The three-score strategy fails to achieve further improvements over the two-score one. We attribute this to the fact that, the latent score measures the similarity between $f^{(3)}_{\mathrm{I2T}}(\boldsymbol{v}_i)$ and $f^{(3)}_{\mathrm{T2I}}(\boldsymbol{t}_i)$. However, we do not use a direct matching loss between them during training CycleMatch. Although adding this latent score for



(a) Flickr30K **(b)** MSCOCO

Figure 6.6: Similarity matrix of 100 image-text pairs from the test set. The related images and texts have the same index numbers. The diagonal line demonstrates high inter-modal correlations for matched image-text pairs. The original cosine scores are re-scaled to be [0,1].



(a) I→T **(b)** T→I **(c)** I→T **(d)** T→I

Figure 6.7: Visualization of adaptive weights for 1000 image queries and 5000 text queries on Flickr30K(a, b) and MSCOCO (c, d). Each dot in the maps is a query sample, having two weights for the adaptive fusion. Note that $w^{(0)} + w^{(1)} = 1$. The weights of query samples are mostly gathered between 0.4 and 0.6. It suggests that both visual and textual scores play an important role in the inference results.

inference will not bring further performance gains, learning the latent embeddings in CycleMatch is still important for improving the entire embedding procedure. As we discussed earlier, CycleMatch performs better than the variant without latent embeddings, namely CycleMatch(w/o latent).

As we can see, the two-score adaptive fusion achieves the best results. In Figure 6.7, we further present and analyze the two adaptive weights (*i.e.* $w^{(1)}(\cdot)$ and $w^{(2)}(\cdot)$), which are learned in the two-score adaptive fusion for visual and textual scores. Figure 6.7(a,b) and (c,d) shows the weights for Flickr30K and MSCOCO, respectively. For I2T retrieval, we illustrate the adaptive weights of 1000 image queries, namely $w^{(1)}(v_q)$ and $w^{(2)}(v_q)$; for T2I retrieval, we show all the weights of 5000 text queries, denoted as $w^{(1)}(t_q)$ and $w^{(2)}(t_q)$. Notice that, each dot in Figure 6.7 represents a query sample that learns individual weights based on its score curves. It can be seen that most samples have weights ranging from 0.4 to 0.6, which suggests that both visual and textual scores have an important impact on the inference results.

Table 6.3: Evaluation on the effect of different inference strategies on the R@K measurements. The two-score strategy based on the adaptive fusion achieves the best results (in bold face).

Inference method	Flickr30K dataset						MSCOCO dataset					
	Image to Text			Text to Image			Image to Text			Text to Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
One-score, without fusion	54.8	82.6	90.1	40.1	70.9	81.0	58.6	85.5	92.6	45.5	78.3	88.7
Two-score, average fusion	57.8	83.3	90.9	43.2	74.8	83.8	60.5	86.3	93.7	47.2	80.3	90.4
Two-score, adaptive fusion	**58.6**	83.6	**91.6**	**43.6**	**75.3**	**84.2**	**61.1**	**86.8**	94.2	**47.9**	80.9	**90.9**
Three-score, average fusion	57.4	83.5	91.0	43.2	74.7	83.9	59.7	86.0	94.0	46.9	80.6	89.8
Three-score, adaptive fusion	57.8	**83.8**	91.2	43.5	74.7	84.0	61.0	86.4	**94.5**	47.8	**81.0**	90.7

6.4.4 Comparisons with state-of-the-art approaches

In Table 6.4 and Table 6.5, we present a comprehensive comparison with previous papers where they reported the cross-modal retrieval performance on Flickr30K and MSCOCO. It can be seen that our CycleMatch (the two-score adaptive fusion) outperforms recent state-of-the-art approaches [64, 67, 211] with promising improvements on both datasets. It is worth noting that these approaches employ different feature encoders that have a significant influence on the performance. For a clear comparison, we further list the image and text encoders used in these approaches. In the following experiments, we will study the effect of different feature encoders on the performance of CycleMatch.

To boost the performance, recent several approaches [69, 77, 200, 211] further fine-tune the image encoders during training their models. Their results with fine-tuning the image encoders achieve better performance on MSCOCO than Flickr30K. We should know that it is feasible to fine-tune the image encoders while training our CycleMatch, which can help to further improve our results. In addition, the fine-tuning process will maintain the findings we mentioned as above. More importantly, our results on the Flickr30K dataset can even compete with the fine-tuned results in [69, 77, 200, 211]. On the MSCOCO dataset, the fine-tuned approaches [69, 77, 200, 211] further merge the validation images into the training set, in order to largely increase the performance. However, we still use the original training set for a fair comparison with other prior approaches.

In addition to the quantitative evaluation, we present our image-to-text and text-to-image retrieval examples in Figure 6.8, which includes both success and failure cases. For each query sample, the top-5 candidates are retrieved, of which the ground-truth samples are highlighted in green. We notice that, the retrieved candidates are semantically related to the query sample in some extent, even for the failure cases.

Table 6.4: Comparison with the state-of-the-art approaches on Flickr30K for image-text retrieval. For the approaches without fine-tuning, we show the best results in blue color; For the ones with fine-tuning, the best results are highlighted with red color. Overall, our results with ResNet show state-of-the-art performance on this dataset.

Method	Image encoder	Text encoder	Image to Text			Text to Image		
			R@1	R@5	R@10	R@1	R@5	R@10
Without fine-tuning image encoders								
DCCA [49]	AlexNet	TF-IDF	16.7	39.3	52.9	12.6	31.0	43
DVSA [55]	AlexNet	RNN	22.2	48.2	61.4	15.2	37.7	50.5
UVSE [66]	VGG-19	RNN	23.0	50.7	62.9	16.8	42.0	56.5
mCNN [52]	VGG-19	CNN	33.6	64.1	74.9	26.2	56.3	69.6
VQA-aware [191]	VGG-19	RNN	33.9	62.5	74.5	24.9	52.6	64.8
GMM-FV [51]	VGG-16	GMM+HGLMM	35.0	62.0	73.8	25.0	52.7	66.0
m-RNN [190]	VGG-16	RNN	35.4	63.8	73.7	22.8	50.7	63.1
RNN-FV [185]	VGG-19	RNN	35.6	62.5	74.2	27.4	55.9	70.0
HM-LSTM [65]	AlexNet	RNN	38.1	-	76.5	27.7	-	68.8
DSPE [53]	VGG-19	HGLMM	40.3	68.9	79.9	29.7	60.1	72.1
sm-LSTM [68]	VGG-19	RNN	42.5	71.9	81.5	30.2	60.4	72.3
VSE++ [211]	ResNet-152	RNN	43.7	-	82.1	32.2	-	72.1
DualCNN [69]	ResNet-152	ResNet-152	44.2	70.2	79.7	30.7	59.2	70.8
RRF-Net [64]	ResNet-152	HGLMM	47.6	77.4	87.1	35.4	68.3	79.9
2WayNet [76]	VGG-16	GMM+HGLMM	49.8	67.5	-	36.0	55.6	-
DAN [67]	ResNet-152	RNN	55.0	81.8	89.0	39.4	69.2	79.1
CycleMatch (Ours)	VGG-19	RNN	51.4	80.6	88.1	38.5	71.0	81.3
CycleMatch (Ours)	ResNet-152	RNN	58.6	83.6	91.6	43.6	75.3	84.2
With fine-tuning image encoders								
DualCNN [69]	ft ResNet-152	ResNet-152	55.6	81.9	89.5	39.1	69.2	80.9
VSE++ [211]	ft ResNet-152	RNN	52.9	-	87.2	39.6	-	79.5
cnp + ctx + gen [200]	ResNet-152, ft VGG-19	RNN	55.5	82.0	89.3	41.1	70.5	80.1

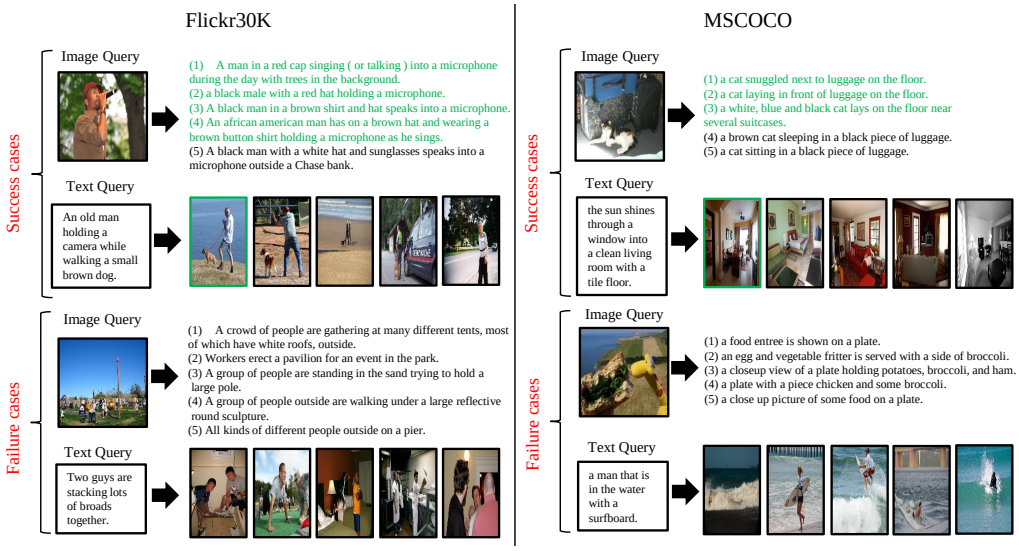


Figure 6.8: Qualitative results of our CycleMatch on Flickr30K and MSCOCO. Given one query, the top-5 candidates are retrieved. In the success cases, the correct matches are highlighted with green. In the failure cases, our method can still retrieve some reasonable false candidates related to the query.

Table 6.5: Comparison with the state-of-the-art approaches on MSCOCO for image-text retrieval. For the approaches without fine-tuning, the best results are highlighted with blue color; For the ones with fine-tuning, we show the best results in red color. Among the approaches without fine-tuning the image encoders, our approach with ResNet can achieve the state-of-the-art performance.

Method	Image encoder	Text encoder	Image to Text			Text to Image		
			R@1	R@5	R@10	R@1	R@5	R@10
Without fine-tuning image encoders								
STV [212]	VGG-19	RNN	33.8	67.7	82.1	25.9	60.0	74.6
DVSA [55]	AlexNet	RNN	38.4	69.9	80.5	27.4	60.2	74.8
GMM-FV [51]	VGG-16	GMM+HGLMM	39.4	67.9	80.9	25.1	59.8	76.6
m-RNN [190]	VGG-16	RNN	41.0	73.0	83.5	29.0	42.2	77.0
RNN-FV [185]	VGG-19	RNN	41.5	72.0	82.9	29.2	64.7	80.4
BiLSTM-Max [210]	ResNet-101	RNN	42.6	75.3	87.3	33.9	69.7	83.8
mCNN [52]	VGG-19	CNN	42.8	73.1	84.1	32.6	68.6	82.8
UVSE [66]	VGG-19	RNN	43.4	75.7	85.8	31.0	66.7	79.9
HM-LSTM [65]	AlexNet	RNN	43.9	-	87.8	36.1	-	86.7
order-embeddings [213]	VGG-19	RNN	46.7	-	88.9	37.9	-	85.9
DSPE [53]	VGG-19	HGLMM	50.1	79.7	89.2	39.6	75.2	86.9
VQA-aware [191]	VGG-19	RNN	50.5	80.1	89.7	37.0	70.9	82.9
DualCNN [69]	ResNet-50	ResNet-50	52.2	80.4	88.7	37.2	69.5	80.6
sm-LSTM [68]	VGG-19	RNN	53.2	83.1	91.5	40.7	75.8	87.4
2WayNet [76]	VGG-16	GMM+HGLMM	55.8	75.2	-	39.7	63.3	-
RRF-Net [64]	ResNet-152	HGLMM	56.4	85.3	91.5	43.9	78.1	88.6
VSE++ [211]	ResNet-152	RNN	58.3	-	93.3	43.6	-	87.8
CycleMatch (Ours)	VGG-19	RNN	55.1	83.5	91.3	43.7	76.7	88.4
CycleMatch (Ours)	ResNet-152	RNN	61.1	86.8	94.2	47.9	80.9	90.9
With fine-tuning image encoders								
DualCNN [69]	ft ResNet-50	ResNet-50	65.6	89.8	95.5	47.1	79.9	90.0
VSE++ [211]	ft ResNet-152	RNN	64.6	-	95.7	52.0	-	92.0
Gen-XRN [77]	ft ResNet-152	RNN	68.5	-	97.9	56.6	-	94.5
cnp + ctx + gen [200]	ResNet-152, ft VGG-19	RNN	69.9	92.9	97.5	56.7	87.5	94.8

6.4.5 Effect of feature encoders

As shown in Figure 6.3, we extract visual and textual features from off-the-shelf feature encoders. The proposed CycleMatch can be compatible with diverse feature encoders, but it is still encouraged to study the effect of different feature encoders on the performance. We report the results in Table 6.6.

Considering the image encoders, we use the VGG-19 and ResNet-152 models to extract the visual features and compare their results. We can see that, ResNet-152 has a considerable improvements over VGG-19 on all measurements, especially for R@1 accuracies. This shows the benefit of using more powerful CNN models for improving the visual embeddings. In addition, the feature dimension with ResNet-152 (*i.e.* 2,048) is lower than that with VGG-19 (*i.e.* 4,096). Therefore, in this work we take the ResNet-152 model as the preferable image encoder.

In terms of the text encoders, we test another two encoders apart from the RNN encoder. The first one is word2vec [188], which describes each word in the sentence with a 300-dimensional feature vector. We then compute the average of all the word features to represent the sentence feature. The second one is an expensive representation based on the Hybrid Gaussian-Laplacian mixture model (HGLMM) [51].

Table 6.6: Evaluation on the effect of different feature encoders on the performance of CycleMatch. By comparison, ResNet-152 is a superior image encoder and RNN is a more powerful text encoder.

Image encoder	Text encoder	Flickr30K						MSCOCO					
		Image to Text			Text to Image			Image to Text			Text to Image		
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
Effect of image encoders													
VGG-19	RNN	51.4	80.6	88.1	38.5	71.0	81.3	55.1	83.5	91.3	43.7	76.7	88.4
ResNet-152	RNN	58.6	83.6	91.6	43.6	75.3	84.2	61.1	86.8	94.2	47.9	80.9	90.9
Effect of text encoders													
ResNet-152	word2vec	48.1	78.7	87.4	37.7	70.8	81.1	55.9	83.8	91.8	44.7	79.1	87.7
ResNet-152	HGLMM	54.5	81.6	90.9	41.3	73.1	82.8	58.4	85.5	93.4	46.2	80.3	89.4
ResNet-152	RNN	58.6	83.6	91.6	43.6	75.3	84.2	61.1	86.8	94.2	47.9	80.9	90.9

Specifically, HGLMM computes a 18,000-dimension feature vector with 30 centers (*i.e.* 300*30*2). Similar to [53], we further reduce it to a 6,000-dimension feature vector in order to decrease the training complexity. As shown in Table 6.6, the RNN encoder is more powerful than both word2vec and HGLMM. In addition, the feature dimension based RNN (*i.e.* 4,096) is feasible and practical during training CycleMatch.

6.5 Chapter Conclusions

In this chapter, we have developed a novel embedding method for the multi-modal task of matching visual and textual representations. We proposed cycle-consistent embeddings to learn both intra-modal correlations and intra-modal consistency. Our approach taking advantage of multiple embedding techniques is able to outperform any single embedding method. The experimental results have demonstrated the superiority of our method over other embedding methods. In addition, we have presented two simple and efficient late-fusion approaches to increase the inference quality. The late-fusion inference can integrate different matching scores together without increasing the training complexity. Finally, our approach has shown state-of-the-art performance for cross-modal retrieval on Flickr30K and MSCOCO.

Future work. we will take into account local relations when matching images and sentences, for example, semantic correlations between visual regions and phases. One potential solution is to exploit the attention mechanism to localize the objects corresponding to the phase description.

Chapter 7

Joint Matching and Classification

In Chapters 2-6, we have proposed several methods to solve the classification and retrieval themes, separately. Unlike many existing approaches which focus only on either multi-modal matching or classification, we aim to study how we can integrate the two tasks together to help promote each other (RQ6).

In this chapter, we propose a unified **Net**work to jointly learn **M**ulti-modal **M**atching and **C**lassification (MMC-Net) between images and texts. The proposed MMC-Net model can seamlessly integrate the matching and classification components. It first learns visual and textual embedding features in the matching component, and then generates discriminative multi-modal representations in the classification component. Combining the two components in a unified model can help in improving their performance simultaneously. Moreover, we present a multi-stage training algorithm by minimizing both of the matching and classification loss functions. Experimental results on four well-known multi-modal benchmarks demonstrate the effectiveness and efficiency of the proposed approach, which achieves competitive performance for multi-modal matching and classification compared to the state-of-the-art approaches.

Keywords
Multi-modal matching, Multi-modal classification, Deep neural networks, Multi-stage training

7.1 Introduction

The problem of multi-modal analytic has attracted increasing attention due to a drastic growth of multimedia data such as image, video and text. Particularly, multi-modal matching has been studied for decades, with the aim of searching for a latent space, where visual and textual features can be unified to be latent embeddings. The hypothesis is that different modalities have semantically related properties that can be distilled into a common latent space. Early approaches to learning latent embeddings are based on the Canonical Correlation Analysis (CCA) [61], which is effective at maximizing the high correlation between visual and textual features in the latent space. Driven by the increasing progress of deep learning, many works [52, 55, 66, 181] have been dedicated to developing deep matching networks to learn discriminative latent embeddings and train the networks by using a bi-directional rank loss function. They have achieved state-of-the-art performance on many well-known multi-modal benchmarks [53, 64, 67, 76].

However, learning latent embeddings is influenced by the notable variance in images or texts. For example, in Figure 7.1, five sentences annotated by humans are provided to describe the same image. The input image and five sentences are projected into a latent space. One can observe that these sentences have significant variance on representing the visual content. Although they can consistently describe the main objects in the scene including 'girl' (or 'child') and 'bicycle' (or 'bike'), they still present great variance in terms of other objects, *e.g.* 'bench', 'table' and 'leaves'. This issue makes it difficult to perform image-text matching.

To address this issue, in this work we aim to introduce a classification component to learn more robust latent embeddings. Our motivation is that object labels can typically provide more consistent and less biased information than sentences. As can be seen in Figure 7.1, object labels contain the most important concepts in the image, such as 'Person' and 'Bicycle' which are commonly mentioned in all of the five sentences. On the other hand, some visual concepts, which are subjectively described in some of the sentences (*e.g.* 'leaves' and 'sweater') will not appear in the ground-truth labels. Hence, using the object labels as additional supervisory signals is beneficial to correct the biased descriptions and improve the matching between images and texts. Motivated by the mutual complements between matching and classification, we raise the research question **RQ 6: How can we design a unified network for joint multi-modal matching and classification?**

To tackle the question, we propose a unified **Net**work for joint **M**ulti-modal **M**atching and **C**lassification (MMC-Net in Figure 7.2). First, the matching component transforms the input visual and textual features, respectively, via a couple of fully-connected layers and a fusion module. The matching loss is imposed on the outputs of the two fusion modules to maximize their correlation. Then, the classification

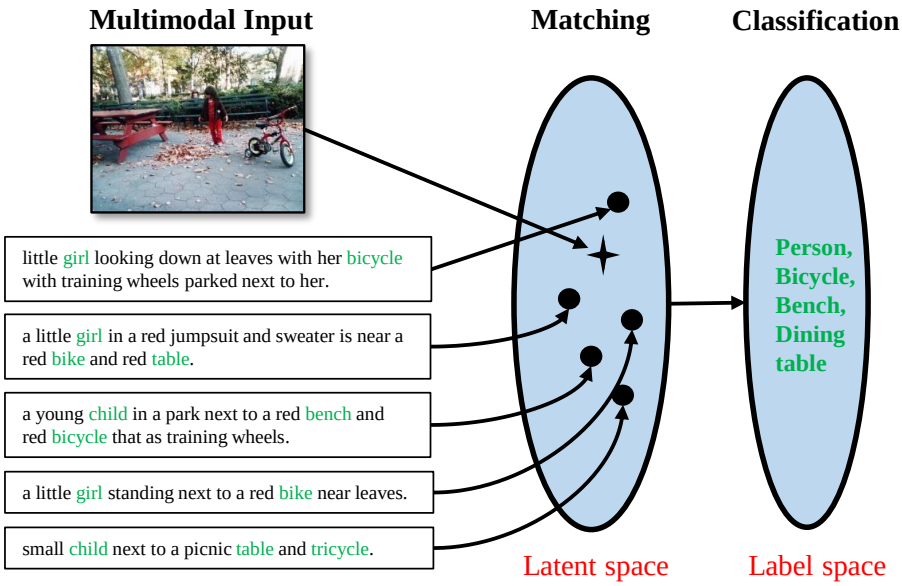


Figure 7.1: Example of joint multi-modal matching and classification. Given one image and its descriptive sentences, they are first co-embedded into a latent space for matching (in red and blue). Then, the visual and textual embedding features are integrated to be a multi-modal representation for classification. In the input sentences, the words related to the ground-truth object labels are in green.

component is built upon the visual and textual embedding features. A compact bilinear pooling module is used to generate a multi-modal representation vector, based on which the classification loss is computed to predict object labels. In this way, the proposed MMC-Net can jointly learn the latent embeddings and the multi-modal representation in a unified model. On the one hand, the classification component is beneficial to alleviate the biased input, so that the model can learn better robust latent embeddings. On the other hand, the matching component is able to bridge the modality gap between vision and language, and therefore combining visual and textual embedding features can produce a discriminative multi-modal representation for classification.

The contributions of this work are as follows:

- We propose a novel deep multi-modal network (*i.e.* MMC-Net), where the matching and classification components can be seamlessly integrated and help promote each other jointly. MMC-Net is a general architecture that is potentially applicable to diverse multi-modal tasks related to matching and classification.
- We present a multi-stage training algorithm by incorporating the matching and classification loss. It can make the matching and classification components more compatible in a unified model.
- Results on four well-known multi-modal benchmarks demonstrate that MMC-Net outperforms the baseline models that are built for either matching or

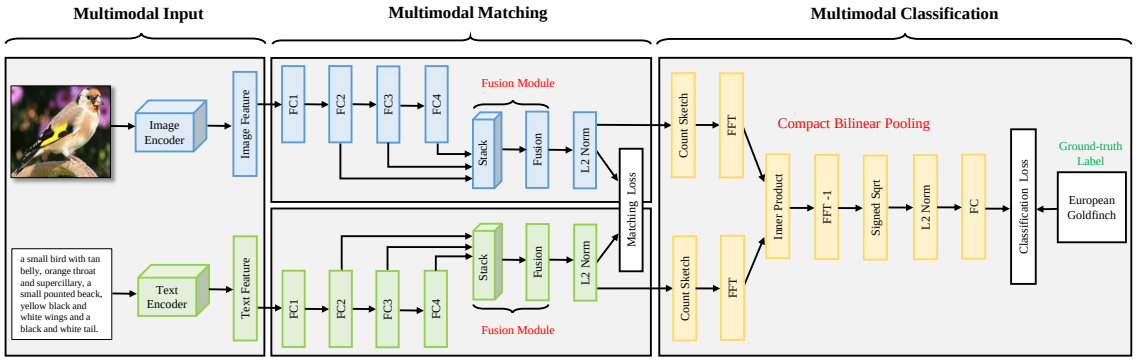


Figure 7.2: The overview architecture of our proposed MMC-Net for joint multi-modal matching and classification. It comprises three key components. (1) The multi-modal input aims to capture visual and textual representations from off-the-shelf encoders (*e.g.* CNN and word2vec). (2) In the matching component, four fully-connected layers in both of the image and text branches are developed to learn the latent embeddings. (3) Based on the visual and textual embedding features, the classification component utilizes a compact bilinear pooling module which can generate a high-order multi-modal representation to perform the prediction. The entire network can be trained with a matching loss and a classification loss.

classification (*i.e.* MM-Net and MC-Net). In addition, our approach achieves competitive performance compared to current state-of-the-art approaches.

The rest of this paper is organized as follows. Section 7.2 introduces the proposed MMC-Net model, and Section 7.3 details its training and inference procedures. Comprehensive experiments in Section 7.4 are used to evaluate the approach. Finally, Section 7.5 concludes the paper and discusses the future work.

7.2 Joint Matching and Classification Network

Overall architecture. Figure 7.2 illustrates the overview architecture of MMC-Net, which mainly consists of three components: multi-modal input, multi-modal matching and multi-modal classification. Given an image and its corresponding text, MMC-Net first utilizes off-the-shelf feature encoders to extract the visual and textual features, respectively. Next, in the multi-modal component, two groups of four fully-connected layers are used in both image and text branches to learn a latent space, where its objective is to minimize the matching loss between the related images and texts. Moreover, the multi-modal classification component is built upon the visual and textual embedding features. We employ a compact bilinear pooling module to generate a high-order and efficient multi-modal representation. The classification loss is computed with respect to the pre-defined ground-truth labels. Next, we will detail each of the three components.

7.2.1 Multi-modal input

In a data collection with N matching image-text pairs, (x_i,y_i) represent the encoded visual and textual features, $i = 1, \ldots, N$. Taking these features as input instead of the raw data enables to train the entire network effectively. Also, any common feature encoders are potentially applicable for this network.

Image encoder: we use the powerful CNN model, ResNet-152 [10], which is pre-trained on ImageNet [5]. First, the CNN model is recast to its fully convolutional network (FCN) counterpart, to extract richer region representations. Then we set the smaller side of the image to 512 and isotropically resize the other side. The last max-pooling layer in ResNet-152 is averaged to generate a 2048-dimensional feature vector. Compared with the widely-used VGG feature [7] (*i.e.* 4096-dim), ResNet-152 can provide more discriminative visual representation, while decreasing the feature dimensions (2048 v.s. 4096). The extracted image feature is then fed into the image branch of the matching component.

Text encoder. we employ the simple yet efficient word2vec [188] to represent sentence-level texts. It provides a 300-dimensional feature vector, which is often called Mean vector. Notably, more informative text encoders can be developed based on word2vec, for example the Hybrid Gaussian-Laplacian mixture model (HGLMM) [51] that computes a 18000-dimensional feature vector with 30 centers (*i.e.* 300*30*2). However, we still use the standard Mean vector due to its high efficiency and low dimensionality. Nevertheless, we clarify that any common text encoders can be potentially adopted to the MMC-Net model.

7.2.2 Multi-modal matching

The multi-modal matching component contains three aspects: latent embedding, fusion module and matching loss.

Latent embedding

As shown in Figure 7.2, the matching component develops two branches of four fully-connected layers to simultaneously project visual and textual features into a discriminative latent space. Note that the parameters of the two branches (drawn in blue and green) are unshared due to the modality specialization. The channels from FC1 to FC4 are set to {2048, 512, 512, 512} in both of the two branches. First, the input visual and textual features are normalized with the batch normalization (BN) [135]. Then FC1 is regularized by a dropout layer with 0.5 probability, and instead other fully-connected layers are regularized with the BN layer. ReLU is used after the fully-connected layers.

Fusion module

Exploiting multi-layer features has been well-studied in many deep neural networks [18, 26, 31, 107], as it allows to take advantage of different levels of hidden representations in the networks. Driven by this, we introduce a fusion module to generate a multi-layer embedding feature. Since the FC2, FC3 and FC4 layers have the same number of channels, it is feasible to stack their feature vectors together. Then we employ a convolutional operation to learn adaptive weights while fusing the three layers.

We denote the stack layer in the two branches as $S(x_i)$ and $S(y_i)$, respectively. The stack layer, a 512×3 matrix, is convolved by the convolutional filter, which has a size of 1×1×3. Note that, the three weights are shared over the spatial dimensions of the stack layer. We can compute the fused visual feature $f(x_i)$ and textual feature $g(y_i)$ by

$$f(x_i) = W_I^{fuse} \odot S(x_i) + b_I^{fuse}, \tag{7.1}$$

$$g(y_i) = W_T^{fuse} \odot S(y_i) + b_T^{fuse}, \tag{7.2}$$

where W_I^{fuse} and W_T^{fuse} are the fusion weights to be learned (*i.e.* 3 elements) b_I^{fuse} and b_T^{fuse} are the bias vectors (*i.e.* 512 elements). The operator $\odot$ represents the convolutional operation.

Although the common element-wise operators such as sum-pooling and inner product are simple to compute, they do not adapt the importance of different layers. Another fusion approach is concatenating the three 512-Dim vectors into one 3*512-Dim vector. However, the concatenation output will increase the feature dimensionality and make it more expensive to compute the matching loss. To summarize, the convolutional fusion module can provide marked performance improvements, while it has a minimal increase to the total parameters used in the network.

Matching loss

As a common practice, the matching distance between $f(x_i)$ and $g(y_i)$ is computed with the cosine distance [52, 53, 76]

$$d(f(x_i), g(y_i)) = 1 - \frac{f(x_i) \cdot g(y_i)}{||f(x_i)|| \cdot ||g(y_i)||}. \tag{7.3}$$

Smaller distances indicate more similar image-text pairs. Both $f(x_i)$ and $g(y_i)$ are L2-normalized before computing their cosine distance. To preserve the similarity constraints in the latent space, we define the matching loss based on an efficient bi-directional rank loss function, similar to [53, 181, 214]. The loss function needs to handle the two triplets, $(x_i, y_i, y_{i,k}^-)$ and $(y_i, x_i, x_{i,k}^-)$, where $x_{i,k}^- \in X_i^-$ and $y_{i,k}^- \in Y_i^-$

are the negative images and texts, $k = 1, \cdots, K$. To exploit more representative non-matching pairs, we pick the top K most dissimilar candidates in each mini-batch. Intuitively, this loss function is designed to decrease the distances of matching pairs (*e.g.* x_i and y_i) and increase the distances of non-matching pairs (*e.g.* x_i and $y^-_{i,k}$, y_i and $x^-_{i,k}$). Formally, the matching loss based on the fused features is:

$$\begin{aligned}\mathcal{L}^{fuse}_{mat} = \sum_{i=1}^{N}\sum_{k=1}^{K} \max\Big[0, d(f(x_i), g(y_i)) - d(f(x_i), g(y^-_{i,k})) + m\Big] \\ + \alpha \max\Big[0, d(f(x_i), g(y_i)) - d(f(x^-_{i,k}), g(y_i)) + m\Big],\end{aligned} \tag{7.4}$$

where m is a margin parameter, and α is used to balance the importance of the two triplets. Minimizing this loss cost will lead to a desirable latent space, where the matching distance $d(f(x_i), g(y_i))$ should be smaller than any of the non-matching ones $d(f(x_i), g(y^-_{i,k}))$ and $d(f(x^-_{i,k}), g(y_i))$, $\forall x^-_{i,k} \in X^-_i, \forall y^-_{i,k} \in Y^-_i$.

In Figure 7.3, we make use of the t-SNE algorithm [207] to visualize our embedding features (*i.e.* $f(x_i)$ and $g(y_i)$). We use the 1,000 images and 5,000 texts from the MSCOCO test set. It can be seen that in the distribution map an image feature (in red) is properly surrounded by several related text features (in green), as each image is annotated by five ground-truth matching texts in the dataset. Therefore, this visualization shows that our embedding model can align the images and texts due to learning their semantic correlation. In addition, some images and texts corresponding to the points are shown in the windows. We can see that the embeddings can cluster related images and texts together.

7.2.3 Multi-modal classification

The classification component aims to incorporate the visual and textual embedding features and then generates a multi-modal representation for predicting object labels. In the following, we detail the classification component including a bilinear pooling module and classification loss.

Bilinear pooling

We take advantage of a bilinear pooling module to incorporate visual and textual embedding features learned in the matching component. The bilinear pooling [215] aims to model the pair-wise multiplicative intersection between all elements of two vectors. It can generate more expressive features than other basic operators such as element-wise sum or product. The standard bilinear pooling is formulated with

$$\mathcal{B}(x_i, y_i) = f(x_i)^T g(y_i), \tag{7.5}$$

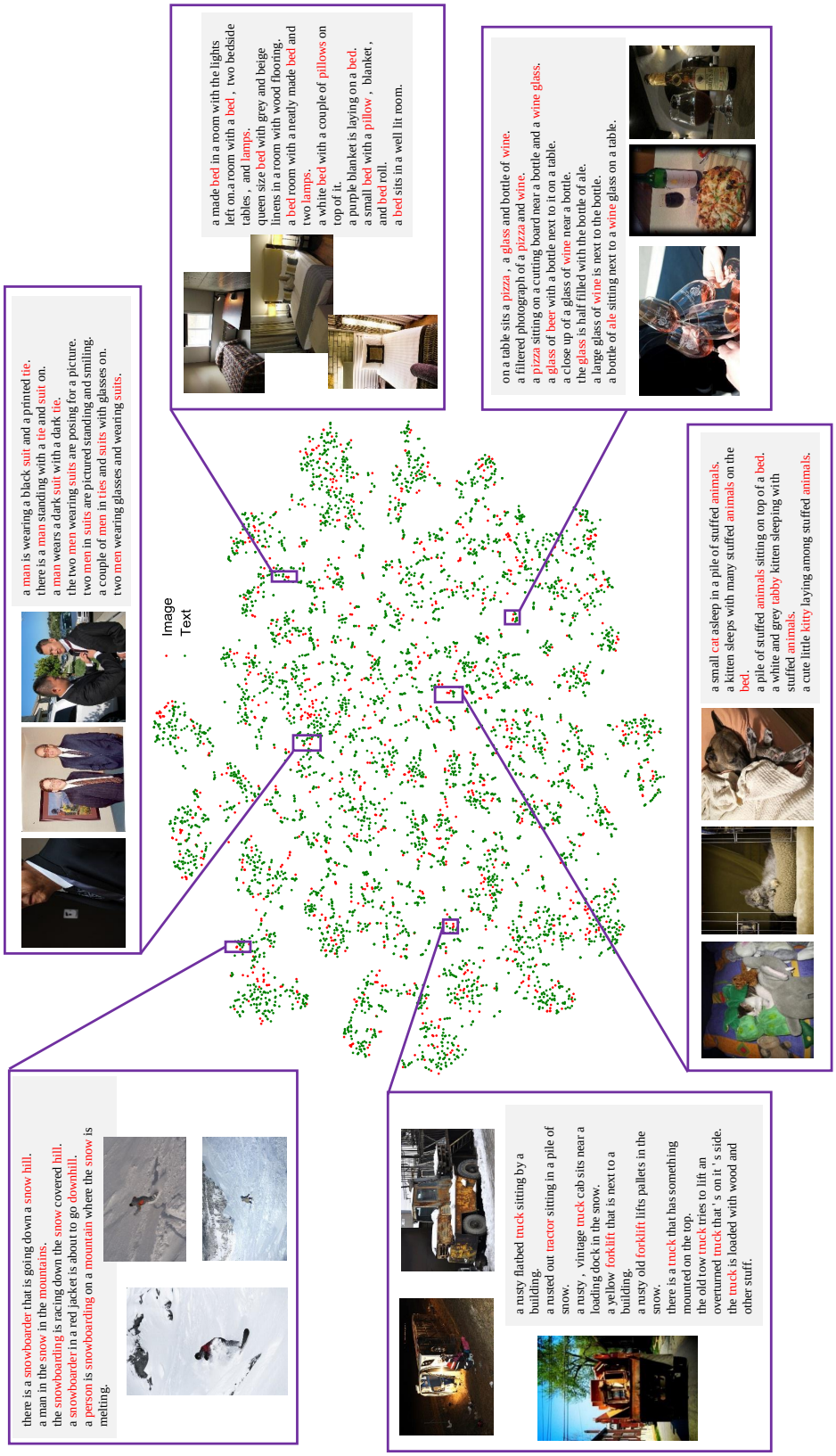


Figure 7.3: Visualization of the visual and textual embedding features learned in the matching component. Each image (in red) is related to several corresponding texts (in green). We present some images and texts corresponding to the points in the distribution map. Some semantic words are highlighted in red.

Algorithm 2: CBP with latent embedding features

1: **Input:** $f(x_i) \in \mathbb{R}^M$, $g(y_i) \in \mathbb{R}^M$
2: **Output:** $\mathcal{B}(x_i, y_i) \in \mathbb{R}^D$
3: **Initialize hash functions:** h_1, s_1, h_2, s_2
  For $j \leftarrow 1 \cdots M$
    sample $h_1[j]$, $h_2[j]$ from $\{1, \cdots, D\}$
    sample $s_1[j]$, $s_2[j]$ from $\{-1, 1\}$
  End for
4: **Compute count sketches:**
  $\hat{f}(x_i) = [0, \cdots, 0]$, $\hat{g}(y_i) = [0, \cdots, 0]$
  For $j \leftarrow 1 \cdots D$
    $\hat{f}(x_i)[h_1[j]] = \hat{f}(x_i)[h_1[j]] + s_1[j] \cdot f(x_i)[j]$
    $\hat{g}(y_i)[h_2[j]] = \hat{g}(y_i)[h_2[j]] + s_2[j] \cdot g(y_i)[j]$
  End for
5: **Convolution of Count Sketches:**
  $\mathcal{B}(x_i, y_i) = \mathrm{FFT}^{-1}(\mathrm{FFT}(\hat{f}(x_i)) \circ \mathrm{FFT}(\hat{g}(y_i)))$,
  where the $\circ$ denotes element-wise multiplication.

Since $f(x_i)$ and $g(y_i)$ are $1 \times M$ vectors (*i.e.* $M = 512$), $\mathcal{B}(x_i, y_i)$ becomes an $M \times M$ matrix that is then reshaped to be a $1 \times M^2$ vector. Due to the high dimensionality of the bilinear vector (*i.e.* M^2), we instead use the compact bilinear pooling (CBP) variant [216], which can decrease the dimensionality to D (where $D \ll M^2$) while retaining the strong discrimination. In contrast to [216, 217] in which they simply perform the CBP module with the input visual or textual features, we build the CBP module based on the latent embeddings to generate a multi-modal feature vector (Figure 7.2).

The computational procedure of the CBP module is detailed in Algorithm 2. At first, we initialize several hashing functions from the pre-defined sets. Then, it computes the count sketches [218] to maintain linear projections of a vector with several random vectors. Finally, we make use of the Fast Fourier Transformation (FFT) to compute the convolution of the count sketches, and produce a bilinear vector $\mathcal{B}(x_i, y_i)$ by an inverse FFT. The count sketches have the properties:

$$E[\langle \hat{f}(x_i), \hat{g}(y_i) \rangle] = \langle f(x_i), g(y_i) \rangle, \tag{7.6}$$

$$Var[\langle \hat{f}(x_i), \hat{g}(y_i) \rangle] \leq \frac{1}{D}(\langle f(x_i), g(y_i) \rangle^2 + \|f(x_i)\|^2 + \|g(y_i)\|^2). \tag{7.7}$$

Next, the bilinear vector $\mathcal{B}(x_i, y_i)$ is processed by a signed square-root layer and an L2 normalization layer. Then, we employ a fully-connected layer to estimate the prediction. Assume that there are C object labels pre-defined in the dataset, the

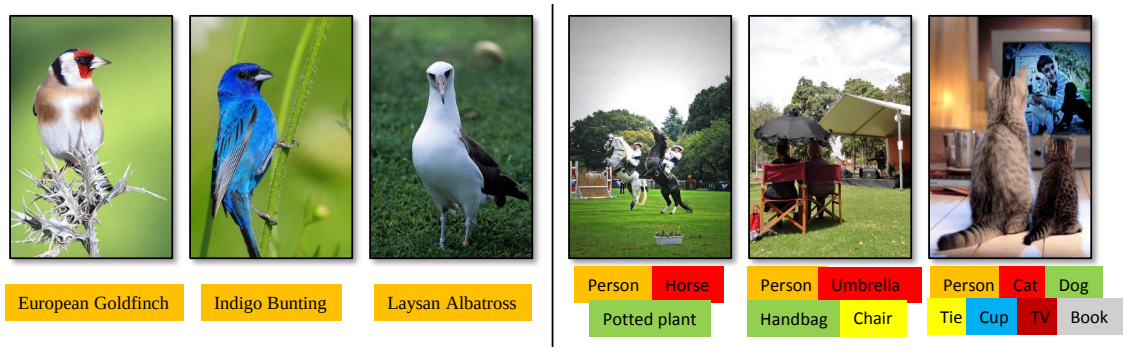


Figure 7.4: Left: Examples of single-label images from CUB-Bird [139]. Right: Examples of multi-label images from MSCOCO [117]. The ground-truth labels are shown under the images.

j-th class probability is predicted with

$$a_{i,j} = \sum_{k=1}^{D} W_{j,k} \mathcal{B}(x_i, y_i)_k, j = 1, \cdots, C. \tag{7.8}$$

where $W_{j,k}$ is the parameter matrix with the size of $D \times C$. For simplicity, we do not show the signed square-root and the L2 normalization in this formulation.

Classification loss

The objective of the classification component is to minimize the loss cost of the prediction with respect to the given ground-truth labels. Figure 7.4 shows some images that are annotated by single label or multiple labels. We need to utilize different loss functions for single-label and multi-label classification, respectively.

1) Single-label classification. For example, the fine-grained classification in the left of Figure 7.4, each image is labelled with a fine bird category. To train the classification component, we use the softmax loss function

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} \delta(g_i = j) \log p_{i,j}, \tag{7.9}$$

$$p_{i,j} = \frac{\exp(a_{i,j})}{\sum_{k=1}^{C} \exp(a_{i,k})}, \tag{7.10}$$

where g_i is the ground-truth label corresponding to x_i. $\delta(g_i = j)$ is 1 when $g_i = j$, otherwise is 0.

2) Multi-label classification. As shown in the right of Figure 7.4, images annotated with multiple labels can provide richer information about the visual content. Although many of these labels may appear in the input text, they can still offer

complementary labels which are ignored in the text due to less visual attention. We employ the sigmoid cross-entropy loss function to supervise the multi-label classification. The total cost sums up K of element-wise loss terms

$$\mathcal{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} g_{i,j}^{'} \log p_{i,j}^{'} + (1 - g_{i,j}^{'}) \log(1 - p_{i,j}^{'}), \tag{7.11}$$

$$p_{i,j}^{'} = \frac{1}{1 + \exp(-a_{i,j})}, \tag{7.12}$$

where $g_{i,j}^{'} \in \{0, 1\}$ is the ground-truth label indicating the absence or presence of the j-th class.

7.3 Training and Inference

This section describes the training procedure of the MMC-Net model. Also, we present the inference manner for multi-modal matching and classification.

Multi-stage training procedure

The optimization objective in the model is to minimize the total training loss which merges the matching and classification loss together

$$\min_{W} \mathcal{L}_{total} = \mathcal{L}_{mat} + \beta \mathcal{L}_{cls}, \tag{7.13}$$

where the parameter β is used to regulate the two loss terms. The parameters W in the network mainly contains W_I and W_T in the image and text branches, and W_{CBP} in the compact bilinear pooling module.

We propose a multi-stage training algorithm to better model the matching and classification components. As summarized in Algorithm 3, the training procedure consists of three stages. During the first stage, we train the matching component with the loss $\mathcal{L}_{mat}$. For the second stage, we use the loss $\mathcal{L}_{cls}$ to train the parameters in the classification component. In this stage, only the parameters in the classification component are updated while the parameters in the matching component are all frozen. In the third stage, the model is initialized by the parameters learned in the first and second stages. It aims to jointly fine-tune the whole network based on the total loss $\mathcal{L}_{total}$. Due to using this multi-stage fashion, it is feasible to promote the training of the entire network and maintain the high performance.

Inference procedure

We present the inference procedure for multi-modal matching and classification.

Algorithm 3: Multi-stage Training Algorithm for MMC-Net.

1: **The first stage:** train the matching component.
initialize: learning rate λ_1, training iterations T_1, $t = 0$.
while $t < T_1$ **do**
$t \leftarrow t + 1$
compute the matching loss $\mathcal{L}_{mat}$ in Eq.(7.4);
update the parameters in the image and text branches:
$W_I^{(t)} = W_I^{(t-1)} - \lambda_1^{(t)} \frac{\partial \mathcal{L}_{mat}}{\partial W_I^{(t-1)}}$;
$W_T^{(t)} = W_T^{(t-1)} - \lambda_1^{(t)} \frac{\partial \mathcal{L}_{mat}}{\partial W_T^{(t-1)}}$;
end while

2: **The second stage:** train the classification component.
initialize: learning rate λ_2 ($< \lambda_1$), training iterations T_2, $t = 0$.
while $t < T_2$ **do**
$t \leftarrow t + 1$
compute the classification loss $\mathcal{L}_{cls}$ in Eq.(7.9) or Eq.(7.11);
update the parameters in the compact bilinear pooling module:
$W_{CBP}^{(t)} = W_{CBP}^{(t-1)} - \lambda_2^{(t)} \frac{\partial \mathcal{L}_{cls}}{\partial W_{CBP}^{(t-1)}}$;
end while

3: **The third stage:** jointly fine-tune the whole network.
initialize: learning rate λ_3 ($< \lambda_2$), training iterations T_3, $t = 0$.
while $t < T_3$ **do**
$t \leftarrow t + 1$
compute the total loss in Eq.(7.13);
update all the parameters in the network:
$W_I^{(t)} = W_I^{(t-1)} - \lambda_1^{(t)} \frac{\partial \mathcal{L}_{total}}{\partial W_I^{(t-1)}}$;
$W_T^{(t)} = W_T^{(t-1)} - \lambda_1^{(t)} \frac{\partial \mathcal{L}_{total}}{\partial W_T^{(t-1)}}$;
$W_{CBP}^{(t)} = W_{CBP}^{(t-1)} - \lambda_2^{(t)} \frac{\partial \mathcal{L}_{total}}{\partial W_{CBP}^{(t-1)}}$;
end while

(1) Multi-modal matching: For the image-to-text matching, given a query image x_q, its purpose is to search for relevant texts *w.r.t.* x_q from a text database Y. Likewise, the text-to-image matching aims to retrieve related images from an image database X, given a query text y_q. In the MMC-Net model, the fused visual and textual features learned in the fusion module are used to compare the matching distance, denoted as $d(f(x_q), g(y_i))$ or $d(f(x_i), g(y_q))$, where $y_i \in Y, x_i \in X$. The k-nearest neighbor (k-NN) search is used to find the top-k most similar candidates.

(2) Multi-modal classification: Its inference is based on the probabilities predicted by the last fully-connected layer in the classification component. For the single-label case, the element that has the maximum probability corresponds to the predicted class. As for the multi-label case, the items whose probabilities in the prediction are more than 0.5 are estimated to contain the corresponding object classes.

Table 7.1: Summary of four multi-modal datasets used in the experiments. TPI indicates the number of matching Texts Per Image.

Dataset	#Total	#Category	#Training	#Test	#TPI
Pascal Sentence	1,000	20	800	100	5
MSCOCO	∼120K	80	82,783	1,000	5
Flowers	8,189	102	2,040	6,149	10
CUB-Bird	11,788	200	5,994	5,794	10

7.4 Experiments

In this section, we evaluate the performance of the proposed MMC-Net on four well-known multi-modal benchmarks. We first introduce the configuration in the experiments, including the datasets, evaluation metrics, parameter settings and baseline models. Then we assess the performance of MMC-Net for tasks of multi-modal matching and classification and compare its results with those of the baseline models. Furthermore, we conduct the ablation study to fully analyze MMC-Net. Lastly, we compare our results with the state-of-the-art approaches.

7.4.1 Experimental setup

Dataset protocols

We perform the experiments on four well-known multi-modal datasets. Some image and text examples are shown in Figure 7.5.

Pascal Sentence [219]. It contains 1,000 images from 20 categories (50 images per category), and one image is described by five different sentences. We pick 800 images for training (40 images per category), 100 images for validation (5 images per category), and 100 images for test (5 images per category). In total, there are $40 * 20 * 5 = 4,000$ image-text training pairs, $5 * 20 * 5 = 500$ validation pairs, and $5 * 20 * 5 = 500$ test pairs.

MSCOCO [117]. It includes 82,783 training images and 40,504 validation images in total. We pick five descriptive sentences for one image and generate $82,783 * 5 = 413,915$ training pairs. For a fair comparison, we use the same 1,000 test images used in recent works [52, 53, 76].

Flowers [138]. This dataset [138] contains 102 classes with a total of 8,189 images. 2,040 images (train+val) are used in the training stage and the rest 6,149 images are for testing. Reed *et al.* [195] collected fine-grained visual descriptions for these images by using the Amazon Mechanical Turk (AMT) platform. One image is described by ten sentence-level descriptions. Therefore, we can obtain $2040 * 10 = 20,400$ training pairs and $6149 * 10 = 61,490$ testing pairs.

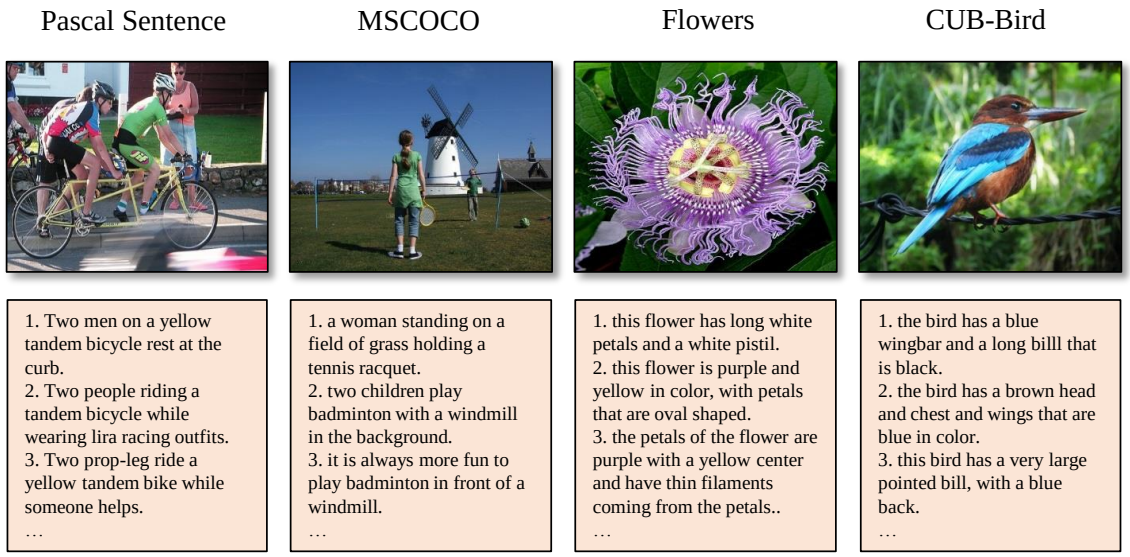


Figure 7.5: Example of four multi-modal datasets. Several textual descriptions are listed for each image.

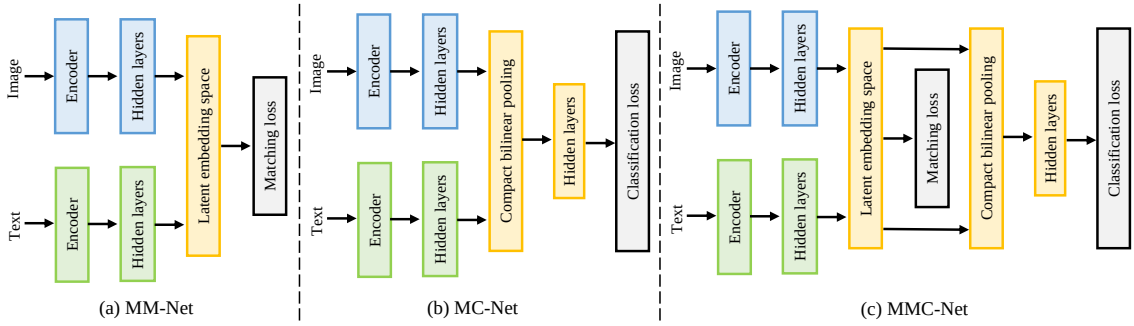


Figure 7.6: Conceptual illustration of three multi-modal networks. (a) Multi-modal Matching Network. (b) Multi-modal Classification Network. (c) Multi-modal Matching and Classification Network. Note that, the parameters in the image and text branches are unshared, as drawn in blue and green.

CUB-Bird [139]. It contains 11,788 bird images from 200 categories. 5,994 images are for training, and 5,794 images are for testing. Similarly, ten sentences are provided to describe one image [195]. As a result, it has $5994 * 10 = 59,940$ pairs for training, and $5794 * 10 = 57,940$ pairs for testing.

Evaluation Metrics

We evaluate the performance of multi-modal matching and multi-modal classification, separately. (1) For multi-modal matching, We employ the widely-used retrieval metric R@K, which is the recall rate of a correctly retrieved ground-truth at top K candidates (e.g. $K = 1, 5, 10$) [55, 190]. It includes results of both image-to-text (I→T) and text-to-image retrieval (T→I). (2) Considering multi-modal classification, We compute the Top-1 classification accuracy for Pascal Sentence, Flowers and

CUB-Bird. Since MSCOCO is a multi-label classification dataset, we evaluate the performance on it using the average precision with the average precision (AP) across multiple classes.

Implementation details

We implemented the proposed approach based on the publicly available Caffe library [130]. It is important to shuffle the training samples randomly during the data preparation stage. The hyper-parameters were evaluated on the validation set of each dataset. For instance, we set $\alpha = 2$ and $m = 0.1$ while computing the matching loss function on all the datasets. The number of non-matching pairs in the negative sets was $K = 20$ for Pascal Sentence, Flowers and CUB-Bird, and $K = 50$ for MSCOCO. We used a mini-batch size of 128 for Pascal Sentence, Flowers and CUB-Bird, and 1500 for MSCOCO. Note that, we use a larger K and mini-batch size for MSCOCO, because it has enormously more training samples, compared to the other three datasets. We trained the model using SGD with a weight decay of 0.0005, a momentum of 0.9. The learning rate was initialized with 0.1 and was divided by 10 when the loss stoped decreasing.

Baseline Models.

To verify the effectiveness of the proposed MMC-Net, we implemented two baseline models. (1) **MM-Net**: a baseline model for multi-modal matching as illustrated in Figure 7.6(a). It only contains the matching component of the MMC-Net (Figure 7.2), which is trained with the matching loss. (2) **MC-Net**: a baseline model for multi-modal classification as illustrated in Figure 7.6(b). It has the similar architecture as the MMC-Net, however, it does not compute the matching loss between visual and textual features. MC-Net is only trained with the classification loss.

7.4.2 Results on multi-modal retrieval

We conduct the cross-modal retrieval experiments on the four datasets. To verify the effectiveness of adding a classification component in MMC-Net, we use the baseline MM-Net for comparison. Table 7.2 and Table 7.3 report the results of image-to-text and text-to-image retrieval, respectively. Overall, MMC-Net can achieve considerable improvements over MM-Net for both I→T and T→I retrieval. These results reveal that the classification component in MMC-Net can help in improving the learning of embedding features in the matching component. Moreover, we can observe more insights from these results as follows:

Table 7.2: Image-to-text retrieval results compared between MMC-Net and MM-Net. The proposed MMC-Net can outperform the baseline MM-Net with considerable gains across all the four datasets.

Method	Pascal Sentence			MSCOCO			Flowers			CUB-Bird		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
MM-Net	47.0	85.0	92.0	55.5	84.2	91.4	58.1	82.5	88.5	32.5	61.4	72.5
MMC-Net	52.0	87.0	93.0	57.0	85.8	92.7	78.7	93.9	96.0	39.2	66.9	76.4

Table 7.3: Text-to-image retrieval results compared between MMC-Net and MM-Net. Compared to MM-Net, MMC-Net can achieve better retrieval results.

Method	Pascal Sentence			MSCOCO			Flowers			CUB-Bird		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
MM-Net	38.4	80.6	88.6	44.7	79.5	89.5	32.7	46.4	52.9	18.3	25.6	28.8
MMC-Net	41.0	81.2	92.5	46.2	80.8	90.5	43.6	54.8	58.6	25.8	31.4	34.5

- By comparison with MM-Net, MMC-Net yields more performance gains on Flowers and CUB-Bird than Pascal Sentence and MSCOCO. For example, the performance gap between MMC-Net and MM-Net is below 5% on Pascal Sentence and MSCOCO, but above 5% on Flowers and CUB-Bird across all the measurements. One reason is that both Flowers and CUB-Bird are fine-grained datasets, and the textual descriptions cannot fully represent the discrimination among different samples. Hence, the results of MM-Net are limited on these two datasets. Instead, MMC-Net can make use of fine-grained class labels to enhance the discriminative abilities when matching images and texts.
- The results of T→I retrieval are lower than those of the I→T retrieval on the four datasets. This is because each image can retrieve several related textual descriptions, but one text is corresponded to only one matched image. We believe that refining the datasets is a favorable solution to narrow the performance gap between the I→T and T→I retrieval.
- For Flowers and CUB-Bird, their results are still not satisfactory, especially for the T→I retrieval. Currently, the fine-grained multi-modal matching still remains challenging, but it is a promising research direction in the field.

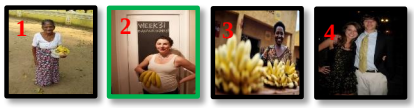
In addition, we present the qualitative retrieval results as shown in Figure 7.7. We can observe that MMC-Net obtains better retrieved candidates than MM-Net, for both I→T and T→I retrieval. Furthermore, we visualize the visual and textual embedding features learned in the matching component of MMC-Net. As mentioned earlier in 7.3, it has shown the embedding map with the MSCOCO test set.

7.4.3 Results on multi-modal classification

Next, we conduct the multi-modal classification experiments on the datasets. To demonstrate the benefit of using a matching component for classification, we compare the MMC-Net model with the baseline MC-Net model. Table 7.4 reports the

Query Image	MM-Net: Retrieved texts	MMC-Net: Retrieved texts
Pascal Sentence 	1. People riding tandem bicycle. 2. Two prop-leg ride a yellow tandem bike while someone helps. 3. Young man wearing jeans and helmet rides his motorcycle in front of a small crowd. 4. A man wearing a helmet does a wheelie on a motorcycle as a crowd watches.	1. Two prop-leg ride a yellow tandem bike while someone helps. 2. People riding tandem bicycle. 3. Two people riding a tandem bicycle while wearing lira racing outfits. 4. Young man wearing jeans and helmet rides his motorcycle in front of a small crowd.
MSCOCO 	1. a man putting together a kite on the floor of a room. 2. man folding banner while holding stick in unfinished carpet. 3. a man folding a giant paper airplane on the floor. 4. a tiny toddler carries a giant bookbag and bag.	1. a man putting together a kite on the floor of a room. 2. man folding banner while holding stick in unfinished carpet. 3. a man folding a giant paper airplane on the floor. 4. a man inside a room putting together a white kite.
Flowers 	1.this flower is pink and white in color, with petals that have pink veins. 2. this pink flower has several filaments sticking out of the receptacle. 3. this flower has pale pink petals with veins and a white center. 4. this flower has petals that are pink with long stamen.	1.this flower is pink and white in color, with petals that have pink veins. 2. this flower has pale pink petals with veins and a white center. 3. this flower has very light pink petals that have darker pink veins, a yellow ovary, and white stamen. 4. this pink flower has several filaments sticking out of the receptacle.
CUB-Bird 	1. a dark brown beak with a long beak and large wingspan. 2. this bird has a dark grey color, with a large bill and long wingspan. 3. this dull colored bird is brown all over, has large wings and a long large bill. 4. a bird with a large, hooked bill, white superciliary and cheek patch, brown crown, and brown body.	1. a dark brown beak with a long beak and large wingspan. 2. large bird that is complete brown, with white stripes littering it's wings and a long blunted bill. 3. a bird with a large, hooked bill, white superciliary and cheek patch, brown crown, and brown body. 4. this dull colored bird is brown all over, has large wings and a long large bill.

(a) Image-to-text retrieval

Query Text	MM-Net: Retrieved images	MMC-Net: Retrieved images
Pascal Sentence: An Swiss-Air flight has just taken off from a runway.		
MSCOCO: a woman in white shirt holding bananas next to door.		
Flowers: the bright orange petals are highlighted by brown spots and the prominent stamen are topped with dark brown anthers.		
CUB-Bird: this bird is light brown, has a long hooked bill, and looks dumb.		

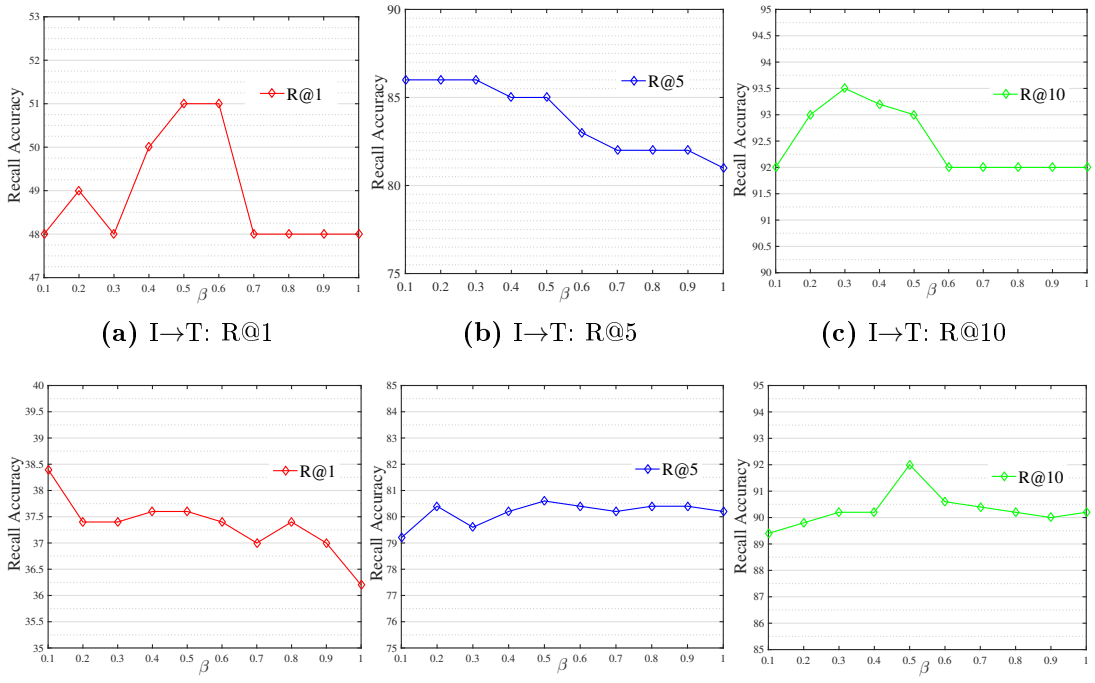
(b) Text-to-image retrieval

Figure 7.7: Image-text retrieval examples on the datasets. For (a) image-to-text retrieval, the ground-truth matching texts are in green. For (b) text-to-image retrieval, the red number in the upper left corner of one image is the ranking order, and the green frame corresponds to the ground-truth matching image. For the I→T and T→I retrieval, MMC-Net can retrieve more accurate candidates than MM-Net.

classification results, where MMC-Net achieves consistent improvements over MC-Net across all the four datasets. It shows that the matching component is able to promote the classification component due to combining the embedding features to generate more discriminative multi-modal representations. Also, MMC-Net has a generalization ability for different types of classification datasets, including either natural images or fine-grained images.

	Pascal Sentence	MSCOCO	Flowers	CUB-Bird
	 A striped sofa and office chairs are near a ping pong table.	 a tennis player wiping his face off with a towel.	 the petals of the flower are purple in color and have green stems with green sepals.	 a bird with a medium yellow bill, white body webbed feet and gray wings.
MC-Net	1. chair 2. tv/monitor 3. sofa 4. diningtable 5. bottle	1. person 2. chair 3. sports ball 4. tennis racket 5. dining table	1. bolero deep blue 2. garden phlox 3. canterbury bells 4. bougainvillea 5. snapdragon	1. Glaucous winged Gull 2. Ring billed Gull 3. California Gull 4. Herring Gull 5. Heermann Gull
MMC-Net	1. sofa 2. chair 3. Diningtable 4. tv/monitor 5. potted plant	1. person 2. tennis racket 3. chair 4. bench 5. sports ball	1. canterbury bells 2. bolero deep blue 3. foxglove 4. stemless gentian 5. garden phlox	1. Herring_Gull 2. California_Gull 3. Western_Gull 4. Ring_billed_Gull 5. Slaty_backed_Gull

Figure 7.8: Multi-modal classification examples on the datasets. Given an input image-text pair, the Top-5 predictions are estimated based on MC-Net and MMC-Net. The ground-truth classes are in green. By comparison, MMC-Net obtains more accurate predictions than MC-Net.



(a) I→T: R@1 **(b)** I→T: R@5 **(c)** I→T: R@10

(d) T→I: R@1 **(e)** T→I: R@5 **(f)** T→I: R@10

Figure 7.9: Effect of the parameter β on the performance of MMC-Net. The retrieval results on Pascal Sentence are reported. We select $\beta = 0.5$ by comparing these results.

7.4.4 Parameter analysis

Next, we aim to analyze the effects of three key parameters in MMC-Net.

Table 7.4: Comparison of the multi-modal classification accuracy between MMC-Net and MC-Net. For the four datasets, MMC-Net can outperform MC-Net with consistent performance gains.

Method	Pascal Sentence	MSCOCO	Flowers	CUB-Bird
MC-Net	71.0	77.6	94.0	80.7
MMC-Net	74.0	79.3	95.2	82.4

Table 7.5: Effect of the mini-batch size on the performance of MMC-Net. We train the model with different mini-batch sizes and compare their retrieval results on MSCOCO.

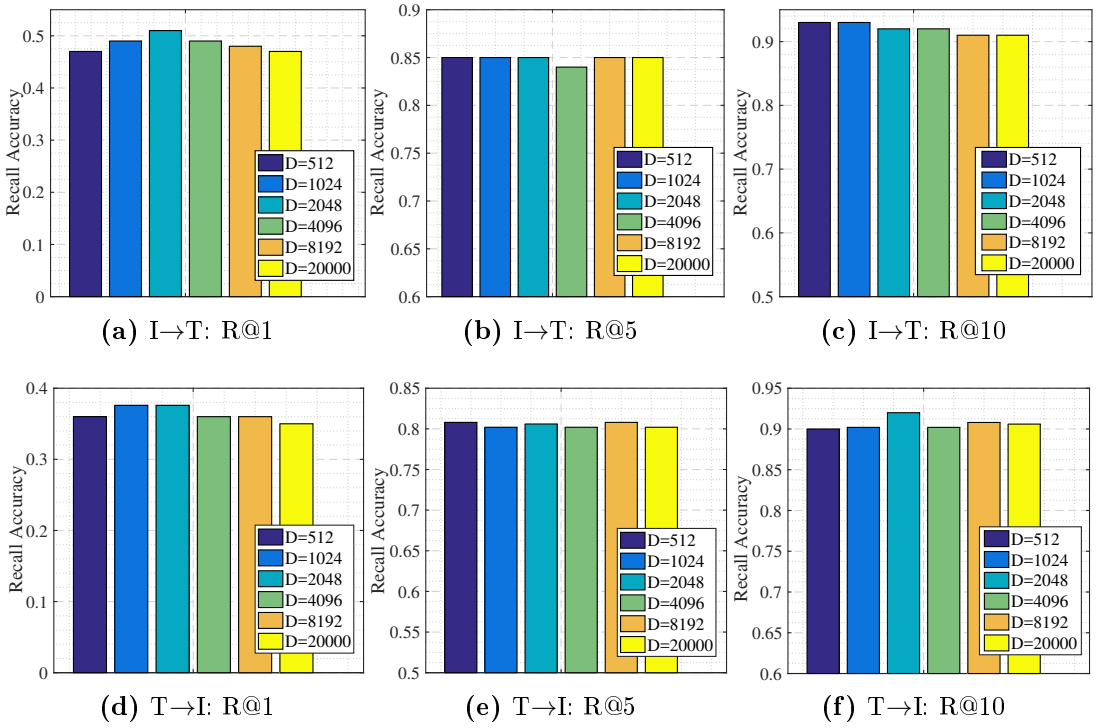
Method	Image-to-Text			Text-to-Image		
	R@1	R@5	R@10	R@1	R@5	R@10
batch size=100	42.5	74.6	87.4	36.6	73.8	86.8
batch size=250	52.6	83.3	91.7	43.0	79.5	89.4
batch size=500	56.6	85.3	92.7	46.0	80.5	90.1
batch size=1000	56.2	85.8	93.0	46.5	80.5	90.1
batch size=1500	57.0	85.8	92.7	46.2	80.8	90.5
batch size=2000	56.7	85.5	92.8	46.7	80.6	90.4

Effect of the mini-batch size.

Since the loss function for multi-modal matching aims to search for hard negative samples, it is essential to define a large mini-batch to increase the search space. For example, we selected a mini-batch size of 1500 for MSCOCO due to its large-scale data. To study the effect of varying different batch sizes, we used different batch sizes to train MMC-Net and tested their performance. Considering the number of negative pairs in each mini-batch is $K = 50$ for MSCOCO, we varied the batch size with 100, 250, 500, 1000, 1500 and 2000. Table 7.5 compares the retrieval results on MSCOCO with different batch sizes. We can observe that the performance is low when the batch size is 100. By increasing the size to 500, it can achieve significant gains across all the measurements. We further raise the size to 2000, however there is no important influence on the results. Finally, we select batch size=1500 due to its slightly superior results.

Effect of the parameter β.

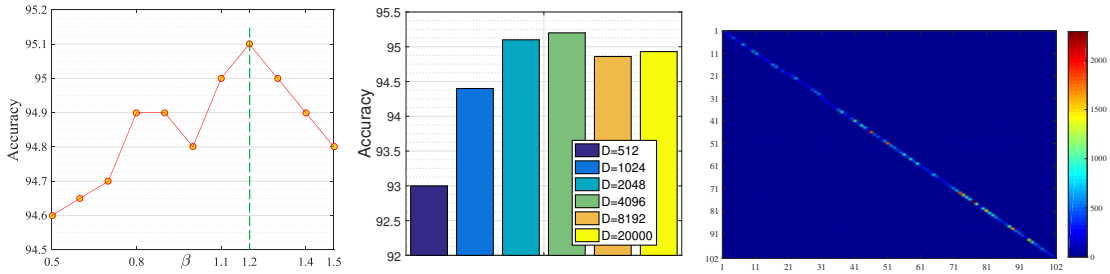
Recall that MMC-Net is trained by integrating the matching and classification loss, we use the parameter β to balance the weights of the two loss functions as defined in Eq. 7.13. This experiment aims to analyze the effect of β on the performance. Figure 7.9 shows the cross-modal retrieval results on Pascal Sentence. The R@1, R@5 and R@10 results are shown separately, when β varies from 0.1 to 1. We pick $\beta = 0.5$ by fully comparing these results.



(a) I→T: R@1 **(b)** I→T: R@5 **(c)** I→T: R@10

(d) T→I: R@1 **(e)** T→I: R@5 **(f)** T→I: R@10

Figure 7.10: Effect of the parameter D on the performance of MMC-Net. We present the retrieval results on Pascal Sentence by using different sizes of D. We select $D = 2048$ that can bring better results.



(a) **(b)** **(c)**

Figure 7.11: Effect of the parameters on the performance of MMC-Net. We report the Top-1 classification results on Flowers. (a) Analysis of the parameter β. (b) Analysis of the parameter D. (c) Confusion matrix of 102 Flowers classes. The diagonal line demonstrates the high accuracy per flower class.

Effect of the parameter D.

In the classification component, a CBP module can integrate visual and textual embedding features into a D-dimension multi-modal vector. In this experiment, we analyze D with $\{512, 1024, 2048, 4096, 8192, 20000\}$, which are all significantly lower than the original bilinear pooling vector (*i.e.* $512 \times 512 = 262,144$). In Figure 7.10, we present the compared results on Pascal Sentence. When $D = 2048$, MMC-Net can achieve better results compared to others.

Since MSCOCO is also composed of scene images like Pascal Sentence, it is straightforward and general to employ the same parameters β and D. In contrast, Flowers and CUB-Bird are commonly used for fine-grained recognition. It is needed to evaluate their parameters separately for Pascal Sentence and MSCOCO. To this end, we estimated the effects of the parameters on the classification accuracy of Flowers, and then applied the same parameters to CUB-Bird for generalization. Figure 7.11 presents the analysis of parameters on Flowers. As for the parameter β shown in Figure 7.11a, the best precision accuracy is achieved with 95.1% for $\beta = 1.2$. As shown in Figure 7.11b, the accuracy is maximized (*i.e.* 95.2%) when $D = 4096$. In the experiments, we set $\beta = 1.2$ and $D = 4096$ for Flowers and CUB-Bird. Additionally, we show the confusion matrix of 102 Flowers categories in Figure 7.11c.

7.4.5 Component analysis

Furthermore, we show ablation study to provide in-depth analysis.

Analysis of the fusion module

This test aims to verify the effectiveness of using the fusion module in the matching component. We build a convolutional fusion module in MMC-Net, which can also be applied on the baseline MM-Net. In Table 7.6, we report the results for both MMC-Net and MM-Net on the Pascal Sentence test set. We can see that using a fusion module can bring considerable performance improvements on all R@K measurements by considerable improvements, compared to the counterparts without using any fusion module. For an additional comparison, we further implement two simple fusion modules: element-wise sum and multiplication. Their results are inferior to those of the convolutional fusion, because they do not consider the weights of different layers. Instead, the convolutional fusion can learn adaptive weights to produce a superior fused feature while spending only three parameters. All the weights can be learned dynamically and adaptively with other network parameters without any manual tuning.

Analysis of the CBP module

We conduct this experiment to test the use of the CBP module in MMC-Net. For comparison, we present two other methods to integrate the visual and textual features. The first method starts by the concatenation of the two features to consturct a multi-modal representation and then feed it into a fully-connected (FC) layer to perform the classification. The second one is using the traditional bilinear pooling (BP) to produce a high-order multi-modal representation. Table 7.7 reports the compared results of different classification modules. The model with CBP can

Table 7.6: Analysis of the fusion module used in MM-Net and MMC-Net. The R@K results on Pascal Sentence are reported. By comparison, the convolutional fusion module can achieve better results than others.

Method	Fusion module	Image to Text			Text to Image		
		R@1	R@5	R@10	R@1	R@5	R@10
MM-Net	No	45.0	82.0	91.0	35.6	75.8	87.0
MM-Net	Sum	46.0	83.0	91.0	36.8	77.6	87.6
MM-Net	Multiplication	46.0	84.0	91.0	37.2	78.4	87.6
MM-Net	Convolution	47.0	85.0	92.0	38.4	80.6	88.6
MMC-Net	No	51.0	85.0	92.0	37.6	80.6	92.0
MMC-Net	Sum	51.0	86.0	92.0	38.4	81.0	92.0
MMC-Net	Multiplication	51.0	86.0	92.0	39.0	81.0	92.0
MMC-Net	Convolution	52.0	87.0	93.0	41.0	81.2	92.5

Table 7.7: Analysis of the CBP module in MMC-Net. The R@K results on Pascal Sentence are reported, which demonstrate the effectiveness and efficiency of using the CBP module.

Method	Dimension	Image to Text			Text to Image		
		R@1	R@5	R@10	R@1	R@5	R@10
MMC-Net with FC	1024	50.0	86.0	92.0	39.6	80.4	90.0
MMC-Net with BP	262144	53.0	88.0	93.0	41.5	81.5	92.5
MMC-Net with CBP	2048	52.0	87.0	93.0	41.0	81.2	92.5

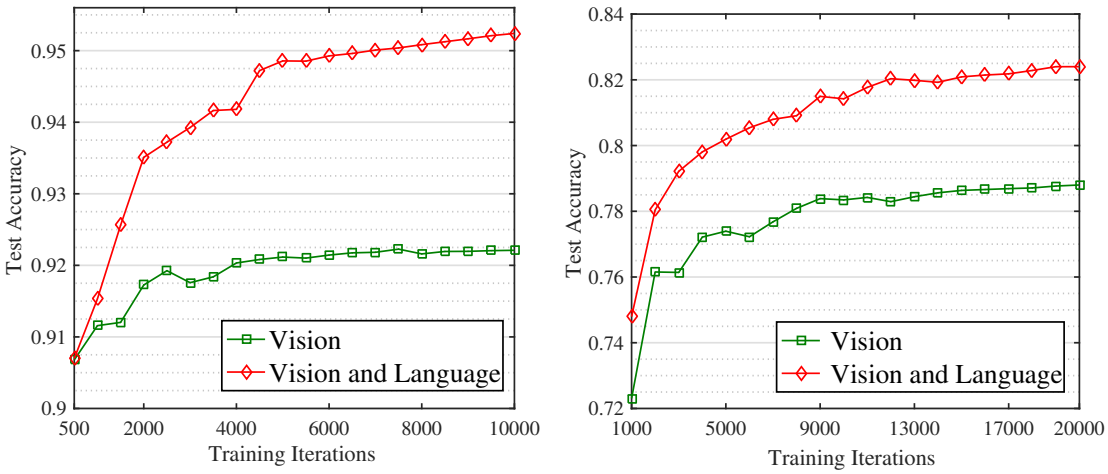
obtain considerable improvements over the one with FC. The MMC-Net with BP achieves better results than other methods, while its multi-modal representation has higher dimensionality. Instead, CBP can maintain both accuracy and efficiency.

Analysis of combining vision and language

This experiment is used to verify the advantage of incorporating visual and textual representations. As reported in Table 7.8, we compare the results between combining visual and textual features (*i.e.* MMC-Net) and using only visual features. We can observe that combining vision and language can achieve significantly superior accuracies on Flowers and CUB-Bird. Although visual features can enable the models to achieve promising performance, the informative textual features can further help improve the classification accuries. This shows the effectiveness of capturing multi-modal representations from both vision and language. Furthermore, Figure 7.12 analyzes the test rates during the training iterations. It can be seen that the vision and language model can consistently outperform the vision model in the entire training stage.

Table 7.8: Analysis of combining vision and language. We report the Top-1 classification rates on Flowers and CUB-Bird. The model with both vision and language outperforms the model with only vision.

Method	Flowers	CUB-Bird
Only Vision	92.2	78.8
Vision and Language	95.2	82.4



(a) Flowers **(b)** CUB-Bird

Figure 7.12: Illustration of the test classification rates during the training iterations. Incorporating language and vision is significant to improve the performance, compared to only using visual information.

Analysis of image encoders

As aforementioned in Section 7.2, we employ the ResNet-152 model to encode the input image. In this experiment, we aim to study the effect of different image encoders. For a fair comparison with DSPE [53], we provide the results of MMC-Net with VGG-19. Also, we implement the DSPE with ResNet-152. Table 7.9 reports the compared results on MSCOCO. For both VGG-19 and ResNet-152, our MMC-Net can outperform DSPE across all the measurements. We should realize that the improvements of MMC-Net come from two aspects. First, the matching component in MMC-Net has more layers than that of DSPE, *i.e.* four layers v.s. two layers. Second, MMC-Net utilizes a classification component to help improve the matching performance. This is the main motivation in this work. Note that, both MMC-Net and DSPE in Table 7.9 use the Mean vector to encode the input text. In [53], they also present another expensive textual representation using the Hybrid Gaussian-Laplacian mixture model (HGLMM) [51], *i.e.* a 18000-dimension vector. Currently, we do not introduce HGLMM to MMC-Net, even though it can help increase the performance.

Table 7.9: Analysis of image encoders. The image feature dimensions are also presented. MMC-Net has better matching results on MSCOCO than DSPE [53].

Method	Image encoder	Dimension	Image to Text			Text to Image		
			R@1	R@5	R@10	R@1	R@5	R@10
DSPE	VGG-19	4096	40.7	74.2	85.3	33.5	68.7	83.2
MMC-Net	VGG-19	4096	46.0	79.7	89.2	38.9	73.5	87.5
DSPE	ResNet-152	2048	53.1	82.7	90.2	43.5	78.2	88.9
MMC-Net	ResNet-152	2048	57.0	85.8	92.7	46.2	80.8	90.5

Table 7.10: Comparison with other state-of-the-art approaches on the Pascal Sentence dataset for image-text retrieval. Best results are in bold face.

Method	Image encoder	Text encoder	Image to Text		Text to Image	
			R@1	R@5	R@1	R@5
SDT-RNN [220]	AlexNet	DT-RNN	23.0	45.0	16.4	46.6
kCCA [220]	AlexNet	word2vec	21.0	47.0	16.4	41.4
DeViSE [214]	AlexNet	skip-gram	17.0	57.0	21.6	54.6
SDT-RNN [220]	RCNN	DT-RNN	25.0	56.0	25.4	65.2
DFE [181]	RCNN	word2vec	39.0	68.0	23.6	65.2
Mean Vector [51]	VGG-16	word2vec	52.5	83.2	**44.9**	84.9
GMM+HGLMM [51]	VGG-16	HGLMM	**55.9**	86.2	44.0	**85.6**
Proposed MMC-Net	ResNet-152	word2vec	52.0	**87.0**	41.0	81.2

7.4.6 Comparison with other approaches

For Pascal Sentence and MSCOCO, we compare our matching results with other state-of-the-art approaches. As reported in Table 7.10 and 7.11, MMC-Net can achieve competitive performance with the state-of-the-art. To be more specific, the method in [51] is effective on small-scale datasets, so it can obtain state-of-the-art results on Pascal Sentence. However, it does not have a strong generalization on large-scale datasets, for example their results on MSCOCO are not quite competitive. In contrast, the proposed MMC-Net maintains the high performance on both of small-scale and large-scale datasets. Moreover, we show the image and text encoders used in different approaches. Both of DSPE [53] and 2WayNet [76] extracted the visual features based on the VGG-19 model, while they rely on a more complicated HGLMM textual representation [51] than the Mean vector used in MMC-Net. As discussed earlier (Section 7.2), we did not use the HGLMM representation in order to maintain the training efficiency. For a fair comparison, MMC-Net with VGG-19 and Mean vector (see Table 7.9) can outperform DSPE with significant improvements, and can compete with 2WayNet while it uses the HGLMM representation. Lastly, we clarify that any common feature encoders for images and texts can be potentially adopted to MMC-Net. Exploring more efficient feature encoders is a fundamental and promising work.

For Flowers and CUB-Bird, we compare the fine-grained classification results with the state-of-the-art. Table 7.12 reports the comparison details. Since the compared methods do not utilize textual representations, we instead show the CNN model

Table 7.11: Comparison with other state-of-the-art approaches on the MSCOCO dataset for image-text retrieval. Best results are in bold face.

Method	Image encoder	Text encoder	Image to Text			Text to Image		
			R@1	R@5	R@10	R@1	R@5	R@10
DVSA [55]	RCNN	RNN	38.4	69.9	80.5	27.4	60.2	74.8
Mean vector [51]	VGG-16	word2vec	33.2	61.8	75.1	24.2	56.4	72.4
GMM+HGLMM [51]	VGG-16	HGLMM	39.4	67.9	80.9	25.1	59.8	76.6
m-RNN [190]	VGG-16	RNN	41.0	73.0	83.5	29.0	42.2	77.0
RNN-FV [185]	VGG-19	RNN	41.5	72.0	82.9	29.2	64.7	80.4
mCNN(ensemble) [52]	VGG-19	CNN	42.8	73.1	84.1	32.6	68.6	82.8
DSPE [53]	VGG-19	word2vec	40.7	74.2	85.3	33.5	68.7	83.2
DSPE [53]	VGG-19	HGLMM	50.1	79.7	89.2	39.6	75.2	86.9
2WayNet [76]	VGG-16	HGLMM	55.8	75.2	-	39.7	63.3	-
Proposed MMC-Net	ResNet-152	word2vec	**57.0**	**85.8**	**92.7**	**46.2**	**80.8**	**90.5**

Table 7.12: Comparison with other approaches on the Flowers and CUB-Bird datasets. Best results are in bold face. The methods in the upper part fine-tune the original CNN models, however, the ones in the lower part do not perform the fine-tuning process. We do not use the bounding box annotations in the datasets. Note that, we use the numbers to describe the depth of the image encoders. The dimension of MMC-Net indicates the multi-modal representation extracted from CBP.

Method	Image encoder	Finetune	Dimension	Flowers	CUB-Bird
Deep Optimized [224]	CNN-16	Yes	4096	91.3	67.1
Part R-CNN [225]	DeCAF-8	Yes	4096	-	76.5
Two-level attention [226]	AlexNet-8	Yes	4096	-	77.9
Deep LAC [227]	AlexNet-8	Yes	12288	-	80.3
NAC-const [221]	AlexNet-8	Yes	4096	91.7	68.5
NAC-const [221]	VGG-19	Yes	4096	**95.3**	81.0
Bilinear CNN [222]	VGG-16	Yes	250k	-	84.0
PD+FC+SWFV-CNN [223]	VGG-16	Yes	70k	-	**84.5**
MsML+ [228]	DeCAF-8	No	134016	89.5	67.9
BoSP [229]	VGG-16	No	5120	94.0	-
RI-Deep [230]	VGG-19	No	4096	94.0	72.6
ProCRC [231]	VGG-19	No	5120	94.8	78.3
MG-CNN [232]	VGG-19	No	12288	-	81.7
Proposed MMC-Net	ResNet-152	No	4096	**95.2**	**82.4**

used in the image encoder and the network depth. Note that, these approaches are divided into two groups based on whether the CNN model is fine-tuned on the target dataset. First, it can be seen that, MMC-Net achieves better results than other approaches without performing the fine-tuning step. Second, MMC-Net can even compete with the approaches with the fine-tuning step. For example, our results on Flowers is competitive with NAC-const [221]. Also, our approach is superior over most approaches on CUB-Bird, except Bilinear CNN [222] and PD+FC+SWFV-CNN [223]. However, we can see that both [222] and [223] produce a significantly more expensive feature vector than MMC-Net. We should realize that additional fine-tuning techniques have potential to improve performance, but are not the focus of this work. Our competitive results are partly due to the use of the ResNet-152 model, while we believe this should not decrease the effectiveness of our approach.

Table 7.13: Summary of the parameters used in the MMC-Net for matching and classification, and the time for running the multi-stage training algorithm.

Dataset	#Params for matching	#Params for classification	Time (hours)
Pascal Sentence	∼8 millions	∼41,000	∼0.3
MSCOCO	∼8 millions	∼164,000	∼7.0
Flowers	∼8 millions	∼418,000	∼0.5
CUB-Bird	∼8 millions	∼820,000	∼1.3

7.4.7 Computational cost

We conducted the experiments on a NVIDIA TITAN X card with 12 GB memory. In practice, we first extracted visual and textual features for all training samples using the off-the-shelf feature encoders. Then, we take as input these input features for the matching and classification components. Since the network parameters in MMC-Net are not expensive, it is feasible and rewarding to use a large mini-batch size to improve the training. In Table 7.13, we show the training parameters in the matching and classification component, and the multi-stage training time cost on the four datasets. The MSCOCO dataset consumes more training time due to its large-scale data. In summary, MMC-Net is an efficient network with a decent model complexity.

7.5 Chapter Conclusions

In this work, we proposed a unified network for joint multi-modal matching and classification. The proposed MMC-Net could simultaneously learn latent embeddings in the matching component, and generate a multi-modal representation vector in the classification component. Consequently, the two components could help promote each other by combining their loss functions together. We evaluated our approach on four well-known multi-modal datasets. The experimental results demonstrated the robustness and effectiveness of the MMC-Net model, compared to the baseline models. In addition, our approach achieved competitive results with the state-of-the-art approaches. The results showed its promising generalization for diverse multi-modal tasks related to matching or classification.

Future work. Currently, we use the class labels to train the classification component in MMC-Net. One potential improvement is to use more detailed information to guide the classification, like attributes. Compared to the class labels, attributes can discover more clues (*e.g.* sit, run, blue and small) about the visual content and text description. Hence, using attributes is beneficial for narrowing the gap between visual features and language words.

Chapter 8

Applications of Image Synthesis

After classification and retrieval, in this chapter we turn to address the third research theme: *synthesis*. In particular, we focus on two practical applications: image-to-image translation and fashion style transfer.

Image-to-image translation between different domains aim to arbitrarily manipulate the source image content given a target one. For RQ7, we need to study what factors influence the performance of cycle-consistent generative networks (CycleGAN), which have become a fundamental approach for general-purpose image-to-image translation, while few work investigate the important factors within it. To this end, we present an extensive and empirical study on cycle-consistent generative networks. We exploit two extended models which can promote the generation quality. Then, we conduct comprehensive experiments to evaluate these models for several translation tasks.

As for fashion style transfer, we aim towards developing a novel approach to perform the problem of person-to-person clothing swapping (RQ8). It is challenging due to varying pose deformations between different person images. We address this challenge by proposing a novel multi-stage generative network (SwapGAN) that integrates three generators based on different synthesis conditions. The SwapGAN model is end-to-end trainable with adversarial loss and mask-consistency loss. We demonstrate the effectiveness of our approach through both quantitative and qualitative evaluations on the DeepFashion dataset. This work can serve as a benchmark for future research on this task.

Keywords

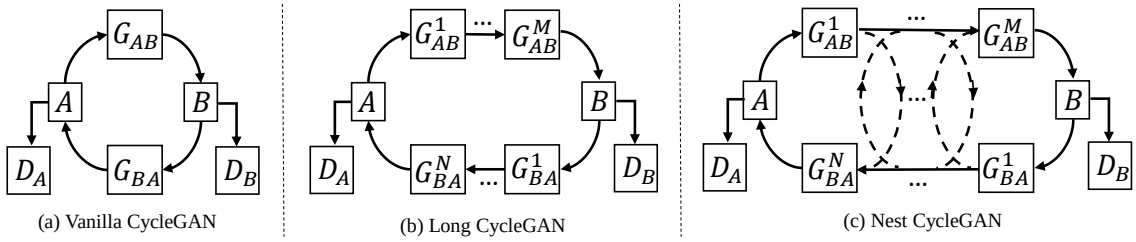
Image synthesis, Image-to-image translation, Fashion style transfer, Generative adversarial networks

8.1 Image-to-Image Translation

Image-to-image translation has achieved increasing attention in recent research. This task learns to synthesize the translated image in the target domain, given one image in the source domain. With the emergence of generative adversarial networks (GANs) [79] in recent years, some efforts have been made to employ unpaired image samples to model mapping functions between two different domains [89, 90, 91]. The translation task therefore becomes an unsupervised problem as the corresponding ground-truth images in the target domain are unknown. In addition, these approaches make use of the adversarial mechanism involved in GANs, to make the generated images undistinguished from real ones in the target domain. One challenging problem is that the domain mappings in these unsupervised approaches are under-constrained due to lack of ground-truth labels. To tackle the challenge, CycleGAN [94] introduces a cycle-consistency loss by reconstructing the generated image back to the source domain. In conjunction with the original adversarial loss, the cycle-consistency loss is beneficial to aid the unsupervised domain mappings. Moreover, this additional loss can help the model in avoiding mode collapse, from which the original GANs often suffer. Figure 8.1(a) illustrates the conceptual architecture of CycleGAN. Due to its high effectiveness and generalization ability, CycleGAN has been a fundamental model to address the task of unsupervised image translation, while few works have examined what factors may influence its performance. This fact motivates our research question **RQ 7: What factors will affect the performance of generative models on the translation tasks?**

Driven by this, in this work we extend the vanilla CycleGAN with new improvements, which can present more insights into what factors promote its performance on unsupervised image-to-image translation. Specifically, our improved models focus on studying the effects of two key factors in CycleGAN: one is the number of generators and another is the number of cycles. For the first factor, we build an extended model called **Long CycleGAN**, which can cascade more generators to perform the translation within a long cycle. For example, in Figure 8.1(b), we can incorporate M and N of different generators for A-to-B and B-to-A translation, respectively. Advantageously, the long cycle can leverage more generators to further increase the generation abilities of the model and improve the quality of the synthesized images. In terms of the second factor, another extended model with additional nested cycles is developed, namely **Nest CycleGAN**. As illustrated in Figure 8.1(c), this model attempts to exploit many inner cycles nested within the outer cycle. In this way, the inner cycles are able to directly connect the intermediate generators and provide more cycle-consistency losses to guide the domain mappings. Nest CycleGAN is used to demonstrate the benefit of adding more cycles among generators.

The contributions of this work are as follows:



(a) Vanilla CycleGAN (b) Long CycleGAN (c) Nest CycleGAN

Figure 8.1: Illustration of three cycle-consistent generative adversarial networks. Based on the (a) Vanilla CycleGAN [94], we build two extended models: (b) Long CycleGAN and (c) Nest CycleGAN. Long CycleGAN can promote the generative abilities by cascading more generators, and Nest CycleGAN is able to add extra inner cycles to enhance the mapping constraints.

- We propose two extended models to explore the important factors in CycleGAN. In addition, we present the initialization networks for the extended models. We conduct qualitative and quantitative evaluation to assess these models, for translation tasks including photo↔label and photo↔sketch.
- Our results witness the superiority of the extended models over the vanilla one. The results can act as an indication that CycleGAN equipped with more generators and cycles would achieve better generation quality.

The rest is structured as follows. Section 8.1.1 describes the vanilla CycleGAN and two extended models. The initialization networks are introduced in Section 8.1.2. The experiments are shown from Section 8.1.3 to Section 8.1.5.

8.1.1 Methodology

Problem Formulation

Assume that there are two unpaired image sets: $\{a_i\}_{i=1}^N$ in domain A and $\{b_j\}_{j=1}^M$ in domain B. The task aims to learn bi-directional mapping functions to map any $a_i \in A$ to $b_j \in B$, and vice versa. We omit the subscript i and j for notational simplicity. Notably, the images in the two sets are unaligned with each other, and the input images lack of ground-truth images to provide correct correspondences.

To tackle this problem, GANs [79] are used to generate realistic-looking target samples by incorporating a generator G and a discriminator D. Taking the A-to-B mapping for example, G_{AB} learns to simulate real images in domain B given the images in domain A. Then D_B need to distinguish real images b from synthetic images $G_{AB}(a)$. The original GANs compute the adversarial loss based on the negative log likelihood. Instead, we employ the least square loss designed in LSGAN [233], due to its proper stability of training and quality of generated images. The adversarial

loss for translating a to b is expressed with

$$\mathcal{L}_{GAN}(G_{AB}, D_B) = \mathbb{E}_{b\sim p_{data}(b)}[(D_B(b)-1)^2] + \mathbb{E}_{a\sim p_{data}(a)}[D_B(G_{AB}(a))^2]. \quad (8.1)$$

Here, p_{data} is the empirical distribution of training images. The generator and discriminator are trained for a minimax objective: $\min_{G_{AB}} \max_{D_B} \mathcal{L}(G_{AB}, D_B)$. Similarly, we can employ another generator and discriminator for the B-to-A mapping, and compute its corresponding adversarial loss: $\mathcal{L}_{GAN}(G_{BA}, D_A)$.

Vanilla Cycle-consistent GAN

Unsupervised image translation relies on adversarial loss to ensure the synthesized images in accordance with the target domain. However, it is important to add extra losses to enhance the constraints of unsupervised mapping functions. CycleGAN [94] develops a cycle-consistent loss by coupling two generators G_{AB} and G_{BA} in a reconstruction-based cycle. To be specific, the generated image $G_{AB}(a)$ is further fed into G_{BA} to obtain the reconstructed image $\hat{a} = G_{BA}(G_{AB}(a))$. Similarly, we can have $\hat{b} = G_{AB}(G_{BA}(b))$. Then, the difference between the input images and their reconstructed ones is computed with the L1 norm :

$$\begin{aligned}\mathcal{L}_{Rec}(G_{AB}, G_{BA}) =& \mathbb{E}_{a\sim p_{data}(a)}[||G_{BA}(G_{AB}(a)) - a||_1] \\ &+ \mathbb{E}_{b\sim p_{data}(b)}[||G_{AB}(G_{BA}(b)) - b||_1].\end{aligned} \quad (8.2)$$

Finally, the full objective in CycleGAN considers minimizing both the adversarial loss and the cycle-consistent loss:

$$\begin{aligned}\mathcal{L}_{Cycle}(G_{AB}, G_{BA}, D_A, D_B) =& \mathcal{L}_{GAN}(G_{AB}, D_B) + \mathcal{L}_{GAN}(G_{BA}, D_A) \\ &+ \lambda \mathcal{L}_{Rec}(G_{AB}, G_{BA}),\end{aligned} \quad (8.3)$$

where λ adjusts the weight of the reconstruction loss. As suggested in CycleGAN [94], the cycle-consistent constraint can help avoid the mode collapse problem, that is, the generated samples may only come from several modes of the real data distribution, but discard many other modes.

Long Cycle-consistent GAN

A key purpose of generative models is improving the quality of synthesized image samples. One favorable solution is introducing more generators to promote the generative abilities of the whole model. Driven by this, we extend CycleGAN by stacking a few generators, and investigate its effects on the generation quality. In Figure 8.1(b), we illustrate the first extended model called Long CycleGAN. Assume that there are M generators translating image samples from domain A to B, and at the same time N generators to map image samples from B to A. The whole mapping procedure can be performed in a chained fashion: the output of the current generator

is taken as input of the next generator. Formally, we can compute the output of each generator with

$$G_{AB}^{m}(a) = F(G_{AB}^{m-1}(a), W_{AB}^{m}), m = 1, \ldots, M, \tag{8.4}$$

$$G_{BA}^{n}(b) = H(G_{BA}^{n-1}(b), W_{BA}^{n}), n = 1, \ldots, N. \tag{8.5}$$

We define F and H as the mapping functions for A-to-B and B-to-A. W_{AB}^{m} and W_{BA}^{n} correspond to their mapping weights. Finally, we can rewrite the full objective for Long CycleGAN

$$\begin{aligned}\mathcal{L}_{Long}(\sum_{m=1}^{M} G_{AB}^{m}, \sum_{n=1}^{N} G_{BA}^{n}, D_A, D_B) =& \mathcal{L}_{GAN}(G_{AB}^{M}, D_B) + \mathcal{L}_{GAN}(G_{BA}^{N}, D_A) \\ &+ \lambda \mathcal{L}_{Rec}(G_{AB}^{M}, G_{BA}^{N}).\end{aligned} \tag{8.6}$$

We note that, when $M = N = 1$, $G_{AB}^{0}(a) = a$ and $G_{BA}^{0}(b) = b$. In this case, Long CycleGAN is the same as the vanilla one and therefore can be viewed as a generalized model.

Nest Cycle-consistent GAN

Furthermore, we present another extended model by nesting more inner cycles apart from a single outer cycle used in Long CycleGAN. T additional cycle-consistency losses based on new inner cycles can help constrain the mapping space between two domains. The extended model named by Nest CycleGAN is shown in Figure 8.1(c). On the one hand, the outer cycle in Nest CycleGAN (in solid line) performs the complete mappings between two domains by using all generators. On the other hand, the inner cycles (in dashed line) aim to build additional connections to bridge intermediate generators in the two chains. Notably, each inner cycle can be viewed as an auto-encoder model that can reconstruct the input image based on latent representations learned from intermediate generators. For instance, the m-th inner cycle for domain A is associated with two sets of generators, *i.e.* $\{G_{AB}^{1}, \ldots, G_{AB}^{m}\}$ and $\{G_{BA}^{N-m+1}, \ldots, G_{BA}^{N}\}$. In addition, we task the output of G_{AB}^{m} as input of G_{BA}^{N-m+1}, which can be denoted as

$$G_{BA}^{N-m+1}(G_{AB}^{m}(a)) = H(G_{AB}^{m}(a), W_{BA}^{N-m+1}). \tag{8.7}$$

After that, the image sample further passes from G_{BA}^{N-m+1} to G_{BA}^{N}, and the reconstructed image based on the m inner cycle can be formulated as

$$\hat{a}_m = G_{BA}^{N}(G_{AB}^{m}(a)). \tag{8.8}$$

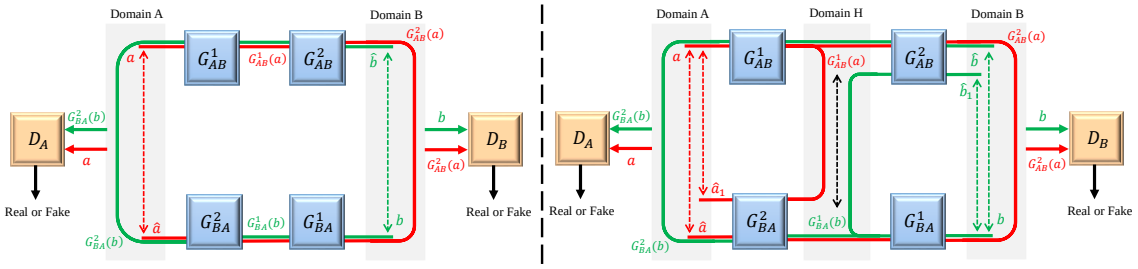


Figure 8.2: Instantiation Networks. Left: Long CycleGAN; Right: Nest CycleGAN. Details can be seen in Section 8.1.2.

Similarly, we can obtain $\hat{b}_n = G^M_{AB}(G^n_{BA}(b))$ for the n-th inner cycle with respect to domain B. Finally, the reconstruction loss with additional inner cycles is

$$\begin{aligned}\mathcal{L}_{Inner}(\sum_{m=1}^{M} G^m_{AB}, \sum_{n=1}^{N} G^n_{BA}) = &\sum_{m=1}^{M} \mathbb{E}_{a\sim p_{data}(a)}[||\hat{a}_m - a||_1] \\ &+ \sum_{n=1}^{N} \mathbb{E}_{b\sim p_{data}(b)}[||\hat{b}_n - b||_1].\end{aligned} \tag{8.9}$$

Particularly, when $m = M$ and $n = N$, the inner cycle turns to be the outer cycle, which can be included in the formulation. The objective of Nest CycleGAN is

$$\begin{aligned}\mathcal{L}_{Nest}(G^M_{AB}, G^N_{BA}, D_A, D_B) =&\mathcal{L}_{GAN}(G^M_{AB}, D_B) + \mathcal{L}_{GAN}(G^N_{BA}, D_A) \\ &+\lambda\mathcal{L}_{Inner}(\sum_{m=1}^{M} G^m_{AB}, \sum_{n=1}^{N} G^n_{BA}).\end{aligned} \tag{8.10}$$

8.1.2 Instantiation network

To assess the effectiveness of the three CycleGAN variants, we build their instantiation networks as follows.

Vanilla CycleGAN. We reproduce the standard CycleGAN with the generator and discriminator in [94]. (1) *Generator*: it consists of an encoder, several residual blocks and a decoder. The encoder module contains three convolutional layers; each residual block adds a skip connection on two 3×3 convolutional layers; the decoder module has two deconvolutional layers using stride-$\frac{1}{2}$ convolutions to upsample, and one stride-1 convolutional layer to output the synthesized image. The convolutional layers are followed by instance normalization [83] and ReLU [4]. (2) *Discriminator*: it is based on the Markovian network from PatchGANs [82, 88], which can run convolutationally across an image to classify if overlapping patches are real or fake. It contains four convolutional layers and the last layer produces a 1-dimensional feature map as the predicted output.

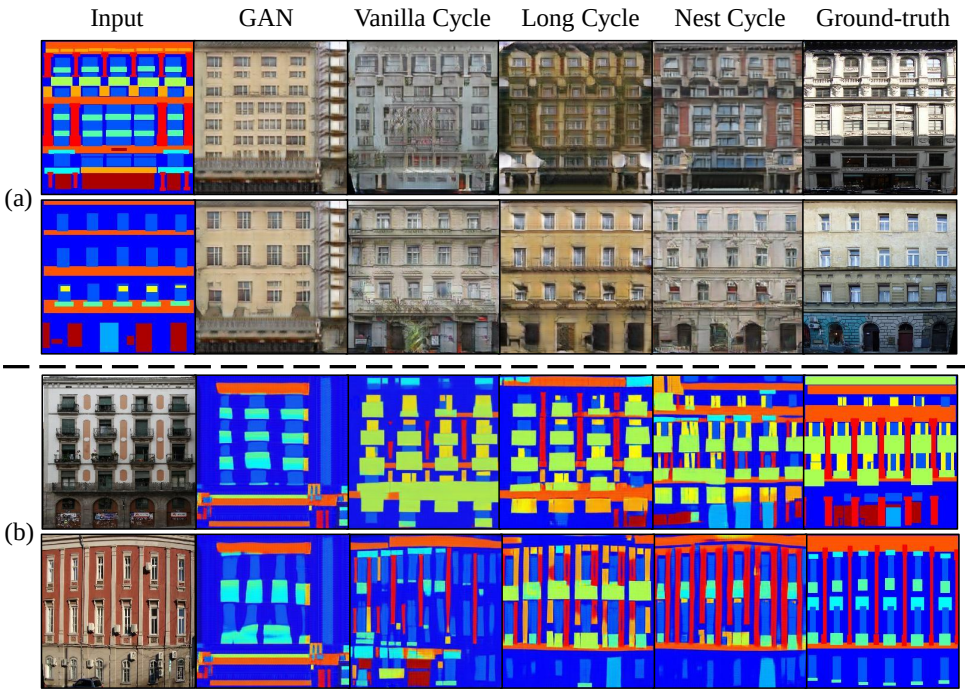


Figure 8.3: Generated samples of (a) the label→photo translation and (b) the photo→label translation evaluated on the CMP-Facade dataset.

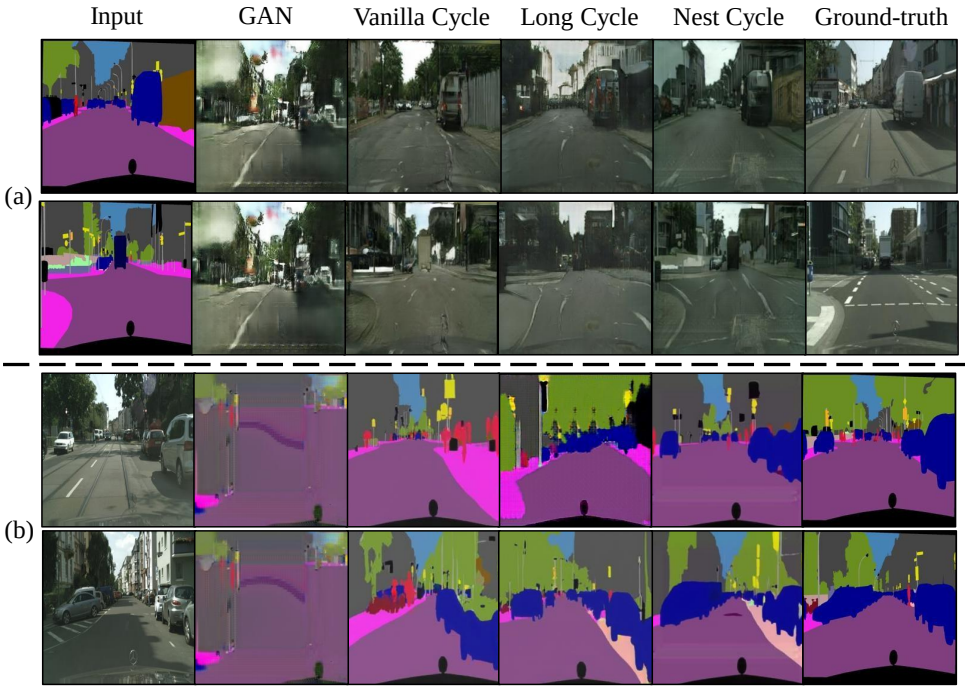


Figure 8.4: Qualitative results of (a) the label→photo translation and (b) the photo→label translation on the Cityscapes dataset.

Long CycleGAN. On top of the vanilla CycleGAN, we instantiate a Long CycleGAN by cascading two generators (*i.e.* $M = N = 2$), but the extension with more generators is straightforward. For fairness, all the generators and discriminators in

Long CycleGAN use the same networks with the vanilla CycleGAN. As illustrated in Figure 8.2 (Left), the model consists of two cycles which can be trained jointly. The red cycle starts with the input a in domain A and translates it to be $G^2_{AB}(a)$ in domain B. D_B learns to distinguish the fake image $G^2_{AB}(a)$ from the real image b. Then, $G^2_{AB}(a)$ is translated back to be the reconstructed image $\hat{a}$ in domain A. Likewise, the green cycle beginning from b performs an inverse translation.

Nest CycleGAN. Next, we build a Nest CycleGAN upon the above Long CycleGAN. In Figure 8.2 (Right), we exploit two additional inner cycles within the outer cycles. The inner cycles can also reconstruct the input images a and b, which are denoted by $\hat{a}_1$ and $\hat{b}_1$. We can see that, the first generated images, *i.e.* $G^1_{AB}(a)$ and $G^1_{BA}(b)$, act as intermediate states between A and B, then they should have implicit semantic similarities in some extend. Hence, we consider adding an extra loss to correlate them with

$$\mathcal{L}_{Sim}(G_{AB}, G_{BA}) = \mathbb{E}_{a\sim p_{data}(a)}[||G^1_{AB}(a) - G^1_{BA}(b)||_1]. \tag{8.11}$$

During training, $\mathcal{L}_{Sim}(G_{AB}, G_{BA})$ is added with $\mathcal{L}_{Nest}(G^M_{AB}, G^N_{BA}, D_A, D_B)$. Consequently, G^1_{AB} and G^1_{BA} can tend to gather in a common domain H between A and B, even though the inputs a and b are unpaired.

8.1.3 Experiment setup

To assess the three CycleGAN variants, we perform three image translation tasks, including photo↔label and photo↔sketch. The input and output images were scaled to 256×256. For fairness, some training parameters were consistent with CycleGAN [94], including mini-batch size of 1, learning rate of 0.0002, and weight decay of 0.0005. All the models were trained with 200 epoches and we fixed $\lambda = 10$ in the experiments, and optimized with the Adam optimizer [234]. Notice that, we randomly shuffled two domain-specific datasets to make sure they are totally unpaired. We implemented the models with TensorFlow [235] on a Titan X GPU card.

8.1.4 Results on photo↔label

For this translation task, we employed two semantic segmentation datasets: CMP-Facade [236] and Cityscapes [237]. CMP-Facade contains 606 images in total. We randomly select 400 images for training, and the remaining 206 images for testing. In Cityscapes, there are 2975 images for training and 500 images for testing. There are 12 and 19 semantic labels in CMP-Facade and Cityscapes, respectively.

Qualitative results. In Figure 8.3 and Figure 8.4, we compare the quality of generated images. For the label→photo task, three cycle-consistent GANs can synthesize more realistic images than the original GAN. It can be seen that, GAN suffers from

Table 8.1: Quantitative results of the label→photo translation evaluated on the CMP-Facade dataset. Higher numbers are better.

Method	CMP-Facade dataset			Cityscapes dataset		
	Per-pixel acc.	Per-class acc.	Class IOU	Per-pixel acc.	Per-class acc.	Class IOU
GAN	0.32	0.12	0.07	0.50	0.11	0.07
Vanilla CycleGAN	0.35	0.15	0.10	0.51	0.17	0.12
Long CycleGAN	0.43	0.19	0.13	0.54	0.18	0.13
Nest CycleGAN	0.49	0.22	0.15	0.57	0.20	0.14
Oracle	0.66	0.51	0.39	0.86	0.45	0.37

mode collapse, where the generated labels look almost identical for different input photos. However, the other three models can avoid this problem due to using cycle-consistency constraints. In addition, the two extended models can produce superior images over the vanilla one.

Quantitative results. In addition to the above qualitative evaluation, we further conduct quantitative experiments for this translation task. Considering the fact that the two datasets are not large scale, it is inappropriate to use the inception score (IS) to measure the generation quality. Instead, we used the FCN-score, *i.e.* a quantitative measurement as suggested in [88], to assess the label→photo task. First, a fully convolutional network (FCN) [26] for semantic segmentation was pre-trained using the real training photos and ground-truth labels. Then, each generated photo was fed into the FCN model to produce the predicted labels. The comparison with the ground labels can assess the generation photos. Commonly, FCN-score includes three standard metrics: per-pixel accuracy, per-class accuracy, and mean class intersection-over-union (IOU).

Table 8.1 reports quantitative results on CMP-Facade and Cityscapes. Comparably, all the three cycle-consistency models outperform the original GAN model. However, we can observe that the performance gap between Vanilla CycleGAN and GAN is not significant, while Long CycleGAN can improve the performance with more considerable gains. This demonstrates the benefit of employing more generators for raising the generative ability. Moreover, Nest CycleGAN can achieve better accuracy than Long CycleGAN due to adding new inner cycles. For a full comparison, we also provide the Oracle results by testing real photos, which can be seen as the upper-bound performance. Our results on CMP-Facade narrow the gap with Oracle.

8.1.5 Results on photo↔sketch

We conducted this task with the SBIR dataset [238] which includes two subsets: one for shoes and the other for chairs. In the shoe dataset, we used 304 samples for training and 115 ones for testing. The chair dataset consists of 200 training samples and 97 testing ones. Figure 8.5 and Figure 8.6 present the generated image samples on the two datasets. We can see that the two extended models are advantageous to the original GAN and Vanilla CycleGAN. It is worth noting that, the sketch→photo

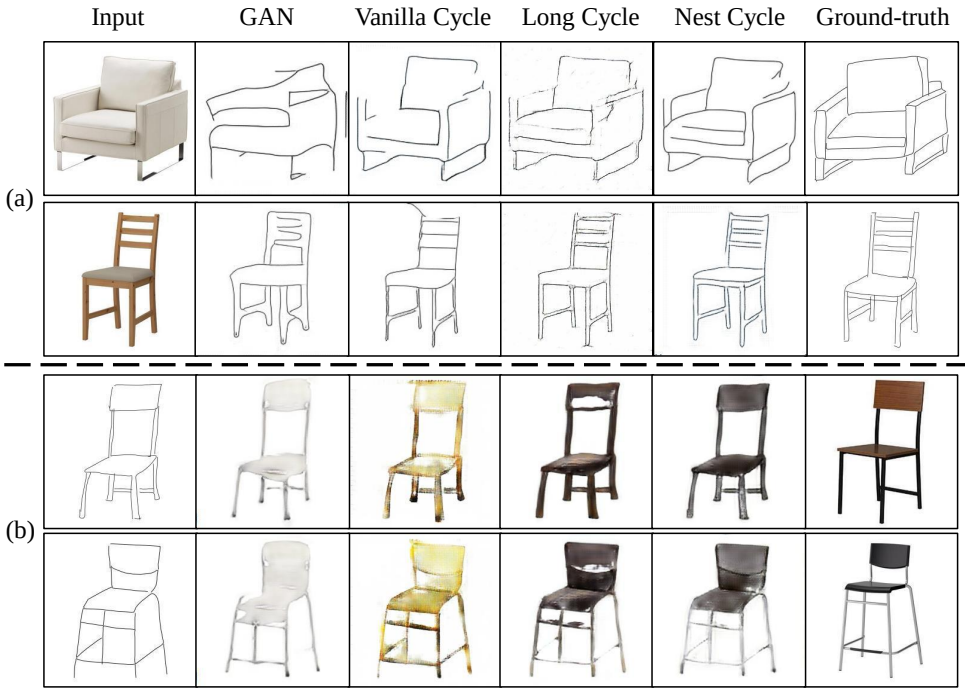


Figure 8.5: Qualitative results of (a) the photo→sketch translation and (b) the sketch→photo translation on the SBIR chairs dataset.

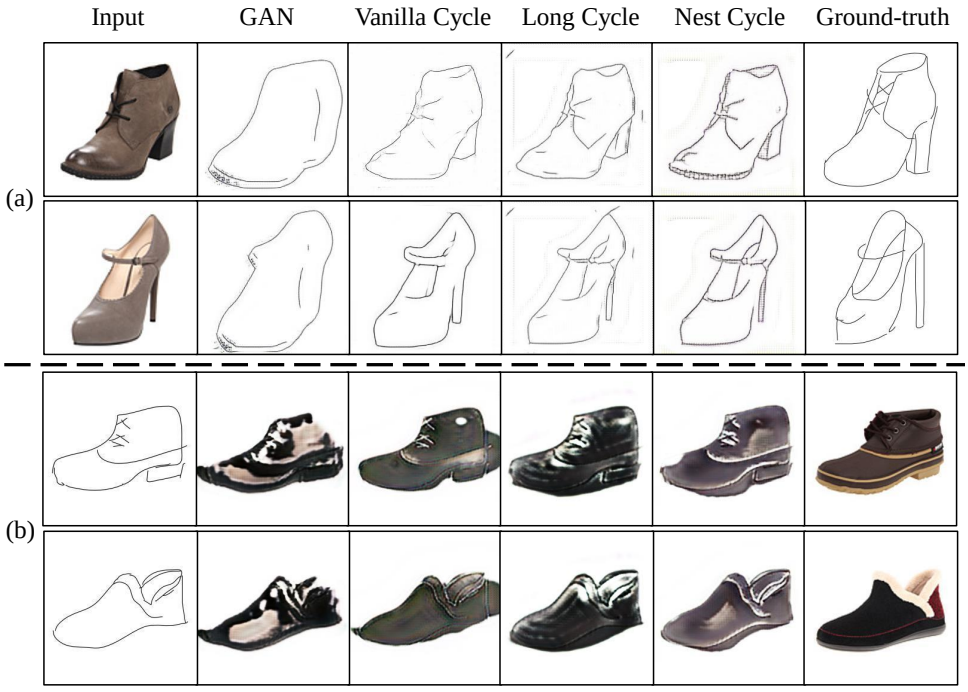


Figure 8.6: Qualitative results of (a) the photo→sketch translation and (b) the sketch→photo translation on the SBIR shoes dataset.

translation is more challenging than the photo→sketch translation. The main reason is that the sketch→photo mapping functions are more under-constrained, and one sketch image therefore may be synthesized with a variety of colors.

8.2 Fashion Style Transfer

Nowadays, online shopping has become an indispensable experience in our daily lives. Consequently, the huge market brought by fashion clothing shopping motivates an increasing variety of fashion relevant research, such as fashion clothing retrieval [95, 239], fashion recommendation [96, 240], fashion parsing [97, 241] and fashion aesthetics [242, 243]. In this work, we deal with the problem of fashion clothing swapping, which aims to visualize what the person would look like with the target clothes. From the practicality perspective, fashion clothing swapping is a useful experience for online consumers who need to virtually try on different clothes instead of wearing them physically. From the research perspective, fashion clothing swapping can be viewed as a specific task belonging to fashion style transfer. The challenge in this task is how to transform the target clothes fitting for the wearers while preserving their pose and body shape.

Traditionally, non-parametric methods [98, 101, 103, 244] are exploited to address this problem. They need to segment the target clothes from the condition image and then employ 2D image warping algorithms or 3D graphics methods to model the deformations between the clothes and the reference person body. However, these traditional methods rely on extra information (*e.g.* 3D measurements and geometric constraints) and complicated optimization algorithms (*e.g.* dynamic programming and dynamic time warping). In addition, non-parametric methods are not general, which means they need to estimate individual deformations for different image pairs. Also, it is non-tractable to match humans' key points due to non-rigid pose deformations.

In contrast to non-parametric methods, recent research [105, 106] turns to recast the clothing swapping as a 2D image synthesis problem. It is mainly driven by the rapid developments of deep generative networks, which have succeeded in many tasks involving synthesizing plausible images [84, 88, 245, 246]. Deep generative networks are able to synthesize the target images without requiring matching key points. Recently, FashionGAN [85] employs a textual description as condition to perform the clothing swapping (Figure8.7(a)). The methods in [105, 106] uses a stand-alone and flat clothing image to re-dress the reference person (Figure 8.7(b)). However, the target clothe is always worn on another person in practical scenarios, rather than is shown in a separate image. In this work, we aim to perform the person-to-person clothing swapping by transferring the clothes on the condition person images to the reference ones (Figure8.7(c)). It becomes more challenging due to the varying deformations among different human poses. Considering the challenge, we need to tackle the last research question **RQ 8: How can we exploit a generative model to directly transfer the fashion style between two person images?**

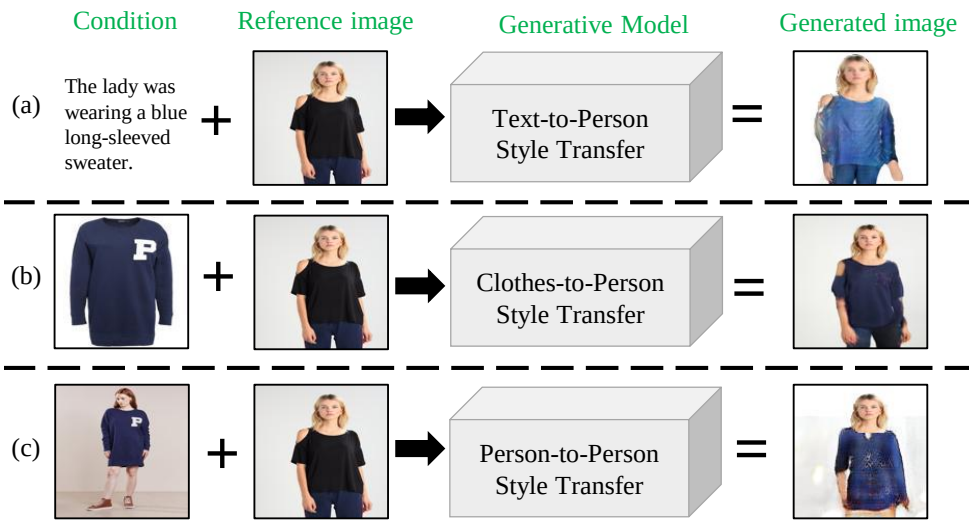


Figure 8.7: Three tasks of fashion clothing swapping conditioned on (a) textual description [85], (b) clothing image [106] and (c) person image, respectively. All the three cases aim to re-dress up the woman in the reference image with a long-sleeved sweather, while preserving her original pose and body shape. (c) shows the synthesized image based on our proposed SwapGAN.

To this end, we propose a multi-stage generative framework (SwapGAN), consisting of three generation stages conditioned on different priors. In the first stage, we interpret this problem as a pose-based person image synthesis process. We therefore exploit a pose-conditioned generative network (*i.e.* Generator I), which can manipulate the person in the condition image to have the same pose and body shape as the person in the reference image. Consequently, the new synthesized image can be viewed as the desired target image where the reference person wears the target clothes while preserving the original pose and body shape. Second, we further exploit a segmentation-conditioned generative network (*i.e.* Generator II) built on top of Generator I. The pose map in Generator I may mistake the clothing style (*e.g.* changing long sleeves to short sleeves), however, the segmentation in Generator II is used to retain the style due to its rich semantic information. To be specific, we take the segmentation map of the condition image into Generator II, to make sure that the synthesized image is consistent with the original condition image. Our hypothesis is that, *if a person image can be well transformed based on an arbitrary pose, then it should be feasible to reconstruct it based on its original segmentation map.* Moreover, we perform the third generation stage by using a mask generative network (*i.e.* Generator III). Generator III is used to explicitly constrain the body shape of the synthesized person images from both Generator I and Generator II. During the training procedure, we can train the entire SwapGAN end-to-end by integrating the adversarial loss from Generator I and Generator II and the mask-consistency loss from Generator III.

The contributions of this work are as follows:

- We prose a multi-stage generative framework for addressing a task of fashion style transfer, *i.e.* person-to-person clothing swapping. This is the first attempt to study it with a deep generative approach, to the best of our knowledge.
- In addition, our approach present the benefit of integrating multiple conditional GANs based on different priors. It can motivate tackling other research problems involved in deep generative networks.
- Furthermore, the experiments on the DeepFashion dataset verify the effectiveness of SwapGAN in terms of qualitative and quantitative evaluations. Our work can be a benchmark study to drive future research on this task. Also, it can enrich the application of deep generative approaches for solving practical problems.

The rest is structured as follows. Section 8.2.1 describes the proposed multi-stage generative model for person-to-person clothing swapping. The network architecture is detailed in Section 8.2.2. We report and discuss experimental results from Section 8.2.3 and Section 8.2.6.

8.2.1 Methodology

Problem Definition

We define the problem of person-to-person clothing swapping to be a conditional person image generation process. Its goal is to manipulate the person in the condition image to have the same pose and body shape as the person in the reference image. Additionally, we paste the head of the reference person onto the new synthesized image, in order to preserve the person identity. In this way, the reference person in the synthesized image can wear the target clothes in the condition image, while retaining the original pose and body shape.

Given a condition person image and a reference one, it may be infeasible to find the ground-truth target image in the dataset to supervise the synthesized image. Instead, we consider training the synthesis process using two images of the same person. To be specific, we have a training dataset of N image pairs, each of which is composed of two images of the same person with **the same clothes**, but with **different poses** (Figure 8.8). We randomly select one of the two images as a reference image, and the other one as a condition image. The reference and condition images are denoted with $X_r^{(i)}$ and $X_c^{(i)}$, $i = 1, \ldots, N$. Taking $X_c^{(i)}$ and the pose map of $X_r^{(i)}$ as input, our generator learns to create a fake $X_r^{(i)}$ during the training procedure. The discriminator needs to distinguish the fake $X_r^{(i)}$ from the real one. Ideally, when the discriminator cannot tell the differences between the real and fake images, the generators should be able to generate high-quality images.

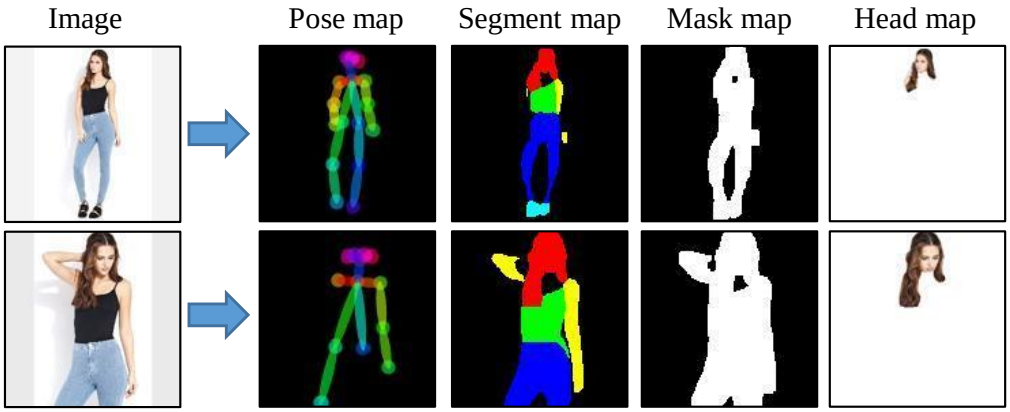


Figure 8.8: Representations for a pair of person images that have the same clothes but show different poses.

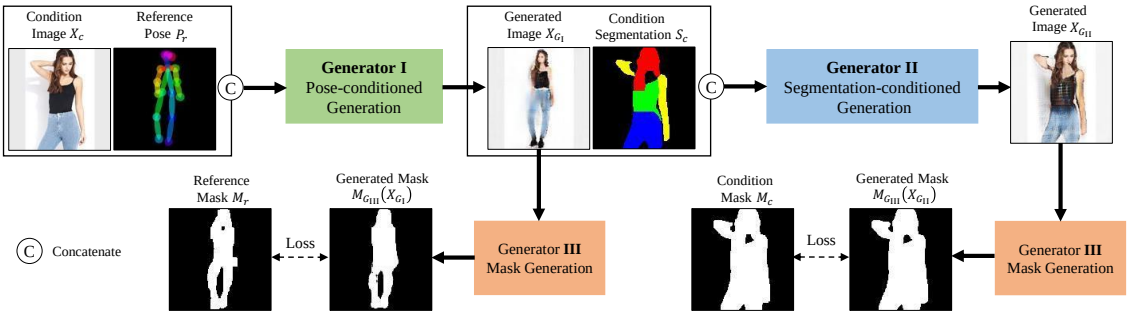


Figure 8.9: Overview architecture of the multi-stage generative framework in the proposed SwapGAN. **Generator** I can synthesize a new image X_{G_I} by manipulating the condition person image X_c based on the reference pose P_r. Then, **Generator** II takes as input X_{G_I} to produce a reconstructed X_c based on the segmentation map S_c. Moreover, **Generator** III is used to explicitly constrain the body shape during the synthesis process.

Person Representation

To specify the synthesis process, we need to extract a couple of person representations based on the person images. As shown in Figure 8.8, we utilize four feature maps described as follows:

1) *Pose map:* We employ one of the state-of-the-art pose estimators, OpenPose [247], to capture person pose information. For each person image, the pose estimator can localize 18 key-points in a pose map. In addition, the key-points are connected by color lines that can present the orientation of limbs. The pose map is used in Generator I.

2) *Segmentation map:* An off-the-shelf human semantic parser [248] is adopted to extract a person segmentation map. The original map can predict 20 fine classes for semantic segmentation. We further re-group the fine classes into five coarse classes, including head, arms, legs, upper-body clothes and lower-body clothes. We employ

this segmentation map in Generator II.

3) *Mask map:* Based on the above segmentation map, it is straightforward to obtain the binary mask of the person by merging all segmented regions. In contrast to the segmentation map, this mask map is used to retain the body shape without involving the semantic clues about the person. The mask maps of both the reference and condition person images are used for Generator III.

4) *Head map:* During the synthesis process, the details of the human face are hard to preserve due to its small size. However, it is needed to restore the identity of the reference person after swapping the clothes. To this end, we capture the head region (face and hair) based on the segmentation map, and paste it onto the new synthesized person image. This similar post-processing step is also used in FashionGAN [85].

For $X_r^{(i)}$ and $X_c^{(i)}$, we denote their four feature maps as $\{P_r^{(i)}, S_r^{(i)}, M_r^{(i)}, H_r^{(i)}\}$ and $\{P_c^{(i)}, S_c^{(i)}, M_c^{(i)}, H_c^{(i)}\}$, respectively. Subsequently, we will omit the superscript i for notational simplicity. We should mention that, these person representations are simple and efficient to extract without extra manual tuning. Note that, our representations are semantically richer than previous works [85, 105, 106].

Overview architecture

To render clothes from a person image on to another one, we propose an image synthesis framework (SwapGAN) based on conditional generative adversarial networks. Figure 8.9 illustrates the overview of SwapGAN, which has three different generators for pose-conditioned generation, segmentation-conditioned generation and mask generation, respectively.

Pose-conditioned generation

We begin to introduce the first generative stage conditioned on the pose map. As illustrated in Figure 8.9, we concatenate the condition image X_c and the reference pose map P_r together, and take them as input into the pose-based generative network, *i.e.* Generator I. We can express the synthesized image with

$$X_{G_\mathrm{I}} = G_\mathrm{I}(X_c, P_r). \tag{8.12}$$

We should mention that, the pose map can not only localize the human key-points, but also constrain the body shape of the synthesized person image to be the same as the reference person.

Next, X_{G_I} and X_c are integrated together to fake the discriminator D. Compared with the real pair of X_r and X_c, G_I learns to produce more realistic-looking images

similar to X_r. Following the original GANs [79], we use the negative log likelihood to compute the adversarial loss *w.r.t.* G_{I}

$$\mathcal{L}_{G_{\mathrm{I}}} = \mathbb{E}_{X_c \sim p_{data}(X_c), P_r \sim p_{data}(P_r)}[\log(D(X_{G_{\mathrm{I}}}, X_c))], \tag{8.13}$$

where $p_{data}(\cdot)$ indicates the empirical distributions of training data. As suggested in LSGAN [233], the least square loss is efficient to improve both the stability of training and the quality of generated images. Driven by this, we turn to use the least-square adversarial loss to represent $\mathcal{L}_{G_{\mathrm{I}}}$:

$$\mathcal{L}_{G_{\mathrm{I}}} = \mathbb{E}_{X_c \sim p_{data}(X_c), P_r \sim p_{data}(P_r)}[(D(X_{G_{\mathrm{I}}}, X_c) - 1)^2], \tag{8.14}$$

The objective for Generator I is to minimize $\mathcal{L}_{G_{\mathrm{I}}}$.

Segmentation-conditioned generation

Given two arbitrary person images, Generator I can synthesize new images by exchanging the clothes and its results therefore can meet the goal of this task. However, the key-points in the pose map are mainly used to measure the localization information of body parts, but pay little attention to the style of the target clothes in the condition image. To address this limitation, we propose to leverage the person segmentation map, which can take into consideration semantic information about the clothes.

Empirically, if $X_{G_{\mathrm{I}}}$ has derived the target clothes from X_c, it should be possible to return the clothes back to the condition person again. In this way, the fashion style of the clothes can be reconstructed well during the synthesis process. This idea motivates the second generative stage that aims towards synthesizing another new image as similar as the condition image X_c. Specifically, we build a segmentation-based generative network (*i.e.* Generator II in Figure 8.9), on top of the output of Generator I. Generator II takes as input the concatenation of the synthesized image $X_{G_{\mathrm{I}}}$ and the condition segmentation map S_c. As a result, we can obtain a new synthesized image from the output of Generator II:

$$X_{G_{\mathrm{II}}} = G_{\mathrm{II}}(X_{G_{\mathrm{I}}}, S_c) = G_{\mathrm{II}}(G_{\mathrm{I}}(X_c, P_r), S_c). \tag{8.15}$$

Ideally, $X_{G_{\mathrm{II}}}$ should be as similar as the original input X_c. From X_c to $X_{G_{\mathrm{II}}}$, the integration of the first and second generative stages actually construct an auto-encoder paradigm. It can help improve the quality and semantics of the generated image $X_{G_{\mathrm{I}}}$. For instance, Generator I may mistake the fashion style by transferring long sleeves to be short sleeves. However, Generator II is capable of correcting the mistake, because the segmentation map includes the lost information about the long sleeves. Next, we incorporate X_r and $X_{G_{\mathrm{II}}}$ into the same discriminator D, and

compute the generative loss function of G_{II}

$$\mathcal{L}_{G_{\mathrm{II}}} = \mathbb{E}_{X_r \sim p_{data}(X_r), S_c \sim p_{data}(S_c)}[(D(X_r, X_{G_{\mathrm{II}}}) - 1)^2]. \tag{8.16}$$

Minimizing this loss can jointly optimize Generator II and Generator I.

Mask generation

Although the pose map and segmentation map have provided some information about the body shape, it is encouraged to learn another generative network to explicitly constrain the synthesized images. As shown in Figure 8.9, we employ a shared Generator III to perform the mask generation for both $X_{G_{\mathrm{I}}}$ and $X_{G_{\mathrm{II}}}$. Different from Generator I and Generator II, Generator III takes only one image as input without specifying other conditions. The two generated masks, denoted as $M_{G_{\mathrm{III}}(X_{G_{\mathrm{I}}})}$ and $M_{G_{\mathrm{III}}(X_{G_{\mathrm{II}}})}$, should consistently match the reference mask M_r and the condition mask M_c, respectively. We define their mask-consistency loss as follows:

$$\begin{aligned}\mathcal{L}_{G_{\mathrm{III}}} =& \mathbb{E}_{M_r \sim p_{data}(M_r)}[||M_{G_{\mathrm{III}}(X_{G_{\mathrm{I}}})} - M_r||_1] \\ &+ \mathbb{E}_{M_c \sim p_{data}(M_c)}[||M_{G_{\mathrm{III}}(X_{G_{\mathrm{II}}})} - M_c||_1].\end{aligned} \tag{8.17}$$

Both G_{I} and G_{II} can benefit from the loss $\mathcal{L}_{G_{\mathrm{III}}}$ to update the synthesis process. Note that, $\mathcal{L}_{G_{\mathrm{III}}}$ will not update the parameters of the discriminator D, because the generated masks are unnecessary to feed into the discriminator. In Figure 8.9, it can be seen that, after training, the generated masks end up similar to the reference and condition mask maps.

Full Objective

The SwapGAN model including three generators and one discriminator can be trained end-to-end. The total generation loss combines the adversarial loss (*i.e.* $\mathcal{L}_{G_{\mathrm{I}}}$ and $\mathcal{L}_{G_{\mathrm{II}}}$) and the mask-consistency loss (*i.e.* $\mathcal{L}_{G_{\mathrm{III}}}$)

$$\mathcal{L}_G = \mathcal{L}_{G_{\mathrm{I}}} + \mathcal{L}_{G_{\mathrm{II}}} + \lambda \mathcal{L}_{G_{\mathrm{III}}}, \tag{8.18}$$

where λ adjusts the weight of $\mathcal{L}_{G_{\mathrm{III}}}$, which we set to 5 in the experiments.

Figure 8.10 shows the structure of the discriminator D. Compared to prior work [246] comparing one real pair and one fake one, our discriminator is able to distinguish one real pair from two fake pairs. Formally, the discrimination loss in D can be

defined with

$$\begin{aligned}\mathcal{L}_D =& \mathbb{E}_{X_r \sim p_{data}(X_r), X_c \sim p_{data}(X_c)}[(D(X_r, X_c) - 1)^2] \\ &+ \mathbb{E}_{X_c \sim p_{data}(X_c), P_r \sim p_{data}(P_r)}[D(X_{G_{\mathrm{I}}}, X_c)^2] \\ &+ \mathbb{E}_{X_r \sim p_{data}(X_r), S_c \sim p_{data}(S_c)}[D(X_r, X_{G_{\mathrm{II}}})^2]. \end{aligned} \tag{8.19}$$

During the training procedure, it is a common practice to iteratively update the parameters of the generators and the discriminator. The full objective in the model is to minimize both $\mathcal{L}_G$ and $\mathcal{L}_D$. The generators attempt to generate more realistic-looking fake images to fool the discriminator. Once the discriminator cannot tell fake images from real ones, then the generators are supposed to properly accomplish the synthesis process. In the testing phase, taking a condition image and the pose map of a reference image as input, the synthesized image from Generator I, *i.e.* $X_{G_{\mathrm{I}}}$, can be used as the desired target image. Additionally, we need to paste the reference head map H_r onto $X_{G_{\mathrm{I}}}$ to make sure the person's identity is preserved.

8.2.2 Network architecture

This section introduces the details about the network architecture of the generators and the discriminator in the SwapGAN.

Generator I and II

By integrating several existing techniques, we design a new generative network for G_{I} and G_{II}. As shown in Figure 8.11, it consists of an encoder, several residual blocks and a decoder. (1) In the encoder, we use four consecutive convolutional layers to represent the input data. (2) There are totally six residual blocks, each of which has two 3×3 convolutional layers and a residual connection on them [10, 80]. (3) As for the decoder, we employ a nearest neighbor interpolation manner to upsample the feature maps, and then transfer the resized feature maps with a 1×1 convolutional layer. Compared with the deconvolution manner based on stride-$\frac{1}{2}$ convolutions, the interpolation manner is simple and efficient to alleviate the checkerboard artifacts, which often occur in generated images [249]. Figure 8.12 visibly compares the generated images by using the two upsampling manners.

In addition, we add skip connections to link the feature maps in the encoder and decoder. As suggested in U-Net [28], the skip connections allow to bridge the down-sampled feature maps directly with the up-sampled ones. They can help retain the spatial correspondences between the input pose/segmentation map and the synthesized image.

Generator III

Since the mask generation is less complicated than the pose-conditioned generation and the segmentation-condition generation, we can make use of a simple U-Net [28] to build G_{III}. Specifically, Generator III learns eight convolutional layers in the encoder and eight deconvolutional layers in the decoder. Similarly, the symmetric skip connections are added between the encoder and the decoder. The residual blocks are not used in G_{III}. Notably, G_{III} can be built as well with the same generative network as G_{I} and G_{II}, however, we find that it cannot bring further improvements for the generated masks.

Discriminator

We build the discriminator D based on the Markovian network from PatchGANs [88], which is encouraged to preserve local high-frequency features. As shown in Figure 8.10, D uses four consecutive layers to convolve the concatenated real or fake image pairs. Lastly, an additional convolutional layer can output a 1-dimensional feature map to classify the patches on the input images are real or fake.

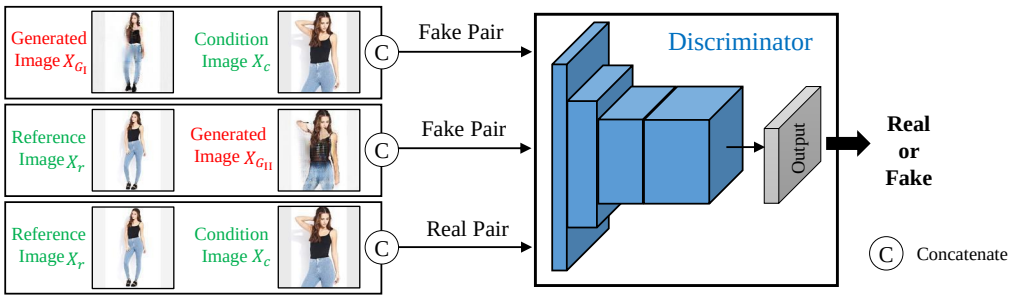


Figure 8.10: Overview of the discriminator D in SwapGAN. It aims to distinguish two fake image pairs from the real pair.

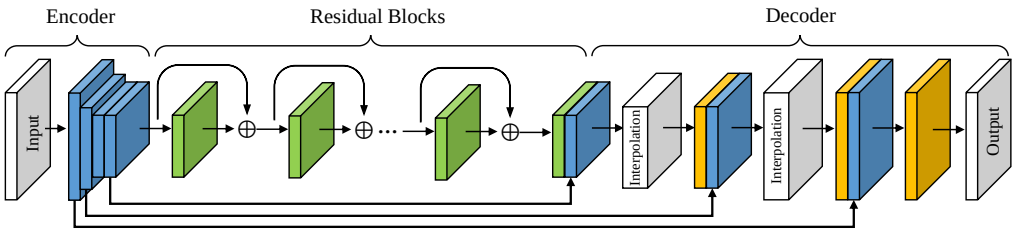
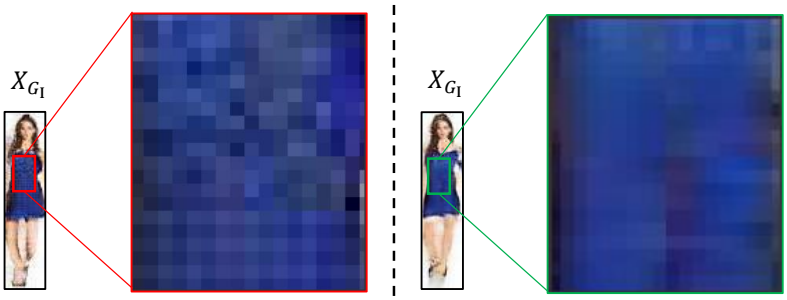


Figure 8.11: Network architecture of both Generator I and II. It is composed of three parts: encoder, residual blocks and decoder. We use additional skip connections to couple the feature maps in the encoder and decoder. In the decoder, we perform the upsampling with an interpolation manner instead of the traditional deconvolution manner.



(a) Deconvolution upsampling (b) Interpolation upsampling

Figure 8.12: Comparison of using two different upsampling manners in the generator. The deconvolution manner results in more checkerboard artifacts that will decrease the generation quality. To alleviate this issue, we use the interpolation manner to generate smooth images. See more details when zoomed-in.

(a) inappropriate person images

(b) appropriate person images

Figure 8.13: Examples of (a) inappropriate and (b) appropriate person images. Considering the goal of person-to-person clothing swapping, we collect the front-view images with both upper-body and lower-body clothes visible.

8.2.3 Experiment setup

Dataset protocol

Currently, DeepFashion [241] is one of the largest datasets for fashion oriented research. We used its In-shop Clothes Retrieval Benchmark, which has a number of in-shop person images with various poses and scales. However, many of the images are inappropriate to the clothing swapping task, due to some issues like missing human faces, back-view images and only upper-body clothes visible. To avoid these issues, we selected front-view person images where the clothing items are shown clearly. In Figure 8.13, we show some examples of inappropriate and appropriate person images. In the training set, we collected 6,000 person images corresponding to 3,000 image pairs, each of which has two images of the same person wearing the same clothes but showing different poses. The testing set contains 1,372 images.

Figure 8.14: Qualitative results of our SwapGAN on the test set. We show four reference images in the first row and four condition images in the first column. The reference person can wear the desired clothes in the condition image while preserving the original pose and body shape.

Implementation Details

We employed the Adam algorithm [234] to optimize the entire SwapGAN with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The initial learning rate for the generators and discriminator was 0.0002, and was linearly decayed after 50 epoches. The entire training procedure was terminated after 100 epochs. All the images were re-scaled to 128×128 pixels. We used a mini-batch size of 8. We implemented the method on the TensorFlow library [235] with a NVIDIA TITAN X GPU card.

Compared methods

We compare our SwapGAN with other three methods described as follows.

Poisson image blending [85]: it is the 2D non-parametric method that uses the Poisson image blending algorithm to apply the target clothes in the condition person image on the person of the reference image. This method is used as a baseline in FashionGAN [85].

TPS warping [106]: this is another non-parametric method. It first estimates a thin plate spline (TPS) transformation and then pastes the warped clothes on the reference image. This is a baseline method in VITON [106].

VITON [106]: in contrast to non-parametric methods, it proposes an encoder-decoder network to generate a new reference person image wearing the target clothes.

We note that, all the three compared methods require segmenting the target clothes from the condition person images. By this way, they can learn the transformations between two different images.

8.2.4 Results and discussion

First, we compare our SwapGAN with other compared methods in terms of both qualitative and quantitative evaluations. Then, we perform ablation study to provide deep insights into SwapGAN.

Qualitative evaluation

This experiment aims to qualitatively show the effectiveness of our method for person-to-person clothing swapping. Figure 8.14 shows our new synthesized images. As for each row, the clothes in the condition image are worn on different reference persons. Also, each column indicates that the same reference person is re-dressed with different clothes. It can be seen that all the reference persons can properly wear the target clothes in the condition images and retain their original poses and body shapes as well. Since we paste the reference head map to ensure the person's identity, some generated images therefore seems a little unnatural.

Next, we compare our results with those of the compared methods. In Figure 8.15, we present a reference image and three condition images. To assess the robustness for different pose deformations, the persons in the three condition images have small, moderate and large pose deformations, respectively, compared to the person in the reference image. From the results, we can see that the Poisson image blending method fails to perform this task. The similar observation is also presented in [85]. Instead of generating a new image, the TPS warping method learns to transform the target clothes and simply pastes it on the reference person. Although the color information can be well preserved in its results, we can notice obvious inconsistency between the warped target clothes and the body of the reference person. The results

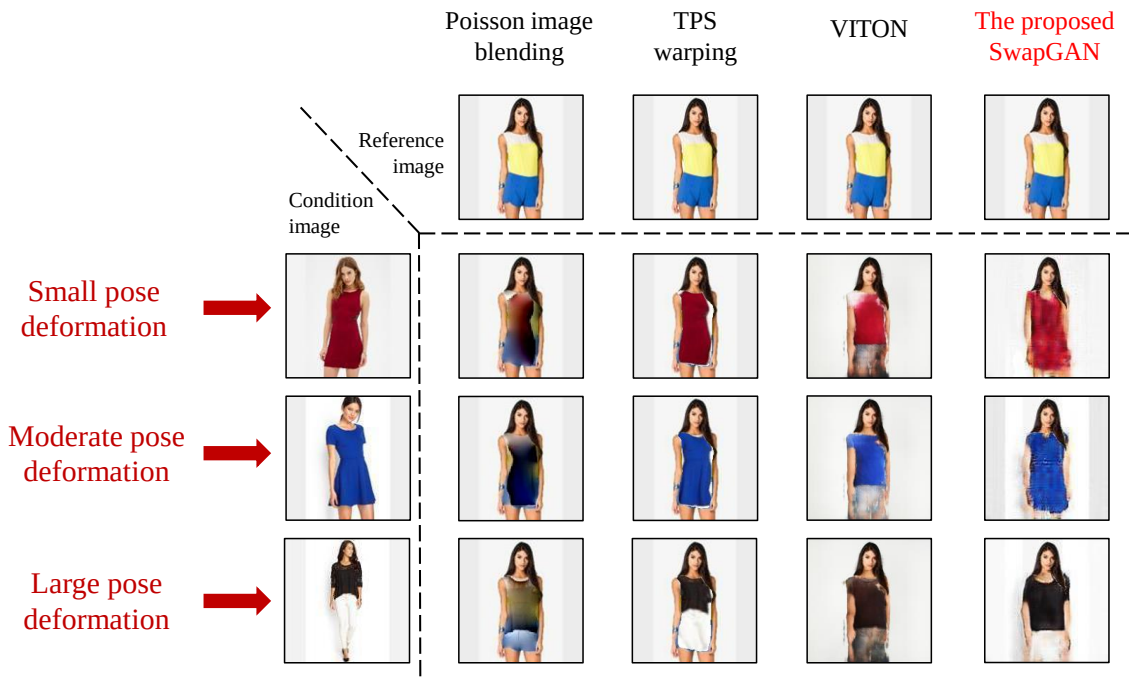


Figure 8.15: Qualitative comparison of different methods. When comparing with the person in the reference image, the persons in the three condition images have small, moderate and large pose deformations, respectively. Compared to other methods, our SwapGAN can visibly provide superior images. Our method is robust to different pose deformations, even the large case in the last row.

of VITON are not satisfactory, because their model is trained with simple stand-alone and flat clothes images, rather than various warped clothes on the condition persons. Compared to the above methods, SwapGAN can generate superior new images for all the condition images. In addition, our method is robust to different pose deformations, however, the three compared methods are weak in the robustness.

Quantitative evaluation

In addition to qualitative results, we adopt a common quantitative metric, Inception Score (IS) [250], to assess the methods. IS is based on the Google's inception CNN model [11], which predicts a distribution $p(y|x)$, measuring the probability assigned to image x to belong to class y. Formally, the computation of IS is expressed by

$$IS = exp\left(\mathbb{E}_{x\sim p_g}\left[KL\left(p(y|x)\|p(y)\right)\right]\right), \tag{8.20}$$

where p_g indicates the distribution of a generative model. $KL(p(y|x)\|p(y))$ measures The Kullback-Leibler divergence [251] between $p(y|x)$ and $p(y)$:

$$KL\left(p(y|x)\|p(y)\right) = \sum_{k=1}^{K} p_k(y|x)\log\frac{p_k(y|x)}{p_k(y)}. \tag{8.21}$$

Table 8.2: Quantitative comparison of different approaches with inception scores (higher is better). Our SwapGAN can outperform the other three compared methods with considerable gains.

Method	Inception score
Poisson image blending	2.10 ± 0.14
TPS warping	2.45 ± 0.12
VITON	2.40 ± 0.05
SwapGAN	2.65 ± 0.09

For the 1,372 images in the test set, we iteratively make each image as the reference image, and then randomly select another 25 images to be its corresponding condition images. As a result, we can collect about 34,000 reference-condition pairs, each of which can produce an image to evaluate. Table 8.2 reports the inception scores towards the 34,000 images. Interestingly, the TPS warping method has a greater score than VITON, because it simply pastes the warped clothes on the reference image, which can help preserve the color information. However, it cannot generate a new image like VITON and SwapGAN. In [106], they also discuss the limitation of the TPS warping method. Overall, SwapGAN achieves a higher score than the other three methods.

8.2.5 Ablation study

We demonstrate ablation results about SwapGAN and analyze the effects of its generators on the performance. To be more specific, we implement two ablation models, which are variants of the full SwapGAN model. The first ablation model is named by Generator I&III, which excludes the segmentation-conditioned generation. The second one, called Generator I&II, keeps the first and second generations but excludes the mask generation. Figure 8.16 shows two generated image samples, from which we have the following observations:

(1) Effect of Generator II. As can be seen in the first row, Generator I&III mistakes the fashion style of the target clothes, because it changes the short sleeves in the condition image to be long sleeves in the new generated image. However, both the Generator I&II model and the full SwapGAN model can avoid this semantic inconsistency due to using the segmentation map in Generator II. It verifies the effectiveness of Generator II for maintaining the style.

(2) Effect of Generator III. Considering the generated images from the Generator I&II model, some parts of the human body are not preserved well, for example, the right arms. By running the mask generation, the full SwapGAN model can produce a more complete body shape similar with the reference image. This demonstrates the benefit of Generator III for our method.

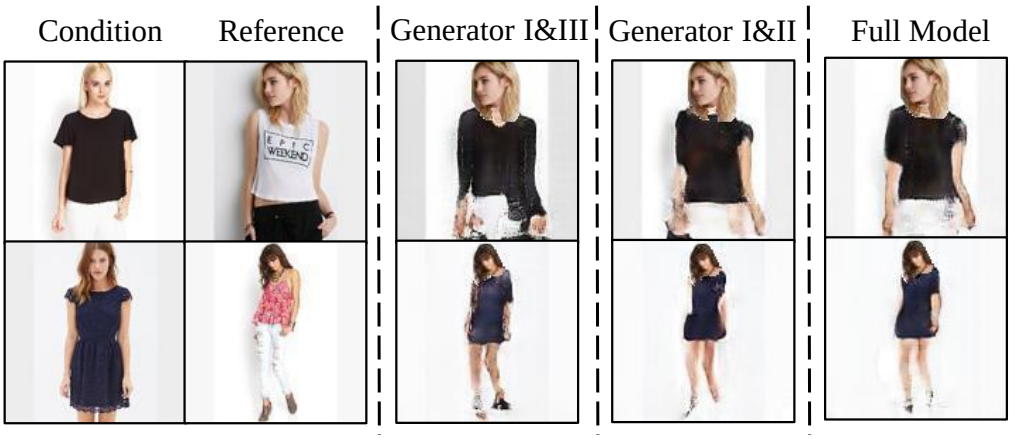


Figure 8.16: Ablation study on different variants of our method. Comparably, the full model can outperform the other two baseline models in terms of generation quality and semantics.

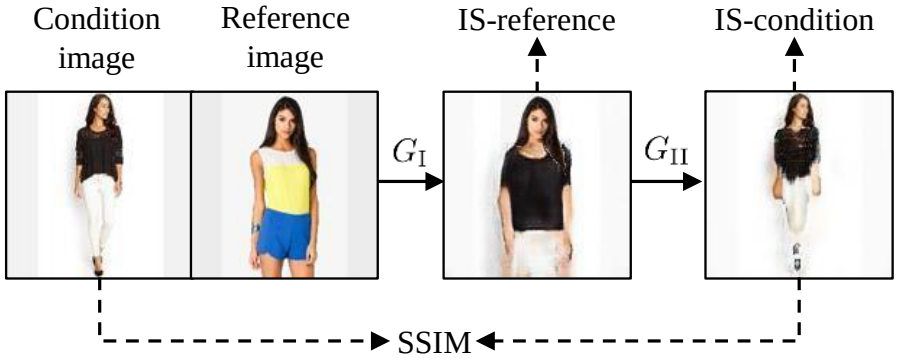


Figure 8.17: Pipeline of our testing procedure with computing two inception scores and a SSIM accuracy.

Figure 8.18: Failure cases of our method for synthesizing complicated color and texture on the clothes.

In terms of quantitative results, we exploit a new test procedure as shown in Figure 8.17. Since SwapGAN can synthesize two new images from G_{I} and G_{II}, we can compute their inception scores respectively, denoted as IS-reference and IS-condition. In addition, the synthesized image from G_{II} is a reconstructed image of the original condition image. Hence, we can adopt another quantitative metric, Structural Similarity (SSIM) [252], to measure the reconstructed similarity.

In Table 8.3, we compare the quantitative results between two ablation models and the full SwapGAN model. Notably, the Generator I&III model has no IS-

Table 8.3: Quantitative results of our different models.

Method	IS-reference	IS-condition	SSIM
Generator I&III	2.47 ± 0.11	–	–
Generator I&II	2.36 ± 0.14	2.66 ± 0.12	0.708
Full Model	2.65 ± 0.09	2.85 ± 0.12	0.717

condition and SSIM accuracy, because it excludes G_{II}. We can see that the full model consistently outperforms the other two ablation models by a considerable margin, in terms of both IS-reference and IS-condition metrics. Moreover, the full model achieves a higher SSIM accuracy than Generator I&II. These quantitative results are consistent with our observation achieved from the qualitative evaluation.

8.2.6 Limitations and discussion

Our method has achieved promising results in many cases, but still has some limitations. First, human faces become blurred in the synthesis process, because it is hard for the generator to restore the detailed face of the reference person. To alleviate this limitation, we employ a post-processing step by pasting the reference head map onto the synthesized image. Second, our method may fail to capture rich color and texture information of the clothes, for example, the failure cases in Figure 8.18. This problem is caused by the limited capability of the adversarial loss. One approach for solving it is to impose additional losses like the perception loss [80], but it will increase the memory cost and training time.

8.3 Chapter Conclusions

First, this work provided an extensive and empirical study on the cycle-consistent generative networks for unsupervised image translation. The comprehensive results demonstrated the effectiveness of our designed models. Besides, the insights observed in this work could help in designing other new cycle-consistency models. In the future, it is straightforward and promising to develop Long and Nest CycleGAN with more generators and cycles. Also, it is interesting to employ a weight-sharing mechanism to avoid increasing memory.

Second, we proposed a novel multi-stage generative adversarial framework to address the problem of person-to-person clothing swapping. Advantageously, it could render the clothing style and preserve the pose and body shape within a multi-stage model. In addition, our model was able to train end-to-end. Qualitative and quantitative results in the experiments demonstrated the effectiveness of our approach. In the future, we plan on developing our approach for images in the wild.

Chapter 9

Conclusions

In this thesis, we have devoted previous seven research chapters to address the eight research questions regarding three themes: classification, retrieval and synthesis. In this chapter, we derive main findings from our approaches and results. In addition, we discuss limitations of our approaches and possible solutions to address them. Lastly, we point out several directions for future work.

9.1 Main Findings

In each research chapter, we have proposed a new approach to answer the corresponding research question. In the next, we will conclude these approaches and present main findings inspired by experimental results and empirical analysis.

(1) We began the research part in **Chapter 2** by focusing on exploiting deep fusion networks for classification. We built a novel deep fusion architecture (*i.e.* CFN) on top of plain CNNs, and witnessed its effectiveness for diverse tasks ranging from image-level to pixel-level classification. In addition, it is promising to apply CFN to more applications such as object detection and visual tracking.

(2) In **Chapter 3** we further exploited CNNs to improve its robustness for edge detection. In contrast to using a general supervision, we proposed to develop relaxed deep supervision (RDS) to guide different intermediate layers. We observed that hierarchical supervisory signals with additional relaxed labels could be consistent with the diversities in different layers. We believe that it is feasible to adapt RDS to other pixel-level predictions, such as image segmentation and saliency detection.

(3) After investigating the classification theme, we then turned to address the questions about the retrieval theme in Chapters 4-7. In **Chapter 4**, we provided a good attempt to incorporate deep features into the inverted index scheme and exploited a novel DeepIndex framework for accurate and efficient image retrieval. In addition, we extended DeepIndex by integrating different deep features and built a 2-D DeepIndex structure that consists of two kinds of variants: intra-CNN and inter-CNN. We found that, Intra-CNN was simpler to build than Inter-CNN, but Inter-CNN could be viewed as a solution to bridge the gap between mid-level and high-level deep feature representations.

(4) Driven by the increasing popularity of large-scale multi-media data, we began to study the cross-modal retrieval task in **Chapter 5**. Specifically, we developed a deep matching network using recurrent residual fusion (RRF) as building blocks for improving visual-textual embeddings. Our work showed that RRF could recurrently improve feature embeddings while retaining the number of network parameters. In addition, the fusion module was efficient to integrate intermediate outputs during the recurrent stage. Potentially, RRF-Net would be seamlessly integrated into other multi-modal applications like image captioning and visual question answering.

(5) In **Chapter 6**, we proposed cycle-consistent embeddings in an image-text matching network, which could incorporate both inter-modal correlations and intra-modal consistency for learning robust visual and textual embeddings. During training, we integrated several ranking losses jointly to optimize the whole embedding learning. For a robust inference, we further leveraged two late-fusion approaches to integrate the matching scores of multiple embedding features. From the experimental results, we showed that cycle-consistency embeddings could effectively promote the cross-modal retrieval performance, compared to a single embedding.

(6) In an effort to accomplish both classification and retrieval, in **Chapter 7** we exploited a unified network for joint multi-modal matching and classification (MMC-Net). The experimental results demonstrated the robustness and effectiveness of the MMC-Net model, compared to the baseline models. On the one hand, the classification component was beneficial to alleviate the biased annotations, so that the model could learn more robust embedding features. On the other hand, the matching component was able to bridge the modality gap between vision and language, and thus combining visual and textual embedding features could produce a more discriminative multi-modal representation.

(7) After focusing on the classification and retrieval themes, our attention moved to the synthesis theme. In **Chapter 8**, we focused on addressing two research questions. The first one was what factors would affect the performance of generative models on the translation tasks. To answer this question, we extended the vanilla CycleGAN with new improvements and showed two extended models. First, we found that the long cycle could leverage more generators to further increase the generation abilities of the model and improve the quality of synthesized images. In addition, the additional inner cycles were able to directly connect the intermediate generators and provided more cycle-consistency losses to constrain the translation. The findings in this work could help in designing other cycle-consistent generative networks for solving image-to-image translation tasks.

(8) The second question we considered in **Chapter 8** was how we can exploit a generative model to transfer the fashion style between two person images. To this end, we interpreted the clothing swapping as a problem of pose-based person image generation and proposed a novel multi-stage generative framework (SwapGAN) to fulfill the clothing swapping from the condition person image to the reference one. The whole SwapGAN framework could be end-to-end trained with both adversarial loss and mask-consistency loss. Our work could be a benchmark study and help to drive future research on this task.

9.2 Limitations and Possible Solutions

Our methods in this thesis have addressed the eight research questions and achieved promising results in terms of the three research themes. However, they still have some limitations which can be discussed from the following three perspectives.

Algorithmic perspective

In Chapter 2, the proposed CFN uses a 1×1 kernel filter in the locally-connected fusion module. It can independently consider each spatial location over the feature maps, while may omit the relationships between different spatial locations. To solve it, a potential solution is to utilize larger kernel sizes such as 1×2 and 1×3, which can incorporate the contextual information in the feature maps. In addition, the adaptive weights learned in the fusion module are the same for all the images. An alternative is to learn dynamical weights conditioned on different input images. For example, Brabandere *et al.* [146] propose a Dynamic Filter Network (DFN), where filters are dynamically generated conditioned on an input image.

In Chapter 4, DeepIndex is designed for accurate and efficient retrieval, however, we can find its performance gap with recent state-of-the-art approaches [48]. It is straightforward to improve our results by using more powerful CNNs like ResNet-152. Besides, it is suggestive to extend multiple DeepIndex with three or more deep features, compared to the 2-D case.

In Chapter 8, the extended CycleGAN models, *i.e.* Long CycleGAN and Nest CycleGAN, can improve the generated quality, however, they will increase the training cost due to using more generators. One promising solution is to introduce a weight-sharing mechanism to avoid increasing the cost. In terms of the proposed SwapGAN for person-to-person clothing swapping, it is hard to preserve rich color and texture information in the clothes. This problem may be caused by the limited capability of the original adversarial loss. To overcome it, we can make use of additional losses (*e.g.* perception loss [80]) to help enhance the synthesis process. However, they will increase the memory cost and training time.

Theoretical perspective

In Chapter 3, we have discussed our motivation for exploiting relaxed deep supervision for robust edge detection. Nevertheless, we should still realize that it still lacks of theoretical insights into interpreting the benefit of diverse supervision for training deep neural networks. Recent works [253, 254] propose theoretical approaches to interpreting deep visual representations learned in CNNs. It is encouraged to use these approaches to achieve deeper insights regarding the utility of diverse supervision.

In Chapter 5, we develop a building block based on recurrent residual fusion (RRF) to advance the visual-textual embedding features. We notice that, using more recurrent steps may decrease the performance. One reason is attributed to the potential over-fitting issue while training the model, however, it is hard to prove it in theory. This issue limits further performance improvements. One alternative is to impose the RRF block on more layers, since RRF is a general structure that can potentially be applied to many existing layers in a deep network.

Practical perspective

In Chapter 6, we apply the proposed CycleMatch to solve the task of cross-modal retrieval between images and texts. Although we witness its promising performance for this task, it is encouraged to transfer our method to other challenging tasks, like visual grounding, visual relationship detection and visual reasoning. In addition to the global image-text matching, we should take into account local similarities between visual regions and phases.

In Chapter 7, the proposed MMC-Net, which can jointly accomplish multi-modal matching and classification, requires ground-truth class labels in addition to the paired information. However, some multi-modal datasets (*i.e.* Flickr30K) do not provide the class labels. Therefore, it is infeasible to train the full MMC-Net model. One potential alternative is to automatically construct a dictionary by parsing all the textual descriptions. Then we can label each image with its key words derived from the dictionary. In this way, it is still feasible to accomplish the classification task based on the word-level labels instead of unavailable class labels.

9.3 Future Research Directions

In the previous seven chapters, we have presented many methods to address the research questions regarding the three research themes. A wide variety of future research is also encouraged to advance these themes. In this section, we briefly discuss future research directions regarding each theme.

Zero-shot classification

Zero-shot classification (ZSC) [255] aims to solve the task where not all the classes are represented in the training set. In ZSC, the training and test class sets are disjoint. It needs to learn a visual classifier based on the seen images and their semantic categories, and then transfers the classify to recognize images of unseen classes. Existing approaches can be summarized in three groups. (1) Direct mapping: learning a mapping function from visual features to semantic representations.

(2) Common space learning: constructing a common embedding space where visual features and semantic representations can be correlated. (3) Model parameter transfer: exploiting the inter-class relationship between seen and unseen classes and then transferring the model parameters of seen classes to the unseen ones.

In recent years, deep neural networks have been widely used for solving the ZSC task due to their powerful representation capabilities [60, 75, 256]. Nevertheless, this task remains challenging in discovering the relations between visual features and semantic knowledge, as well as generalizing the relations to unseen classes. Since ZSC relies on discovering the sematic relations between visual and textual features, it is encouraged to incorporate a visual-textual matching component into a ZSC system. Our research on classification and retrieval is related to this future direction.

Generation for cross-modal retrieval

Recall that cross-modal retrieval needs to overcome the semantic gap between two different modalities like vision and language. To achieve it, one common approach is to project visual and textual features into the same embedding space where we need to compare their correlations. However, in most existing datasets, each matched image-text pair has limited samples, for example one image is labeled with one or five descriptions. This issue will hinder the learning capabilities of deep neural networks. Recently, Zheng *et al.* [257] propose to use generation networks to produce more image samples to extend the datasets. Driven by this idea, it is feasible to use GANs to alleviate the lack of image-text samples for cross-modal retrieval. For example, we can generate more realistic-looking images based on the text description, and also create additional descriptions for each image. In addition to cross-modal relations, we can add intra-modal constraints between the real and generated samples. Integrating both cross-modal and intra-modal matching could be beneficial to learn better embedding features. Our research on retrieval and synthesis can be adopted to this future direction.

Unified image synthesis

Recent studies on image-to-image translation have achieved encouraging results for a range of different domain-specific image sets. However, most of existing approaches are inefficient for jointly modeling multi-domain image translation tasks, because they need to train individual generative networks for every two domains, *i.e.* , in order to learn all mappings among N domains, $N \times (N-1)$ generators need to be learned. To address this problem, StarGAN [258] recently proposes a unified generative adversarial network, which allows to translate a range of image domains by using a single generative network. The key point in StarGAN is that it uses

a label (*e.g.* binary or one-hot vector) to represent the domain information. In addition, StarGAN can incorporate different labels from multiple datasets using a simple mask vector to indicate the dataset information. However, one potential issue may make StarGAN fail when different datasets have some overlapped labels. One potential solution is that, we can extend the mask vector to be consistent with the number of domains, rather than with the number of datasets. In this way, StarGAN can discard overlapped domain labels in different datasets. We believe that exploring a unified image generative network is still a promising future work.

Bibliography

[1] Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61** (2015) 85–117

[2] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016) `http://www.deeplearningbook.org`.

[3] Cun, L., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: NIPS. (1990)

[4] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1106–1114

[5] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115** (2015) 211–252

[6] Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: CVPR workshop. (2014)

[7] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015)

[8] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015) 1–9

[9] Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: NIPS. (2015) 2377–2385

[10] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 770–778

[11] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI. (2017) 4278–4284

[12] Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC. (2016)

[13] Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.: Deep networks with stochastic depth. In: ECCV. (2016) 646–661

[14] Veit, A., Wilber, M.J., Belongie, S.: Residual networks behave like ensembles of relatively shallow networks. In: NIPS. (2016) 550–558

[15] Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: ECCV. (2014) 329–344

[16] Liu, L., Shen, C., van den Hengel, A.: The treasure beneath convolutional layers: cross convolutional layer pooling for image classification. In: CVPR. (2015) 4749–4757

[17] Sermanet, P., Chintala, S., LeCun, Y.: Convolutional neural networks applied to house numbers digit classification. In: ICPR. (2012)

[18] Yang, S., Ramanan, D.: Multi-scale recognition with DAG-CNNs. In: ICCV. (2015) 1215–1223

[19] Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV. (2003) 1470–1477

[20] Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: CVPR. (2010) 3304–3311

[21] Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV. (2010) 143–156

[22] Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale orderless pooling of deep convolutional activation features. In: ECCV. (2014) 392–407

[23] Yue-Hei Ng, J., Yang, F., Davis, L.S.: Exploiting local features from deep networks for image retrieval. In: CVPR, Deep Vision workshop. (2015)

[24] Wei, X.S., Gao, B.B., Wu, J.: Deep spatial pyramid ensemble for cultural event recognition. In: ICCV Workshops. (2015)

[25] Yoo, D., Park, S., Lee, J.Y., Kweon, I.S.: Multi-scale pyramid pooling for deep convolutional representation. In: CVPR, DeepVision workshop. (2015)

[26] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015) 3431–3440

[27] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. In: ICLR. (2015)

[28] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. (2015) 234–241

[29] Bertasius, G., Shi, J., Torresani, L.: Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: CVPR. (2015) 4380–4389

[30] Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In: CVPR. (2015) 3982–3991

[31] Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV. (2015) 1395–1403

[32] Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS. (2014) 2366–2374

[33] Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015) 2650–2658

[34] Wang, X., Fouhey, D.F., Gupta, A.: Designing deep networks for surface normal estimation. In: CVPR. (2015) 539–547

[35] Li, G., Yu, Y.: Visual saliency based on multiscale deep features. In: CVPR. (2015) 5455–5463

[36] Wang, L., Lu, H., Ruan, X., Yang, M.H.: Deep networks for saliency detection via local estimation and global search. In: CVPR. (2015) 3183–3192

[37] Wang, L., Wang, L., Lu, H., Zhang, P., Ruan, X.: Saliency detection with recurrent fully convolutional networks. In: ECCV. (2016) 825–841

[38] Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based multimedia information retrieval: State of the art and challenges. TOMCCAP **2** (2006) 1–19

[39] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)

[40] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60** (2004) 91–110

[41] Wengert, C., Douze, M., Jégou, H.: Bag-of-colors for improved image search. In: ACM Multimedia. (2011) 1437–1440

[42] Wan, J., Wang, D., Hoi, S.C.H., Wu, P., Zhu, J., Zhang, Y., Li, J.: Deep learning for content-based image retrieval: A comprehensive study. In: ACM Multimedia. (2014) 157–166

[43] Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. In: ECCV. (2016) 241–257

[44] Radenović, F., Tolias, G., Chum, O.: Cnn image retrieval learns from bow: Unsupervised

fine-tuning with hard examples. In: ECCV. (2016) 3–20

[45] Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.S.: Neural codes for image retrieval. In: ECCV. (2014) 584–599

[46] Zheng, L., Wang, S., He, F., Tian, Q.: Seeing the big picture: Deep embedding with contextual evidences. CoRR **abs/1406.0132** (2014)

[47] Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV. (2008) 304–317

[48] Zheng, L., Yang, Y., Tian, Q.: SIFT meets CNN: A decade survey of instance retrieval. TPAMI **40** (2018) 1224–1244

[49] Yan, F., Mikolajczyk, K.: Deep correlation for matching images and text. In: CVPR. (2015) 3441–3450

[50] Ranjan, V., Rasiwasia, N., Jawahar, C.V.: Multi-label cross-modal retrieval. In: ICCV. (2015) 4094–4102

[51] Klein, B., Lev, G., Sadeh, G., Wolf, L.: Associating neural word embeddings with deep image representations using fisher vectors. In: CVPR. (2015) 4437–4446

[52] Ma, L., Lu, Z., Shang, L., Li, H.: Multimodal convolutional neural networks for matching image and sentence. In: ICCV. (2015) 2623–2631

[53] Wang, L., Li, Y., Lazebnik, S.: Learning deep structure-preserving image-text embeddings. In: CVPR. (2016) 5005–5013

[54] Wei, Y., Zhao, Y., Lu, C., Wei, S., Liu, L., Zhu, Z., Yan, S.: Cross-modal retrieval with cnn visual features: A new baseline. IEEE Transactions on Cybernetics **47** (2017) 449–460

[55] Karpathy, A., Li, F.F.: Deep visual-semantic alignments for generating image descriptions. In: CVPR. (2015) 3128–3137

[56] Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: CVPR. (2015) 3156–3164

[57] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., Parikh, D.: VQA: Visual question answering. In: ICCV. (2015) 2425–2433

[58] Malinowski, M., Rohrbach, M., Fritz, M.: Ask your neurons: A neural-based approach to answering questions about images. In: ICCV. (2015) 1–9

[59] Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: CVPR. (2015) 2927–2936

[60] Xian, Y., Akata, Z., Sharma, G., Nguyen, Q., Hein, M., Schiele, B.: Latent embeddings for zero-shot classification. In: CVPR. (2016) 69–77

[61] Hotelling, H.: Relations between two sets of variates. Biometrika **28** (1936) 321–377

[62] Andrew, G., Arora, R., Livescu, K., Bilmes, J.: Deep canonical correlation analysis. In: ICML. (2013) 1247–1255

[63] Gong, Y., Ke, Q., Isard, M., Lazebnik, S.: A multi-view embedding space for modeling internet images, tags, and their semantics. IJCV **106** (2014) 210–233

[64] Liu, Y., Guo, Y., Bakker, E.M., Lew, M.S.: Learning a recurrent residual fusion network for multimodal matching. In: ICCV. (2017) 4107–4116

[65] Niu, Z., Zhou, M., Wang, L., Gao, X., Hua, G.: Hierarchical multimodal lstm for dense visual-semantic embedding. In: ICCV. (2017) 1881–1889

[66] Kiros, R., Salakhutdinov, R., Zemel, R.S.: Unifying visual-semantic embeddings with multimodal neural language models. In: NIPS workshop. (2014)

[67] Nam, H., Ha, J.W., Kim, J.: Dual attention networks for multimodal reasoning and matching. In: CVPR. (2017) 299–307

[68] Huang, Y., Wang, W., Wang, L.: Instance-aware image and sentence matching with selective multimodal lstm. In: CVPR. (2017) 2310–2318

[69] Zheng, Z., Zheng, L., Garrett, M., Yang, Y., Shen, Y.: Dual-path convolutional image-text

embedding. CoRR **abs/1711.05535** (2017)

[70] Wang, B., Yang, Y., Xu, X., Hanjalic, A., Shen, H.T.: Adversarial cross-modal retrieval. In: ACM Multimedia. (2017) 154–162

[71] Feng, F., Wang, X., Li, R.: Cross-modal retrieval with correspondence autoencoder. In: ACM Multimedia. (2014) 7–16

[72] Habibian, A., Mensink, T., Snoek, C.G.: Videostory: A new multimedia embedding for few-example recognition and translation of events. In: ACM Multimedia. (2014) 17–26

[73] Rastegar, S., Soleymani, M., Rabiee, H.R., Mohsen Shojaee, S.: Mdl-cw: A multimodal deep learning framework with cross weights. In: CVPR. (2016) 2601–2609

[74] Vukotić, V., Raymond, C., Gravier, G.: Bidirectional joint representation learning with symmetrical deep neural networks for multimodal and crossmodal applications. In: ICMR. (2016) 343–346

[75] Kodirov, E., Xiang, T., Gong, S.: Semantic autoencoder for zero-shot learning. In: CVPR. (2017) 3174–3183

[76] Eisenschtat, A., Wolf, L.: Linking image and text with 2-way nets. In: CVPR. (2017) 4601–4611

[77] Gu, J., Cai, J., Joty, S., Niu, L., Wang, G.: Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In: CVPR. (2018)

[78] Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2face: Real-time face capture and reenactment of rgb videos. In: CVPR. (2016) 2387–2395

[79] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS. (2014) 2672–2680

[80] Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV. (2016) 694–711

[81] Shuhui Jiang, Y.F.: Fashion style generator. In: IJCAI. (2017) 3721–3727

[82] Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: ECCV. (2016) 702–716

[83] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: CVPR. (2017) 4105–4113

[84] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: ICML. (2016) 1060–1069

[85] Zhu, S., Fidler, S., Urtasun, R., Lin, D., Chen, C.L.: Be your own prada: Fashion synthesis with structural coherence. In: ICCV. (2017) 1689–1697

[86] Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR **abs/1411.1784** (2014)

[87] Yan, X., Yang, J., Sohn, K., Lee, H.: Attribute2image: Conditional image generation from visual attributes. In: ECCV. (2016) 776–791

[88] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR. (2017) 5967–5976

[89] Liu, M.Y., Tuzel, O.: Coupled generative adversarial networks. In: NIPS. (2016) 469–477

[90] Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NIPS. (2017) 700–708

[91] Taigman, Y., Polyak, A., Wolf, L.: Unsupervised cross-domain image generation. In: ICLR. (2017)

[92] Benaim, S., Wolf, L.: One-sided unsupervised domain mapping. In: NIPS. (2017) 752–762

[93] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., Krishnan, D.: Unsupervised pixel-level domain adaptation with generative adversarial networks. In: CVPR. (2017) 95–104

[94] Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV. (2017) 2223–2232

[95] Kiapour, M.H., Han, X., Lazebnik, S., Berg, A.C., Berg, T.L.: Where to buy it: Matching street clothing photos in online shops. In: ICCV. (2015) 3343–3351

[96] Al-Halah, Z., Stiefelhagen, R., Grauman, K.: Fashion forward: Forecasting visual style in fashion. In: ICCV. (2017) 388–397

[97] Liu, S., Liang, X., Liu, L., Lu, K., Lin, L., Cao, X., Yan, S.: Fashion parsing with video context. IEEE Transactions on Multimedia **17** (2015) 1347–1358

[98] Guan, P., Reiss, L., Hirshberg, D.A., Weiss, E., Black, M.J.: Drape: Dressing any person. ACM Trans. Graph. **31** (2012) 35:1–35:10

[99] Zhou, Z., Shu, B., Zhuo, S., Deng, X., Tan, P., Lin, S.: Image-based clothes animation for virtual fitting. In: SIGGRAPH Asia. (2012)

[100] Movania, M.M., Farbiz, F.: Depth image based cloth deformation for virtual try-on. In: ACM SIGGRAPH. (2013)

[101] Yang, S., Ambert, T., Pan, Z., Wang, K., Yu, L., Berg, T.L., Lin, M.C.: Detailed garment recovery from a single-view image. CoRR **abs/1608.01250** (2016)

[102] Hauswiesner, S., Straka, M., Reitmayr, G.: Virtual try-on through image-based rendering. IEEE Transactions on Visualization and Computer Graphics **19** (2013) 1552–1565

[103] Pons-Moll, G., Pujades, S., Hu, S., Black, M.J.: Clothcap: Seamless 4d clothing capture and retargeting. ACM Trans. Graph. **36** (2017) 73:1–73:15

[104] Gultepe, U., Gudukbay, U.: Real-time virtual fitting with body measurement and motion smoothing. Computers & Graphics **43** (2014) 31–43

[105] Jetchev, N., Bergmann, U.: The conditional analogy gan: Swapping fashion articles on people images. In: ICCV Workshop. (2017)

[106] Han, X., Wu, Z., Wu, Z., Yu, R., Davis, L.S.: VITON: an image-based virtual try-on network. In: CVPR. (2018)

[107] Liu, Y., Guo, Y., S. Lew, M.: On the exploration of convolutional fusion networks for visual recognition. In: MMM. (2017) 277–289

[108] Liu, Y., Guo, Y., Georgiou, T., Lew, M.S.: Fusion that matters: convolutional fusion networks for visual recognition. Multimedia Tools and Applications (2018)

[109] Canny, J.: A computational approach to edge detection. TPAMI **8** (1986) 679–698

[110] Xiaofeng, R., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: NIPS. (2012) 593–601

[111] Leordeanu, M., Sukthankar, R., Sminchisescu, C.: Generalized boundaries from multiple image interpretations. TPAMI **36** (2014) 1312–1324

[112] Sironi, A., Lepetit, V., Fua, P.: Projection onto the manifold of elongated structures for accurate extraction. In: ICCV. (2015) 316–324

[113] Kivinen, J.J., Williams, C.K.I., Heess, N.: Visual boundary prediction: A deep neural prediction network and quality dissection. In: AISTATS. (2014)

[114] Liu, Y., Lew, M.S.: Learning relaxed deep supervision for better edge detection. In: CVPR. (2016) 231–240

[115] Liu, Y., Guo, Y., Wu, S., Lew, M.S.: Deepindex for accurate and efficient image retrieval. In: ICMR. (2015) 43–50

[116] Liu, Y., Guo, Y., Liu, L., Bakker, E.M., Lew, M.S.: Cyclematch: A cycle-consistent embedding network for image-text matching. (2018)

[117] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., DollÃąr, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. (2014) 740–755

[118] Liu, Y., Liu, L., Guo, Y., Lew, M.S.: Learning visual and textual representations for multimodal matching and classification. Pattern Recognition **84** (2018) 51–67

[119] Liu, Y., Guo, Y., Chen, W., Lew, M.S.: An extensive study of cycle-consistent generative networks for image-to-image translation. In: ICPR. (2018)

[120] Liu, Y., Chen, W., Liu, L., Lew, M.S.: Swapgan: A multi-stage generative approach for person-to-person fashion style transfer. (2018)

[121] Babenko, A., Lempitsky, V.S.: Aggregating local deep features for image retrieval. In: ICCV. (2015) 1269–1277

[122] Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR. (2014)

[123] Gregor, K., LeCun, Y.: Emergence of complex-like cells in a temporal product network with local receptive fields. CoRR **abs/1006.0448** (2010)

[124] Sun, Y., Wang, X., Tang, X.: Deeply learned face representations are sparse, selective, and robust. In: CVPR. (2015) 2892–2900

[125] Lee, C., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS. (2015)

[126] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI **33** (2011) 898–916

[127] Dollár, P., Zitnick, C.L.: Fast edge detection using structured forests. TPAMI **37** (2015) 1558–1570

[128] Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto. (2009)

[129] Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV **111** (2015) 98–136

[130] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding. In: ACM Multimedia. (2014) 675–678

[131] Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A.C., Bengio, Y.: Maxout networks. In: ICML. (2013) 1319–1327

[132] Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: ICLR. (2015)

[133] Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: CVPR. (2015) 3367–3375

[134] Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., Yan, S.: Deep learning with s-shaped rectified linear activation units. In: AAAI. (2016) 1737–1743

[135] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015) 448–456

[136] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178

[137] Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: CVPR. (2009) 413–420

[138] Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: ICVGIP. (2008) 722–729

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

[140] Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV. (2008) 304–317

[141] Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006) 2161–2168

[142] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2** (2011) 27:1–27:27

[143] Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV. (2011) 991–998

[144] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr,

P.H.S.: Conditional random fields as recurrent neural networks. In: ICCV. (2015) 1529–1537

[145] Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: NIPS. (2015) 2017–2025

[146] De Brabandere, B., Jia, X., Tuytelaars, T., Van Gool, L.: Dynamic filter networks. In: NIPS. (2016) 667–675

[147] Ferrari, V., Fevrier, L., Jurie, F., Schmid, C.: Groups of adjacent contour segments for object detection. TPAMI **30** (2008) 36–51

[148] Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV **59** (2004) 167–181

[149] Dollár, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: CVPR. (2006) 1964–1971

[150] Isola, P., Zoran, D., Krishnan, D., Adelson, E.H.: Crisp boundary detection using pointwise mutual information. In: ECCV. (2014) 799–814

[151] Hallman, S., Fowlkes, C.C.: Oriented edge forests for boundary detection. In: CVPR. (2015) 1732–1740

[152] Lim, J., Zitnick, C.L., Dollár, P.: Sketch tokens: A learned mid-level representation for contour and object detection. In: CVPR. (2013) 3158–3165

[153] Hwang, J., Liu, T.: Pixel-wise deep learning for contour detection. In: ICLR. (2015)

[154] Bertasius, G., Shi, J., Torresani, L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In: ICCV. (2015) 504–512

[155] Ganin, Y., Lempitsky, V.S.: N^4 -fields: Neural network nearest neighbor fields for image transforms. In: ACCV. (2014) 536–551

[156] Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: CVPR. (2014) 891–898

[157] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105

[158] Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. (2015) 5188–5196

[159] Cun, L., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: NIPS. (1990)

[160] Yan, Z., Zhang, H., Piramuthu, R., Jagadeesh, V., DeCoste, D., Di, W., Yu, Y.: Hd-cnn: Hierarchical deep convolutional neural network for large scale visual recognition. In: ICCV. (2015) 2740–2748

[161] Sironi, A., TÂÍÂźretken, E., Lepetit, V., Fua, P.: Multiscale centerline detection. TPAMI (2015)

[162] Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV. (2012) 746–760

[163] Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013) 1841–1848

[164] Zheng, L., Wang, S., Liu, Z., Tian, Q.: Packing and padding: Coupled multi-index for accurate image retrieval. In: CVPR. (2014) 1947–1954

[165] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)

[166] Sun, S., Zhou, W., Li, H., Tian, Q.: Search by detection: Object-level feature for image retrieval. In: ICIMCS. (2014) 46–49

[167] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178

[168] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: ICML. (2014) 647–655

[169] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR. (2015)

[170] Jégou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. IJCV **87** (2010) 316–336

[171] Babenko, A., Lempitsky, V.S.: The inverted multi-index. In: CVPR. (2012) 3069–3076

[172] Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: ECCV. (2014) 329–344

[173] Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR. (2012) 2911–2918

[174] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR. (2008)

[175] Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006) 2161–2168

[176] Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP. (2009) 331–340

[177] Yang, M., Wang, X., Lin, Y., Tian, Q.: Semantic-aware co-indexing for near-duplicate image retrieval. In: ICCV. (2014)

[178] Tolias, G., Avrithis, Y., Jégou, H.: To aggregate or not to aggregate: selective match kernels for image search. In: ICCV. (2013) 1401–1408

[179] Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: CVPR. (2009)

[180] Gong, Y., Wang, L., Hodosh, M., Hockenmaier, J., Lazebnik, S.: Improving image-sentence embeddings using large weakly annotated photo collections. In: ECCV. (2014) 529–545

[181] Karpathy, A., Joulin, A., Li, F.: Deep fragment embeddings for bidirectional image sentence mapping. In: NIPS. (2014) 1889–1897

[182] Mineiro, P., Karampatziakis, N.: A randomized algorithm for cca. In: NIPS workshop. (2014)

[183] Michaeli, T., Wang, W., Livescu, K.: Nonparametric canonical correlation analysis. In: ICML. (2016) 1967–1976

[184] Hardoon, D.R., Szedmak, S.R., Shawe-taylor, J.R.: Canonical correlation analysis: An overview with application to learning methods. Neural Computation **16** (2004) 2639–2664

[185] Lev, G., Sadeh, G., Klein, B., Wolf, L.: RNN fisher vectors for action recognition and image annotation. In: ECCV. (2016) 833–850

[186] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS. (2014) 3104–3112

[187] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9** (1997) 1735–1780

[188] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS. (2013) 3111–3119

[189] Young, P., Lai, A., Hodosh, M., Hockenmaier, J.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. TACL **2** (2014) 67–78

[190] Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z., Yuille, A.: Deep captioning with multimodal recurrent neural networks (m-rnn). ICLR (2015)

[191] Lin, X., Parikh, D.: Leveraging visual question answering for image-caption ranking. In: ECCV. (2016) 261–277

[192] Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I., Torralba, A.: Learning

cross-modal embeddings for cooking recipes and food images. In: CVPR. (2017) 3020–3028

[193] Li, S., Xiao, T., Li, H., Zhou, B., Yue, D., Wang, X.: Person search with natural language description. In: CVPR. (2017) 1970–1979

[194] Xu, H., Saenko, K.: Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In: ECCV. (2016) 451–466

[195] Reed, S., Akata, Z., Lee, H., Schiele, B.: Learning deep representations of fine-grained visual descriptions. In: CVPR. (2016) 49–58

[196] Bucher, M., Herbin, S., Jurie, F.: Improving semantic embedding consistency by metric learning for zero-shot classiffication. In: ECCV. (2016) 730–746

[197] Rohrbach, A., Rohrbach, M., Hu, R., Darrell, T., Schiele, B.: Grounding of textual phrases in images by reconstruction. In: ECCV. (2016) 817–834

[198] Zhang, Y., Yuan, L., Guo, Y., He, Z., Huang, I.A., Lee, H.: Discriminative bimodal networks for visual localization and detection with natural language queries. In: CVPR. (2017) 557–566

[199] He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T., Ma, W.Y.: Dual learning for machine translation. In: NIPS. (2016) 820–828

[200] Huang, Y., Wu, Q., Wang, L.: Learning semantic concepts and order for image and sentence matching. In: CVPR. (2018)

[201] Wang, F., Huang, Q., Guibas, L.: Image co-segmentation via consistent functional maps. In: ICCV. (2013) 849–856

[202] Zhou, T., Krähenbühl, P., Aubry, M., Huang, Q., Efros, A.A.: Learning dense correspondence via 3d-guided cycle consistency. In: CVPR. (2016) 117–126

[203] Godard, C., Mac Aodha, O., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR. (2017) 270–279

[204] Yi, Z., Zhang, H., Tan, P., Gong, M.: Dualgan: Unsupervised dual learning for image-to-image translation. In: ICCV. (2017) 2849–2857

[205] Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: ICML. (2017) 1857–1865

[206] Chen, X., Zitnick, C.L.: Mind's eye: A recurrent visual representation for image caption generation. In: CVPR. (2015) 2422–2431

[207] van der Maaten, L., Hinton, G.: Visualizing high-dimensional data using t-sne. JMLR **9** (2008) 2579–2605

[208] Nandakumar, K., Chen, Y., Dass, S.C., Jain, A.: Likelihood ratio-based biometric score fusion. IEEE TPAMI **30** (2008) 342–347

[209] Zheng, L., Wang, S., Tian, L., He, F., Liu, Z., Tian, Q.: Query-adaptive late fusion for image search and person re-identification. In: CVPR. (2015) 1741–1750

[210] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: EMNLP. (2017) 670–680

[211] Faghri, F., Fleet, D.J., Kiros, R., Fidler, S.: VSE++: improved visual-semantic embeddings. CoRR **abs/1707.05612** (2017)

[212] Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: NIPS. (2015) 3294–3302

[213] Vendrov, I., Kiros, R., Fidler, S., Urtasun, R.: Order-embeddings of images and language. In: ICLR. (2016)

[214] Frome, A., Corrado, G.S., Shlens, J., Bengio, S., Dean, J., Ranzato, M.A., Mikolov, T.: Devise: A deep visual-semantic embedding model. In: NIPS. (2013) 2121–2129

[215] Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural Computation **12** (2000) 1247–1283

[216] Gao, Y., Beijbom, O., Zhang, N., Darrell, T.: Compact bilinear pooling. In: CVPR. (2016) 317–326

[217] Fukui, A., Park, D.H., Yang, D., Rohrbach, A., Darrell, T., Rohrbach, M.: Multimodal compact bilinear pooling for visual question answering and visual grounding. In: EMNLP. (2016) 457–468

[218] Pham, N., Pagh, R.: Fast and scalable polynomial kernels via explicit feature maps. In: SIGKDD. (2013) 239–247

[219] Rashtchian, C., Young, P., Hodosh, M., Hockenmaier, J.: Collecting image annotations using amazon's mechanical turk. In: Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk. (2010) 139–147

[220] Socher, R., Karpathy, A., Le, Q., Manning, C., Ng, A.: Grounded compositional semantics for finding and describing images with sentences. TACL **2** (2014) 207–218

[221] Simon, M., Rodner, E.: Neural activation constellations: Unsupervised part model discovery with convolutional networks. In: ICCV. (2015) 1143–1151

[222] Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: ICCV. (2015) 1449–1457

[223] Zhang, X., Xiong, H., Zhou, W., Lin, W., Tian, Q.: Picking deep filter responses for fine-grained image recognition. In: CVPR. (2016) 1134–1142

[224] Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: Factors of transferability for a generic convnet representation. TPAMI **38** (2016) 1790–1802

[225] Zhang, N., Donahue, J., Girshick, R., Darrell, T.: Part-based rcnn for fine-grained detection. In: ECCV. (2014) 834–849

[226] Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z.: The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In: CVPR. (2015) 842–850

[227] Lin, D., Shen, X., Lu, C., Jia, J.: Deep lac: Deep localization, alignment and classification for fine-grained recognition. In: CVPR. (2015) 1666–1674

[228] Qian, Q., Jin, R., Zhu, S., Lin, Y.: Fine-grained visual categorization via multi-stage metric learning. In: CVPR. (2015) 3716–3724

[229] Guo, Y., Liu, Y., Lao, S., Bakker, E.M., Bai, L., Lew, M.S.: Bag of surrogate parts feature for visual recognition. IEEE Trans. on Multimedia (2017)

[230] Xie, L., Wang, J., Lin, W., Zhang, B., Tian, Q.: Towards reversal-invariant image representation. IJCV **123** (2017) 226–250

[231] Cai, S., Zhang, L., Zuo, W., Feng, X.: A probabilistic collaborative representation based approach for pattern classification. In: CVPR. (2016) 2950–2959

[232] Wang, D., Shen, Z., Shao, J., Zhang, W., Xue, X., Zhang, Z.: Multiple granularity descriptors for fine-grained categorization. In: ICCV. (2015) 2399–2406

[233] Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: ICCV. (2017) 2813–2821

[234] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR. (2015)

[235] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015)

[236] Tyleček, R., Šára, R.: Spatial pattern templates for recognition of objects with regular structure. In: German Conference on Pattern Recognition. (2013) 364–374

[237] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U.,

Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR. (2016) 3213–3223

[238] Yu, Q., Liu, F., SonG, Y.Z., Xiang, T., Hospedales, T., Loy, C.C.: Sketch me that shoe. In: CVPR. (2016) 799–807

[239] Liang, X., Lin, L., Yang, W., Luo, P., Huang, J., Yan, S.: Clothes co-parsing via joint image segmentation and labeling with application to clothing retrieval. IEEE Transactions on Multimedia **18** (2016) 1175–1186

[240] Zhang, X., Jia, J., Gao, K., Zhang, Y., Zhang, D., Li, J., Tian, Q.: Trip outfits advisor: Location-oriented clothing recommendation. IEEE Transactions on Multimedia **19** (2017) 2533–2544

[241] Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: CVPR. (2016) 1096–1104

[242] Zhang, L., Liu, M., Chen, L., Hu, Y., Zhang, L., Zimmermann, R.: Online modeling of aesthetic communities using deep perception graph analytics. IEEE Transactions on Multimedia (2017)

[243] Garg, V., Banerjee, R.H., Rajagopal, A.K., Thiruvambalam, S., Warrier, D.: Sales potential: Modeling sellability of fashion product. KDD (2017)

[244] Zheng, Z.H., Zhang, H.T., Zhang, F.L., Mu, T.J.: Image-based clothes changing system. Computational Visual Media **3** (2017) 337–347

[245] Yoo, D., Kim, N., Park, S., Paek, A.S., Kweon, I.S.: Pixel-level domain transfer. In: ECCV. (2016) 517–532

[246] Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., Van Gool, L.: Pose guided person image generation. In: NIPS. (2017) 405–415

[247] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: CVPR. (2017) 1302–1310

[248] Gong, K., Liang, X., Zhang, D., Shen, X., Lin, L.: Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In: CVPR. (2017) 6757–6765

[249] Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016)

[250] Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: NIPS. (2016) 2226–2234

[251] Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. **22** (1951) 79–86

[252] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Trans. on Image Processing **13** (2004) 600–612

[253] Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: CVPR. (2017) 3319–3327

[254] Zhang, Q., Wu, Y.N., Zhu, S.C.: Interpretable convolutional neural networks. In: CVPR. (2018)

[255] Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for image classification. TPAMI **38** (2016) 1425–1438

[256] Lei Ba, J., Swersky, K., Fidler, S., salakhutdinov, R.: Predicting deep zero-shot convolutional neural networks using textual descriptions. In: ICCV. (2015) 4247–4255

[257] Zheng, Z., Zheng, L., Yang, Y.: Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In: ICCV. (2017) 3774–3782

[258] Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: CVPR. (2018)

List of Abbreviations

Abb.	Full Name
AP	Average Precision
BN	Batch Normalization
BoW	Bag of Words
CBIR	Content-based Image Retrieval
CBP	Compact Bilinear Pooling
CCA	Canonical Correlation Analysis
CFN	Convolutional Fusion Networks
cGAN	Conditional Generative Adversarial Networks
CNN	Convolutional Neural Networks
DSN	Deeply Supervised Networks
FC	Fully-connected Layer
FCFN	Fully Convolutional Fusion Networks
FCN	Fully Convolutional Networks
FFT	Fast Fourier Transformation
FV	Fisher Vector
GAN	Generative Adversarial Networks
GAP	Global Average Pooling
HGLMM	Hybrid Gaussian-Laplacian Mixture Model
I2I	Image-to-Image Translation
I2T	Image-to-Text Translation
IoU	Intersection over Union
IS	Inception Score

Abb.	Full Name
LC	Locally-connected Layer
LSGAN	Least Square Generative Adversarial Networks
LSTM	Long Short-Term Memory
MA	Multiple Assignment
mAP	Mean Average Precision
MC-Net	Multi-modal Classification Network
MDS	Multi-Dimensional Scaling
MM-Net	Multi-modal Matching Network
MMC-Net	Multi-modal Matching and Classification Network
NMS	Non-maximal Suppression
ODS	Fixed Contour Threshold
OIS	Per-image Best Threshold
PCA	Principal Component Analysis
RDS	Relaxed Deep Supervision
ReLU	Rectified Linear Units
RNN	Recurrent Neural Networks
RRF	Recurrent Residual Fusion
SGD	Stochastic Gradient Descent
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine
T2I	Text-to-Image Translation
TF	Term Frequency
TPS	Thin Plate Spline
t-SNE	t-Distributed Stochastic Neighbor Embedding
VLAD	Vector of Aggregate Locally Descriptor

English Summary

A core mission of computer vision research is endowing machines with the ability to understand visual data. Driven by it, in this thesis we present research on exploring and analyzing images from three themes: classification, retrieval and synthesis.

Our first theme focuses on image-level and pixel-level classification. Firstly, we propose an efficient convolutional fusion network that can learn adaptive weights by fusing different intermediate layers with adding only a few parameters. In addition, our proposed neural network can be extended for pixel-level classification such as semantic segmentation and edge detection. Our work suggests the superiority of deep fusion networks over plain convolutional neural networks. Secondly, we further study the pixel-level classification task for edge detection. In contrast to prior works that use a fixed supervision for all intermediate layers, we develop diverse supervision that can adapt to the diversities of different layers. Our method can incorporate the diversities into the supervisory signals.

The second theme of this thesis includes image retrieval and cross-modal retrieval. We build a DeepIndex framework by incorporating deep visual features into the inverted index scheme. Subsequently, we can leverage a multiple DeepIndex framework to integrate different deep features at an indexing level. Furthermore, we develop a deep matching network to unify visual and textual features for cross-modal retrieval. The building block in our network integrates the recurrent mechanism, the residual learning and a fusion module. This integration can help promote feature embeddings while retaining the shared parameters. We propose cycle-consistent embeddings which can preserve both inter-modal correlations and intra-modal consistency while matching visual and textual representations. For a robust inference, we leverage two late-fusion approaches to integrate the matching scores of different embedding features. Lastly, in contrast to either multi-modal matching or multi-modal classification, we exploit a unified network for joint matching and classification. The matching component can bridge the modality gap between vision and language, and simultaneously the classification component is used to combine visual and textual embedding features to be a multi-modal representation.

Our third theme studies two applications about image synthesis. Firstly, we extend cycle-consistent generative adversarial networks for image-to-image translation. Our extended models make use of more generators and inner cycles to enhance the

constrains while performing the unsupervised translation between different image domains. Secondly, we propose a novel generative framework for addressing the problem of person-to-person fashion style transfer. It includes three generative networks that are cascaded in a multi-stage paradigm. Our framework can be trained end-to-end to swap the clothes of person images while preserving their pose and body shape.

We have conducted numerous experiments to verify the effectiveness of our proposed methods for the three research themes. Our results demonstrate promising improvements over various baseline methods, and are comparable with the state-of-the-art results from the research community. By performing a wide range of tasks and applications in the field, this thesis provides novel contributions, insights and findings.

Nederlandse Samenvatting

Een van de belangrijkste doelen van het onderzoek op het gebied van Computer Vision is om machines toe te rusten met het vermogen om visuele data te begrijpen. Gedreven door dit doel presenteren we in dit proefschrift onderzoek op het gebied van de exploratie en analyse van beelden binnen de drie thema's: classificatie, retrieval en synthese.

Ons eerste thema concentreert zich op classificatie op beeld- en pixel-niveau. Ten eerste introduceren we een efficiënt convolutioneel fusie netwerk dat in staat is om adaptieve gewichten aan te leren door middel van de fusie van verschillende tussenliggende lagen van het netwerk waarbij slechts een klein aantal nieuwe parameters benodigd zijn. Daarenboven kan ons neurale netwerk eenvoudig uitgebreid worden om classificaties op pixel-niveau, zoals semantische segmentatie en edge-detectie, uit te voeren. Ons werk is een sterke aanwijzing dat diepe fusie netwerken superieur zijn aan gewone convolutionele netwerken. Ten tweede bestuderen we de classificatie-taak op pixel-niveau voor edge-detectie. In tegenstelling tot eerdere studies welke een supervisie gebruikten die constant blijft voor alle tussenliggende lagen, ontwikkelen wij een diverse supervisie welke zich kan aanpassen aan de diversiteit van de verschillende lagen van het netwerk. Onze methode is daarbij in staat om de diversiteiten in het supervisie-signaal in te lijven.

Het tweede thema van dit proefschrift omvat image retrieval en cross-modal retrieval. We construeren een DeepIndex-framework door het opnemen van deep visual features in het geïnverteerde index-schema. We kunnen dan een meervoudig DeepIndex-framework gebruiken om verschillende deep features op het index niveau te integreren. Verder ontwikkelen we een deep matching netwerk voor de unificatie van visuele en tekstuele features ten behoeve van cross-modal retrieval. De bouwsteen van ons netwerk integreert het recurrente mechanisme, het residuale leren en een fusie-module. Deze integratie kan helpen bij de ontwikkeling van feature embeddings terwijl gedeelde parameters behouden blijven. We introduceren cycle-consistent embeddings welke bij het matchen van visuele en tekstuele representaties zowel intermodale correlaties als intra-modale consistentie kan behouden. Voor robuuste afleidingen gebruiken we twee late-fusion benaderingen om matching scores van verschillende embedding features te integreren. Ten slotte exploiteren we een unified netwerk voor zowel matching als classificatie, dit in tegenstelling tot netwerken voor

ofwel multimodale matching of multimodale classificatie. De matching component is in staat om het modaliteits-gat tussen beeld en taal te overbruggen terwijl tegelijkertijd de classificatie component visuele en tekstuele embedding features combineert tot een multimodale representatie.

In ons derde thema bestuderen we twee beeldsynthese applicaties. Ten eerste breiden we cycle-consistente generatieve adversarial netwerken uit voor beeld-naar-beeld vertalingen. Onze uitgebreide modellen maken gebruik van meer generatoren en inwendige cycli om de constraints te versterken terwijl de unsupervised vertaling tussen verschillende beelddomeinen wordt uitgevoerd. Ten tweede introduceren we een nieuw generatief kader om het probleem van persoon-persoon fashion style transfer aan te pakken. Het omvat drie generatieve netwerken welke in een cascade een multi-stage paradigma vormen. Ons kader kan end-to-end getraind worden om de kleren van een persoon te verwisselen met behoud van pose en lichaamsvorm.

We hebben talloze experimenten uitgevoerd om de effectiviteit van onze voorgestelde methoden binnen de drie onderzoeksthema's te verifiëren. Onze resultaten laten veelbelovende verbeteringen zien in vergelijking met verschillende baseline methoden en zijn vergelijkbaar met de beste resultaten uit de research community. Door het uitvoeren van een wijd scala aan taken en applicaties uit het onderzoeksveld geeft dit proefschrift nieuwe contributies, inzichten en bevindingen.

Curriculum Vitae

Yu Liu was born in Mishan, Heilongjiang, China on September 4, 1988. In 2007, he started his study at the Dalian University of Technology in Dalian, Liaoning, China, and received the B.S. degree in 2011, and the M.S degree in 2014 under the supervision of Prof. He Guo.

In September 2014, he started his PhD supported by the China Scholarship Council and worked at the Media Lab in Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands, under the supervision of Prof. Dr. J.N. Kok and Dr. M.S. Lew. His research interests include computer vision, multimedia retrieval and deep learning. Specifically, he is focusing on image recognition (*e.g.* image classification, edge detection), vision & language (*e.g.* image captioning and cross-modal retrieval) and image synthesis (*e.g.* image-to-image translation and fashion style transfer). He has published papers in international conferences and journals, including CVPR, ICCV, ICMR, ICIP, ICPR, Pattern Recognition and MTAP. In addition, he obtained the best paper award at the 23rd International Conference on MultiMedia Modeling in 2017. Currently, he is a reviewer for some conferences and journals, such as CVPR, ACCV, IJCV, Neurocomputing, IEEE Access and IJMIR.