CLOTHING IMAGE RETRIEVAL WITH TRIPLET CAPSULE NETWORKS

A Thesis

by

O. Furkan Kınlı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
August 2019

CLOTHING IMAGE RETRIEVAL WITH TRIPLET CAPSULE NETWORKS

Approved by:

Asst. Dr. M. Furkan Kıraç, Advisor
Department of Computer Science
Özyeğin University

Prof. Dr. Lale Akarun
Department of Computer Engineering
Boğaziçi University

Asst. Dr. Reyhan Aydoğan
Department of Computer Science
Özyeğin University

Date Approved: 19 August 2019

To my family, my prospective wife Buse and my best friend Alex.

ABSTRACT

Clothing image retrieval has become more important after some major developments in Computer Science and the emergence of e-commerce. Recent studies generally attack this problem by using Convolutional Neural Networks (CNNs). Despite their popularity, CNNs, by nature, have some intrinsic limitations such as losing the hierarchical spatial relationship between the parts of an image, and not being robust to affine transformations. Most recently proposed network architecture, namely Capsule Networks, has the ability to overcome these limitations by preserving the part-whole relationship and pose information in the images. In this thesis, we investigate in-shop clothing retrieval performance of densely-connected Capsule Networks with dynamic routing. To achieve this, we propose Triplet-based designs of Capsule Network architecture with two different feature extraction methods: Stacked-convolutional (SCCapsNet) and Residual-connected (RCCapsNet) Capsule Networks. Experimental results of our proposed designs on in-shop clothing retrieval show that SCCapsNet achieves 32.1% Top-1, 81.8% Top-20, and 90.0% Top-50 recall-at-K scores; whereas RCCapsNet has even better performance with 33.9% Top-1, 84.6% Top-20, and 92.6% Top-50 recall-at-K scores. These figures demonstrate that both of our designs outperform the baseline study and the earlier approaches by a wide margin without using any extra supportive information besides to the images. Moreover, when compared to the SOTA architectures on clothing retrieval, our proposed Triplet Capsule Networks achieve comparable recall rates with only half of the parameters used in the SOTA architectures. In the future, our designs may inherit extra performance boost due to advances in the relatively new Capsule Network research.

ÖZETÇE

Kıyafet resmi erişimi Bilgisayar Bilimleri'ndeki bazı önemli gelişmelerden ve e-ticaret'in doğuşundan sonra daha da önemli hale gelmiştir. Yakın dönemdeki çalışmalarda bu problemin çözümü için genel olarak Evrişimli Sinir Ağları (ESA) kullanılmıştır. Popülaritesine rağmen, ESAlar, doğaları gereği, parçalar arası hiyerarşik konumsal bilgi kaybı ve afin dönüşümlerine dayanıklı olmama gibi bazı yerleşik sınırlamalara sahiptir. Yeni önerilen Kapsül Ağları mimarisi, resimlerdeki parça-bütün ilişkisini ve poz bilgisini koruyarak bu sınırlamaları ortadan kaldırabilme özelliğine sahiptir. Bu tezde, dinamik yönlendirme algoritması ile çalışan, nöronları yoğun bağlı Kapsül Ağları'nın vitrin kıyafet resimlerine erişim performansını araştırdık. Buradan yola çıkarak, iki farklı öznitelik çıkartma metoduyla tasarlanmış Triplet-bazlı Kapsül Ağ mimarileri önerdik: İstifli-evrişimsel (SCCapsNet) ve Artık-bağlı (RCCapsNet) Kapsül Ağları. Vitrin kıyafet resmi erişimine yönelik önerilen mimari tasarımlarımızın deneysel sonuçları, SCCapsNet'in %32.1 en-yüksek-1, %81.8 en-yüksek-20 ve %90.0 en-yüksek-50 recall-at-K skorlarına ulaştığını gösterirken; RCCapsNet ise %33.9 en-yüksek-1, %84.6 en yüksek-20 ve %92.6 en-yüksek-50 recall-at-K skorlarıyla daha da iyi bir performans ortaya koymuştur. Bu rakamlar referans çalışmasının ve daha öncül yaklaşımların performanslarıyla karşılaştırıldığında, resimlere ek olarak hiçbir ilave destekleyici bilgi kullanmayan her iki tasarımımız da önemli bir farkla daha önde bir performans sergilemiştir. Ayrıca, önerdiğimiz Triplet Kapsül Ağları, modern mimarilerde kullanılan parametre sayısının sadece yarısı kadar parametre kullanarak, modern mimariler ile kıyaslanabilir sonuçlar elde etmiştir. İlerleyen dönemde, tasarımlarımız, nispeten yeni Kapsül Ağları araştırmalarındaki gelişmelerden yola çıkarak ekstra performans artışı alabilir.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Asst. Dr. M. Furkan Kıraç for the continuous support of my M.Sc. study and research, for his motivation, enthusiasm, patience and immense knowledge. His guidance helped me throughout my research., and I could not have imagined having a better advisor and mentor for my M.Sc. study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Dr. Lale Akarun and Asst. Dr. Reyhan Aydoğan, for their encouragement, insightful comments, and hard questions.

I would like to thank my fellows in Video, Vision and Graphics Lab (VVGL) at Özyeğin University: Barış Özcan, Mahir Atmış, Yakup Görür and Ahmet Tavlı; and in Artificial Intelligence Lab (AILab) at Özyeğin University: Emir Arditi, Ammar Raşid, Umut Çakan and Cihan Eran. Special thanks in here goes to Barış for his invaluable advice and feedback on my research and for always being so supportive of my work. I am also very grateful to Emir for helping me in numerous ways during various stages of my M.Sc. study and research.

I would also like to say a heartfelt thank you to my mother Fatma Hanım, my father Muammer, and my brother Efecan for always believing in me and encouraging me to follow my dreams. The only reason for making all these happen is to make sure that they are proud of me. I believe to achieve sort of it.

And finally to Buse, who has been by my side throughout this M.Sc. study, living every single minute of it, and without her support, I would not have had the courage to embark on this journey at first glance. And the final boss, Alex, thank you for being such a lovely jumbo-sized friend of me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

Online shopping is a highly growing market. In-store sales were the leading arms for the profit margin of companies in the previous years. However, with the impact of digital transformation, e-commerce recently becomes a new trend, and in-store sales start to lose the leading share of market volume to online sales. Online retail sales worldwide are expected to reach 3.5 trillion dollars by the end of 2019 [7], and clothing sales provide a significant portion of this volume. According to a research conducted by Statista in 2018 [8], 57% of global internet users had purchased fashion-related products through the Internet. In addition, the global fashion e-commerce market has a volume of approximately 480 billion dollars, and it is expected to almost double in the next 4 years [9].

In such an emerging market, one of the most prominent priorities of a company that wants to take part in the competition is to improve the shopping experience of its customers. At this point, one of the most significant problems that customers may encounter when shopping online is that they cannot find the desired product in stock by searching through the company's search engine. Traditional fashion search engines allow to search by only the words that describe the products. In principle, this is simply a similarity ranking mechanism that matches the textual input from the customer to the meta-data of the product such as title, label and category information. However, to accurately rank the products with this mechanism, it is required to have fully-defined meta-data information and an unbiased vocabulary. Therefore, online retailers have to provide their products by combining with richly annotated meta-data

and diverse catalog information, so that their customers can find the product they are looking for. Although having complete meta-data and catalog information leads to increase the conversion rate, it can be exhaustive to collect for the companies.

Next, competitors in the fashion market attempts to add a value to the shopping experience of their customers by adapting the developing technologies to the sales process. Incorporating visual information of the products through the search engines enhances the shopping experience of the customers in a sophisticated way. In other words, the fact that search engines have the ability to measure the similarity between query images taken by customers and in-shop product images helps the customers to find what they are exactly looking for. As a consequence, this leads to increase the conversion rates for the companies. Due to the limitations of the existing machine learning approaches, it was difficult to acquire this feature in search engines up to a few years ago. In the last couple of years, it becomes easier to solve with the help of novel techniques combining Computer Vision and Deep Neural Networks.

In regard to the previous studies, the problem of learning the similarity between images is attacked with several different techniques in different domains [10, 11, 12, 13, 14]. In recent years, as in the case of almost all image-related tasks [15, 16, 17, 18], convolution-based neural network architectures achieved most of the SOTA performances in image retrieval tasks [19, 20, 21]. However, convolutional neural networks (CNNs) have some limitations by their nature, and most recently proposed neural network architecture [4], called Capsule Networks, claims to overcome these limitations. To the best of our knowledge, nobody attacks to in-domain image retrieval task by using Capsule Networks up to now. In this thesis, we mainly investigate the performance of Capsule Networks on clothing image retrieval task, and propose Triplet-based [13] design of Capsule Networks which learns how similar the images in fashion domain are.

1.2 Definition

The main objective of this thesis is to observe the similarity learning performance of Triplet-based Capsule Network architecture on more realistic, diverse and rich data sets. Our study has five goals following the thesis hypotheses.

1.2.1 Goals

The goals of this thesis can be expressed as follows:

- Studying the brand-new deep learning architecture called Capsule Networks to understand how it internally functions
- Designing a novel Triplet-based Capsule Network architecture by adjusting the original architecture for similarity learning task
- Seeking more powerful feature extraction recipe than a 1-layer convolutional network for the input of Capsule Networks
- Performing experiments by using our proposed Capsule Network architecture on in-shop partition of DeepFashion data set [1]
- Combining all above, and investigating the SOTA research on clothing image retrieval and Capsule Networks

1.2.2 Hypotheses

- Capsule Networks can be designed as Triplet-based to solve similarity learning of the clothing images.
- Capsule Networks can achieve even better performance than CNN-based architectures without using any side information for in-shop image retrieval task.
- Capsule Networks can get comparable results to the SOTA methods that utilize different side information and extra modules during their training phase.

1.2.3 Summary

In this thesis, it is mainly aimed to design a novel Triplet-based Capsule Network architecture, and to investigate its performance on in-shop clothing image retrieval task. To achieve this, our design should be capable of measuring the similarity between two image embeddings produced by a network that shares its weights for each embedding. Moreover, since a shallow CNN falls short of extracting features from more realistic images like clothing images, more complex methods are employed for feature extraction phase such as stacked-convolutional blocks and residual blocks. In this design, our model is not supported by any side information (i.e. landmarks and attributes) that helps to fill the deficiency of pose configuration in CNN-based architectures since Capsule Networks are capable of learning the pose configuration of the objects by itself.

1.3 Contributions

This thesis introduces Triplet-based design of Capsule Networks for measuring the similarity between the latent capsule vectors. Our proposed architecture is employed for in-shop image retrieval task. At this point, while this study can be considered as a benchmark for clothing image retrieval, also it is an experimental study of Capsule Networks on a more realistic, diverse and rich data set.

This study mainly shows that Capsule Networks are capable of getting promising results on in-shop image retrieval, even without receiving any extra support as in the case of the SOTA CNN-based methods to mitigate the limitations of CNNs. In addition, in this study, we extract the features in real-like clothing images by more powerful structures in order to figure out better training strategy for Capsule Networks.

To the best of our knowledge, this thesis is the first research that employs Capsule Networks for a similarity learning task in fashion domain.

1.4 Thesis Outline

The outline of this thesis can be shown as follows:

In Chapter 2, we will recall the previous studies related to our research on Similarity Learning (2.2), Clothing Image Retrieval (2.3) and Capsule Networks (2.4).

Chapter 3 investigates technical background behind well-known CNN architectures (3.2), and gives a brief explanation of the capsule idea and routing mechanism (3.3). Moreover, Triplet-based objective functions and the performance metrics that we employ for this study (3.4) are presented in this chapter.

Chapter 4 introduces the baseline study (4.2), reveals our proposed Triplet-based Capsule Network architectures in details (4.3), shows up in-shop partition of DeepFashion data set (4.4), and explains the implementation details of our work (4.5).

In Chapter 5, we will show the experimental results of our proposed architectures on in-shop image retrieval task (5.2). At the end of this chapter, we will compare the performances of our proposed architecture, the baseline study utilizing landmark information and attributes to recover the pose configuration, and the existing SOTA methods using in-shop partition of DeepFashion data set in their studies (5.3).

Chapter 6 gives a brief summary of overall approach and the contributions, and concludes this thesis with revealing the possible improvements on this research.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

In this chapter, we concisely review the previous studies on similarity learning and clothing image retrieval tasks, and indicate the similar approaches to our study and the distinctive techniques in the literature. In addition, we demonstrate some recent works on the brand-new deep learning architecture, namely Capsule Networks, in a wide variety of domains.

2.2 Similarity Learning

The task of finding how similar two images are is one of the challenging tasks in Computer Vision. Most of the SOTA methods have attacked to similarity learning by measuring the similarity of the images with various distance metrics, and trying to optimize the objective functions with several different strategies. In recent studies, Nguyen *et al.* [10] (2010) shows that cosine similarity can be considered as an effective alternative of Euclidean distance for metric learning problems. With the help of the regularization term on the objective function, measuring the distance between the manifold vectors with cosine similarity significantly improves the generalization ability of the model. As a result, this study achieves the SOTA performance on face verification domain.

Wang *et al.* [11] (2014) demonstrates that deep learning architectures are capable of learning to rank fine-grained image similarities directly from images, and perform better than the architectures based on hand-crafted feature descriptors. The proposed multi-scale network model is designed for finding out the relationship between triplets

that contain an anchor image, a similar image (positive), and a dissimilar image (negative) by ranking the relative distance between the anchor and the others in Euclidean space.

In analogy to above cited study, Zagoruyko *et al.* [12] (2015) investigates several CNN-based architectures to learn a general similarity function from directly annotated pairs of raw images without utilizing any manually-designed features. In this study, as a result of the comparison among different combinations of dual-channel-based [12], Siamese-based [22] and 2-stream-based [12] architectures, it is indicated that extracting the features jointly from pairs has notable impact on matching the similar images.

Schroff *et al.* [13] (2015) presents a system, called FaceNet, that directly learns to map face images to embeddings as feature vectors in Euclidean space. The distance between feature vectors directly corresponds to how dissimilar these images are. This approach achieves the SOTA performance in face recognition and face verification domains by encoding the images with only 128-bytes in Euclidean space.

Gordo *et al.* [14] (2016) focuses on learning deep representations of the images for retrieval tasks by aggregating different region of interests in the images into compact feature vectors that are robust to scaling and translation. This is an effective method of similarity learning by packing Siamese Network [22] and Regional Proposal Network [23] architectures together to generate the encodings. In the view of such combination of different deep learning architectures, it is possible to compare the compact global representations by projecting them onto a different sub-space. To extend this study, Gordo *et al.* [24] (2017) discusses possible reasons for getting underwhelming results on image retrieval tasks when using deeper architectures. At this point, it is claimed that cleaning noisy samples from training data, intentive selection of network architecture and optimal strategies during training lead to achieve better results on image retrieval tasks.

2.3 Clothing Image Retrieval

To survey remarkable image retrieval studies in fashion domain, Kiapour *et al.* [25] (2015) introduces an excessively challenging task, namely *Exact Street to Shop* where the goal is to match street photos that are captured in uncontrolled settings to the same item in online shop photos that are captured by professionals. In this study, it is experimentally shown that learning the similarity between street and online shop photos directly is the best way to solve cross-domain image matching task when compared to the existing methods that learn the embedded representations by traditional deep learning approaches.

Huang *et al.* [26] (2015) seeks another solution for the cross-domain image matching task as in the case of aforementioned study. In this study, Dual Attribute-aware Network (DARN) is proposed to address this problem. This architecture consists of two sub-networks that are structurally similar, yet extract the feature representations for street photos and online shop photos separately. The main objective of this approach is to create a powerful and domain-specific semantic representations of clothes with the help of the fine-grained clothing attributes.

Liu *et al.* [1] (2016) introduces a new data set, namely DeepFashion, which has a vast amount of large-scale clothing images annotated with numerous attributes, landmark information and cross-domain image correspondences. Moreover, in this comprehensive study, FashionNet [1] is proposed for clothing category and attribute classification, in-shop and consumer-to-shop retrieval tasks. In this design, first, FashionNet simultaneously predicts landmarks and attributes. Thereafter, it employs the estimated landmark locations to pool the learned features so that the discriminative features can be brought to the forefront.

Corbière *et al.* [27] (2017) demonstrates that it is possible to achieve promising results on image retrieval tasks on fashion domain by integrating bag-of-words approach to weakly supervised learning process. In this method, the proposed model encodes

weakly-annotated noisy data using the bag-of-words descriptors to generate separable visual concepts, and also to provide meaningful similarity between images. Along with that, the learned representations are suitable for both classification and retrieval tasks, and this study essentially addresses the issue of finding a large, rich-annotated and clean-labeled data set for training of deep neural networks.

Wang *et al.* [28] (2017) proposes a Visual Attention Model (VAM) and a novel Dropout-like connection between attention layers and the main network. To reduce the dependency of having massive amount of side information on getting more accurate results, VAM is employed to extract the attention maps from clothing images. These attention maps are gated to intermediate feature maps extracted by main network by performing randomized Dropout-like operation on the feature maps. The main goal of utilizing attention-based design is to simplify focusing on important regions in the images and to diminish the effect of the background clutter.

Yuan *et al.* [29] (2017) addresses the issue of defining a model with right complexity and choosing hard samples carefully during training. To alleviate this, a novel framework, namely Hard-aware Deeply Cascaded Embeddings (HDC), is proposed in this study. This framework ensembles a set of models with different complexities in cascaded manner, and figures out hard samples at multiple complexity levels for training phase. After the forward pass, the main model is back-propagated only if the sample is considered as a hard case.

Opitz *et al.* [20] (2018) shows how to improve the robustness of the feature embeddings by exploiting the independence within ensembles. To achieve this, a new ensemble learning approach for metric learning problem, called Boosting Independent Ensembles Robustly (BIER), is presented in this study. In this approach, the last layer of the embeddings is divided into multiple non-overlapping groups, and each group is considered as a separate metric learning network that shares the features in the previous layers of the main network.

Ge *et al.* [19] (2018) introduces hierarchical triplet loss (HTL) to address the random sampling issue during training a triplet loss. The main idea behind this study is to learn the hierarchical class structure of the images in order to encode global context in the images. The class-level hierarchical tree controlled by a violate margin that dynamically changes with the structure of the hierarchical tree guides the sampling operation for training data by comparing the data distribution on a manifold sphere.

Kim *et al.* [21] (2018) proposes multiple-way attention-based ensemble architecture that learns the feature embeddings with multiple attention masks. In this design, each learner contributes to the generation of the feature embeddings by attending to different parts of the objects. This framework encourages diversity in the feature embeddings with the help of divergence loss that differentiates this embeddings from different learners.

2.4 Capsule Networks

Capsules are defined as local entities of a neural network which encapsulate the results of some complicated internal computations between neurons into more informative vector-formed output. The preliminary idea related with capsules is suggested by Hinton *et al.* [30] (2011). The key part of the capsule idea is that Convolutional Neural Networks (CNNs) output single scalar neuron activation to deliver the viewpoint invariance after several sub-sampling stages, but high-level features extracted by CNNs are incapable of solving certain tasks that require to reason the spatial relationship between high-level parts of an instance. To address this, Hinton *et al.* [30] indicates that neural networks containing capsules, instead of neurons, can learn feature representations as a whole feature vector of instantiation parameters of an instance. Therefore, the capsule idea introduces an unprecedented approach to overcome the variations in pose information of the instance.

Sabour *et al.* (2017) [4] proposes a novel Capsule Network architecture that uses dynamic routing mechanism to route the vector-formed capsule output to an appropriate capsule (parent capsule) in the next layer. In principle, the capsule computes the prediction vector for each possible parent by multiplying its output with a weight matrix, and this gives a feedback about the coupling coefficient of the parent capsules. The coupling coefficients can be considered as the votes of a child capsule for all parent capsules, and the parent capsule with the largest coefficient value is activated. In this study, it is experimentally shown that Capsule Networks have considerably better performance than CNNs on tasks of recognizing hand-written digits (MNIST [2]) and segmenting highly overlapping digits (MultiMNIST [4]).

Zhao *et al.* (2018) [31] investigates the performance of Capsule Networks with dynamic routing for text classification on several well-known benchmarks by using three different implementation strategies. To alleviate some disturbance factors of noisy capsule outputs, first, an additional category that represents the stop words or unrelated words is added to the network. Moreover, the softmax operation while assigning the coupling coefficients between child capsule and possible parent capsules is transformed into leaky form. Lastly, the coupling coefficients are weighted by the probability of the existence of child capsule. These manipulations on dynamic routing leads to notable boost on the performance of Capsule Networks for text classification.

LaLonde *et al.* (2018) [32] demonstrates that Capsule Networks can be effectively exploited in object segmentation task by altering the architecture and dynamic routing algorithm. This study extends the capsule idea with locally-connected routing mechanism and the concept of deconvolutional (transposed-convolutional) structure to be able to operate on large-sized images with the less number of parameters than the original architecture. The performance of this modified version of Capsule Networks is observed on an application of pathological lung segmentation from computed tomography (CT) images.

Hinton *et al.* (2018) [33] introduces matrix capsules which have activation units that refer to the presence of an entity; and 4x4 matrix, namely pose matrix, to learn the relationship between the entity and pose information. At this point, instead of dynamic routing algorithm, Expectation-Maximization (EM) [34] is applied to the coefficients in an iterative manner to route the output of each capsule to parent capsule in the next layer. Matrix capsules with EM routing significantly outperforms the SOTA on the smallNORB benchmark [35], and also it is experimentally shown that this new version of capsules is more robust to white box adversarial attacks than CNNs and fully-connected capsules.

Rawlinson *et al.* (2018) [36] demonstrates that the distinctive features of Capsule Networks disappear when the last capsule layer (latent capsules) is not masked with a supervision for specializing an identity on each latent capsule. In addition, supervised capsule architecture is constraint to be designed in shallow form since the aforementioned degenerate capsule behaviour without masking would even occur in deeper designs of Capsule Networks. To mitigate the negative effects of these limitations, unsupervised sparsening method for capsules is proposed in this study. The main objective of this study is to recover the distinctive features of capsules (e.g. equivariances) through the instrument of fully unsupervised capsules.

Zhang *et al.* (2018) [37] proposes Capsule Projection Network (CapProNet) that learns an orthogonal projection matrix for capsule subspaces where each is updated until the input feature vector refers to corresponded class. In this approach, the capsule projection is used for multi-dimensional weight normalization in capsule subspaces, instead of normalizing the capsule projections onto 1-d subspaces. This study has better performances on well-known benchmark data sets with the same level of computational and memory expenses as in the SOTA methods.

Lenssen *et al.* (2018) [38] introduces guaranteed equivariance and invariance properties to the Capsule idea with two main contributions. The first one is that providing

provable equivariance properties under transformations by grouping the elements, and the latter is that connecting the outputs of Capsule Networks to combine the strengths of both ideas of Capsule Networks and grouping the elements in a single deep neural network architecture. As a result, this structure allows to generate sparse evaluation of groups of the elements and to control particular equivariance and invariance properties of an instance.

Zhang *et al.* (2018) [39] addresses the problem of expensive computational process during dynamic routing which creates the bottleneck preventing widespread applications of Capsule idea. In this study, the framework of weighted kernel density estimation is exploited by two fast routing methods with different optimization strategies in order to generalize the existing routing methods. With the help of these methods, it leads to construct a Capsule Network architecture that handles larger-sized images in a time-efficient manner.

In analogy to above cited study, Wang *et al.* (2018) [40] formulates routing mechanism as in [4] in optimization perspective, and combines an agglomerative clustering-like loss and KL regularization term between the distribution of current coupling coefficients and its previous form. According to its results, this idea improves the convergence behavior of Capsule layers, and allows to use more routing iterations between Capsules in each step. At the end, it leads to get better results on unsupervised perceptual grouping task.

Jaiswal *et al.* (2018) [41] presents a framework that uses Capsule Network architecture as discriminator within the Generative Adversarial Networks (GANs) [18], instead of CNNs. To achieve this, Capsule design guides the images, and the objective function for GANs is updated by margin loss for the discriminator part. This study shows that GANs with Capsule discriminators can achieve better results than convolutional-GANs for modelling the image data distribution on MNIST and CIFAR data sets.

Xinyi *et al.* (2019) [42] presents a framework that adapts the capsule idea to Graph Neural Networks (GNNs) to mitigate the problems on GNN-based graph embeddings algorithms. Basic node features are extracted by a capsule-formed GNN, thereafter, high-quality graph embeddings are produced by routing mechanism and attention module. In principle, attention module leads to eliminate the issue of various-sized graphs. The proposed framework achieves the SOTA performance on 6 out of 10 benchmark data sets in domains of biology and social network.

Kosiorek *et al.* (2019) [43] presents an unsupervised version of Capsule Networks where a neural encoder looks at all parts of an instance, and infers the presence and pose configuration of the instance. Moreover, a decoder predicts the pose configuration by using a mixture of pose predictions. By packing the encoder and decoder parts together, this framework has the capacity for learning the inference of the objects and their parts without any supervision. As a results, it achieves SOTA and comparable results for unsupervised classification on SVHN and MNIST data sets, respectively.

Kınlı *et al.* (2019) [44] compares the performances of CNNs and Capsule Networks with dynamic routing on more realistic, diverse and rich data set in fashion domain. In this design, convolutional layers are stacked without any pooling operation before Capsule layers for extracting low-level features and routing them to the Capsules as input. The result of this experimental study on DeepFashion data set [1] shows evidence of that Capsule Networks trained only on images, not supported by any side information, can perform better than CNN-based deep learning architectures supported by a kind of side information such as landmarks and attributes.

CHAPTER III

BACKGROUND INFORMATION

3.1 Introduction

Convolutional Neural Networks (CNNs) are one of the most commonly used architectures for image-related deep learning studies [45, 6, 46, 23, 47, 18]. Although CNNs have outstanding performances in different domains, due to the nature of CNNs and pooling operations following the convolutional layers, most CNN-based architectures have some limitations such as losing the hierarchical spatial relationship in the images and not being robust to affine transformations. On the other hand, Capsule Networks are composed of groups of neurons, and with the help of its novel routing algorithms, they have the capability for learning high dimensional pose configuration of the objects as well. In this thesis, we investigate the performance of brand-new Capsule Networks using dynamic routing algorithm on in-shop image retrieval task. In this chapter, before revealing the details of our work, we concisely review the basic structure of CNNs frequently used in many Computer Vision tasks, the capsule idea and the structure of Capsule Networks.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) is a very specialized class of Neural Networks (NNs). As in the case of all kinds of NNs, they are composed of neurons with trainable weights and biases, and each neuron in the first layer takes some signals coming from some inputs (e.g. image, audio, 1-dimensional array). These signals are multiplied with the weights of the neurons, and then flows throughout the hidden layers by this way. Before transmitting the output of a neuron to the next layer, a non-linear

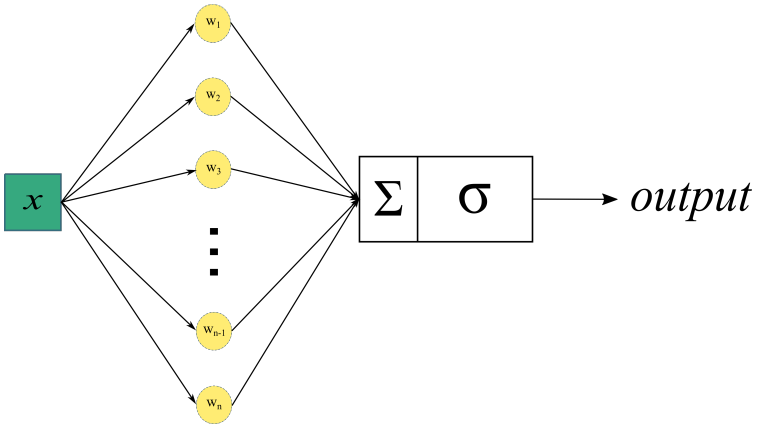


Figure 1: Single layer activation work flow.

activation function is applied to the sum of all weights in this layer in order to produce a nonlinear decision boundary on non-linear combinations of the weights and the inputs. The formula of the output of a single neuron (i.e. perceptron) can be shown as

$$f(x) = \sigma(Wx + b) \tag{1}$$

where x represents the input signal, W and b are the weights and biases respectively, and σ is a non-linear activation function. In regular NNs, each neuron is fully connected to all neurons in the previous layer, and where the neurons in a single layer take completely independent actions from the other neurons in the same layer and does not share any connections between them. At the end, the network predicts class labels or some continuous values for the raw inputs with the help of a differentiable loss function.

CNNs have a different structure than regular NNs, which generally contain a certain number of convolutional layers followed by a non-linear activation function and down-sampling layers (e.g. max-pooling). In this design, all neurons in a particular layer are *locally* connected to a region of the input volume. At this point, convolution operation extracts the features by sliding the weight kernels (i.e. filters) over the local regions of the input. After applying a non-linearity to the output, the spatial

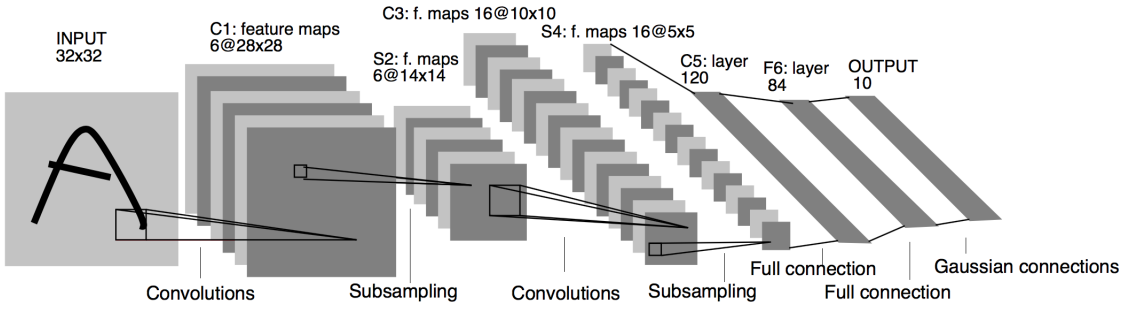


Figure 2: The architecture of LeNet-5 [2].

connections between the region of the input and corresponding pixels on the output compose the receptive fields [48]. In this design, each single pixel in the output essentially represents a large region centered roughly the same part of the input, and thus, the next convolution operation on the current output will substantially operate on the pixels that represent larger region of the original input. For that matter, this mechanism, called effective receptive fields, can propagate their information to the output in more sophisticated way, so that CNNs can extract the features from higher-dimensional data better than regular NNs. Moreover, due to the consecutive convolutional layers in this architecture, low-level features such as corner, texture and edge are extracted from the input data by sharing the weights. Then, these features are combined in deeper layers to compose higher level features. Therefore, this architecture has the capacity for extracting more powerful features from high-dimensional inputs (e.g. images). The representation of one of the earliest CNN architectures in the literature, namely LeNet-5 [2], can be seen in Figure 2.

3.2.1 Convolution Operation

In purely mathematical context, convolution is a function derived from two functions f and g by integration that represents how the shape of one is transformed by the other. In linear systems, this operation is used to explain the relationship between three signals of interest: the input signal, the impulse response, and the output signal.

The formula of convolution operation as follows:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2}$$

2-dimensional space, convolution operation incorporates a 2-dimensional input data and a small matrix of weights, namely *kernel*, to produce a new matrix derived from these two matrices. This kernel basically slides over the local parts of the input data. During this process, an element-wise multiplication is applied to the overlapping parts of the input data and the kernel, and then the results of this multiplication are summed up to a single output value. By this way, after completing the sliding process, the input data is transformed into another 2-dimensional sub-space by using the kernel matrix. In the context of machine vision, the input data and the kernel matrix correspond to the image and a feature extraction filter respectively, and the feature maps as output are essentially the weighted sum of the input features that are located at the same region of the image.

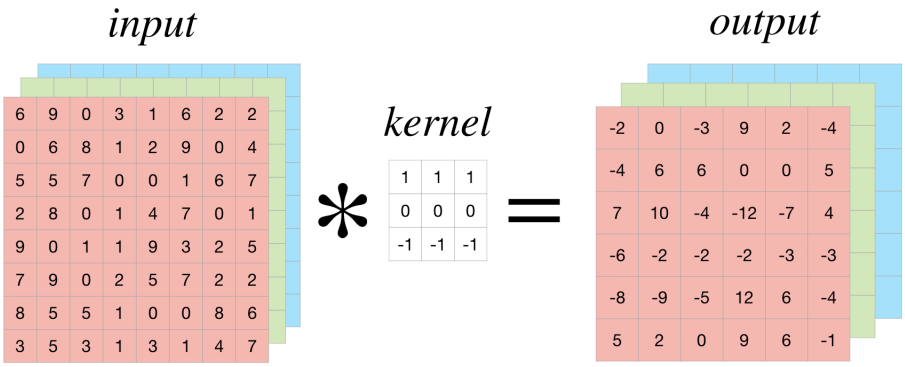


Figure 3: Performing convolution operation by sliding the kernel over the input data to form the receptive fields.

There are two techniques commonly used in order to improve the quality of the features extracted by convolution operations. First, in the original convolution operation, the edge pixels on the input cannot be a center when sliding the kernel, and this

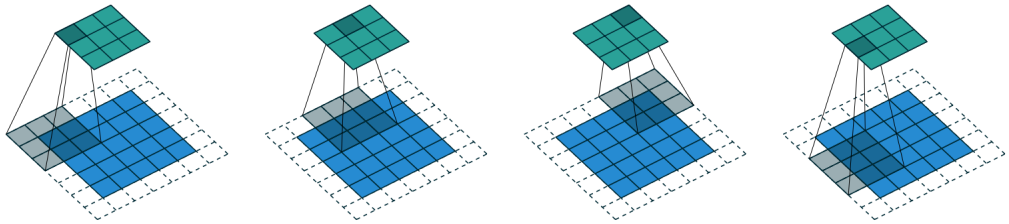


Figure 4: Convolving a 3×3 kernel over a 5×5 input data with padding 1 and stride 2 [3]

leads to lose some information at borders. Padding can be applied to alleviate this problem. In this method, the input data is extended beyond the edges by a certain number of the extra *fake* pixels which is usually zero, so that the edges in the original input can be now a center, and it is possible to keep the information that may be important for the context, yet at borders. Secondly, the common approach for designing a CNN architecture is to reduce the size of spatial dimensions, and to increase the depth of the layer throughout the network. One of the methods to achieve it is to increase the gap between the slide steps of the kernel. This is called as *stride*. In most of modern CNN architectures, increasing the stride is one of the methods to apply for reducing the size of spatial dimensions, in addition to the other methods such as using pooling operations between each two consecutive convolutional layers.

3.2.2 Pooling Operation

Pooling is an operation that contains fixed-size window, namely *pooling kernel*, sliding over all regions in the input data, and computing an output for each region traversed by the pooling kernel. During this process, it makes some computations with all elements in a region overlapped with the kernel in order to route the output to the next layer. In most well-known CNN-based architectures, it is preferred to take the maximum or the average of the values of all elements in the region, and they are called as *maximum pooling* (as shown in Figure 5) and *average pooling*, respectively.

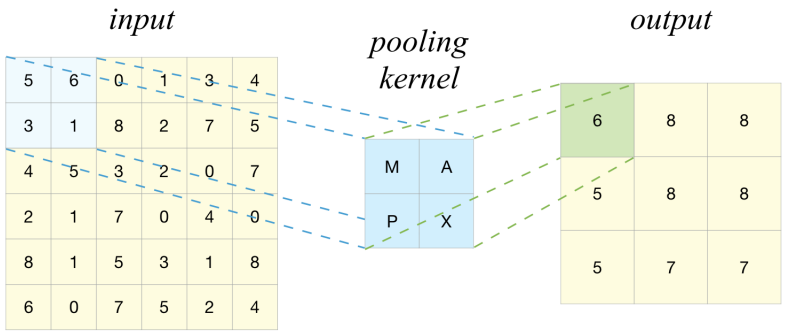


Figure 5: Performing maximum pooling on a 6×6 input data by a 2×2 pooling kernel with stride 1.

Pooling operators are deterministic, and do not contain any parameters. The main objective of pooling operations is to aggregate the information from the input data by reducing the size of spatial dimensions of hidden representations in the network. Therefore, the network can learn a translation and rotation invariant global representation of the input data. As a side effect of reducing the size of spatial dimensions, the number of parameters computed in the following layers dramatically decreases since the total size of the input data is *nearly* halved after pooling operations.

Although there are indisputable advantages of applying one of the pooling operations between successive convolutional layers, as aforementioned before, pooling has two main negative effects on the performance of CNN-based architectures. First, pooling operations are kinds of rudimentary routing methods, where the neurons are picked by a heuristic (e.g. maximum, average) without considering the relationship between pixels. As a result of this inherent limitation of pooling operations, CNNs classify an image by joining some components of the object in the image regardless of the spatial relationship between them. Therefore, CNNs can easily confuse an object with the fake one that contains some components of the object with improper alignment. Secondly, CNNs are not robust to affine transformations since the output of pooling operations completely throws away pose information that may be important

for recognizing the object. In other words, CNNs are not able to capture different pose information of the images if this information is not seen on the training set.

3.2.3 Activation Functions

Activation function is basically a mathematical gate that determines whether a neuron is activated for the next layer or not. In Artificial Neural Networks (ANNs), the main objective of the activation functions is to introduce non-linearity into the output of neurons. These functions have major effects on the computational efficiency of training process and the convergence behaviour of the objective functions.

As mentioned in Section 3.2, the input is fed into the neurons in the current layer, and each neuron has weights W and biases b. The output is formed by multiplying the input and weights, then adding the bias term to it, as shown in Equation 1. This form can be considered as the weighted linear combination of the input. Without applying a function that injects non-linearity to this output, the network is essentially a linear regression model. However, to transfer the information to the next layers in such a way may cause a problem on training, and the network may not perform well when the task contains more complex input representations, or in other words, it is closer to the real-world scenarios. Therefore, ANNs need to append the activation functions to the end of their layers to generate non-linear transformation of the initial output of the neurons in each layer. This makes the network capable to learn more complex representations in the data.

Being differentiable is one of the most important features which activation functions should have. Note that it is assumed that being at least one-sided differentiable at a certain point is sufficient for training process (*e.g.* ReLU [49]). For training of ANNs, due to the chain rule, being able to compute the gradients of the objective function with respect to the weights during backpropagation rests upon the constraint of that the activation functions are differentiable.

Table 1: Various forms of non-linear activation functions.

Name	**Formula**	**Derivative**
Identity	$f(x) = x$	$f'(x) = 1$
Binary Step	$f(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$	$f'(x) = \begin{cases} 0 & x \neq 0 \\ ? & x = 0 \end{cases}$
Logistic	$f(x) = \dfrac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Hyperbolic Tangent	$f(x) = tanh(x)$	$f'(x) = 1 - f(x)^2$
ReLU [49]	$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$
Leaky ReLU [50]	$f(x) = \begin{cases} 0.1x & x < 0 \\ x & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.1 & x < 0 \\ 1 & x \geq 0 \end{cases}$
ELU [51]	$f(x) = \begin{cases} \alpha(e^x - 1) & x < 0 \\ x & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & x < 0 \\ 1 & x \geq 0 \end{cases}$
SoftPlus [52]	$f(x) = ln(1 + e^x)$	$f'(x) = \dfrac{1}{1+e^{-x}}$
Swish [53]	$f(x) = \dfrac{x}{1+e^{-x}}$	$f'(x) = \dfrac{(e^{-x}(x+1)+1)}{(1+e^{-x})^2}$

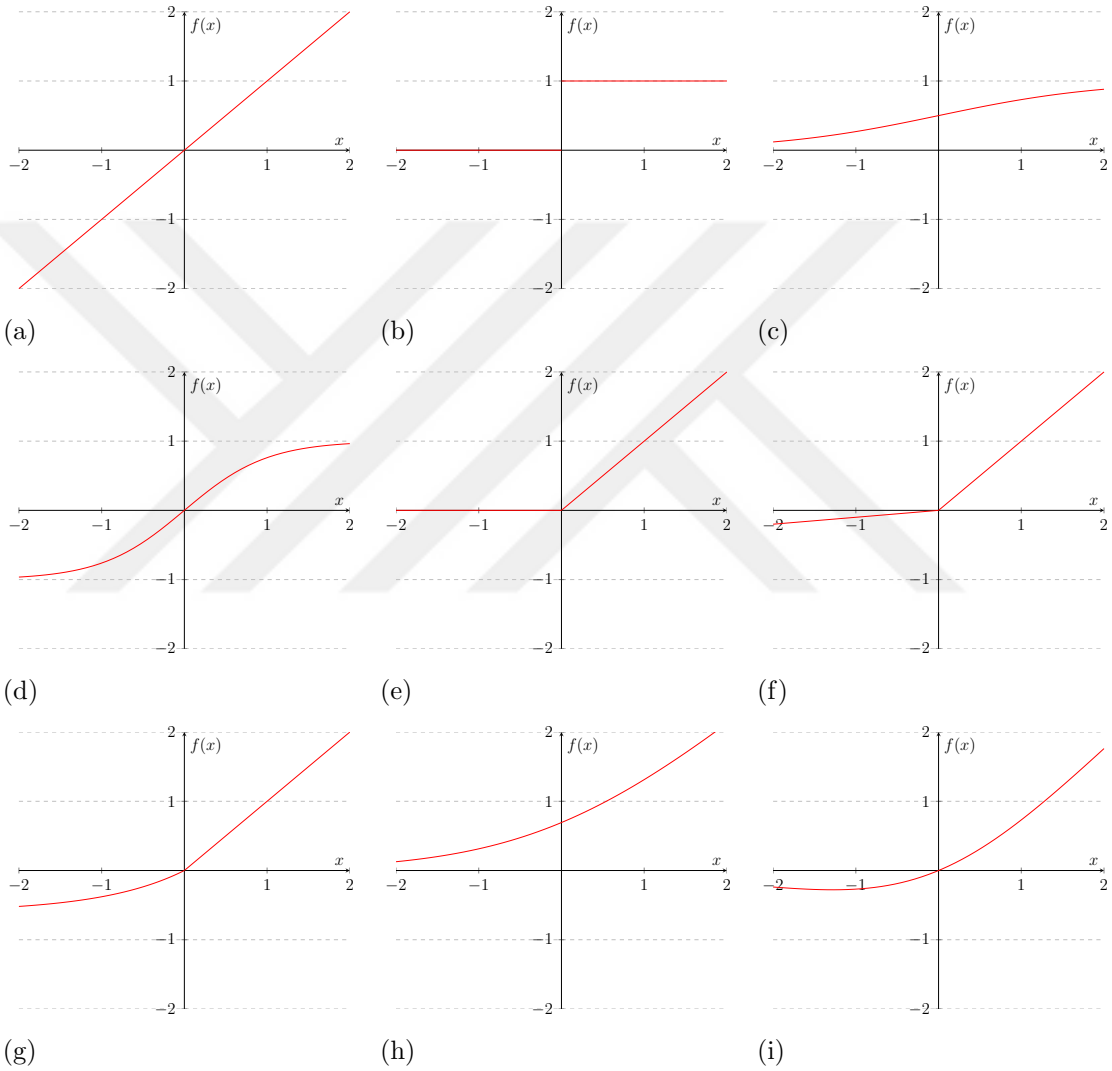


Figure 6: The plots of different activation functions. (a): Identity, (b): Binary Step, (c): Logistic, (d): Hyperbolic Tangent, (e): ReLU, (f): Leaky ReLU, (g): ELU, (h): SoftPlus, (i): Swish.

While Table 1 demonstrates the formulas and the derivatives of different activations functions used for ANNs, the plots of aforementioned functions are shown in Figure 6.

3.2.4 Regularization Methods

In the domain of Machine Learning, one of the most challenging issues in the design of a model is to find the best possible fit not only to the training data, but also to the test data that is not seen by the model during training. This issue is called as *over-fitting*. The model is over-fitted on the training data when the model performs well on the training data while it performs poorly on the test data. To reduce the over-fitting, the model has to be designed in such a way that it generalizes over the training data by using several different regularization methods.

Regularization is a collection of strategies that is employed to a ML model in order to reduce the generalization error of the model. These strategies lead to a better learning process for the model without memorizing the training data by penalizing the objective functions or constraining the decision boundaries of the objective functions.

One of the most common approach to regularize the model is to add a parameter norm penalty to the objective function. As shown in Equation 3, the objection function is denoted as J, the regularizer function as Ω, and α is a hyper-parameter for adjusting the impact of the norm penalty on the weights. The regularizer function Ω is commonly picked as L1-norm to scale the gradients by a constant, or L2-norm to decay the weights gradually. To have stronger regularization effect on the model, α should be increased, and setting it to 0 means no regularization on the model.

$$J(\Theta; X, y) = J(\Theta; X, y) + \alpha\Omega(\Theta) \tag{3}$$

The other useful regularization method is Dropout [54]. For each training step, it randomly drops out some hidden units in the network, and different combinations of

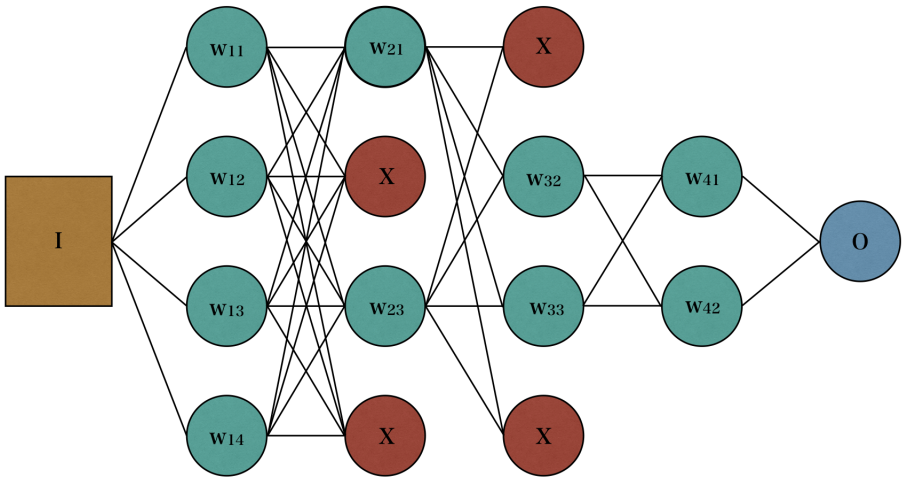


Figure 7: Basic representation of Dropout mechanism.

the hidden units generate different sub-networks during training in order to prevent complex co-adaptations of the hidden units. Therefore, the model have the capacity to learn more generalized representations in the input data. This method is computationally inexpensive, but very powerful to regularize the model. In principle, dropping out a unit can be considered as multiplying the activation of this unit by 0.

NNs are starve for the data. The larger amount of data provided for training *mostly* improves the performance of the model. However, in most of time, it is not possible to collect task-specific data, and to annotate it manually since this is an exhaustive process to complete. Augmenting the training data is another regularization method of training a neural network model. Especially for Computer Vision tasks, samples in the training set may have enormous variety of factors of variation, and by this way, the model can be generalized on the training data by fitting the augmented samples applied many different transformations. Some examples of these transformations can be considered as rotating the images by a certain degree, cropping the images randomly, and shifting the pixels in a particular direction.

3.3 Capsule Architecture

CNNs are applicable for a wide range of Computer Vision tasks such as image classification [45, 6, 46], object detection [23, 47], object localization [55, 56], synthetic image generation [18]. This popularity stems from the fact that they can automatically learn by deriving the identical properties of the data without needing any prior knowledge about the domain. However, due to the pooling operations and the nature of CNNs, this architecture has two significant intrinsic limitations which are referred as losing the hierarchical spatial relationship in the images and not being robust to affine transformations.

Recently, an alternative Deep Learning approach called Capsule Networks (CapsNets), with a novel routing algorithm between capsules, has been proposed by Sabour and Hinton *et al.* [4]. In this design, it is supposed to learn the information about the object and the intrinsic spatial relationship between the parts of the object by harnessing the routing-by-agreement algorithm. Thus, CapsNets are able to recognize the objects regardless of the viewing angle and without needing different transformations of them during training.

3.3.1 Fully-connected Capsule Networks

Basically, a capsule could be considered as a group of neurons who together pack a high dimensional information. This information refers to the existence of the entity and pose configuration describing the underlying behavior of the entity in a more refined way. The activation vector within a capsule represents several features of a specific entity such as position, size, orientation, deformation and texture, while the overall length of this vector states the probability of the existence of that specific entity. Capsule output in a layer is routed to the capsules in the next layer by multiplying it with the weight matrix (*i.e.* transformation matrix). The magnitude of the coupling coefficient represents the strength of a parent capsule to be routed.

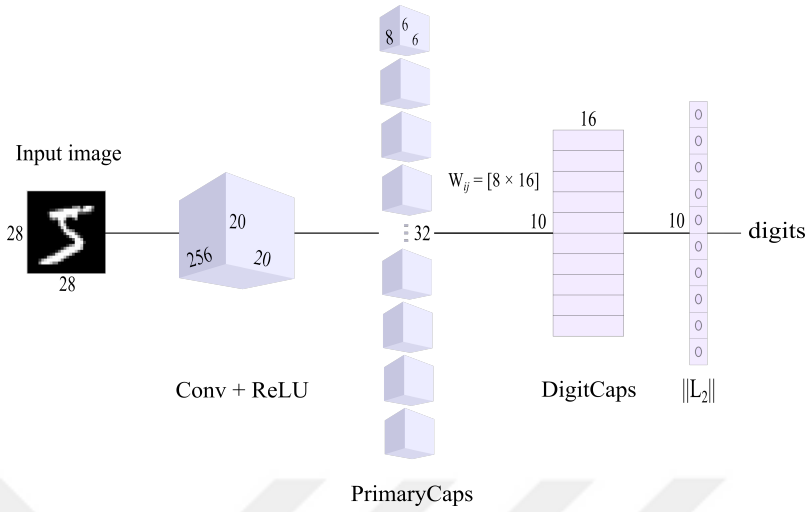


Figure 8: Capsule Network architecture proposed by Sabour and Hinton *et al.* [4].

In other words, this algorithm is kind of a top-down feedback mechanism where the predictions in lower levels determine which capsule in the higher level is activated. This is called "routing-by-agreement" [4]. This algorithm is a far more powerful routing algorithm than pooling variants that pick the neurons by a heuristic.

3.3.2 Dynamic Routing between Capsules

Considering u_i as the output of capsule i, and W_{ij} as trainable transformation matrix

$$\hat{u}_{j|i} = W_{ij} u_i \tag{4}$$

where $\hat{u}_{j|i}$ is the vector that predicts the output of the parent capsule j by capsule i. The relationship between capsules in the previous layer and the possible parent capsule is encoded to a coefficient c_{ij} as *routing soft-max* whose initial logits b_{ij} are the log prior probabilities of routing i^{th} capsule in the previous layer to j^{th} capsule in the next layer. The logits of all capsules in each layer are initialized to 0 at the beginning of the routing-by-agreement algorithm.

$$c_{ij} = \frac{e^{b_{ij}}}{\sum e^{b_{ij}}} \tag{5}$$

The input for the parent capsule j is calculated as weighted sum over all prediction vectors from the capsules in the previous layer.

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{6}$$

A non-linear function called *squashing* is applied to the input for the parent capsule j to ensure that the values in this vector are compressed in a range between 0 and slightly below 1. Note that epsilon value (10^{-7}) is added to the denominator of unit scaling of the input vector since we observed that the gradients vanish at the early stage of our experiments. The final version of squashing formula is calculated as follows.

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\| + \epsilon} \tag{7}$$

Therefore, the magnitude of the dot product of v_j and $\hat{u}_{j|i}$ decides which capsule in the next layer is likely to route (agreement).

$$a_{ij} = v_j \hat{u}_{j|i} \tag{8}$$

For Capsule Networks, the loss is the sum of the losses of all category capsules that are calculated as separate margin loss L_k, for each category capsule k

$$L_k = T_k max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k) max(0, \|v_k\| - m^-)^2 \tag{9}$$

where T_k represents the existence of the instantiation in category capsule k; and m^+, m^- and λ hyper-parameters that control the loss value by the existence, and set to 0.9, 0.1 and 0.5 respectively as proposed in [4].

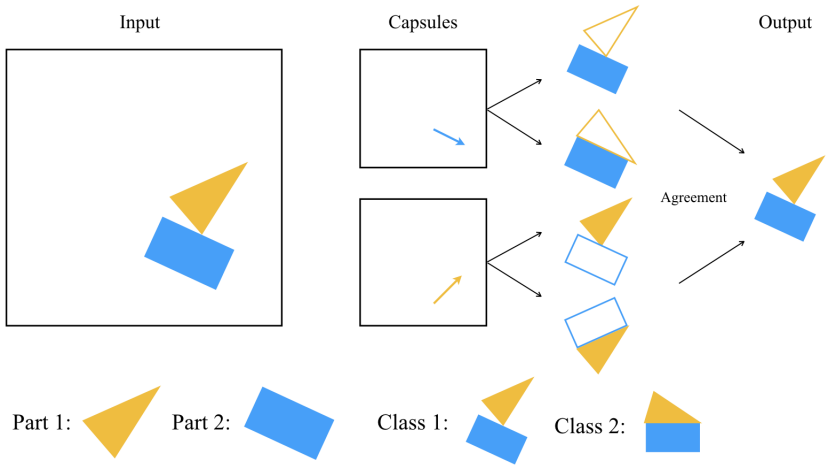


Figure 9: Illustration of the intuition behind routing-by-agreement algorithm [5]

3.3.3 Comparison of CNNs and Capsules

CNNs are capable of learning some different representations directly from sample images by obtaining regional spatial information with local receptive fields [48]. Although CNNs have outstanding performance on image-related deep learning tasks, in real life, there are some intrinsic limitations of this architecture. First, Pooling layer is used for down-sampling the output of the previous layer and routing it to the next layer. With the help of this layer, CNNs are translation-invariant. However, this operation leads to lose significant information during routing the output of a layer to the next layer by ignoring the spatial relationship between some parts of the image. Moreover, when the network goes deeper, the information loss gradually increases throughout the network. For this reason, CNNs cannot gather the hierarchical information between important pieces that identify the object. At this point, it ends up with the loss of the spatial information, and regional information can only be learned by the network. As a result, CNNs does not perform well on the data that contains more complex representations of the objects without supporting the data by an additional side information such as landmarks (i.e. the location information of specific parts of an object)

Likewise, CNNs are not robust to affine transformations. Due to pooling operations, this architecture cannot employ pose information for recognizing the objects (i.e. position, size, orientation). An image with pose configuration that is not encountered during training could be misclassified by CNNs on testing phase. Therefore, the training data has to include different kinds of transformations of the sample images to get better performances in CNN-based architectures. Based upon these reasons, a new deep learning architecture called Capsule Networks is proposed by Sabour and Hinton *et al.* [4]. The idea behind this approach [30] goes back decade ago, but it recently starts to work well after inventing dynamic routing algorithm.

Capsule Networks can also perform well by using only small amount of data that provides limited amount of transformation information of the data to feed the network. The main reason behind this is that the weights of transformation matrix W_{ij} are back-propagated to learn the affine transformation of the entity represented by i^{th} capsule in the primary capsule layer. This means that W_{ij} learns the rotational transformations for a given entity. In addition, in contrast to scalar activations flowing on the other types of NNs, Capsule Networks allows to apply sophisticated calculations that consider more details (i.e. pose, deformation, velocity, albedo, hue, texture) of more complex images with the help of flowing vectors between layers. Therefore, it is possible to emphasize the underlying linear relationship between visual entities that have many different geometric variations.

There are some shortcomings of Capsule Networks. This brand-new Deep Learning architecture just has not been engineered enough to be able to achieve CNN-like performances. Moreover, due to the dynamic routing algorithm, to train a Capsule Network needs much more memory, and it is completed in longer time when comparing CNN-based networks with the same number of parameters. Despite all, Capsule idea has a ton of potential, and the further research on this architecture may directs to much more improvements than its potential.

3.4 Triplet-based Design

A Triplet Network [13] inspired by *Siamese Networks* [22] is a design of learning the similarity between the pairs. In this design, there are 3 instances for the same feed-forward Neural Network that shares the weights throughout the network, and they are denoted as anchor instance x, positive instance x^+ and negative instance x^-. The main idea behind designing triplets is to learn to generate such embedded representations of these instances that minimizes the distance between the embeddings l and l^+, both of which have the same identity, and maximizes the distance between the embeddings l and l^- that has a different identity. The formula of Triplet-based design of a Neural Network as follows

$$Network(x, x^+, x^-) = L(d(l, l^+), d(l, l^-)) \tag{10}$$

where $d(l_1, l_2)$ is the distance metric that measures the distance between the embedding representation of two instances, and $L(d_1, d_2)$ is the objection function that learns to minimize d_1 and to maximize d_2.

The embeddings generated by the network is represented by $f(x) \in \mathbf{R}^d$. This function projects the instance x onto d-dimensional Euclidean space. Herewith, the embeddings are constraint to be confined in d-dimensional hyper-sphere by normalizing them [57]. As illustrated in Figure 10, in a Triplet-based design, it makes the positive instance close as much as possible to the anchor instance while discriminating the negative instance at least a certain distance in Euclidean space. This distance is defined as margin α. The mathematical representation of this relationship is shown in Equation 11.

$$\begin{aligned} d(l, l^+) &= \|f(x) - f(x^+)\|_2^2 \\ d(l, l^-) &= \|f(x) - f(x^-)\|_2^2 \\ d(l, l^+) + \alpha &< d(l, l^-) \end{aligned} \tag{11}$$

The objective function that learns to separate the instances in a triplet into Euclidean space in such a way is formalized as follows

$$\sum_{i}[d(l_i, l_i^+) - d(l_i, l_i^-) + \alpha] \tag{12}$$

The most critical part of training a Triplet Network is to select triplets that obey the constraint in Equation 11. Generating all possible triplets that obey this constraint does not contribute to the training phase at all, and it leads to slower convergence behavior of the objective function [13]. Therefore, there is a need for more clever idea to generate triplet set. For that matter, triplets have to be selected as *hard*, so that they can improve the learning process. Selecting the furthest one of the positive instances to the anchor instance is one of the triplet selection strategies, and called as *hard positive sampling*. The opposite idea, selecting the closest one of the negative instances to the anchor instance, can be considered more powerful method to sample triplets, namely *hard negative sampling*.

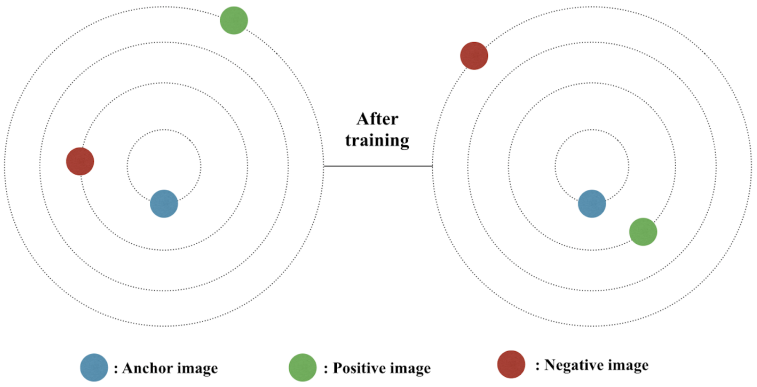


Figure 10: Learning phase of Triplet Networks.

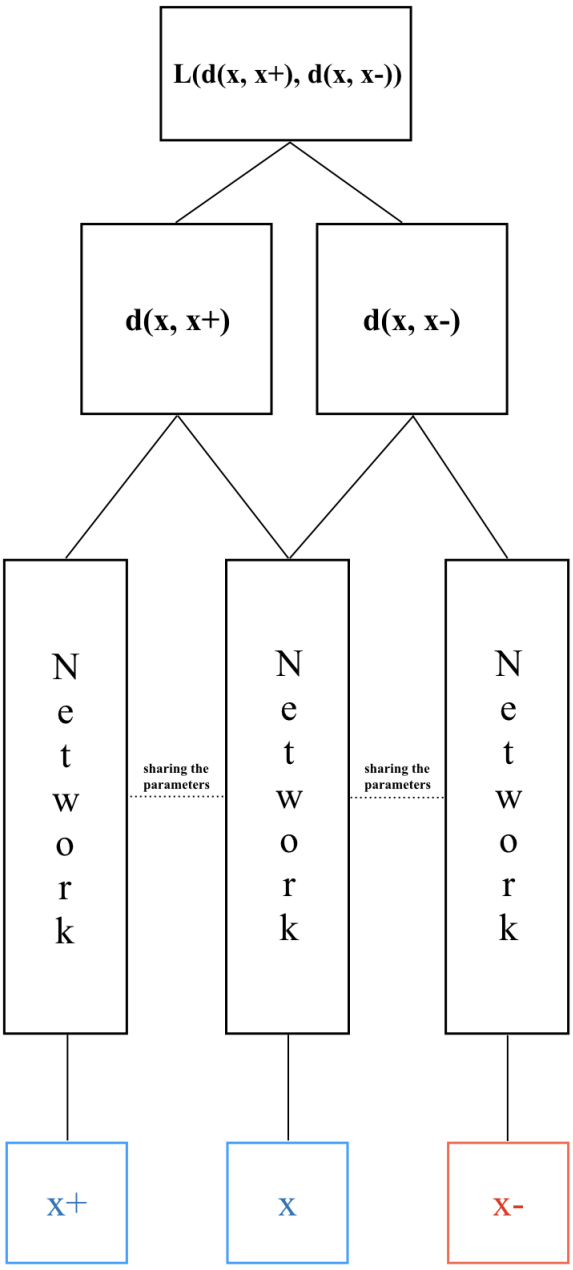


Figure 11: Triplet Network structure. x: anchor, x^+: positive, x^-: negative, $d(x_1, x_2)$: the distance metric and $L(d_1, d_2)$: the objective function.

CHAPTER IV

EXPERIMENTAL STUDY

4.1 Introduction

In this thesis, the main objective is to observe the similarity learning performance of Capsule Networks designed Triplet-based on realistic, diverse and large-sized clothing images. To achieve this, we propose two different Capsule Network architectures using the dynamic routing algorithm. In these architectures, it is tried to improve the feature extraction of the network structure in the original design, which is not suitable for larger-sized images. We used both N-layer stacked convolutional layers and N-layer residual blocks to extract more complex features from the images. Thereafter, we train our proposed architectures on in-shop image retrieval partition of DeepFashion [1] data set, which has 25k training images, 14k query images and 12k gallery images. Finally, we examine the best performance of our Triplet-based Capsule Network architectures on in-shop image retrieval task, and compare the results both the baseline study (i.e. FashionNet) [1] and the other SOTA methods trained on the same partition of DeepFashion data set.

As an ablation study, we modify our proposed architectures according to the classification task, and train them on 210k training images on the category classification partition of DeepFashion data set. Along the same line, we compare the category classification performance of Capsule Networks with more advanced feature extraction strategies with both the baseline study (i.e. FashionNet) and the other SOTA studies attacked to the same task.

4.2 Baseline: FashionNet

We picked FashionNet proposed in Liu *et al.* [1] as the representative of CNN-based architectures. The limitations of traditional CNN architectures are addressed in FashionNet [1] by taking advantage of hand-crafted landmark information.

FashionNet is built on VGG-16 model [45] by ramifying the last layer into 3 different branches. The first branch in the intermediate layer, named as *pose branch*, is responsible for learning the location value and the visibility of the key-points on the structure of clothes from the images. Separately, *local branch* is the second branch that captures the local features by passing over the output of pose branch. The last branch in this design directly learns the global features from images. At the end, as shown in Figure 12, these branches are concatenated into a single output layer in order to predict the clothing category and attributes, as well as to learn the pair-wise relationships between clothing images. In addition, there are several different variants of FashionNet utilizing different number of attributes (100, 500 or 1000) to create comprehensive profiles of different clothing variations. Liu *et al.* [1], first, pre-train FashionNet by using a large subset of DeepFashion as training and validation data. Thereafter, another (smaller) subset of DeepFashion is used for fine-tuning the pre-trained FashionNet model, where any item in the smaller subset overlaps with the larger one.

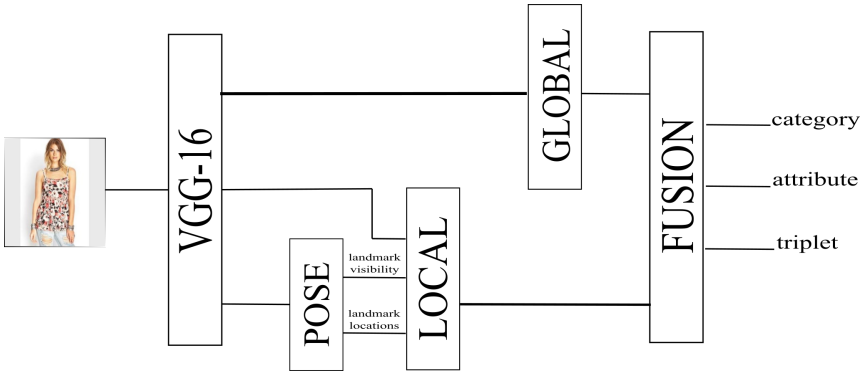


Figure 12: FashionNet [1] model architecture.

FashionNet is optimized by weighted sum of four different loss functions with iterative training strategy. In the first step of this strategy, the location and visibility information for landmarks is estimated with the help of local and global features of the images. Then, by utilizing the estimated landmark locations to gate the local features, the clothing category and attributes are predicted, and the clothing image pairs are retrieved accordingly. At this point, it clearly shows that FashionNet tries to cope with the lack of pose information in CNN-based models by supporting the model with the extra information extracted from the hand-crafted landmark annotations.

4.3 Proposed Architectures

In this study, we design a novel Triplet-based Capsule Network architecture to investigate the image retrieval performance of Capsules in fashion domain. In this design, we modify the original Capsule Network structure to a Triplet-based version, so that the network can learn the similarity between two images by feeding the objective function with the embedded representations of the images extracted by Capsules. The input of this structure is a triplet that has 3 images denoted as anchor image x, positive image x^+ and negative image x^-. Moreover, we have 3-stream Capsule Networks sharing the weights along the whole network. The images in the triplet are fed into each Capsule Network stream one by one, and the network generates 368-dimensional embeddings for each stream that represent the images in Euclidean sub-space. During forming these embeddings, we normalize all Capsules by L2-norm, and then we masked all Capsules but the one that belongs to the correct class in order to generate sparse encodings, and lastly flattening out them to 368-dimensional vector representation. The embeddings are respectively denoted as anchor embedding l, positive embedding l^+ and negative embedding l^-.

As shown in Figure 8, Capsule Networks essentially contain two main blocks: feature extraction block and Capsules. In the default methodology proposed by Sabour

and Hinton *et al.* [4], the feature extraction block has a single convolutional layer with 64 filters. Extracting the features by such a shallow structure may be enough for 1-channel handwritten digit images with the size of 28×28 [4]. However, Capsules need more complex features as the input to achieve better results on the data sets with 3-channel larger-sized images. Therefore, we design two different feature extraction blocks to generate the Capsule input formed by more powerful features. The first idea is to stack a number of convolutional layers with different number of filters without using any pooling operation between these layers. In this way, it is mainly aimed to use the power of convolutional layers in extracting the features from the images, and to combine these features by preserving the hierarchical spatial relationship between pixels in the higher levels. Moreover, adding residual blocks [6] before Capsules is the other idea to extract more powerful features from the images. Residual blocks are basically 2-layer stacked-convolutional blocks with residual connections between the input and the output of the blocks. In certain circumstances (e.g. going too much deeper), non-linearity may cause vanishing or exploding the gradients. These connections allow the gradients to flow through the network directly, without passing through non-linear activation functions, and it leads to propagate larger gradients to the earlier layers, so that they can learn as fast as the layers at the end.

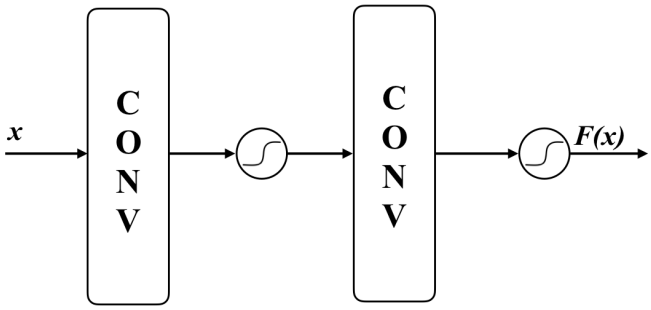


Figure 13: Illustration of 2-layer stacked-convolutional structure.

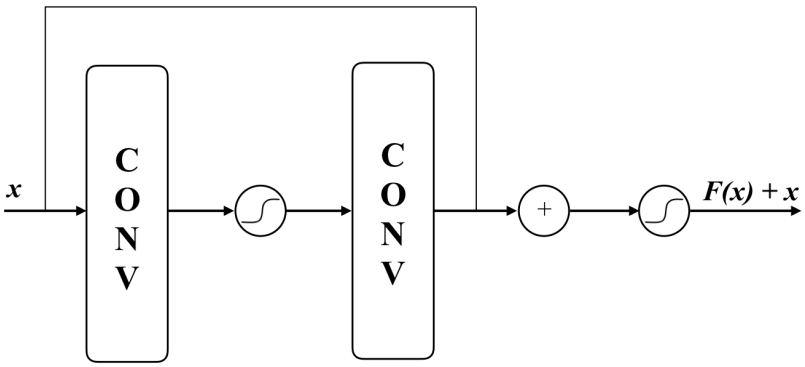


Figure 14: Simple residual block [6]

Between convolutional layers in both design, it is used leaky rectified linear unit (Leaky ReLU) [50] as activation function that allows for a small, non-zero gradient (when the unit is saturated and not active) and batch normalization [58] between convolutional layers for regularization purposes. As a side effect, adding more convolutional layers before capsules reduces the number of trainable parameters of fully-connected Capsule layers, so that it can provide feasible training of such a large data set within limited computational resources.

Furthermore, Capsule structures are identical in both designs. There are two fully-connected Capsule layers which are called *Primary Capsule* and *Class Capsule* respectively. Primary Capsule layer has 32 channels of 16-dimensional fully-connected Capsules, and the feature extraction block is connected to this layer. In this layer, Capsules are activated by *squashing* function as shown in Equation 7. Activated Capsule outputs are routed to 16-dimensional Class Capsule layer after iterating dynamic routing algorithm 3 times. Any kind of reconstruction methods, as in the original architecture [4], is not applied to our Capsule Network design. Stacked-convolutional (i.e. namely SCCapsNet) and residual-connected (i.e. namely RCCapsNet) architectures are shown in Figure 15 and Figure 16, respectively.

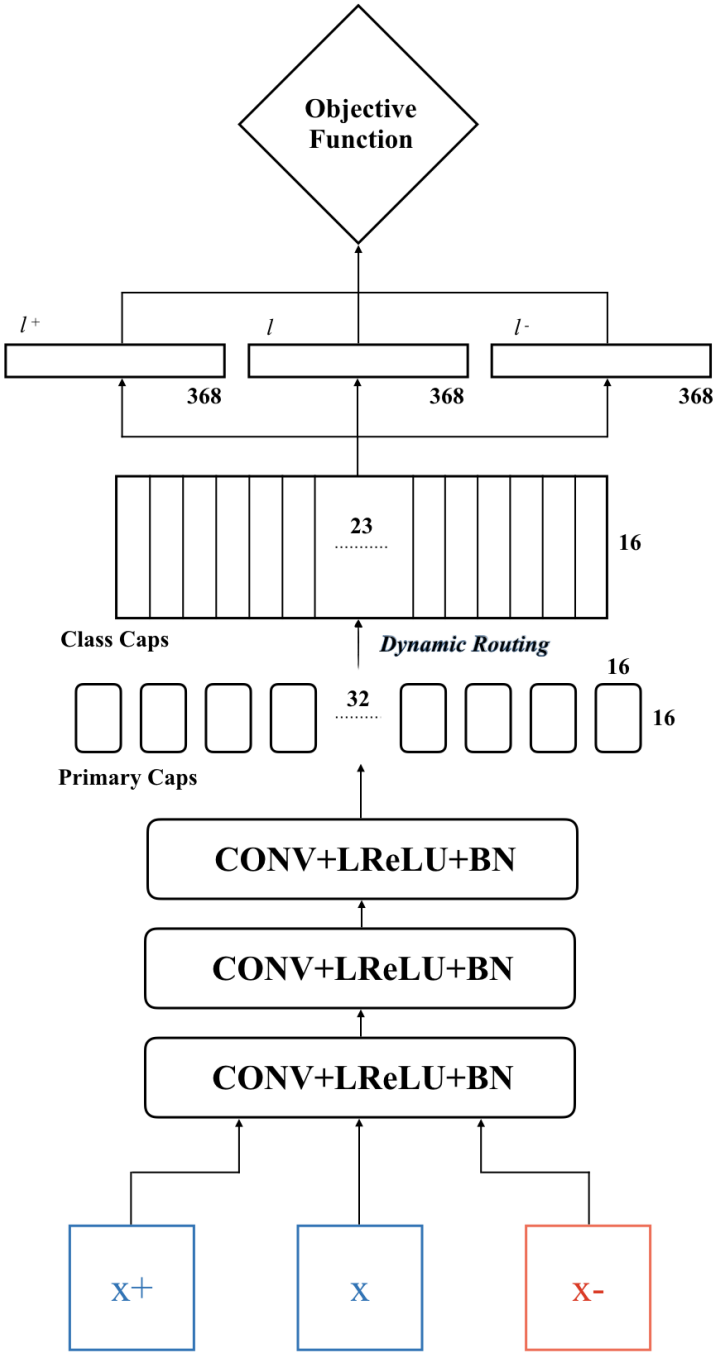


Figure 15: Triplet-based stacked-convolutional Capsule Network architecture (SC-CapsNet).

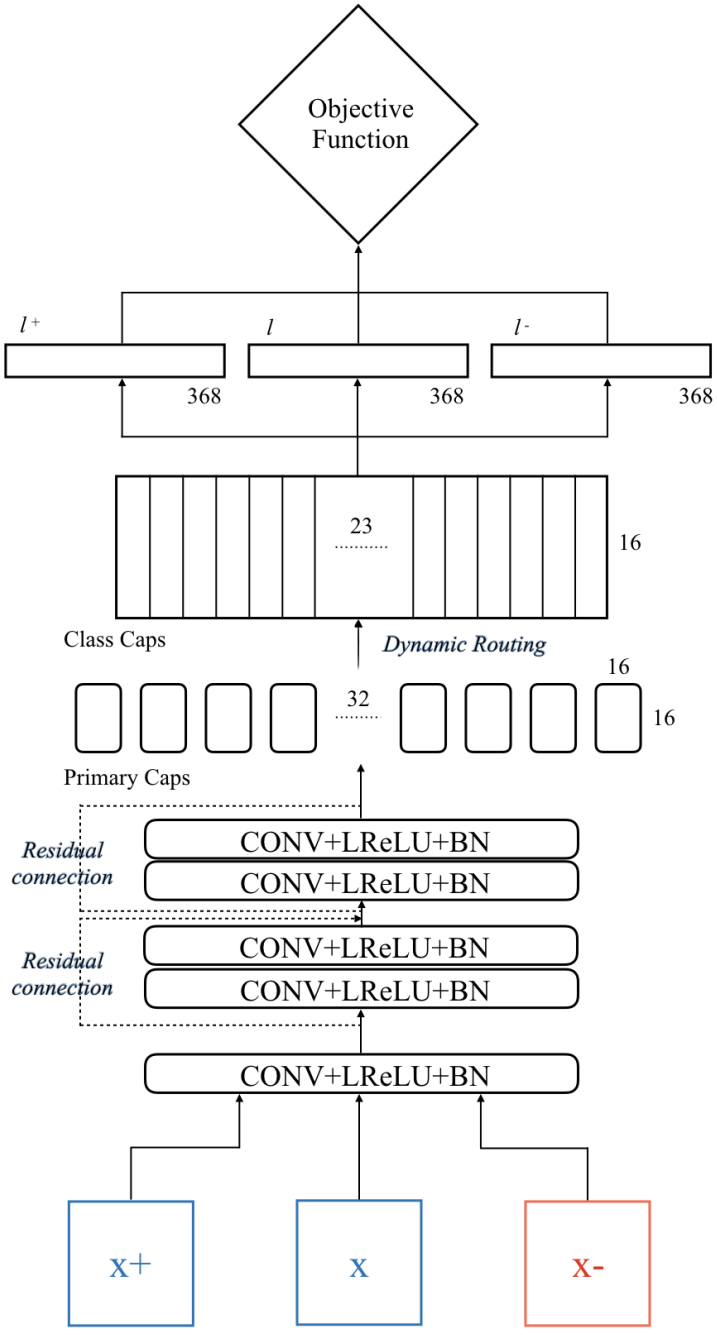


Figure 16: Triplet-based residual-connected Capsule Network architecture (RCCapsNet).

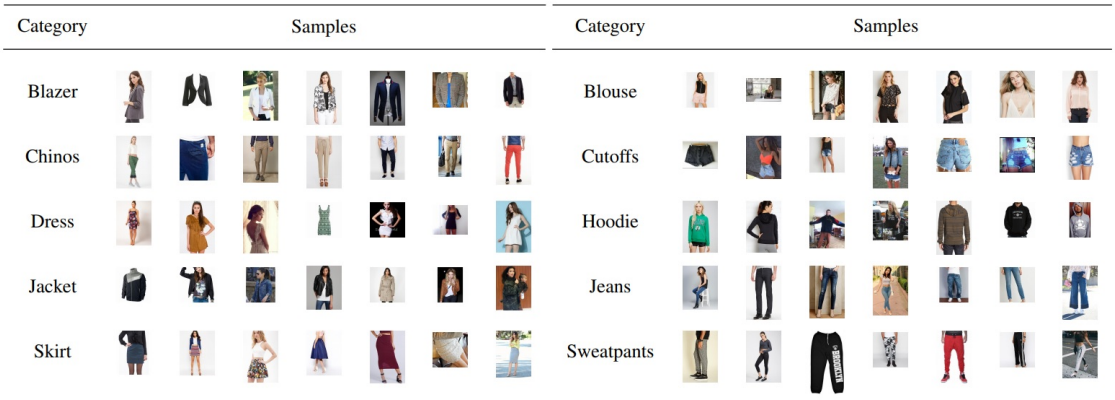
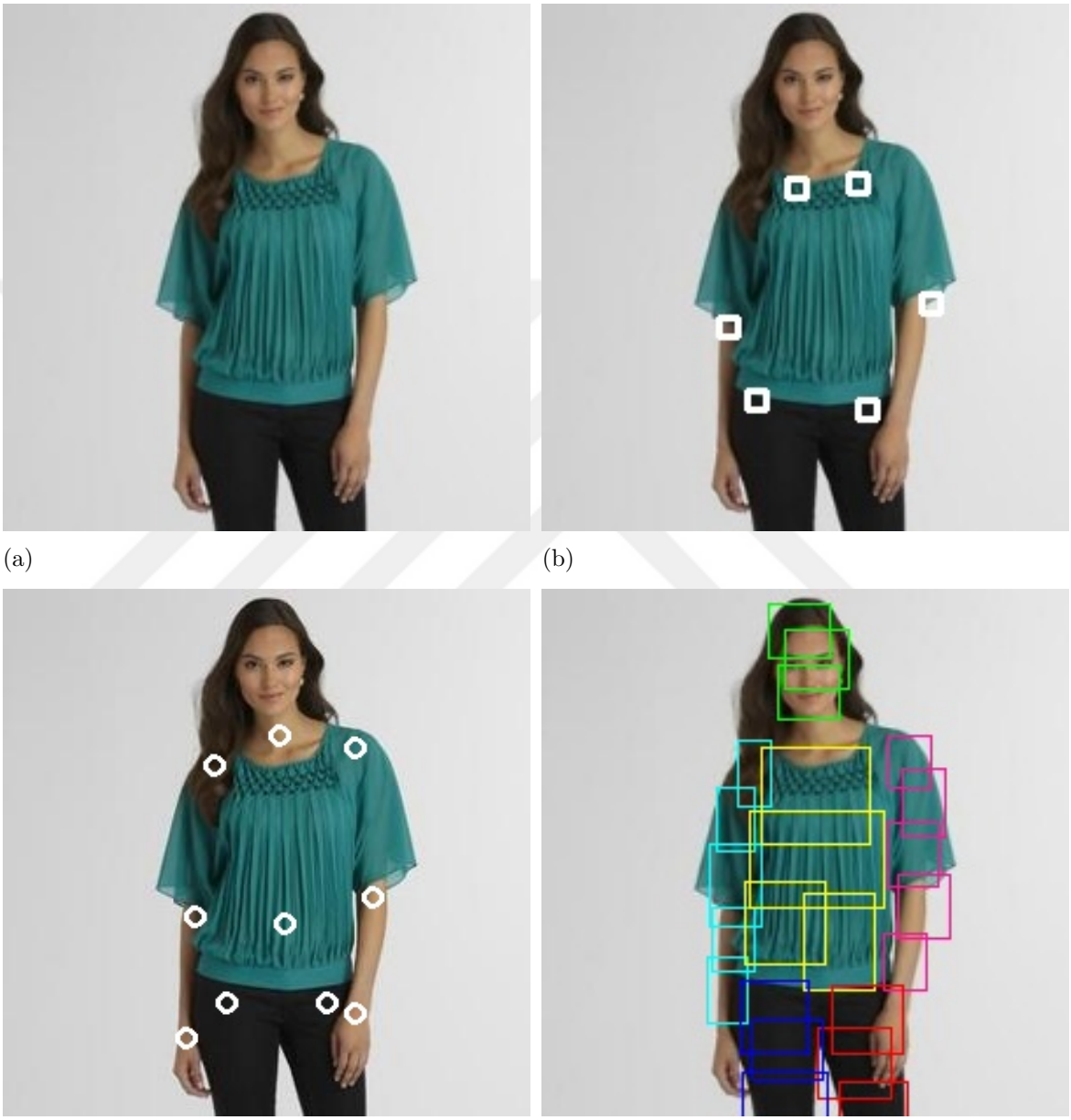


Figure 17: Example images from DeepFashion [1] data set.

4.4 Data set

DeepFashion [1] is a data set of 800K high or mid-resolution images that belong to 50 fine-grained categories. The images in this data set are collected by Multimedia Laboratory of The Chinese University of Hong Kong, from two representative online shopping Web pages and user-generated contents on blogs and forums by querying from Google Images. Sample images of DeepFashion data set with category labels can be seen in Figure 17.

DeepFashion is an extensively annotated clothing data set that contains numerous attributes, localization parameters and correspondence of images shot under different scenarios ranging from well-posed online shopping photos to unstructured customer photos. For image retrieval task, there are 25k training images, 14k query images and 12k gallery images with 23 different categories in in-shop partition of this data set. Moreover, the attributes form five groups: texture, fabric, shape, part, and style, and an image can have +8 landmark locations. We specifically did not include these hand-crafted landmarks or attribute information in the data set (shown in Table 2) to our training process since our proposed architectures have the capacity for learning pose information by itself.



(a) (b)

(c) (d)

Figure 18: Example images from DeepFashion [1] data set with (a) and without (b) landmarks employed; with (c) human joints and with (d) poselets, a part of pose.

Table 2: Forming five main attribute groups, and the examples of the attributes in each group.

Groups	**Attributes**
Texture	Floral, Stripe, Paisley, Distressed, Dot, Plaid, Panel, Raglan, ...
Fabric	*Lace, Denim, Chiffon, Pleated, Woven, Leather, Cotton, Linen, ...*
Shape	*Crop, Maxi, Fit, Longline, Boxy, Mini, Skinny, Midi, Pencil, ...*
Part	*Sleeveless, Pocket, V-Neck, Hooded, Racerback, Peplum, Strappy, ...*
Style	*Graphic, Muscle, Tribal, Peasant, Surplice, Polka, Retro, Yoga, ...*

Triplet selection on the data set is one of the most crucial parts of training a triplet-based model to ensure fast convergence [13]. In our design, we did not apply any hard positive sampling method (using all possible positive images in the data set)— to our design; whereas the negative samples are picked in accordance with the distance to the anchor image. In other words, the closest image to the anchor image in Euclidean space provided that it belongs to different category is selected as the negative instance of triplet. The logic of selecting hard samples for the triplets as follows

$$\underset{x^-}{\operatorname{argmin}} \|f(x) - f(x^-)\|_2^2 \tag{13}$$

(a) (b) (c)

Figure 19: An example of a triplet [1] that contains an anchor image (a), a positive image (b) and a negative image (c).

4.5 Implementation Details

Our experiment environment includes Intel Core i7-8700K CPU with 3.70GHz, 32 GB RAM and 2 MSI GTX 1080 Ti Armor OC 11GB GPUs. Our proposed architectures are implemented in Keras framework with Tensorflow backend. Source code is published on GitHub[1].

In hyper-parameter settings, we pick Adam [59] to optimize our objective function with the learning rate 10^{-3}, and decay rate 5×10^{-4}. Image batch for each gradient step contains 32 different samples. Dynamic routing algorithm is iterated three times between Capsule layers in each step. Pixel-wise normalization is applied to all images in the data set. Moreover, we apply some data augmentation techniques to the images after generating the triplets. All hyper-parameter settings is shown in Table 3, and all augmentation techniques used in our design is shown in Table 4.

We use in-shop partition of DeepFashion data set for our experiments, and recall-at-K metric for our evaluation. During testing, we compute the feature embeddings from our network for all images in both query and gallery sets. For each image in query set, we then retrieve top-K similar images from the gallery set. Recall score for each query is 1 if at least one image out-of-K retrieved images has the same clothing identity number as the gallery images. Finally, we compute the average overall score (recall-at-K) on query set.

[1] https://github.com/birdortyedi/image-retrieval-with-capsules

Table 3: Hyper-parameters settings in our design.

Hyper-parameter	**Value**
Optimizer	*Adam* [59]
Learning Rate	*0.001*
Decay Rate	5×10^{-4}
Batch Size	*32*
Routings	*3*
Normalization	*Pixel-wise*

Table 4: Data augmentation methods applied to our design.

Augmentation Methods	**Applied**	**Range**
Feature-wise Centering	✗	None
Sample-wise Centering	✗	None
Feature-wise STD Norm.	✗	None
Sample-wise STD Norm.	✗	None
ZCA Whitening	✗	None
Rotation	✓	[0°-30°]
Width Shifting	✓	[0-0.1]
Height Shifting	✓	[0-0.1]
Channel Shifting	✗	None
Brightness	✓	[0.5-1.5]
Shearing	✓	[0-0.1]
Zoom	✓	[0-0.1]
Horizontal Flipping	✓	None
Vertical Flipping	✗	None

(a) (b) (c) (d) (e) (f) (g)

Figure 20: Example augmented images from DeepFashion data set. (a): Original image, (b): Horizontal Flipping, (c): Rotation, (d): Brightness, (e): Height Shifting, (f): Width Shifting, (g): Zooming.

CHAPTER V

RESULTS AND DISCUSSION

5.1 Introduction

Although CNNs have a proven track record of accomplishments in Computer Vision tasks, as a matter of fact, CNNs neglect pose configuration of the objects, and they are not robust to affine transformations. To try to mitigate these negative effects, in the literature, several different studies attacked to clothing retrieval problem by using different approach such as using semantic attributes [25, 26], textual image descriptors [27], alternative objective functions [19, 20], different sampling strategies [29], network ensembling [20, 21] and attention-based mechanisms [28, 21]. However, Capsule Networks inherently learn pose information without needing any side information or extra modules. Therefore, we attack to the same problem by using Triplet-based design of Capsule Networks.

In our design, we train our both architectures on 25k training samples of in-shop partition of DeepFashion data set [1]. For triplet selection process, we pick the closest image with different category to the anchor image in Euclidean space as the negative image; whereas we pick each possible positive image in the data set as the positive image. As mentioned in Schroff *et al.* [13], applying negative hard sampling to the triplet training improves the convergence behavior of the model significantly. Moreover, this task is an information retrieval task, hence we measure the performances of our models by recall-at-K metric, where K is 1 or multiplies of 10 up to 50. Lastly, we test our models on 14k query and 12k gallery sets of in-shop clothing images in DeepFashion data set. Some examples of retrieved images from gallery set are shown in Table 5.

Table 5: Some examples of retrieved images from gallery set.

Query	Retrieved Images
	

Table 6: The details of our architectures, the baseline study and the other SOTA methods.

Model Name	**Backbone Architecture**	**Side Information (SI) Extra Module (EM)**	**# of (M) Params**
WTBI [25]	AlexNet [60]	Category-specific Similarity (SI)	60
DARN [26]	Custom NiN [61]	Visual Similarity (SI)	105
FashionNet [1]	VGG-16 [45]	Landmark Information (SI)	134
Corbiére *et al.* [27]	ResNet50 [6]	Bag-of-words Descriptors (EM)	25
SCCapsNet *(ours)*	CapsNet [4]	**No SI/EM Used**	2.5
RCCapsNet *(ours)*	CapsNet [4]	**No SI/EM Used**	4.5
HDC [29]	GoogLeNet [46]	Hard-Aware Cascaded Embedding (EM)	5
VAM [28]	GoogLeNet [46]	Attention with Impdrop Connection (EM)	6
BIER [20]	GoogLeNet [46]	Embedding Boosting (EM)	5
HTL [19]	GoogLeNet [46]	Hierarchical Triplet Loss (EM)	5
A-BIER [20]	GoogLeNet [46]	Embedding Boosting with Adversarial Loss (EM)	5
ABE [21]	GoogLeNet [46]	Attention-based Ensembling (EM)	10

In this chapter, we examine the results of our proposed architectures on in-shop clothing retrieval. First, we report the performances of Stacked-convolutional and Residual-connected Triplet Capsule Networks on in-shop partition of DeepFashion data set [1], and then we compare our results with the baseline study, namely FashionNet [1], and the other SOTA methods trained on the same data set. Furthermore, we examine the category-specific retrieval performances of our models for six clothing categories containing most of samples in the data set (i.e. Dresses, Blouses/Shirts for Women, Tees/Tanks for Women, Shorts for Women, Sweaters for Women, and Jackets/Coats for Women). Finally, we present the classification scores of slightly modified version of our proposed architectures as an ablation study. The backbone architectures and the number of parameters of all studies included ours are represented in Table 6.

5.2 Experimental Results

As a result of our experiments on in-shop clothing retrieval task, we report that Stacked-convolutional design (*i.e.* SCCapsNet) achieves 32.1% Top-1, 81.8% Top-20, and 90.9% Top-50 recall-at-K scores, whereas Residual-connected design (*i.e.* RCCapsNet) has unsurprisingly better performance than SCCapsNet. The figures for RCCapsNet as follows: 33.9% Top-1, 84.6% Top-20, and 92.6% Top-50 recall-at-K scores. The main observation for our results is that the residual connections between convolutional layers in the feature extraction blocks improve the overall retrieval performance by 2%. As explained in [6], the residual connections lead to efficiently propagate the gradients to the earlier layers, and prevent the degradation of the training accuracy. Thus, from here we can say that Capsules are enriched by more complex feature extraction methods in order to be suitable for training on more realistic and diverse data sets.

5.2.1 Comparison with the Baseline

In the course of the second experimental comparison, we discuss the results of our proposed Capsule Network architectures, and the baseline study, namely *FashionNet* [1]. Table 7 summarizes the performances of SCCapsNet, RCCapsNet, and the variants of FashionNet. As in the case of all studies mentioned in Section 2, all results are compared based on Top-20 recall-at-K performances.

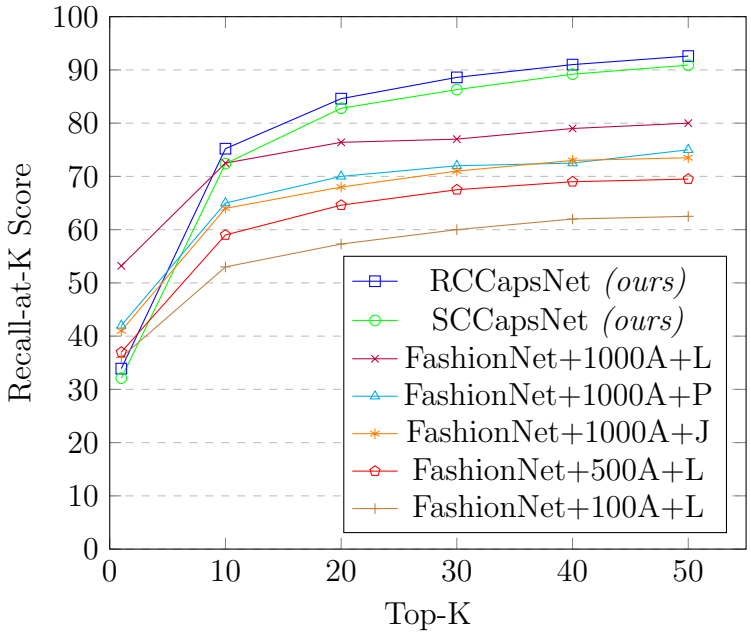
As shown in Table 7, both of our designs outperform all variants of FashionNet including building blocks that employ a smaller number of attributes, and use human joints or poselets instead of landmarks. RCCapsNet gets 5%, and SCCapsNet 3%, better Top-20 recall-at-K scores when compared to the best variant of FashionNet, which utilizes 1000 attributes and the landmark information during training. This result shows that Capsule Network model seeing only the images during training have better performance than, CNN-based architectures supported by hand-crafted

landmarks and attributes. At this point, CNNs may utilize various kinds of side information to deal with the lack of pose information, however, Capsules can inherently learn pose configurations of the images with the help of the activation vectors flowing in the network.

Table 7: Recall-at-K performances of the variants of the baseline study [3] and our proposed models. FashionNet has different building blocks where the model has different numbers of attributes (A) (i.e. 100, 500 and 1000), or fashion landmarks (L) are replaced with human joints (J) or poselets (P). SCCapsNet and RCCapsNet do not use any extra side information during training.

Models	**Top-1** (%)	**Top-10** (%)	**Top-20** (%)	**Top-30** (%)	**Top-40** (%)	**Top-50** (%)
FashionNet+100A+L	36.0	53.0	57.3	60.0	62.0	62.5
FashionNet+500A+L	37.0	59.0	64.6	67.5	69.0	69.5
FashionNet+1000A+J	41.0	64.0	68.0	71.0	73.0	73.5
FashionNet+1000A+P	42.0	65.0	70.0	72.0	72.5	75.0
FashionNet+1000A+L	**53.2**	72.5	76.4	77.0	79.0	80.0
SCCapsNet *(ours)*	32.1	72.4	81.8	86.3	89.2	90.9
RCCapsNet *(ours)*	33.9	**75.2**	**84.6**	**88.6**	**91.0**	**92.6**

Recall-at-K performance of FashionNet variants and our proposed models



5.2.2 Comparison with the SOTA

In this section, we show the next comparison for results of our proposed models with the other SOTA methods in the literature. Table 8 summarizes in-shop clothing retrieval results of SCCapsNet, RCCapsNet, and the SOTA methods. These figures indicate how successful our proposed models are, and what the main limitations of them are when compared to the SOTA CNN-based architectures. First, both of our designs outperform the earlier methods (*i.e.* WTBI [25] and DARN [26]) which both disparately use semantic attributes to improve the overall performance, but neglect pose configurations of the images during training. According to Top-20 recall-at-K scores, while SCCapsNet improves their scores by 31% and 14%, RCCapsNet achieves these improvements with a margin of 34% and 17% respectively.

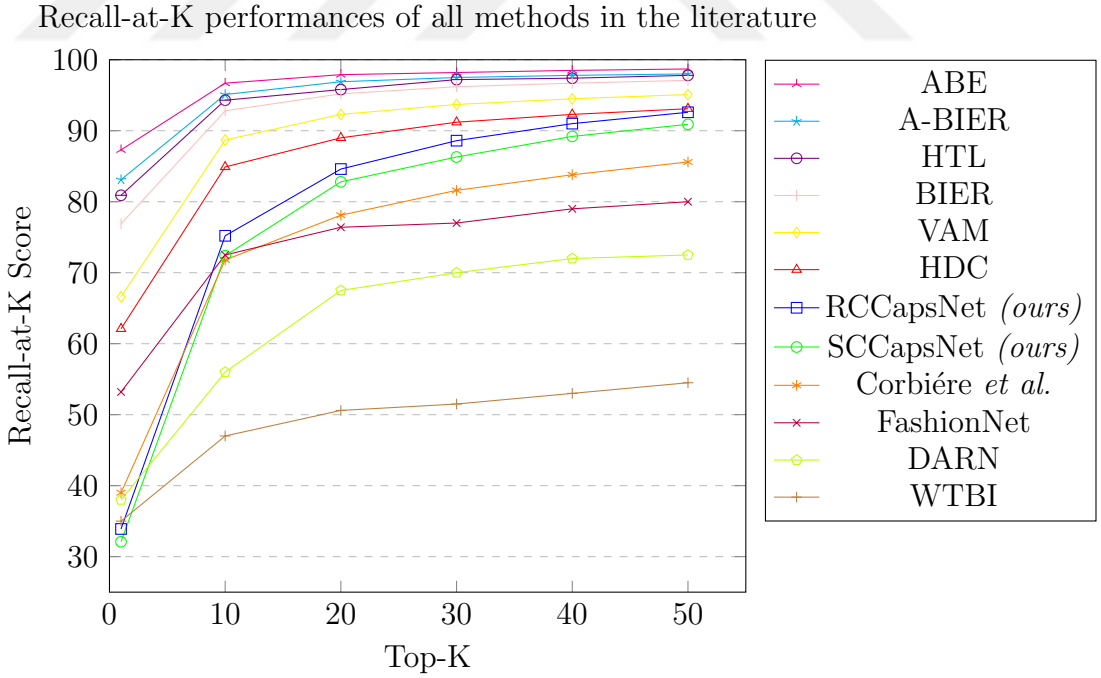
Moreover, as aforementioned in Section 5.2.1, our proposed architectures get better retrieval scores than the best variant of FashionNet (*i.e.* supported by 1000 attributes and 8 landmarks) on in-shop clothing retrieval task. Although FashionNet tries to recover pose configuration thrown away by the nature of CNNs by employing landmark information to their training process, Capsule Networks perform better by using only images with less complex feature extraction parts, since they can inherently learn pose configuration by preserving the hierarchical spatial relationship between pixels. Next, the other approach whose the performance is worse than ours is the method of leveraging weakly-annotated textual descriptors of the images for resolving clothing image retrieval proposed by Corbiere *et al.* [27]. In this design, these textual descriptors (*i.e.* bag-of-words) represent different coarse semantic concepts such as texture information, color and shape. Capsules can directly learn these concepts from the images in a more sophisticated way, hence, SCCapsNet and RCCapsNet can achieve higher recall-at-K scores than this approach without taking advantage of bag-of-words descriptors.

In addition to all these, our proposed architectures cannot achieve the performances of more advanced CNN-based architectures. In these designs, there are various techniques applied to CNNs to boost the overall performances, which are different hard sampling strategies [29], more advanced objective functions [19, 20], ensembling the networks [20, 21] and attention-based mechanisms [28, 21]. Although these techniques may significantly improve the overall performance in CNNs, in principle, they increase the model complexities by a wide margin, or increase training time considerably. First, the numbers of trainable parameters in our proposed architectures, SCCapsNet and RCCapsNet, are respectively $\sim$2.5 and $\sim$4.5 million, while the SOTA methods have twice as many trainable parameters in their models. Moreover, Capsule Networks need more time for training than CNNs since dynamic routing algorithm is a relatively slow routing mechanism when compared to the pooling variants. Therefore, within limited computational resources, these techniques are not applied to our models to boost the overall performance of Capsule Networks, and left as future research ideas.

Capsule idea [30] is not a new notion in Machine Learning, but the first working implementation of this idea [4] includes a novel routing mechanism flowing the information between Capsule layers. However, encapsulating the neurons as a densely-connected layer is completely different, but not a complex structure. Therefore, Capsules in this form have such structural limitations. On the other hand, there are some remarkable improvements on Capsule Networks (*i.e.* Matrix-formed Capsules [33] and Stacked Capsule Auto-Encoders [43]), but densely-connected Capsule Network architecture is the only version whose achievements have been reproduced in the literature. In the future, our proposed models, SCCapsNet and RCCapsNet, may inherit extra performance boost on in-shop clothing retrieval, due to advances in the relatively new Capsule Network research.

Table 8: Experimental results of in-shop image retrieval task on DeepFashion data set.

Models	**Top-1** (%)	**Top-10** (%)	**Top-20** (%)	**Top-30** (%)	**Top-40** (%)	**Top-50** (%)
WTBI [25]	35.0	47.0	50.6	51.5	53.0	54.5
DARN [26]	38.0	56.0	67.5	70.0	72.0	72.5
FashionNet [1]	53.2	72.5	76.4	77.0	79.0	80.0
Corbiére *et al.* [27]	39.0	71.8	78.1	81.6	83.8	85.6
SCCapsNet *(ours)*	32.1	72.4	81.8	86.3	89.2	90.9
RCCapsNet *(ours)*	33.9	75.2	84.6	88.6	91.0	92.6
HDC [29]	62.1	84.9	89.0	91.2	92.3	93.1
VAM [28]	66.6	88.7	92.3	-	-	-
BIER [20]	76.9	92.8	95.2	96.2	96.7	97.1
HTL [19]	80.9	94.3	95.8	97.2	97.4	97.8
A-BIER [20]	83.1	95.1	96.9	97.5	97.8	98.0
ABE [21]	87.3	96.7	97.9	98.2	98.5	98.7



5.2.3 Category-specific Comparison

This section demonstrates the category-specific results of our proposed architectures on in-shop clothing retrieval task. Table 10 summarizes the results of SCCapsNet and RCCapsNet under six different clothing categories. These categories are *Blouse/Shirts*, *Tees/Tanks*, *Dresses*, *Shorts*, *Sweaters* and *Jackets/Coats*, and they are picked based on sample intensity of each category in the data set.

First, RCCapsNet gets more accurate results than SCCapsNet on all specific categories. This is unsurprisingly in line with the overall performances. The next observation is that the categories containing much more samples are more sensitive to the change in feature extraction block than the categories with less samples. For example, the maximum change observed on Top-20 recall-at-K performances of both of our designs appears in Dresses category, which is third most intense category, by 9% margin. On the other hand, the difference between our designs for less intense categories is little if any (*e.g.* for Sweaters, it is only 0.4%; for Jackets/Coats, it is 0.6%). This indicates that picking more powerful feature extraction block lets the model perform better for more diverse categories.

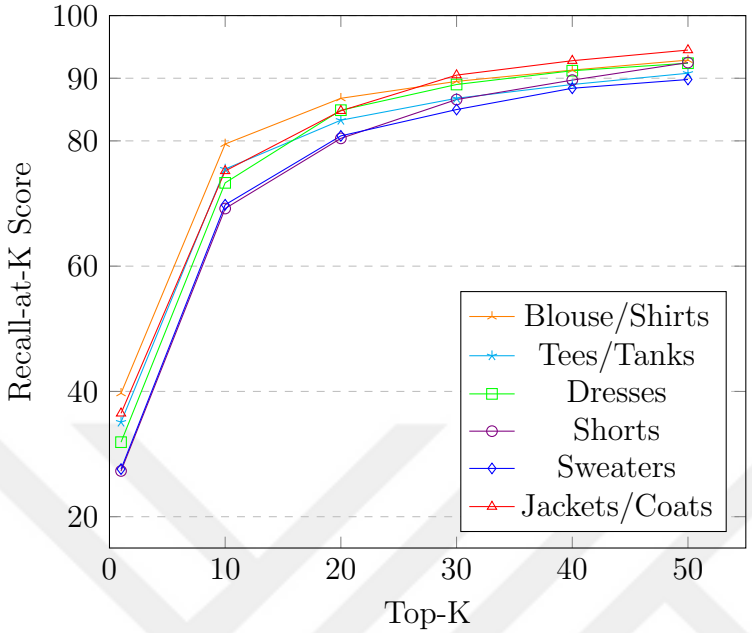
Table 9: The number of samples for each category in query set.

Category Name	**Number of Unique Items**	**Number of Total Items**
Blouse/Shirts	697	2.094
Tees/Tanks	673	2.955
Dresses	624	1.091
Shorts	246	988
Sweaters	212	735
Jackets/Coats	195	545

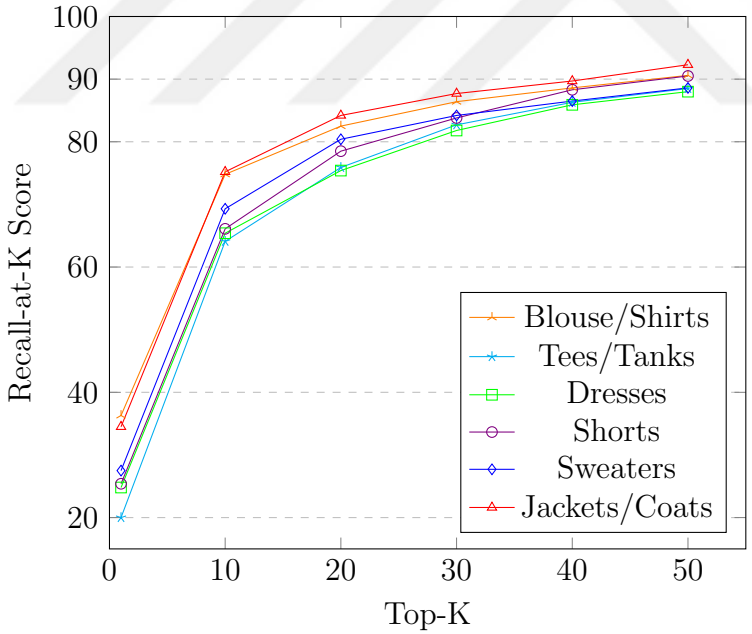
Table 10: Category-specific results of in-shop image retrieval task on DeepFashion data set. Table is ordered by sample intensity of each category.

Models	**Category**	**Top-1 (%)**	**Top-10 (%)**	**Top-20 (%)**	**Top-30 (%)**	**Top-40 (%)**	**Top-50 (%)**
SCCapsNet	Blouse/Shirts	36.3	74.8	82.5	86.4	88.6	90.6
	Tees/Tanks	20.0	64.1	75.9	82.7	86.3	88.5
	Dresses	24.8	65.4	75.4	81.8	85.9	88.0
	Shorts	25.4	66.1	78.5	83.8	88.3	90.5
	Sweaters	27.5	69.3	80.4	84.2	86.5	88.6
	Jackets/Coats	34.5	75.2	84.2	87.7	89.7	92.3
RCCapsNet	Blouse/Shirts	39.7	79.5	86.8	89.5	91.3	92.9
	Tees/Tanks	35.1	75.5	83.3	86.8	89.0	90.8
	Dresses	31.9	73.3	84.9	89.0	91.2	92.4
	Shorts	27.3	69.2	80.4	86.6	89.7	92.5
	Sweaters	27.6	69.8	80.8	85.0	88.3	89.8
	Jackets/Coats	36.5	75.2	84.8	90.5	92.8	94.5

Category-specific recall-at-K performances of SCCapsNet



Category-specific recall-at-K performances of RCCapsNet



5.2.4 Ablation Study: Category Classification

In this section, we mainly investigate the performance of our Capsule Network designs on category classification partition of DeepFashion data set [1]. To achieve this, we modify our proposed architectures slightly to adjust the overall structure for a classification task. For this task, the related partition of DeepFashion data set contains 210k training images, 40k validation images and 40k test images with 46 different categories. Moreover, this task is fine-grained category classification (i.e. 46 classes), hence the performance is measured by top-K accuracy metric, where K equals to 3 or 5.

In this design, feature extraction blocks remain same, but Capsule layers differs from our proposed architectures for some certain aspects. First, we use 32 channels of 8-dimensional fully-connected Capsules in Primary Capsule layer. In the meantime, we change the number of Capsules in Class Capsule layer to 46, instead of 23, since each category should be represented by one Capsule, intuitively. Next, the other difference is that the output of Class Capsule is normalized on dimension axis, instead of number of classes axis, so that the length of each Capsule can represent the presence of an instance for each category as a probability. Modified versions of our proposed architectures, namely SCCapsNet-CLS nad RCCapsNet-CLS are shown in Figure 21 and Figure 22, respectively.

As in proposed in [4], the classification loss for Capsule Networks is the sum of the losses of all category capsules that are calculated as separate margin loss L_k, for each category capsule k

$$L_k = T_k max(0, m^+ - \|v_k\|)^2 + \lambda_m (1 - T_k) max(0, \|v_k\| - m^-)^2 \tag{14}$$

where T_k represents the existence of the instantiation in category capsule k; and m^+, m^- and λ_m hyper-parameters that control the loss value by the existence, and set to 0.9, 0.1 and 0.5 respectively as in [4].

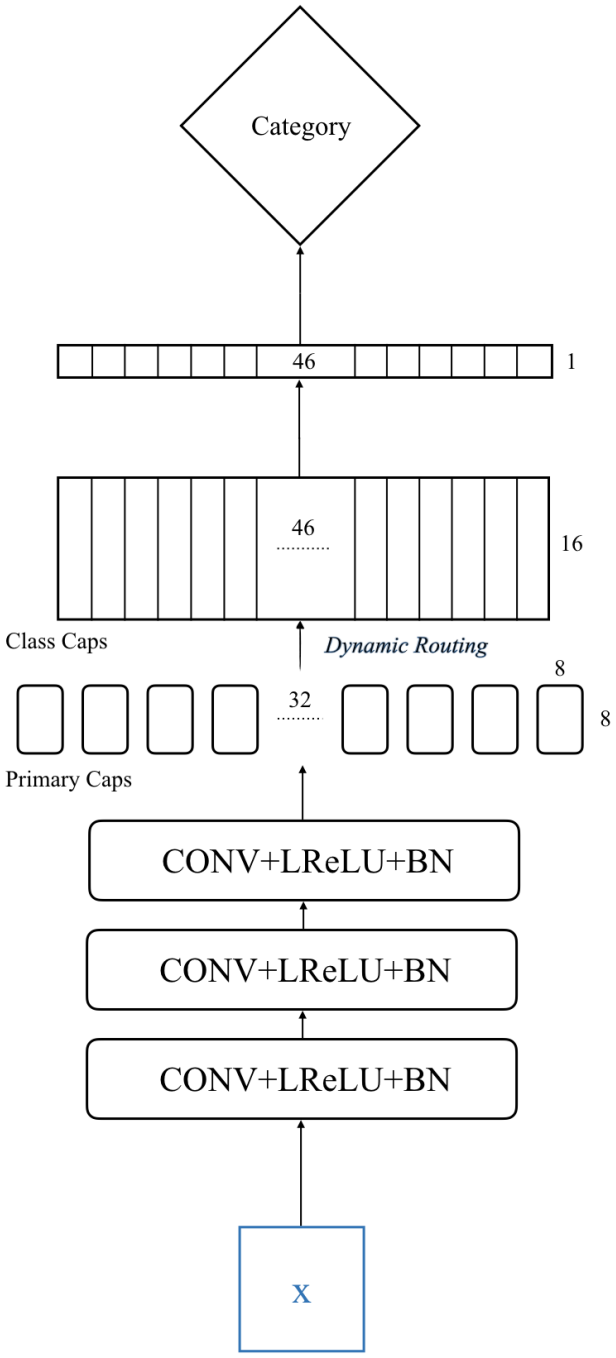


Figure 21: Stacked-convolutional Capsule Network architecture for classification task (SCCapsNet-CLS).

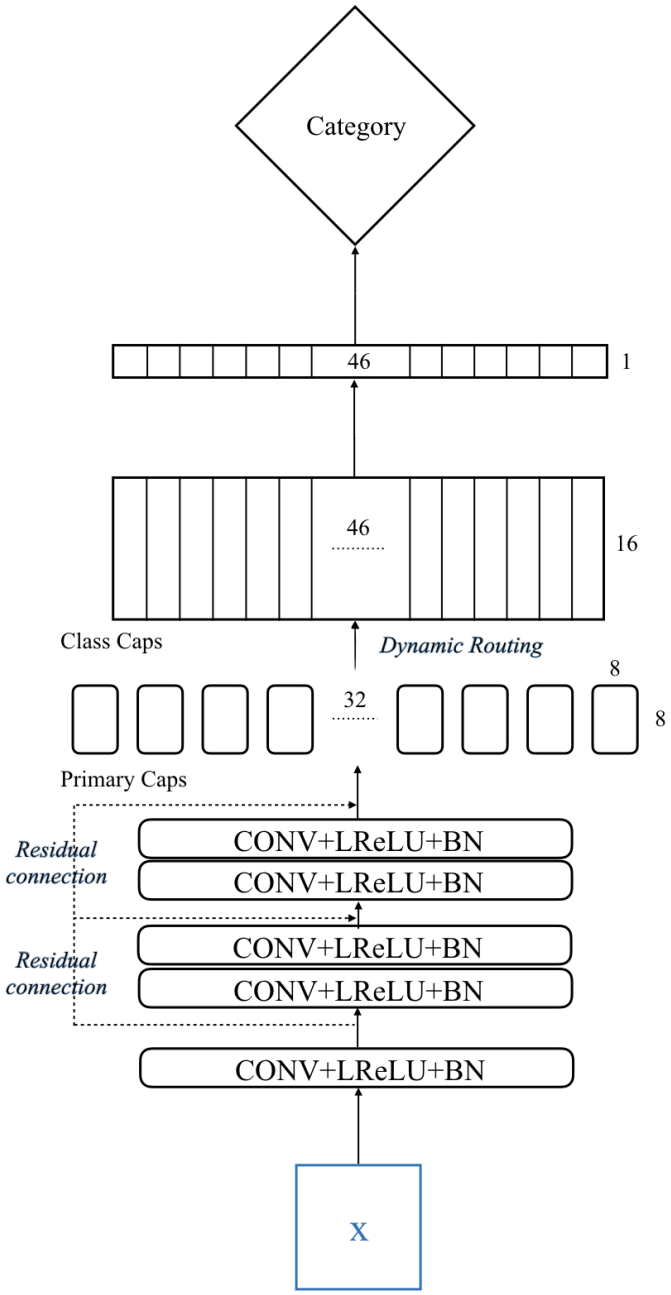


Figure 22: Residual-connected Capsule Network architecture for classification task (RCCapsNet-CLS).

Furthermore, we masked the activity vector of the correct capsule, and used it for reconstructing the images. To achieve this, the output of this design is fed into a decoder network that contains 4 transpose-convolutional layers followed by Leaky ReLU activation function [50] and batch normalization [58]. The mean-squared difference between the original images and the reconstructed ones is added to double margin loss in order to create a regularization effect [4]. Final loss L is calculated as follows:

$$L_r = \frac{1}{N}\sum_{i}^{N}(x_i - r_i) \tag{15}$$

$$L = L_k + \lambda_r L_r \tag{16}$$

In the course of the first experimental comparison, we discuss the results of our proposed Capsule Network architectures, and our baseline study FashionNet [1]. We report that SCCapsNet-CLS achieves 83.81% top-3 accuracy and 89.83% top-5 accuracy, and RCCapsNet-CLS achieves 85.12% top-3 accuracy and 91.41% top-5 accuracy on clothing category classification. As shown in Table 11, these figures demonstrate that our designs outperform all FashionNet variants including different building blocks that employ a smaller number of attributes, and use human joints or poselets instead of landmarks.

Table 11: Top-K accuracy of the variants of the baseline study [1] and our proposed models.

Models + the required side information (if any)	**Top-3 (%)**	**Top-5 (%)**
FashionNet + 100 A + L	47.38	70.57
FashionNet + 500 A + L	57.44	77.39
FashionNet + 1000 A + J	72.30	81.52
FashionNet + 1000 A + P	75.34	84.87
FashionNet + 1000 A + L	82.58	90.17
SCCapsNet-CLS *(ours)*	83.18	89.83
RCCapsNet-CLS *(ours)*	**85.12**	**91.41**

Table 12 summarizes the clothing category classification results of our proposed models and the SOTA methods with the information of applied techniques (*i.e.* backbone, side information and extra module). SCCapsNet-CLS and RCCapsNet-CLS outperform WTBI [4] and DARN [18] which both use semantic attributes disparately to improve the classification performance, but neglect pose configuration during training. Moreover, as aforementioned before, our proposed architectures and the best variant of FashionNet (i.e. supported by 1000 attributes and +8 landmarks) have closely contested classification performances on DeepFashion data set. At this point, while FashionNet employs landmark information to recover pose configuration thrown away due to pooling operations, our designs can learn pose information by preserving the hierarchical spatial relationship between pixels. However, our proposed architectures cannot achieve the performance of more advanced CNN-based architectures. The underlying reason for this is that encapsulating the neurons as a densely connected layer is completely different, but not a complex structure. On the other hand, the SOTA CNN-based methods referred in Table 12 adopt different techniques (e.g. bag-of-words descriptors, dynamic branching and attention mechanisms) to their models to improve the overall clothing classification performance. In the future, our proposed models may inherit extra performance boost on the clothing classification, due to advances in the relatively new Capsule Network research.

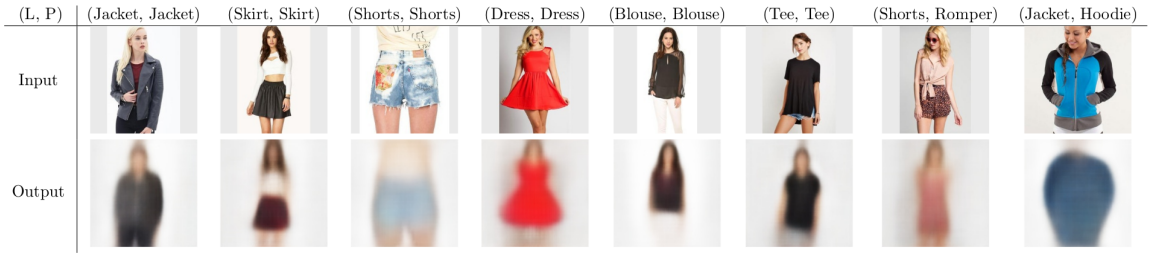


Figure 23: Examples for DeepFashion test reconstructions of RCCapsNet

Table 12: Experimental results on DeepFashion data set for the clothing category classification.

Architectures	**Backbone**	**Side Information (SI) Extra Module (EM)**	**Top-3 (%)**	**Top-5 (%)**
WTBI [25]	*AlexNet* [60]	*Category-specific Similarity (SI)*	43.73	66.26
DARN [26]	*Custom NiN* [61]	*Visual Similarity (SI)*	59.48	79.58
FashionNet [1]	*VGG-16* [45]	*Landmark Information (SI)*	82.58	90.17
SCCapsNet-CLS *(ours)*	*CapsNet* [4]	**No SI / EM Used**	83.18	89.83
RCCapsNet-CLS *(ours)*	*CapsNet* [4]	**No SI / EM Used**	85.12	91.41
Corbiére *et al.* [27]	*ResNet50* [6]	*Bag-of-words Descriptors (EM)*	86.30	92.80
Lu *et al.* [62]	*VGG-16* [45]	*Dynamic Branching (EM)*	86.72	92.51
Wang *et al.* [63]	*VGG-16* [45]	*Two Attention Modules (EM)*	90.99	95.78
Liu *et al.* [64]	*VGG-16* [45]	*Single Attention Module (EM)*	91.16	96.12

CHAPTER VI

CONCLUSION AND FUTURE WORK

In this thesis, we aim to observe the similarity learning performance of Capsule Network architecture. To achieve this, we design Triplet-based version of Capsule Networks by adjusting the original architecture proposed in [4]. Instead of employing this novel design to a toy problem, we perform some experiments on more realistic, diverse and rich data set, hence we use in-shop clothing retrieval partition of DeepFashion data set proposed in [1]. Moreover, Capsule Networks are in need of more powerful feature extraction methods to be able to perform well on this diverse data set. Therefore, we seek for more powerful feature extraction recipes than a 1-layer convolutional network for the input of Capsules. At this point, we investigate the SOTA research that combines in-shop clothing retrieval and densely-connected Capsule Networks to prove the hypotheses in Section 1.2.2.

Next, the results of our proposed architectures on in-shop clothing retrieval show that Stacked-convolutional design achieves 32.1% Top-1, 81.8% Top-20, and 90.0% Top-50 recall-at-K scores; whereas Residual-connected design gets 33.9% Top-1, 84.6% Top-20, and 92.6% Top-50 recall-at-K scores. Both of our designs outperform the earlier approaches [25, 26, 27] and all variants of FashionNet [1] by a significant margin without using any extra supportive information besides to the images. In addition, although these may be evaluated as comparable results according to the number of parameters used in the network, eventually, our designs cannot achieve such performances as in more advanced CNN-based architectures where various techniques [29, 28, 19, 20, 21] are applied to boost the overall performances.

Finally, Capsule Networks have a great potential for improving their performances, in spite of the structural limitations of densely-connected Capsules. There are some remarkable improvements [33, 43] on Capsule Networks, and in the future, our proposed models (*i.e.* SCCapsNet and RCCapsNet) may inherit extra performance boost on in-shop clothing retrieval, due to these improvements. Moreover, the techniques used in the SOTA methods of clothing retrieval task (*i.e.* different hard sampling strategies [29], more advanced objective functions [19, 20], ensembling the networks [20, 21] and employing attention-based mechanisms to the network [28, 21]) may be applied to Capsule Networks after providing that the computational burden on training process is eliminated.

APPENDIX A

SCCAPSNET MODEL SUMMARY

Table 13: Model summary (part 1) of SCCapsNet.

Layer Name	**Layer Type**	**Output shape**	**Param #**
input_2	InputLayer	(None, 256, 256, 3)	0
input_3	InputLayer	(None, 256, 256, 3)	0
input_4	InputLayer	(None, 256, 256, 3)	0
sccapsnet	Model	(None, 23, 16)	2,425,536
l2_norm_1	Lambda	(None, 23, 16)	0
input_5	InputLayer	(None, 23)	0
anchor_mask	Mask	(None, 368)	0
positive_mask	Mask	(None, 368)	0
negative_mask	Mask	(None, 368)	0
concatenate_1	Concatenate	(None, 1104)	0
Total Parameters			2,425,536
Trainable Parameters			2,425,024
Non-trainable Parameters			512

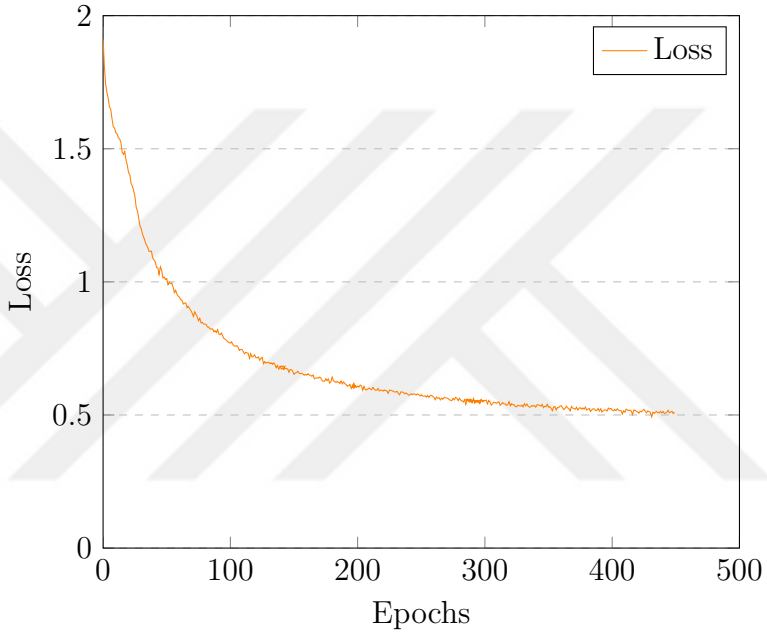
Table 14: Model summary (part 2) of SCCapsNet.

Layer Name	**Layer Type**	**Output shape**	**Param #**
input_1	InputLayer	(None, 256, 256, 3)	0
conv_block_1	Conv2D	(None, 128, 128, 64)	9,472
batch_norm_1	BatchNormalization	(None, 128, 128, 64)	256
leaky_relu_1	LeakyReLU	(None, 128, 128, 64)	0
spatial_dropout2d_1	SpatialDropout2D	(None, 128, 128, 64)	0
conv_block_2	Conv2D	(None, 64, 64, 128)	401,536
batch_norm_2	BatchNormalization	(None, 64, 64, 128)	256
leaky_relu_2	LeakyReLU	(None, 64, 64, 128)	0
spatial_dropout2d_2	SpatialDropout2D	(None, 64, 64, 128)	0
conv_block_3	Conv2D	(None, 32, 32, 64)	401,472
batch_norm_3	BatchNormalization	(None, 32, 32, 64)	256
leaky_relu_3	LeakyReLU	(None, 32, 32, 64)	0
spatial_dropout2d_3	SpatialDropout2D	(None, 32, 32, 64)	0
primarycaps_conv	Conv2D	(None, 16, 16, 512)	1,606,144
primarycaps_reshape	Reshape	(None, 8192, 16)	0
primarycaps_squash	Lambda	(None, 8192, 16)	0
fashioncaps	FashionCaps	(None, 23, 16)	5,888
Total Parameters			2,425,536
Trainable Parameters			2,425,024
Non-trainable Parameters			512

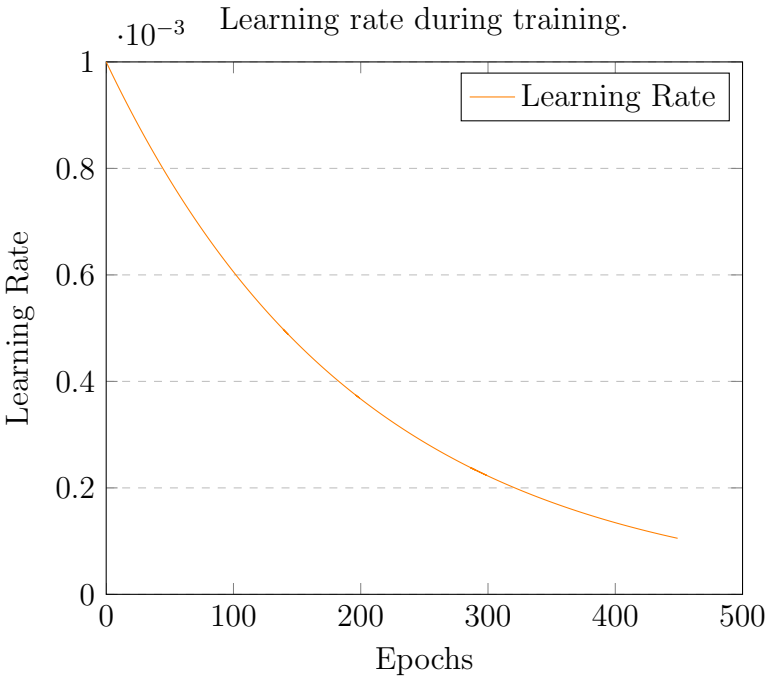
APPENDIX B

SCCAPSNET TRAINING LOGS

Total loss during training.



Learning rate during training.



APPENDIX C

RCCAPSNET MODEL SUMMARY

Table 15: Model summary (part 1) of RCCapsNet.

Layer Name	**Layer Type**	**Output shape**	**Param #**
input_2	InputLayer	(None, 256, 256, 3)	0
input_3	InputLayer	(None, 256, 256, 3)	0
input_4	InputLayer	(None, 256, 256, 3)	0
rccapsnet	Model	(None, 23, 16)	4,845,952
l2_norm_1	Lambda	(None, 23, 16)	0
input_5	InputLayer	(None, 23)	0
anchor_mask	Mask	(None, 368)	0
positive_mask	Mask	(None, 368)	0
negative_mask	Mask	(None, 368)	0
concatenate_1	Concatenate	(None, 1104)	0
Total Parameters			4,845,952
Trainable Parameters			4,840,448
Non-trainable Parameters			5,504

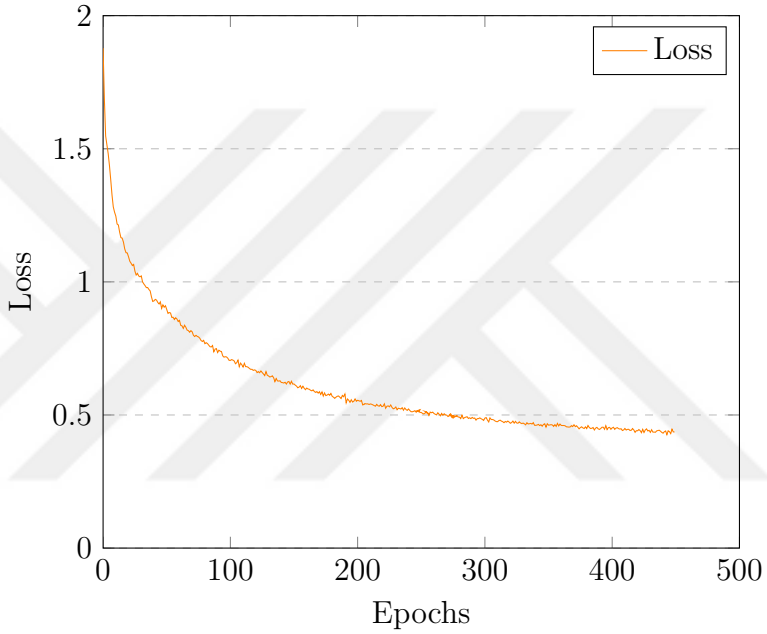
Table 16: Model summary (part 2) of RCCapsNet.

Layer Name	Layer Type	Output shape	Param #
input_1	InputLayer	(None, 256, 256, 3)	0
conv_block_1	Conv2D	(None, 128, 128, 64)	9,472
batch_norm_1	BatchNormalization	(None, 128, 128, 64)	256
relu_1	LeakyReLU	(None, 128, 128, 64)	0
spatial_dropout2d_1	SpatialDropout2D	(None, 128, 128, 64)	0
conv2d_1	Conv2D	(None, 64, 64, 128)	73,856
batch_norm_2	BatchNormalization	(None, 64, 64, 128)	512
leaky_relu_1	LeakyReLU	(None, 64, 64, 128)	0
conv2d_2	Conv2D	(None, 64, 64, 128)	147,584
conv2d_3	Conv2D	(None, 64, 64, 128)	8,328
batch_norm_3	BatchNormalization	(None, 64, 64, 128)	512
batch_norm_4	BatchNormalization	(None, 64, 64, 128)	512
add_1	Add	(None, 64, 64, 128)	0
leaky_relu_2	LeakyReLU	(None, 64, 64, 128)	0
spatial_dropout2d_2	SpatialDropout2D	(None, 64, 64, 128)	0
conv2d_4	Conv2D	(None, 32, 32, 256)	295,168
batch_norm_5	BatchNormalization	(None, 32, 32, 256)	1,024
leaky_relu_3	LeakyReLU	(None, 32, 32, 256)	0
conv2d_5	Conv2D	(None, 32, 32, 256)	590,080
conv2d_6	Conv2D	(None, 32, 32, 256)	33,024
batch_norm_6	BatchNormalization	(None, 32, 32, 256)	1,024
batch_norm_7	BatchNormalization	(None, 32, 32, 256)	1,024
add_2	Add	(None, 32, 32, 256)	0
leaky_relu_4	LeakyReLU	(None, 32, 32, 256)	0
spatial_dropout2d_3	SpatialDropout2D	(None, 32, 32, 256)	0
conv2d_7	Conv2D	(None, 16, 16, 512)	1,180,160
batch_norm_8	BatchNormalization	(None, 16, 16, 512)	2,048
leaky_relu_5	LeakyReLU	(None, 16, 16, 512)	0
conv2d_8	Conv2D	(None, 16, 16, 512)	2,359,808
conv2d_9	Conv2D	(None, 16, 16, 512)	131,584
batch_norm_9	BatchNormalization	(None, 16, 16, 512)	2,048
batch_norm_10	BatchNormalization	(None, 16, 16, 512)	2,048
add_3	Add	(None, 16, 16, 512)	0
leaky_relu_6	LeakyReLU	(None, 16, 16, 512)	0
primarycaps_reshape	Reshape	(None, 8192, 16)	0
primarycaps_squash	Lambda	(None, 8192, 16)	0
fashioncaps	FashionCaps	(None, 23, 16)	5,888
Total Parameters			4,845,952
Trainable Parameters			4,840,448
Non-trainable Parameters			5,504

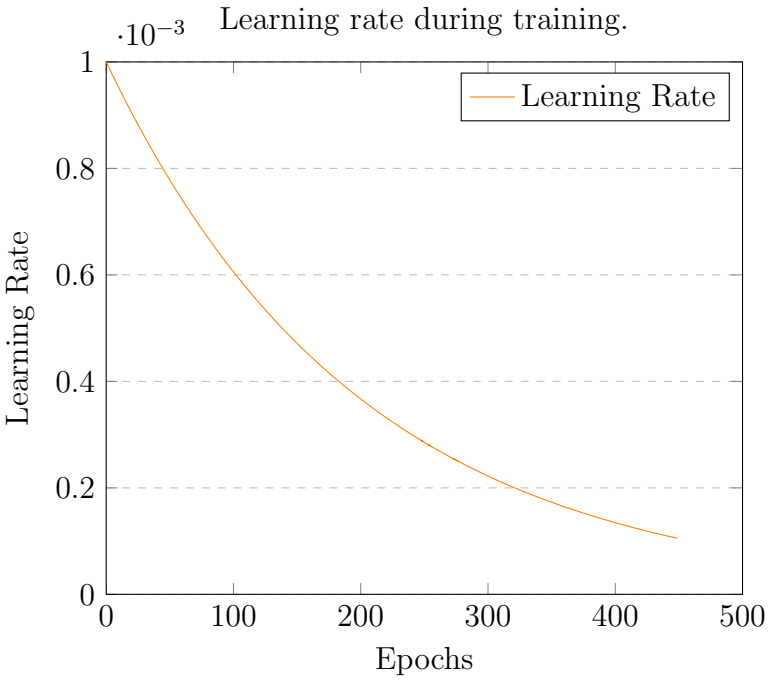
APPENDIX D

RCCAPSNET TRAINING LOGS

Total loss during training.



Learning rate during training.



Bibliography

[1] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "DeepFashion: Powering robust clothes recognition and retrieval with rich annotations," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[3] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," in *arXiv preprint arXiv:1603.07285v2*, 01 2018.

[4] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30*, pp. 3856–3866, 2017.

[5] A. Géron, "Capsule Networks (CapsNets): Tutorial," 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[7] J. Clement, "Retail e-commerce sales worldwide from 2014 to 2021," 2019.

[8] J. Clement, "Share of internet users who have purchased selected products online in the past 12 months as of 2018," 2019.

[9] Statista, "Fashion ecommerce report 2019," tech. rep., Statista Digital Market Outlook, Apr. 2019.

[10] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Proceedings of The Asian Conference on Computer Vision (ACCV)*, 2010.

[11] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[12] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4353–4361, 06 2015.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *CoRR*, vol. abs/1503.03832, 2015.

[14] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *Proceedings of The European Conference on Computer Vision (ECCV)*, vol. 9910, pp. 241–257, 10 2016.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[17] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of The European Conference on Computer Vision (ECCV)*, 2014.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, pp. 2672–2680, 2014.

[19] W. Ge, "Deep metric learning with hierarchical triplet loss," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[20] M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "BIER : Boosting independent embeddings robustly," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5199–5208, 2017.

[21] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[22] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *International Conference on Machine Learning (ICML) Deep Learning Workshop*, vol. 2, 2015.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, June 2014.

[24] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, pp. 237–254, 2016.

[25] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg, "Where to Buy It: Matching street clothing photos in online shops," pp. 3343–3351, 12 2015.

[26] J. Huang, R. Feris, Q. Chen, and S. Yan, "Cross-Domain Image Retrieval with a Dual Attribute-aware Ranking Network," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1062–1070, 2015.

[27] C. Corbière, H. Ben-younes, A. Ramé, and C. Ollion, "Leveraging weakly annotated data for fashion image retrieval and label prediction," *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 2268–2274, 2017.

[28] Z. Wang, Y. Gu, Y. Zhang, J. Zhou, and X. Gu, "Clothing retrieval with visual attention model," *2017 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, 2017.

[29] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 814–823, Oct 2017.

[30] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming Auto-encoders," in *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, ICANN'11, (Berlin, Heidelberg), pp. 44–51, Springer-Verlag, 2011.

[31] M. Yang, W. Zhao, J. Ye, Z. Lei, Z. Zhao, and S. Zhang, "Investigating capsule networks with dynamic routing for text classification," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3110–3119, Oct.-Nov. 2018.

[32] R. LaLonde and U. Bagci, "Capsules for object segmentation," in *arXiv preprint arXiv:1804.04241*, 04 2018.

[33] G. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proceedings of International Conference on Learning Representation (ICLR)*, 2018.

[34] T. Moon, "The Expectation-Maximization Algorithm," *Signal Processing Magazine, IEEE*, vol. 13, pp. 47 – 60, 12 1996.

[35] Y. LeCun, Fu Jie Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. II–104 Vol.2, June 2004.

[36] D. Rawlinson, A. Ahmed, and G. Kowadlo, "Sparse unsupervised capsules generalize better," in *arXiv preprint arXiv:1804.04241*, 04 2018.

[37] L. Zhang, M. Edraki, and G.-J. Qi, "CapProNet: Deep Feature Learning via Orthogonal Projections Onto Capsule Subspaces," in *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 5819–5828, 2018.

[38] J. E. Lenssen, M. Fey, and P. Libuschewski, "Group Equivariant Capsule Networks," in *Advances in Neural Information Processing Systems 31*, pp. 8844–8853, 2018.

[39] S. Zhang, Q. Zhou, and X. Wu, "Fast Dynamic Routing Based on Weighted Kernel Density Estimation," in *Cognitive Internet of Things*, 2018.

[40] D. Wang and Q. Liu, "An Optimization View on Dynamic Routing Between Capsules," in *ICLR*, 2018.

[41] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan, "CapsuleGAN: Generative Adversarial Capsule Network," in *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[42] S. Verma and Z.-L. Zhang, "Graph Capsule Convolutional Neural Networks," in *Joint ICML and IJCAI Workshop on Computational Biology*, (Stockholm, Sweden), 2018.

[43] A. R. Kosiorek, S. Sabour, Y. W. Teh, and G. E. Hinton, "Stacked Capsule Autoencoders," in *arXiv preprint arXiv:1906.06818*, 06 2019.

[44] F. Kınlı and F. Kıraç, "FashionCapsNet: Clothing Classification with Capsule Networks," 08 2019.

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[47] R. Girshick, "Fast R-CNN," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (Washington, DC, USA), pp. 1440–1448, IEEE Computer Society, 2015.

[48] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems 29*, pp. 4898–4906, 2016.

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, 2010.

[50] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[51] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proceedings of the International Conference on Learning Representations*, 01 2016.

[52] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving deep neural networks using softplus units," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–4, July 2015.

[53] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *CoRR*, vol. abs/1710.05941, 2017.

[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[55] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929, June 2016.

[56] F. Yu, V. Koltun, and T. A. Funkhouser, "Dilated residual networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 636–644, 2017.

[57] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L2 hypersphere embedding for face verification," in *Proceedings of the 25th ACM International Conference on Multimedia*, MM '17, (New York, NY, USA), pp. 1041–1049, ACM, 2017.

[58] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[61] M. Lin, Q. Chen, and S. Yan, "Network in network," in *2rd International Conference on Learning Representations (ICLR) 2014, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2014.

[62] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," 2016.

[63] W. Wang, Y. Xu, J. Shen, and S.-C. Zhu, "Attentive fashion grammar network for fashion landmark detection and clothing category classification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[64] J. Liu and H. lu, "Deep fashion analysis with feature map upsampling and landmark-driven attention," 09 2018.