12-4-2019 3:30 PM

A New Method to Solve Same-different Problems with Few-shot Learning

Yuanyuan Han, *The University of Western Ontario*

Supervisor: Charles Ling, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

© Yuanyuan Han 2019

Abstract

Visual learning of highly abstract concepts is often simple for humans but very challenging for machines. Same-different (SD) problems are a visual reasoning task with highly abstract concepts. Previous work has shown that SD problems are difficult to solve with standard deep learning algorithms, especially in the few-shot case, despite the ability of such algorithms to learn abstract features. In this paper, we propose a new method to solve SD problems with few training samples, in which same-different visual concepts can be recognized by examining similarities between Regions of Interest by using a same-different twins network. Our method achieves state-of-the-art results on the Synthetic Visual Reasoning Test SD tasks and outperforms several strong baselines, achieving accuracy above 95% on several tasks and above 85% on average with only 10 training samples. On a few of these challenging SD tasks, our approach even outperforms reported human performance [30]. We further evaluate the performance of our method outside of the synthetic tasks and achieve good performance on the MNIST, FashionMNIST and Face Recognition datasets.

Keywords: Few-shot Learning, Deep Learning, Same-Different, Regions of Interest

Summary for Lay Audience

In recent years, computer vision has witnessed many significant breakthroughs in standard recognition tasks such as image classification, image segmentation, or object detection. Most of these gains are a result of applying deep convolutional neural networks (CNNs). However, visual learning tasks requiring attention to highly abstract concepts such as ”sameness” and ”difference” have proven especially difficult for standard deep CNNs, although it may be simple and obvious for humans. The ability to recognize visual tasks with highly abstract concepts is a ubiquitous human skill that has not seen significant progress for the machine.

Besides, humans can learn these highly abstract visual concepts such as ”sameness” and ”difference” with little supervision. When one person only met someone once, he can remember who they are when he meets them on the street next time. In this thesis, we primarily deal with the same-different classification through few-shot learning. In particular, we try to solve SVRT same-different visual reasoning problems by using few-shot learning and then apply our model to solve more complex same-different problems in real life.

Acknowlegements

To my supervisor, Dr. Charles Ling, I owe an immense debt of gratitude for his support, mentorship, scientific insights and contagious enthusiasm during my studies. His consistent, patient confidence in me was essential in the performance of the work described in this thesis. He provides much help since I came to Canada, I believe he considerably exceeded the expectations of his role as supervisor.

I would like to thank Xuezhi, Yining, Xinxu and Tanner for their encouragement and advices during this research and the life in Canada.

I would also like to show special gratitude to my parents for their constant support throughout my research and life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we provide some background on same-different problems and discuss the importance of the research question and our contribution toward solving it. The thesis layout is introduced at the end.

1.1 Background

In recent years, computer vision has witnessed many significant breakthroughs in standard recognition tasks such as image classification [19], image segmentation [6] (the process of partitioning a digital image into multiple segments) or object detection [26]. Most of these gains are a result of applying deep convolutional neural networks (CNNs). However, visual learning tasks requiring attention to highly abstract concepts such as "sameness" and "difference" have proven especially difficult for standard deep CNNs [30, 17].

Considering the two images in Figure 1.1, the image on the left (a) could have been correctly classified as containing the cars by the deep convolutional neural network after the network was trained on thousands of images. However, similar algorithms struggle to learn the concepts of "sameness" and "difference" for the image on the right (b) even after seeing millions of training examples, although it may be simple and obvious for humans. The ability to recognize visual tasks with highly abstract concepts is a ubiquitous human skill that has not

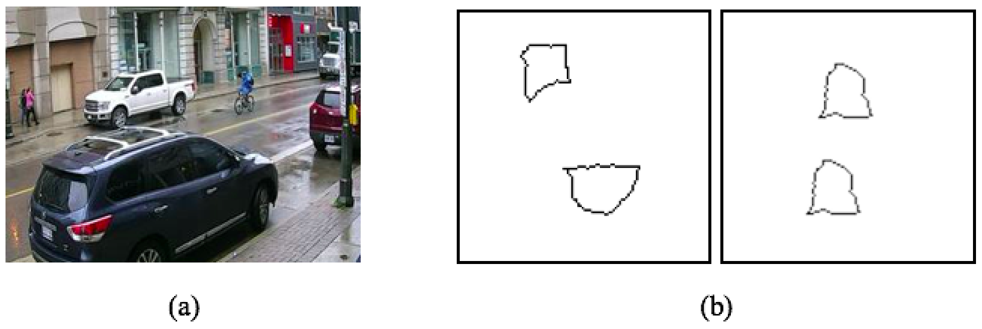


Figure 1.1: The image in (a) can be classified as containing the car by CNNs with high confidence. However, CNN fails to learn the concept of "sameness" and "difference": the first image in (b) has two different curves and the second image in (b) has two same curves. The image in (b) is from the Synthetic Visual Reasoning Test tasks [10].

seen significant progress for the machine.

The study of same-different problems has many practical implications. One of the popular applications is face similarity recognition [15]. Face recognition can help us verify identity or do some face analysis. In modern mobile devices, there are many features developed based on face recognition, such as the ability to turn on our phone or open our bank app via face recognition. Face recognition can also help us target people from many people. The research of same-different problems can significantly improve the performance of face recognition.

Besides, humans can learn these highly abstract visual concepts such as "sameness" and "difference" with little supervision. When a person meets someone once, they can usually remember who the person is when they meet on the street next time. In the Synthetic Visual Reasoning Test tasks (SVRT) [10], it has also been proven that humans can learn the concepts of "sameness" and "difference" with only few examples [30, 10]. However, for most deep learning models, they need large amounts of labeled training data to train their large number of parameters. Considering that (1) there aren't many labeled dataset in real life and much annotation is needed for new classes which is very time-consuming and labor-intensive, (2) in some cases, numerous annotated images may simply never exist for newly emerging categories such as rare animals, recognizing visual categories based on very few labeled examples will be essential and helpful.

All these motivate the study we are interested in: few-shot learning same-different tasks, which aims to recognize “sameness” and “difference” visual categories based on very few labeled examples. It would be a huge challenge to solve the same-different problems due to that: (1) Previous work shows that standard deep learning methods are not capable of solving same-different problems even when provided with millions of training samples [30, 17]. (2) The availability of only one or very few examples challenges the standard ‘fine-tuning’ practice in deep learning [9] and more easily causes over-fitting.

1.2 Research Question

In this thesis, we primarily deal with the same-different classification through few-shot learning. In particular, we try to solve SVRT same-different problems by using few-shot learning and then apply our model to solve more complex same-different problems in real life.

1.2.1 SVRT Visual Reasoning Tasks

Synthetic Visual Reasoning Test tasks include 23 tasks. The 23 SVRT tasks can be split into two groups based on the type of patterns: spatial relation (SR) problems (ex. shapes in a line vs. not in a line) and same-different (SD) problems (ex. two pairs of unique shapes vs. two pairs of identical shapes) [30]. Previous attempts on SVRT problems show deep learning approaches are capable of solving SR problems, at least when provided with plenty of training data (20K training images in [30] and 1 million used in [17]), but fail to solve SD problems even when provided with millions of training samples [30, 17].

A few examples of SD problems taken from the SVRT tasks are given in Figure 1.2. Each pair of images stands for one SVRT task, and the pair is divided into two classes: “sameness” (Class2) and “difference” (Class1). In the “sameness” class, in addition to determining “sameness” and “difference”, other highly abstract visual concepts such as“grouping”, “reflection”,

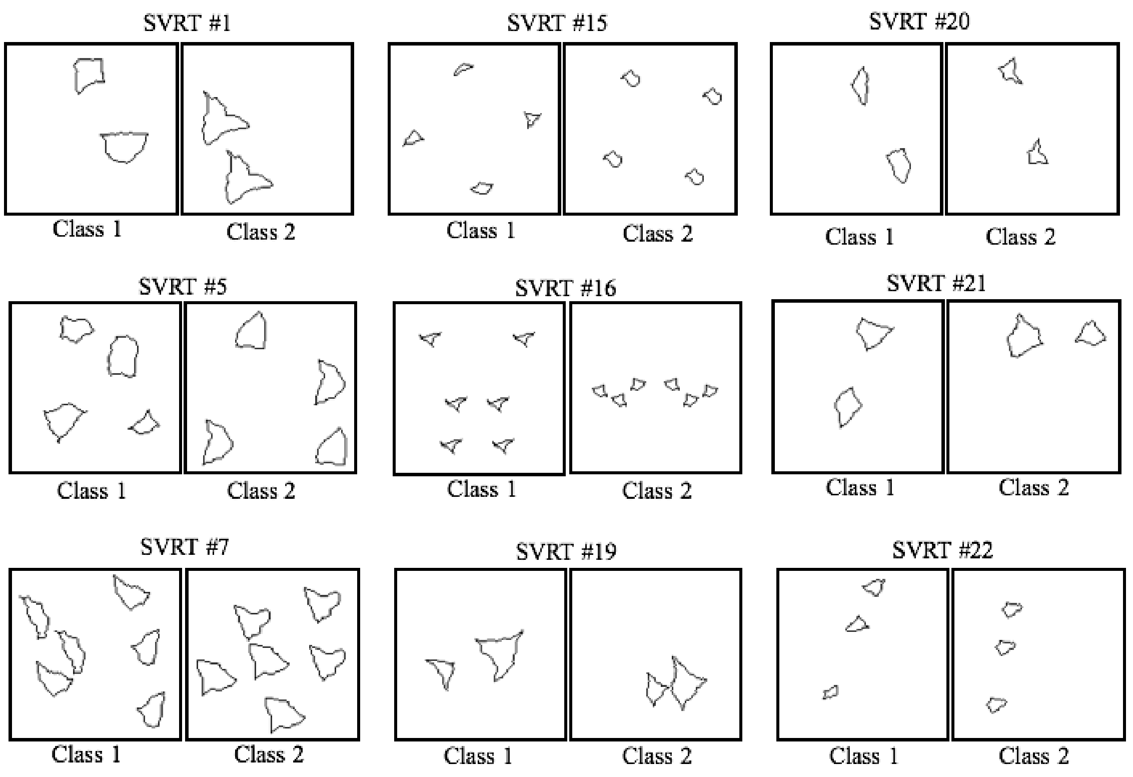


Figure 1.2: A few same-different examples from the SVRT tasks.

and "invariant on scaling and rotating" are also included. Therefore, we not only need to deal with "sameness" and "difference", but also need to solve the visual reasoning based on "sameness" and "difference". Some same-different problems in the SVRT tasks are even very difficult for human beings such as SVRT #5, #7, #16, #21 [10]. These tasks need to deal with some highly abstract visual reasoning such as"grouping", "reflection", and "invariant on scaling and rotating".

For SVRT #5, both classes contain four random objects. Class 1 contains four different objects, while class 2 contains two pairs of the same objects. For SVRT #7, both classes contain six objects. In class 1, objects can be organized into three groups, each containing two same objects. In class 2, objects can be organized into two groups, each consisting of three same objects. SVRT #16 requires the agent to decide whether shapes on the right side are copies of the shapes on the left, or whether they are vertically mirrored. SVRT #21 is a very challenging task. Each image contains two objects. One of the objects in class 2 can be obtained from the other by scaling, translating, and rotating.

Francois Fleuret et al. studied SVRT tasks. In their experiment of few-shot learning for SVRT SD problems, the prediction accuracy remained virtually at 50% for every problem [10]. Many SD tasks only have about 50% accuracy, even trained by plenty of training samples [30]. So it would be a huge challenge to solve the same-different problems with few training samples. In this thesis, we will study how to use few training samples to determine whether two objects the same or different and how to make visual reasoning based on "sameness" and "difference", so as to correctly classify the new image into its category.

1.2.2 Other Same-different Problems

We also apply our model to solve more other SD problems in real life. These problems are generated through other datasets such as MNIST, Fashion-MNIST, and Face datasets, and need to deal with some fuzzy "sameness" and visual reasoning based on "sameness" and "difference". Figure 1.3 shows several "sameness" examples for these same-different problems. Chapter 5 introduces these problems in detail and shows how we solve them by using our model. Through these applications, it has been shown that our model can be used in a wider variety of same-different problems.

1.3 Contributions

In this thesis, we proposed a new method to solve SD visual classification problems when a few training samples are provided. Our new approach is inspired by research on object detection and few-shot learning. This method is divided into three parts: (1) Regions of Interest (RoIs) are obtained through selective search method; (2) Similarities between RoIs are learned through the same-different twins network. (3) The class labels for new images are recognized based on the similarities. The advantage of our approach is that it can turn the problem of

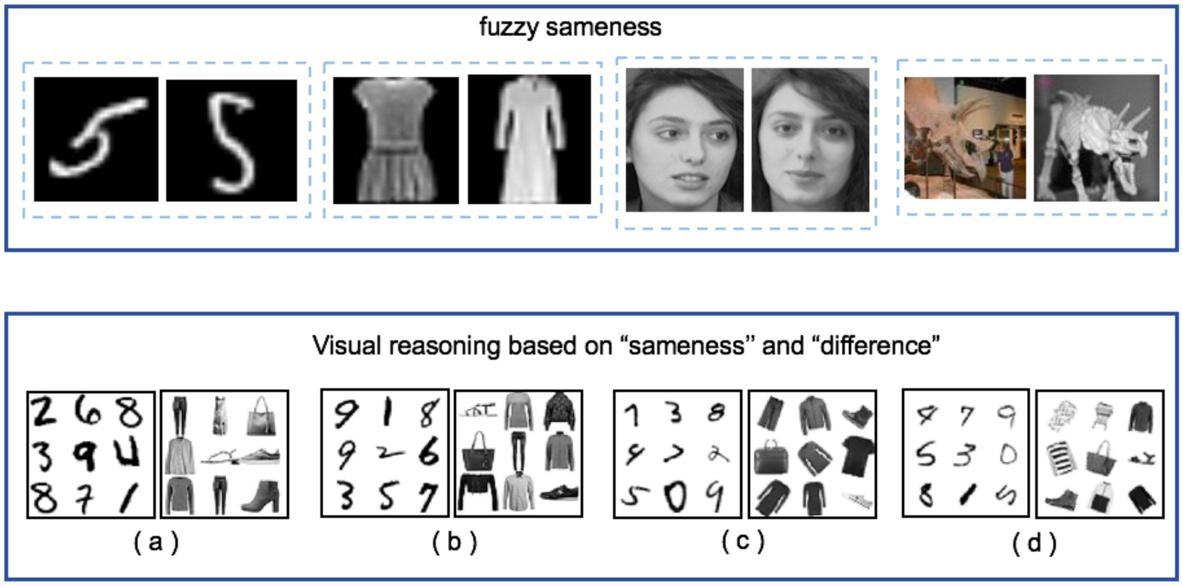


Figure 1.3: A few "sameness" examples generated by MNIST, Fashion-MNIST, miniImagenet and Face datasets. The first line of images needs to deal with fuzzy "sameness" and the second line needs to solve visual reasoning. In second line, (a) contains one exact duplicate. (b) contains one fuzzy duplicate. (c) contains one exact duplicate with some transformations (scaling and rotating). (d) contains one fuzzy duplicate with some transformations

abstract visual reasoning of "sameness" and "difference" into the problem of calculating the similarities between the Regions of Interest. Therefore, the classification tasks of "sameness" and "difference" can be solved more directly and efficiently. By using the region comparison methods, we are able to achieve accuracies above 90% on several tasks and above 85% on average with only ten training samples. Our method even surpasses reported human performance on some SD tasks such as SVRT #5, #15, and #16 [10]. We also evaluate the performance of our approach on other SD tasks generated by using MNIST, Fashion-MNIST, and face dataset. These tasks need to deal with some fuzzy "sameness" and our method also achieves good performance on these tasks.

The main contribution of this thesis is as following:

- We developed a same-different twins network that can be used to solve the same-different classification problems through one-shot or few-shot learning, as showed in Chapter

3.2.2.

- We built a novel model to solve SVRT SD problems with few-shot learning through combining regions of interest module, same-different twins network, and pattern recognition module together, achieving state-of-the-art performance on all SVRT SD problems, seeing Chapter 3.

- We benchmarked several popular few-shot learning algorithms on the SVRT SD problems and demonstrated the strength of our approach on few-shot learning of SD visual tasks, outperforming the previous state-of-the-art as well as several strong baselines, seeing Chapter 4. We believe this work will inspire future development in solving the same-different problems.

- We evaluated the performance of our method on several other SD tasks generated by using MNIST, Fashion-MNIST, and Face dataset, showing that our method could be applied to a broader variety of same-different problems, seeing Chapter 5.

1.4 Structure of this Thesis

This thesis aims to solve the same-different problems through few-shot learning. This Chapter has introduced the background and the research problems in this thesis. The rest of the thesis is organized as follows. In Chapter 2, we provide an overview of the related work, including research on SD visual reasoning tasks and few-shot learning methods and Regions of Interest selection. In Chapter 3, we describe our new approach to solving the same-different visual reasoning problems. In Chapter 4, we discuss the experiments, including the datasets that we use, baseline models, and experiment results. In Chapter 5, we use our model to solve more other SD tasks generated through other datasets such as MNIST, Fashion-MNIST, and Face dataset. Chapter 6 concludes this thesis and outlines potential future work.

Chapter 2

Related Work

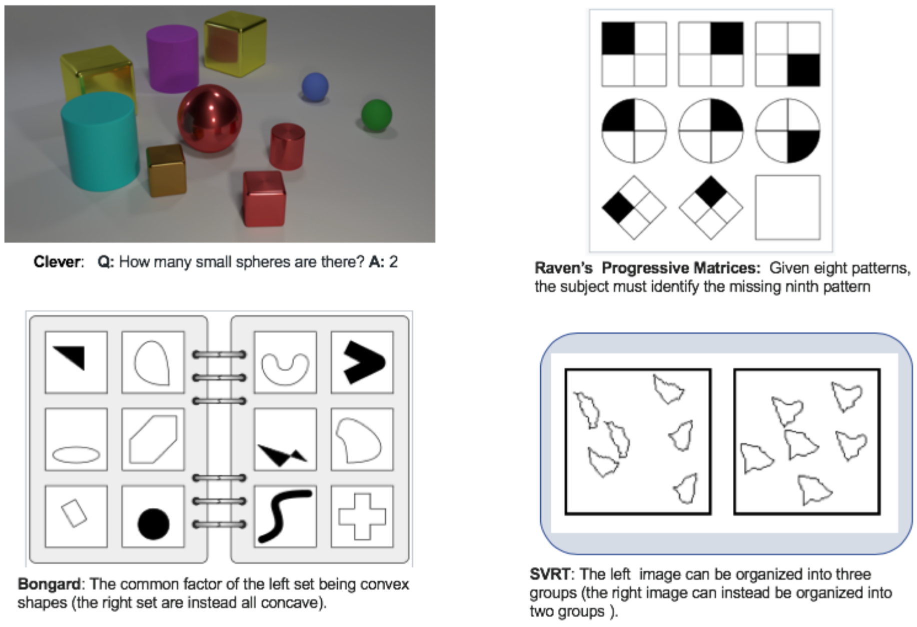
In this chapter, we will briefly review the previous work done on same-different problems, describe various few-shot learning methods in image classification, and provide an overview of approaches for selection of Regions of Interest.

2.1 Research on Same-different Problems

Same-different problems contain standard same-different classification tasks (whether two images belong to the same category) in a broad sense and same-different visual reasoning task (some highly abstract visual concepts are contained in one image).

2.1.1 Same-different Visual Reasoning Tasks

There are several potential sources of abstract visual reasoning tasks. These embody the CLEVER dataset (answer queries based on a scene with multiple objects) [16], Bongard problems (classify the new images through seeing six labeled pictures in each class) [4], Raven's Progressive Matrices (given several patterns, identify the missing pattern) [24], and Synthetic Visual Reasoning Test tasks (SVRT) (given N labeled images, classify unseen new images) [10]. Figure 2.1 shows several examples of these abstract visual reasoning tasks. Unlike Ravens Pro-



Clever: **Q:** How many small spheres are there? **A:** 2

Raven's Progressive Matrices: Given eight patterns, the subject must identify the missing ninth pattern

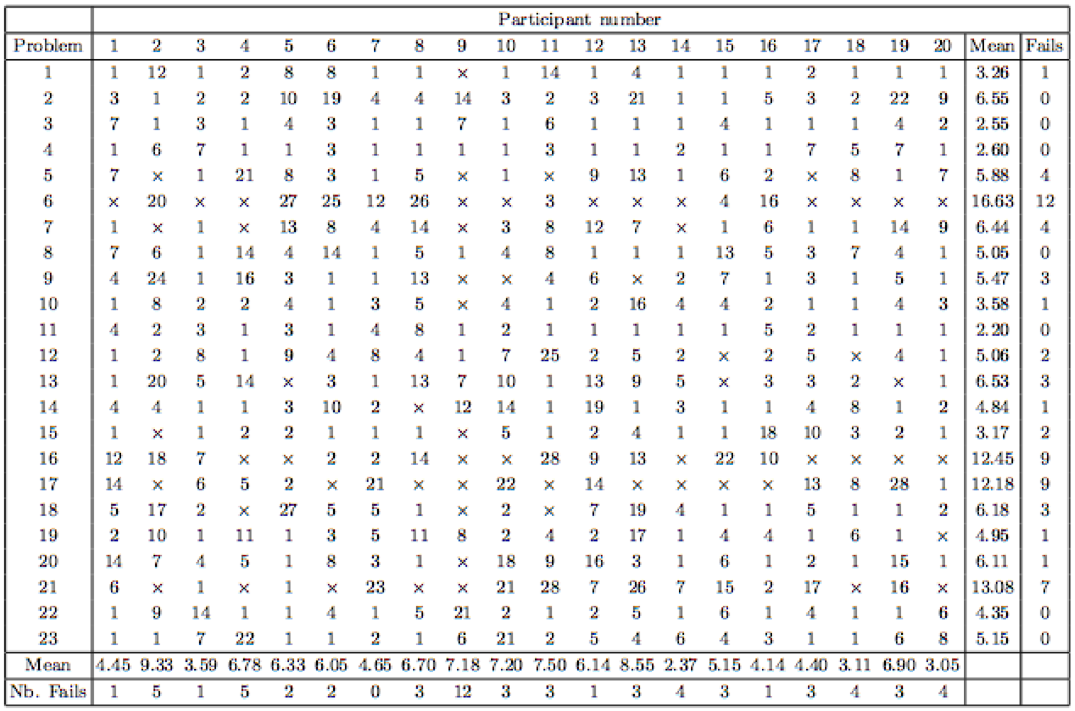
Bongard: The common factor of the left set being convex shapes (the right set are instead all concave).

SVRT: The left image can be organized into three groups (the right image can instead be organized into two groups).

Figure 2.1: Examples for several abstract visual reasoning tasks

gressive Matrices and CLEVR datasets that have seen significant progress [2, 14], SVRT tasks and Bongard problems only have seen partial success. SVRT tasks can be divided into two kinds of tasks: spatial relation tasks and same-different visual reasoning tasks. Perhaps due to the difficulty in learning to solve same-different problems in SVRT tasks, most work in this same-different visual reasoning task has focused on examining how and why machine learning approaches have failed.

Francois Fleuret et al. [10] first introduced the SVRT tasks and showed that humans are much more proficient at these tasks than the presented machine learning approaches. They used SVRT tasks to compare the efficiency in binary image classification between human and machine learning and demonstrated that the SVRT tasks were easy to spot and characterize for humans, but those tasks were challenging to learn for generic machine learning systems. Figure 2.2 shows the human performance on SVRT tasks tested by Fleuret et al [10]. 20 people who participated in Fleuret's experiments would be randomly given one image from the two classes. What they needed to do was to identify to which class the image belonged. When he

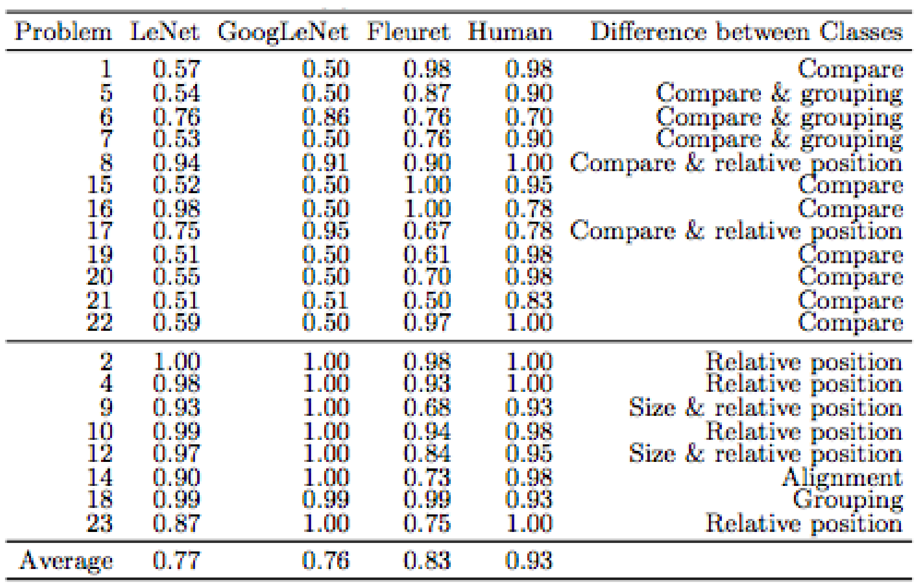


	Participant number																					
Problem	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Mean	Fails
1	1	12	1	2	8	8	1	1	×	1	14	1	4	1	1	1	2	1	1	1	3.26	1
2	3	1	2	2	10	19	4	4	14	3	2	3	21	1	1	5	3	2	22	9	6.55	0
3	7	1	3	1	4	3	1	1	7	1	6	1	1	1	4	1	1	1	4	2	2.55	0
4	1	6	7	1	1	3	1	1	1	1	3	1	1	2	1	1	7	5	7	1	2.60	0
5	7	×	1	21	8	3	1	5	×	1	×	9	13	1	6	2	×	8	1	7	5.88	4
6	×	20	×	×	27	25	12	26	×	×	3	×	×	×	4	16	×	×	×	×	16.63	12
7	1	×	1	×	13	8	4	14	×	3	8	12	7	×	1	6	1	1	14	9	6.44	4
8	7	6	1	14	4	14	1	5	1	4	8	1	1	1	13	5	3	7	4	1	5.05	0
9	4	24	1	16	3	1	1	13	×	×	4	6	×	2	7	1	3	1	5	1	5.47	3
10	1	8	2	2	4	1	3	5	×	4	1	2	16	4	4	2	1	1	4	3	3.58	1
11	4	2	3	1	3	1	4	8	1	2	1	1	1	1	1	5	2	1	1	1	2.20	0
12	1	2	8	1	9	4	8	4	1	7	25	2	5	2	×	2	5	×	4	1	5.06	2
13	1	20	5	14	×	3	1	13	7	10	1	13	9	5	×	3	3	2	×	1	6.53	3
14	4	4	1	1	3	10	2	×	12	14	1	19	1	3	1	1	4	8	1	2	4.84	1
15	1	×	1	2	2	1	1	1	×	5	1	2	4	1	1	18	10	3	2	1	3.17	2
16	12	18	7	×	×	2	2	14	×	×	28	9	13	×	22	10	×	×	×	×	12.45	9
17	14	×	6	5	2	×	21	×	×	22	×	14	×	×	×	×	13	8	28	1	12.18	9
18	5	17	2	×	27	5	5	1	×	2	×	7	19	4	1	1	5	1	1	2	6.18	3
19	2	10	1	11	1	3	5	11	8	2	4	2	17	1	4	4	1	6	1	×	4.95	1
20	14	7	4	5	1	8	3	1	×	18	9	16	3	1	6	1	2	1	15	1	6.11	1
21	6	×	1	×	1	×	23	×	×	21	28	7	26	7	15	2	17	×	16	×	13.08	7
22	1	9	14	1	1	4	1	5	21	2	1	2	5	1	6	1	4	1	1	6	4.35	0
23	1	1	7	22	1	1	2	1	6	21	2	5	4	6	4	3	1	1	6	8	5.15	0
Mean	4.45	9.33	3.59	6.78	6.33	6.05	4.65	6.70	7.18	7.20	7.50	6.14	8.55	2.37	5.15	4.14	4.40	3.11	6.90	3.05		
Nb. Fails	1	5	1	5	2	2	0	3	12	3	3	1	3	4	3	1	3	4	3	4		

Figure 2.2: Completed experimental results with humans (taken from the source: Comparing machines and humans on a visual categorization test, Fleuret et al. [10]). Each row corresponds to one of the 23 problems and each column to one of the 20 participants. Each cell contains the number of attempts before seven consecutive correct categorizations were made. Entries containing "X" indicate that the participant failed to solve the problem, and those cells are not included in the marginal means [10].

correctly identified the class 7 times in a row, it proved that he/she learned this rule, and when 35 consecutive failures occurred to the classification, it proved that he/she failed to learn the rule.

For machine learning methods, Fleuret et al. [10] paired three different feature groups with two classification algorithms, and found that the best performing pair combined Fourier and wavelet coefficients derived from the images with Adaboost [11]. However, in their few-shot experiments (with only ten labeled examples of each problem), the performance on most of the 23 tasks was roughly random, with accuracy about 50%. With a large number of training samples, the difficulty between same-different and spatial relation tasks became more apparent. Given 10,000 labeled images, their best model being was able to obtain 81% accuracy



Problem	LeNet	GoogLeNet	Fleuret	Human	Difference between Classes
1	0.57	0.50	0.98	0.98	Compare
5	0.54	0.50	0.87	0.90	Compare & grouping
6	0.76	0.86	0.76	0.70	Compare & grouping
7	0.53	0.50	0.76	0.90	Compare & grouping
8	0.94	0.91	0.90	1.00	Compare & relative position
15	0.52	0.50	1.00	0.95	Compare
16	0.98	0.50	1.00	0.78	Compare
17	0.75	0.95	0.67	0.78	Compare & relative position
19	0.51	0.50	0.61	0.98	Compare
20	0.55	0.50	0.70	0.98	Compare
21	0.51	0.51	0.50	0.83	Compare
22	0.59	0.50	0.97	1.00	Compare
2	1.00	1.00	0.98	1.00	Relative position
4	0.98	1.00	0.93	1.00	Relative position
9	0.93	1.00	0.68	0.93	Size & relative position
10	0.99	1.00	0.94	0.98	Relative position
12	0.97	1.00	0.84	0.95	Size & relative position
14	0.90	1.00	0.73	0.98	Alignment
18	0.99	0.99	0.99	0.93	Grouping
23	0.87	1.00	0.75	1.00	Relative position
Average	0.77	0.76	0.83	0.93	

Figure 2.3: Experiment results on the SVRT classification tasks (taken from the source: 25 years of cnns: Can we compare to human abstraction capabilities?, Stabinger et al. [30])

on average on same-different tasks and 88% accuracy on average on spatial relation tasks. Some same-different problems could not be solved with even 10,000 training examples with a performance of around 60%.

A unique approach to solving SVRT tasks is presented by Kevin Ellis et al. [7]. They introduced an unsupervised learning algorithm to solve SVRT tasks. Their method synthesized programs from data. First, their model parsed the images into symbolic forms by locating distinct shapes (SVRT tasks are particularly amenable to this). Then, using a grammar tuned for the task, their algorithm searched over a space of drawing programs, looking for those that best reproduce the original image over the space. Once a program had been found for each training image, a classifier was trained to map drawing programs to class labels, where it demonstrated strong performance. The algorithm could learn how abstract structures are represented in vision and achieve excellent performance on SVRT tasks. However, this approach is quite slow, and some images may take hundreds of seconds to synthesize the program.

When studying the SVRT problems with more modern computer vision approaches, Sebastian Stabinger et al. [30] used 20000 SVRT training images per class to train LeNet [21]

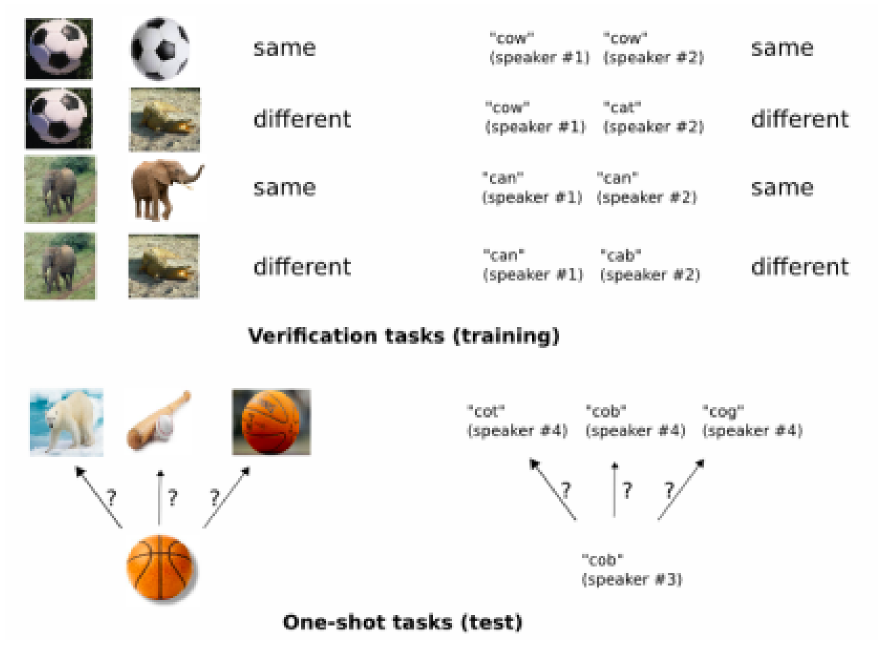

same
different
same
different
"cow" (speaker #1) "cow" (speaker #2) same
"cow" (speaker #1) "cat" (speaker #2) different
"can" (speaker #1) "can" (speaker #2) same
"can" (speaker #1) "cab" (speaker #2) different
Verification tasks (training)
"cot" (speaker #4) "cob" (speaker #4) "cog" (speaker #4)
?
"cob" (speaker #3)
One-shot tasks (test)

Figure 2.4: General strategy for siames neural network(taken from the source: Siamese neural networks for one-shot image recognition, Koch et al. [18]).

and GoogLeNet [32] CNNS. Figure 2.3 shows the experimental results on the SVRT classification tasks from Stabinger et al. They found that near-perfect performance was achievable on roughly half of the tasks, with the other half being significantly more difficult (near-random). By observing the abstract concepts required to solve each SVRT task, they noticed that the easy-difficult split closely corresponded to whether or not tasks required same-different comparisons, with a couple of exceptions (they determined that a couple of same-different problems could be solved by exploiting simple pixel distribution patterns). The authors found that both CNNs perform very similarly, working well on spatial relation problems and not seem to have made much progress on same-different problems, despite training each network on 20,000 images per class. Many same-different tasks only achieved about 50% accuracy even when CNNs were trained by plenty of training sets. The general inability for CNNs to learn to solve SD tasks is also supported by [17], where a more systematic examination of the performance of an array of CNN architectures was performed.

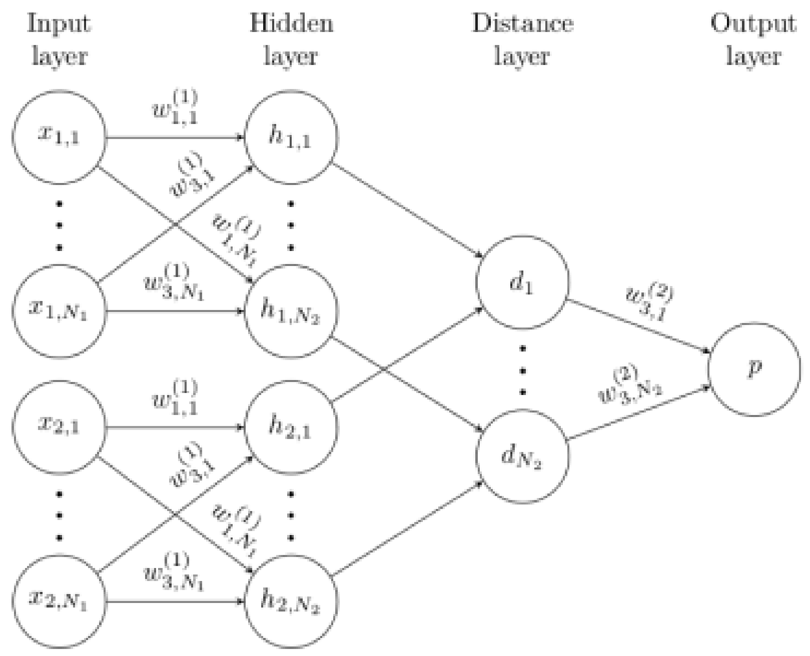


Figure 2.5: A sample 2 hidden layer siamese network for binary classification with logistic prediction *p* (taken from the source: Siamese neural networks for one-shot image recognition, Koch et al. [18]).

2.1.2 Standard Same-different Classification Tasks

Koch et al. [18] built a Convolutional Siamese Neural Network to solve same-different image recognition problems by one-shot learning. Figure 2.4 shows the general strategy for Siamese Neural Network. The author first trained a model to discriminate between a collection of same/different pairs, and then made the model generalize to evaluate new categories based on learned feature mappings for verification. A Siamese Neural Network consists of twin networks that accept distinct inputs but are joined by an energy function at the top. This function computes some metric between the highest-level feature representation on each side, as shown in Figure 2.5. In this thesis, we developed our same-different twins network based on the Siamese Neural Network by changing the feature embedding layer and distance metric layer.

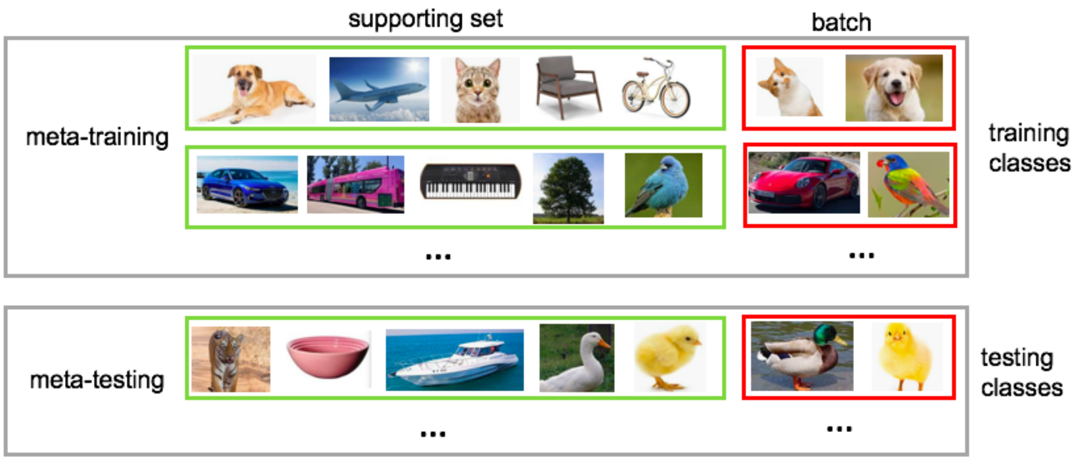


Figure 2.6: An example for meta-learning (learn to learn) based few-shot learning method.

2.2 Few-shot Learning Methods

The study of few-shot learning has become a hot research direction for some time. Few-shot classification [22, 20, 18] is a task in which a classifier must be trained to recognize new classes not seen in the training samples when few training examples are given. A simple method, such as re-training the model on the new training samples, would lead to severe over-fitting. While the problem is quite tricky for the machine, it is straightforward for human beings. It has been demonstrated that humans can perform even one-shot classification where only a single example of each new class is given with high accuracy [20].

Earlier work on few-shot learning mainly introduced generative models by using some complex strategies such as probabilistic models based on the Bayesian approach [8, 20]. With the success of deep learning on large-scale data sets [19, 12, 28], many people contributed to generalizing deep learning-based approaches to solve few-shot learning problems. Many of those approaches use a meta-learning or learning-to-learn strategy, which means that they extract some transferable knowledge from previous tasks or some auxiliary tasks, such as transfer-learning method. This transferable knowledge can help to learn the target few-shot problems well without suffering from the over-fitting that often occurs when applying deep learning models to solve sparse data problems.

Meta-learning can be divided into two phases: meta-training phase and meta-test phase. In the meta-training phase, the training data sets are decomposed into different meta tasks, which help the model to learn the generalization ability in the case of category change. If a meta-task contains C distinct classes and K examples in each class, the target few-shot problem is called C-way K-shot. In the meta-test phase, classification for the new class can be completed without changing the existing model. Figure 2.6 shows an example for meta-learning (learn to learn) based few-shot learning method. There are mainly three kinds of meta-learning models in modern few-shot learning methods: Fine-Tune based, RNN based, and Embedding and Metric Learning based.

2.2.1 Few-shot Learning based on Fine-Tuning

Deep CNNs have recently shown outstanding image classification performance in large-scale visual recognition challenges. However, to train a CNN requires a large number of labeled image samples because CNN has millions of parameters that need to be estimated. For few-shot learning tasks, it is obviously not feasible. A solution is to use a pre-trained CNN as a feature extractor [23]. Transfer Learning is the reuse of a pre-trained model on a new problem. When solving a new problem, most of the layers in front of the model are frozen, and we only need to fine-tune the last few layers. It is currently popular in the field of deep learning because it enables us to train deep neural networks with comparatively little data.

C. Finn et al. [9] proposed Model-Agnostic Meta-Learning (MAML) approach, which aimed to train a given neural network model on a variety of learning tasks to obtain the initial weight configuration, such that it can solve new tasks using only few training samples. The strategy here is to meta-learn an initial set of neural network weights (the parameters of the model), so the neural network models can be effectively fine-tuned to produce excellent generalization performance on the new few-shot learning tasks within few gradient-descent update steps. S Ravi et al. [25] further proposed the few-shot optimization method, which used an LSTM-based meta-learner model. The method not only can obtain a good initial neural net-

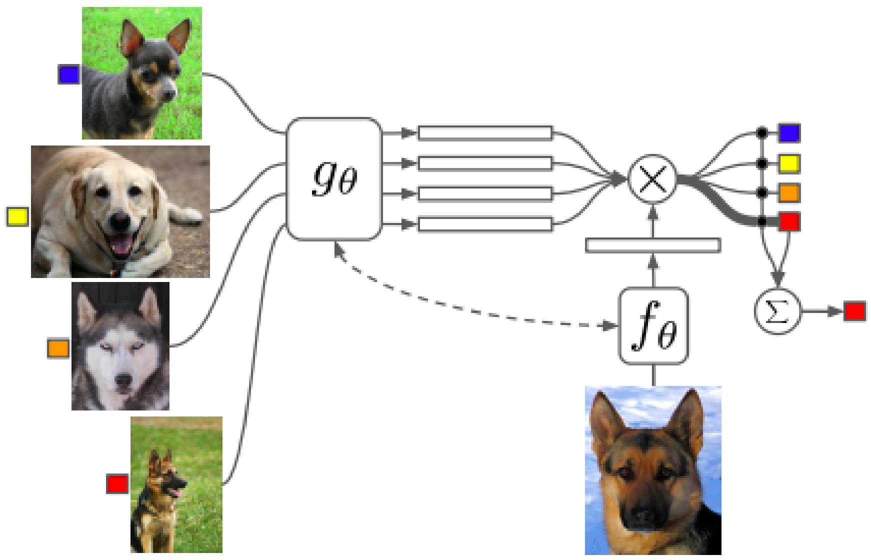


Figure 2.7: Matching Networks architecture(taken from the source: Matching networks for one shot learning, Vinyals et al. [34])
.

work weights but an LSTM-based optimizer. The optimizer can be trained to be specifically effective for fine-tuning. The biggest problem for the fine-tune based meta-learning methods is that these approaches need to fine-tune on the target problems.

2.2.2 Few-shot Learning based on RNN Memory

Adam Santoro et al. proposed meta-learning with memory-augmented neural (MANN) networks [27] that uses recurrent neural networks (RNN) with memories. Meta-learning means that the weights of the RNN are trained by learning many distinct meta tasks. The strategy here is to use RNN to accumulate the knowledge in its hidden activations or external memory through iterating on the meta-training examples. The stored knowledge can help to solve the target few-shot problems. When classifying new classes, it can be done by comparing them with historical information stored in the memory. The main challenge for MANN is to ensure that all the relevant historical information has been reliably stored in the RNN without forgetting.

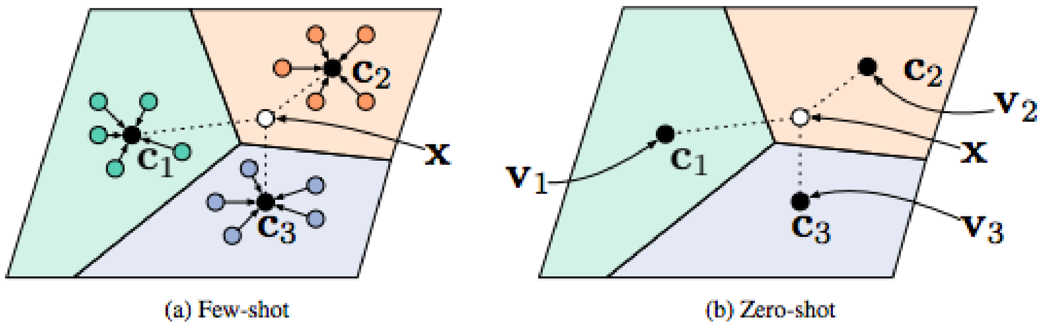


Figure 2.8: Prototypical Networks in the few-shot and zero-shot (no labeled example) scenarios (taken from the source: Prototypical networks for few-shot learning, Jake Snell et al. [29]).

2.2.3 Few-shot Learning based on Embedding and Metric Learning

Another kind of popular meta-learning based few-shot learning method is the embedding and metric learning approach. This method aims to learn a set of projection functions that take C-way K-shot tasks from the target problem and classify new images in a feed-forward manner [34, 29, 3]. The meta-learning here is to train a metric net through learning many distinct meta tasks. The metric net will learn how to parameterize a classifier to classify the new classes in terms of the sparse support training set. Metric-learning based approaches aim to learn a set of projection functions such that when images are represented in this embedding, they are easy to recognize using simple nearest neighbor or linear classifiers [34, 29, 18]. In this case, the meta-learned transferable knowledge is the projection function, and the target few-shot problem is a simple feed-forward computation. Matching Networks [34], Prototypical Networks [29], Relation Network [31], and Convolutional Siamese Network [18] belong to embedding and metric learning approach.

Vinyals et al. [34] proposed Matching Networks, which uses an attention mechanism over the embeddings of the labeled training sets (support set) to predict new classes for the unlabeled test sets (query set). Figure 2.7 shows the Matching Networks architecture. The architecture learns a network that maps a small labeled supporting set and an unlabeled example to its label, obviating the need for fine-tuning to adapt to new class types.

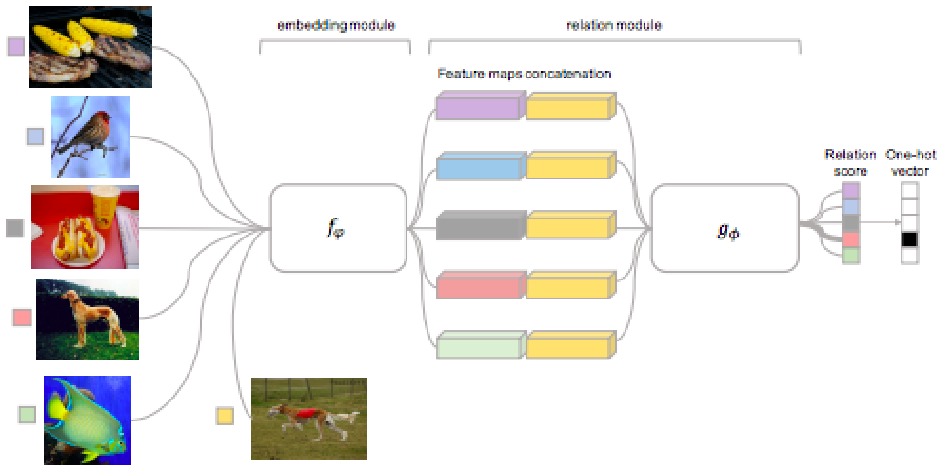


Figure 2.9: Relation Network architecture for a 5-way 1-shot problem with one query example (taken from the source: Learning to compare: Relation network for few-shot learning, Flood Sung et al. [31]).

Jake Snell et al. [29] proposed the Prototypical Networks, which is based on the idea that each class can be represented by the mean of its supporting examples in a representation space learned by a neural network. When one query comes, its classification can be performed by computing distances to prototype representations of each class. Figure 2.8 shows the Prototypical networks in the few-shot and zero-shot scenarios.

Flood Sung et al. [31] proposed the Relation Network, which is designed to learn a deep distance metric to conduct few-shot learning within episodes. In each episode, the network is designed to simulate the few-shot setting. After training, the relation network is able to classify images of new classes by computing relation scores between query images and the few supporting examples of each new class. Figure 2.9 shows the relation network architecture for a 5-way 1-shot problem with one query example.

The most related methods to ours are the Relation Networks [31] and the Convolutional Siamese Networks [18]. These approaches focus on learning embeddings in which images are represented, such that it can be easily recognized using a fixed nearest-neighbor [29] or linear classifier [29, 18]. Our method combines the Convolutional Siamese Network and Relation

Network, replacing the fixed metric on the top of the Convolutional Siamese Network for the relation classifier CNN in Relation Network. Therefore, our method can provide a learnable rather than fixed metric, or non-linear rather than linear classifier.

2.3 Regions of Interest Selection

Regions of Interest selection is also called object proposal in some papers. It has been widely used in the object detection area. Comprehensive surveys and comparisons about RoIs selection methods can be found in these papers [13, 5]. To sum up, Regions of Interest selection methods could be divided into two approaches: one method is based on grouping superpixels (e.g., selective search [33]). Another method is based on sliding windows (e.g., objectness in windows [1], EdgeBoxes [35]). Many object proposal methods, like selective search, have been used as an independent detector module. In this paper, we use selective search method to obtain the Regions of Interest because selective search that is used as an independent detector is straightforward and effective.

Chapter 3

Methodology

In this chapter, we explain our approach for few-shot learning same-different visual problems in detail. Firstly, we analyze the problem that we need to solve and define it as one-shot learning in a broad sense, and then we describe our model and each module in the model in detail. Lastly, we briefly introduce how our model learns on training samples.

3.1 Problem Definition

We try to solve the same-different problems with few-shot learning. According to the previous research on few-shot learning, we need to use three datasets: a training set, a supporting set, and a testing set. The support set and testing set share the same label space. They are from the target few-shot problems. With the support set only, we can, in principle, train a classifier to predict a class label y for each sample x in the test set. However, due to the lack of labeled samples in the support set, it will cause over-fitting, and the performance of such a classifier is usually not satisfactory. Therefore, we need to use the training set to perform the meta-training tasks in advance. The training set has its own label space that is different from the support and testing set, and is mainly used to perform the meta-training such that transferable knowledge can be learned to help to perform better few-shot learning on the support set and classify the test set more effectively. When the support set contains C distinct classes and

K examples in each class, the target few-shot problem is called C-way K-shot. Considering SVRT SD problems consisting of randomly generated objects in each class of each task (e.g. for SVRT #1, although Class 2 contains two same objects in each image, the same objects are different among the images, as shown in Figure 3.1), we need to learn the similarity between the objects for each image when solving this problem. Therefore, we can define the problem we are studying as one-shot learning, which means that the machine needs to predict whether another object is the same as this object that machine only saw once.

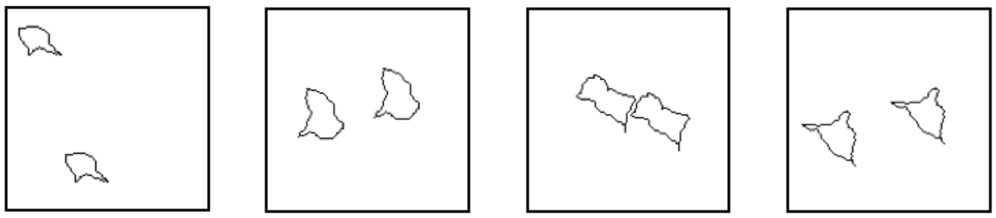


Figure 3.1: A few samples in "sameness" class for SVRT #1

3.2 Model

In order to solve the SVRT SD problems, we built a novel model. Our model consists of three parts: 1) Regions of Interest selection module. 2) same-different twins network. 3) pattern recognition module. Figure 3.2 illustrates the simplified process in solving the SVRT same-different problems through our approach. Firstly, samples x_i are entered into the regions of interest selection module, resulting in a set of regions of interest (r_i, r_j,...). Then randomly grouping these regions of interest into pairs and feeding each pair into the same-different twins network. Same-different twins network is developed based on the Convolutional Siamese Neural Network [18] through replacing the fixed $L1$ distance metric on the top of the siamese network to a learnable CNN metric. Same-different twins network contains two modules: an embedding module f_φ and a relation module g_ϕ. Each pair such as r_i and r_j is fed into the embedding module f_φ, which produces feature maps $f_\varphi(r_i)$ and $f_\varphi(r_j)$. The feature maps $f_\varphi(r_i)$ and $f_\varphi(r_j)$ are combined with operator $C(f_\varphi(r_i), f_\varphi(r_j))$. In this work, we assume $C(.,.)$ to be the concatenation of feature maps in depth. The combined feature map of the pair is fed into the relation module g_ϕ, which eventually produces a scalar in range of 0 to 1 representing the similarity for each pair such as r_i and r_j, which is called relation score. The relation score between r_i and r_j is $g_\phi(C(f_\varphi(r_i), f_\varphi(r_j)))$. After that, we use the pattern recognition module to determine which class samples x_i belongs to based on these relation scores. The pattern recognition module includes a filter module p and a k-nearest classifier. The filter module p sets the relation scores below the threshold to 0 and then adds all theses relation scores to obtain a similarity value. Finally, we use a k-nearest neighbor classifier to predict the class label based on the similarity value for each test image.

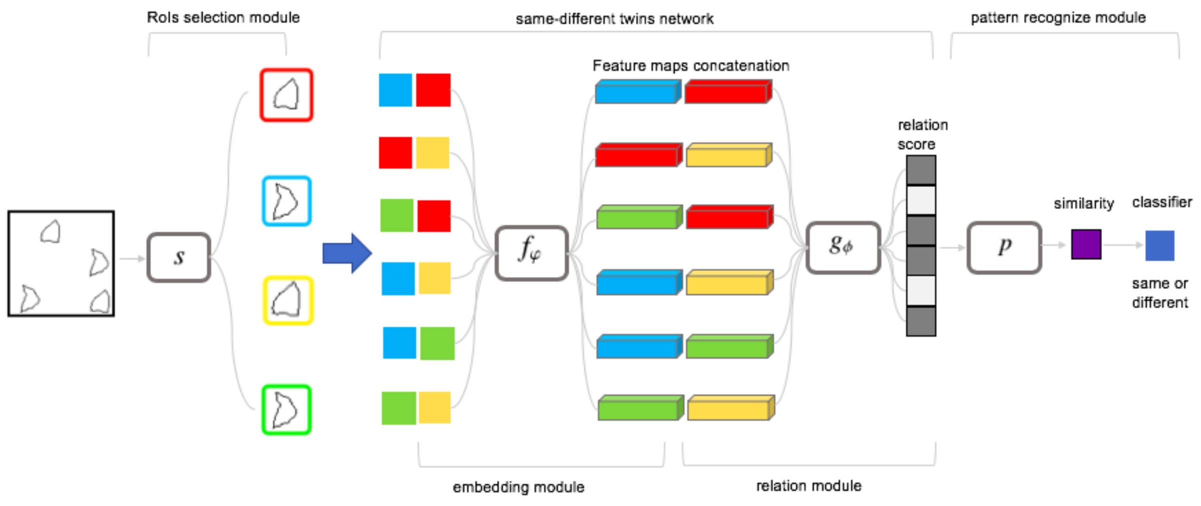


Figure 3.2: The simplified process in solving the SVRT SD problems through our approach

3.2.1 Regions of Interest Selection Module

To locate the objects in the image, we use the selective search method [33] to obtain Regions of Interest and other methods are also possible such as the simple border detection using

Python-OpenCV. Selective search combines the advantages of both an exhaustive search and segmentation, and it can capture all possible object locations with high recall. Selective search has been developed as an independent detector. There are three hyper-parameters in the selective search method: scale, sigma and minSize. Objects can be effectively located through configuring reasonable parameters for the hyper-parameters sigma, scale, minSize. In all detection results, some detection results are inaccurate and highly repetitive. In order to remove duplicate regions and improve detection accuracy, we calculate the Intersection-over-Union (IoU) among regions. In mathematics, IoU is a statistic used for gauging the similarity and diversity of sample sets. In the field of the image, IoU is a standard for measuring the detection accuracy of a corresponding object. The method of calculating IoU in the field of the image is showed in Figure 3.3. It means that the area where the two detection regions overlap is divided by the total area. We set a threshold to remove the inaccurate detection results for SVRT SD problems. When the IoU between two regions is greater than the threshold, the region with the smaller area is removed. Each bounding box is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height h and width w.

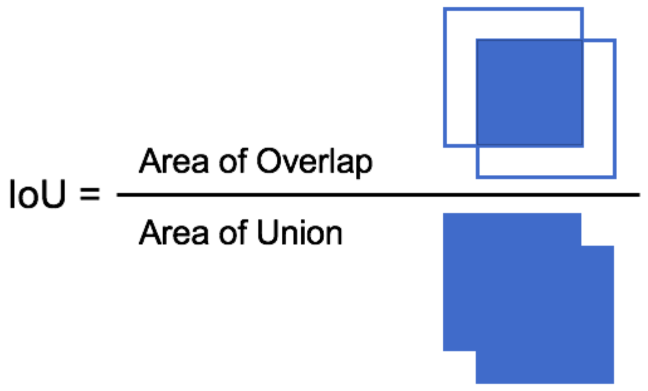


Figure 3.3: The method of calculating IoU in the field of image

3.2.2 Same-different Twins Network

Same-different twins network is developed based on the Convolutional Siamese Neural network [18]. As we have known, metric-learning based approaches aim to learn a set of projection functions such that when images are represented in the embedding space, they are easy to recognize using simple nearest neighbor or linear classifiers [34, 29, 18]. In the Convolutional Siamese Neural network [18], the authors used the $L1$ distance metric to compute the distance between the highest level feature representations of each pair. In this work, we use a learnable CNN metric called relation module to replace the $L1$ distance metric. Compared with $L1$ distance that focuses on learning a shallow (linear) Mahalanobis metric for fixed feature representation, the CNN metric can learn a deep non-linear metric (similarity function) that can better identify matching/mismatching pairs. This has been shown in [31]. We also developed a new embedding module that can accommodate the 84 x 84 SVRT images.

Figure 3.4 shows the convolutional architecture for the embedding module. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in [28]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. We developed a VGG-like architecture for the embedding module. The embedding module consists of a sequence of convolutional layers and max-pooling layers. The convolutional layers have filters of size three and a fixed stride of 1. The number of convolutional filters is specified as a multiple of 16, which refers to the Convolutional Siamese Neural network [18]. This setting can optimize the performance of extracting features. The embedding applies a rectified linear units (ReLU) activation function to the outputted feature maps. Each set of convolutional layers is followed by a 2x2 max-pooling layer with the stride of 2.

After obtaining the feature maps of the twin pairs through the embedding module, the feature maps are concatenated in-depth and inputted into the relation module. Figure 3.5 shows the relation module for few-shot learning same-different problems. The relation module consists of two convolutional blocks and two fully-connected layers. Each convolutional block con-

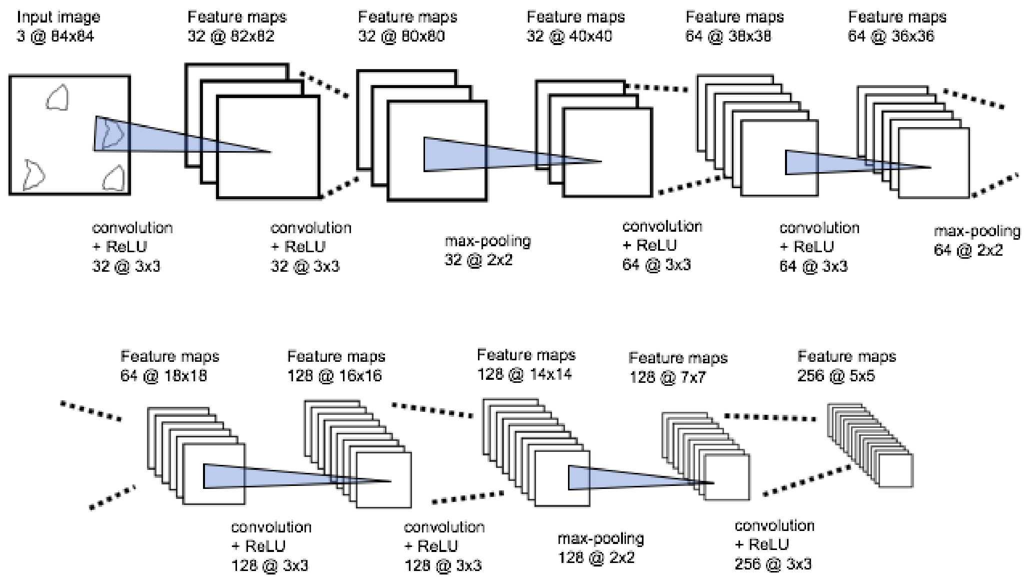


Figure 3.4: Convolutional architecture for the embedding module

tains a convolution layer with 256 filters followed by batch normalization, ReLU non-linearity activation function, and a 2x2 max-pooling layer. Each filter in the convolution layer has the kernel size of 3x3. Two convolutional blocks are followed by two fully-connected layers. The two fully-connected layers are 64 and 1 dimensional, respectively. The first fully-connected layer uses the ReLU activation function, and the output layer uses the Sigmoid activation function (logistic function) in order to generate relation scores in a reasonable range for each twin pair. Between the two fully connected layers, we added a dropout layer that helps alleviate over-fitting.

3.2.3 Pattern Recognition Module

Some SVRT tasks contain more than two objects and need to solve the highly abstract concept of “grouping” such as SVRT #5 and SVRT #7. For SVRT #7, objects in the positive class can be organized into three groups, each containing two same objects. Objects in the negative class can be organized into two groups, each containing three same objects. Therefore, in addition

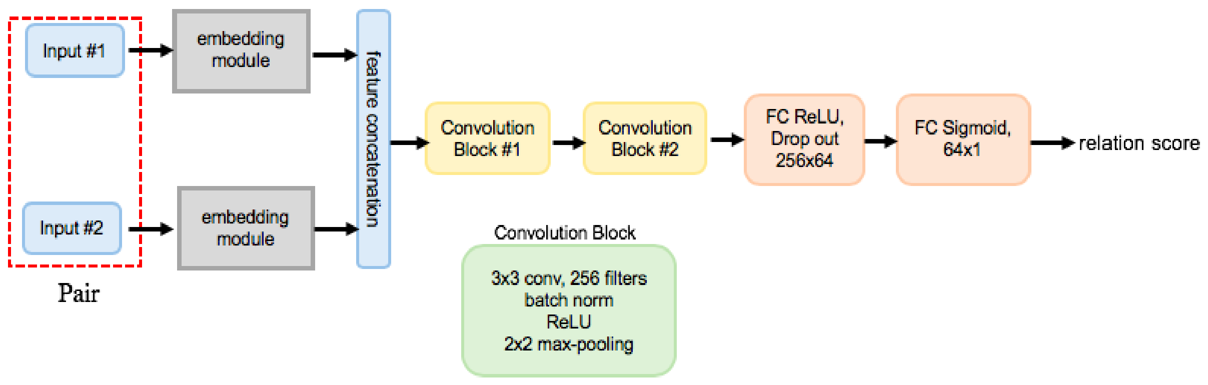


Figure 3.5: Relation module for few-shot learning SD problems

to identifying the sameness and difference for each pair, we also need to learn the pattern in each class. As discussed, through a same-different twins network, we can obtain the relation score for each pair and finally get all relation scores for all pairs. Based on the relation scores, a simple method to recognize the pattern for each class is to get the sum of these relation scores. However, due to that SVRT images are too simple, just composed of lines, many pairs that belong to different classes also have very high relation scores. Therefore, we need to use a threshold, which is obtained through training set and validation set, to distinguish the relation scores. When relation score is below the threshold, this pair has a high probability of belonging to a different class, and we set the relation score of this pair to 0. After that, we obtain the sum of all the relation score as the similarity value. In this way, we can use simple nearest neighbor or linear classifiers to recognize the class due to that the similarity is one-dimensional and can reflect the pattern for each class. Finally, we train a k-nearest neighbor classifier to predict class labels based on the similarities.

3.3 Learning

Loss function. Let i indexes the ith mini-batch (the number of training data during one iteration). Suppose the min-batch size is m. Let $y(x_1^i, x_2^i)$ represents a vector with m length. This vector refers to the true label of the mini-batch. When x_1 and x_2 belong to the same category, $y(x_1^i, x_2^i) = 1$, otherwise $y(x_1^i, x_2^i) = 0$. Let vector $p(x_1^i, x_2^i)$ with m length represents the predicted results for the mini-batch. We used the binary cross-entropy as the loss function. Binary cross-entropy can be calculated as:

$$L(x_1^i, x_2^i) = -y(x_1^i, x_2^i)logp(x_1^i, x_2^i) - (1 - y(x_1^i, x_2^i))log(1 - p(x_1^i, x_2^i)) \quad (3.1)$$

Optimization. Backpropagation is an algorithm widely used in the training of feedforward neural networks for supervised learning. In this work, we also use the standard backpropagation algorithm to optimize our object function by calculating the gradient. We set the mini-batch size to 128 with learning rate α_j, momentum β_j and regulation weights λ_j, so the update rule at epoch T is as follows:

$$w_{kj}^T(x_1^i, x_2^i) = w_{kj}^T + \Delta w_{kj}^T(x_1^i, x_2^i) + 2\lambda_j|w_{kj}| \quad (3.2)$$

$$\Delta w_{kj}^T(x_1^i, x_2^i) = -\alpha_j \nabla w_{kj}^T + \beta_j \Delta w_{kj}^{T-1} \quad (3.3)$$

where ∇w_{kj} is partial derivative with respect to the weight between the jth neuron in some layer and the kth neuron in the successive layer.

Weight initialization. Before the training, we initialized the weights and biases for all convolution layers and fully-connected layers. For all convolution layers, the weights were drawn from a normal distribution with zero mean and a standard deviation of 10^{-2}. The biases were initialized from a normal distribution with the mean 0.5 and a standard deviation of 10^{-2}. The fully-connected layers were initialized in the same way as the convolution layer, but the weights were initialized using a normal distribution with zero mean and a standard deviation of $2x10^{-1}$.

Learning rate. In this work, we set the initial learning rate to α, and the learning rate is gradually decreased following the method of dividing by 10. For example, assume that the initial learning rate is 0.06, it is gradually decreased according to 0.006, 0.0006, 0.00006, ... We monitored the one-shot validation error when training the same-different twins network. When the validation accuracy did not increase for 50 iterations, we stopped and used the parameters of the model at the best iteration step according to the one-shot validation accuracy. Then we updated the learning rate to continue to train the network.

Hyperparameter optimization. For the learning rate and regularization hyperparameters, we set the initial learning rate $\alpha \epsilon [10^{-2}, 10^{-1}]$, momentum $\beta \epsilon [0, 1]$, and regulation weights $\lambda \epsilon [0, 0.1]$. For neural network hyperparameters, we set the number of convolution filters in each layer from 16 to 256 using multiples of 16. For the fully connected layer hyperparameters, we set the units from 16 to 256, also in multiples of 16. These hyperparameters are obtained through maximizing the accuracy of one-shot tasks in the validation sets.

Chapter 4

Experiments

In this chapter, we first discuss the experimental setup, including the datasets that we used and the performance evaluation method, several strong baselines, and some experiment implementation details. Then, we present the experimental results of our approach and compare them against the strong baselines. After that, we analyze the experimental results in detail.

4.1 Datasets

We use the meta-learning (learn to learn) strategy to solve the same-different problems with few-shot learning. According to the previous research on meta-learning, we need to use two kinds of datasets. The first data set is used for meta-training to extract some transferable knowledge from some auxiliary tasks such that it can help the model to learn the generalization ability. The second data set is the dataset for meta-testing, which is the SVRT problems that we currently need to solve.

4.1.1 Meta-training datasets

In this thesis, we use the Omniglot dataset as the meta-training dataset to extract some transferable knowledge. Omniglot dataset was collected by Brenden Lake and his collaborators [20]

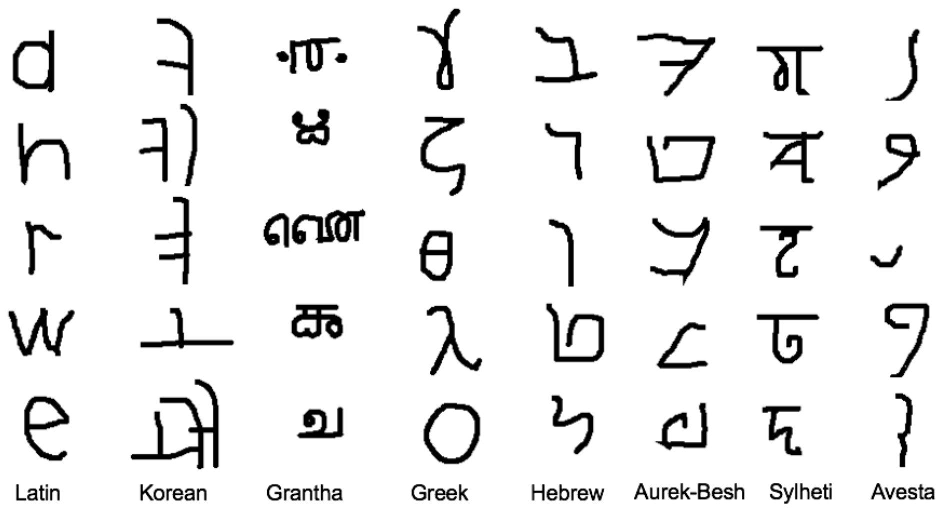


Figure 4.1: The Omniglot dataset contains different images from alphabets across the world.

and was used for studying one-shot learning and for developing more human-like learning algorithms. Omniglot contains over 1600 handwritten characters from 50 different alphabets ranging from familiar Latin to unfamiliar local dialects. Figure 4.1 shows a few examples from the Omniglot dataset. Omniglot dataset can be obtained from the github [1] or downloaded from www.omniglot.com.

The number of characters (classes) in each alphabet varies considerably from about 15 to upwards of 40 characters. Each class contains 20 samples drawn by different people. Lake split the 50 different alphabets into a 40 alphabet background set and a 10 alphabet evaluation set. The background set is used for developing the same-different twins network by learning hyper-parameters and feature mappings. The evaluation set is used to test the one-shot classification performance of the network. Each image in Omniglot dataset is a 105x105 binary-valued image which was drawn by hand on an online canvas. In order to transfer the developed twins network to solve the SVRT problems, we resize the Omniglot dataset to 84x84 to keep consistent with SVRT tasks. We also change the binary-valued image to the RGB image due to images in SVRT tasks have an RGB image format.

Besides, considering that each class only contains 20 samples, we also augmented the back-

[1] https://github.com/brendenlake/omniglot

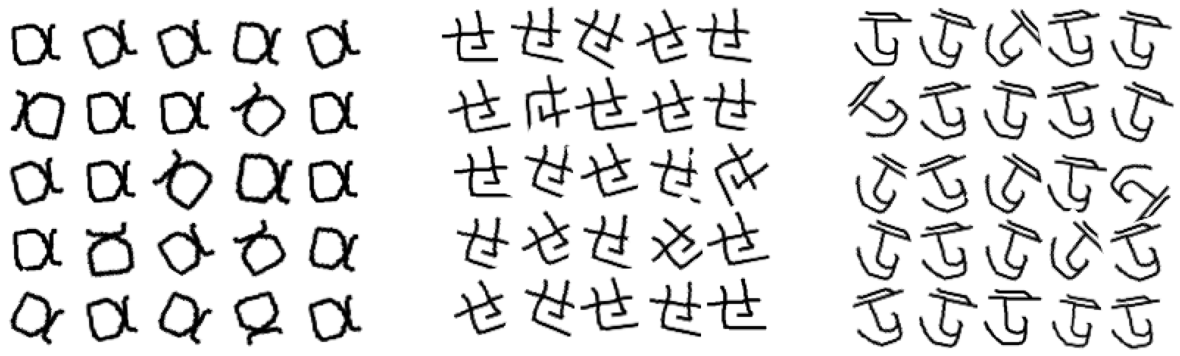


Figure 4.2: A few samples of random data augmentation generated for three images in the Omniglot data set

ground set (training set) with the data augmentation, which is a strategy that can significantly increase the diversity of data available for training models, without actually collecting new data. The data augmentation techniques include scaling, rotation, horizontal flipping, and translation. Figure 4.2 shows a few samples of random data augmentation generated for three images in the Omniglot data set.

4.1.2 SVRT datasets (meta-testing datasets)

In this thesis, we try to solve the SVRT same-different visual reasoning problems. This series of synthetic image recognition problems is developed by Fleuret and Don Geman. The code to generate these SVRT tasks is publicly available at [2]. Figure 1.2 in Chapter 1 shows a few examples of the SVRT SD problems. For each task, we use ten labeled training samples, 1000 validation sets, and 10000 test samples in order to compare our method with the published results by Fleuret et al. [10] where the authors used ten training samples. We also augmented the training sets with small affine distortions. The affine distortions include scaling and rotation. The augmented training sets are mainly for training the pattern recognition module.

[2] http://www.idiap.ch/~fleuret/svrt/

4.2 Performance Evaluation Method

There are various metrics that can be used to evaluate machine learning algorithms. These metrics include classification accuracy, F1 Score, confusion matrix, and so on. In this thesis, we use the classification accuracy to evaluate the performance of our model. Classification accuracy is the ratio of the number of correct predictions to the total number of input samples. For every SVRT SD task, we conducted ten testing trials and used 1000 test samples for each test trial (including 500 positive samples and 500 negative samples). The final performance is the mean of classification accuracy for ten testing trials.

4.3 Implementation Details for Our Method

We solved the SVRT SD problems in three steps. The first step is to select the Regions of Interest by the selective search. The second step is to meta-training the same-different twins network. The last step is to use the trained same-different twins network to do one-shot learning and predict the relation score for each pair of regions of interest. After that, the pattern recognition module is used to do the classification task for SVRT same-different problems based on the predicted relation scores.

For the selection of Regions of Interest, we set the hyper-parameters scale to 1000, sigma to 0.3, and min_size to 10, which helps to locate the objects effectively. For the IoU, we set the threshold to 0.9, which means that when the IoU between two regions of Interest is greater than 0.9, the region with the smaller area is removed.

For the same-different twins network, we used the Omniglot dataset for meta-training, which helped the same-different twins network learn some transferable knowledge. Firstly, randomly grouping the Omniglot background set into pairs, and then these pairs were fed into the same-different twins network to train this network. Hyper-parameters follow the setting below: batch size 128, Adam optimizer (the initial learning rate 0.06, momentum 0.6, regulation weights 0.02). Validation tasks were set to 20-way one-shot following the Siamese network [18], and we set 1000 validation tasks for each evaluation to help us select the optimal parameters. The network with the maximum classification accuracy was used to predict the similarity of Regions of Interest on SVRT SD tasks. The same-different twins network was implemented with PyTorch.

Finally, the pattern recognition module was used to predict the class for new SVRT SD samples based on the similarities of Regions of Interest. Firstly, the pattern recognition module filtered the relation scores and set the relation scores below the threshold to 0. The threshold was learned through the SVRT SD validation set, selecting the value which had maximum classification accuracy on the validation set. After that, the similarity value for each image was obtained by adding the relation scores of all these pairs of regions of interest. Then we trained a k-nearest neighbors classifier to predict class labels for each new SVRT image based on the similarities. The value for k was chosen from {1,2,3,4,5} and the best value was found to be $k = 2$.

4.4 Baselines

Apart from comparing to the few-shot learning results for SVRT same-different tasks obtained by Fleuret et al. [10], we also compare our method against several other approaches known to perform well at other few-shot classification tasks, including fine-tune deep CNNs, Model-Agnostic Meta-Learning (MAML) [9], Prototypical Nets [29], and Relation Nets [31]. For all these models, we used SVRT images of size 84x84, ten training samples,1000 validation samples, and 10000 test samples. Performance is measured with classification accuracy. To produce all experimental results, we average across ten trials, with 1000 different unseen test images each trial. For hyper-parameter tuning, we used 1000 validation images for every task.

In addition, for meta-learning methods Model-Agnostic Meta-Learning (MAML) [9], Prototypical Nets [29], and Relation Nets [31], we also used Omniglot datasets to do the meta-training to extract the transferable knowledge in keeping with our approach and used SVRT

SD tasks in the meta-testing phase. Due to that each SVRT task is a binary classification question and has 10 training samples (5 positive, 5 negative), we defined the meta-learning tasks as 2-way 5-shot. The 2-way 5-shot meta-training tasks were generated through Omniglot datasets. 2-way 5-shot meta-testing tasks were also generated by SVRT SD problems(10 training samples, 10000 testing samples).

4.4.1 Fine-tune deep CNNs

Fine-tune deep CNNs is mainly the reuse of a pre-trained model on a new problem. When solving a new problem, most of the layers in front of the model are frozen, and we only need to fine-tune the last few layers. For this method, we fine-tuned the Vgg-16 architecture pre-trained on ImageNet [28]. Other architectures are also possible such as ResNet, InceptionV3, Xception. The fully connected layers in Vgg-16 architecture were replaced by two fully-connected layers with width 256 and 64 activated by sigmoid function. The softmax classifier was trained with the Adam optimizer provided by Keras, and the loss function used binary cross-entropy loss. We trained the deep CNNs for 200 epochs, and the optimal number of epochs for each SVRT same-different task was chosen based on the classification accuracy on validation set. The optimal number of epochs for each task was chosen from $\{8, 16, 32, 64\}$.

4.4.2 Model-Agnostic Meta-Learning

MAML aims to meta-learn an initial set of neural network weights (the parameters of the model are explicitly trained), so the neural network models can be effectively fine-tuned to produce good generalization performance on the new few-shot learning tasks within few gradient-descent update step. We used the same architecture for this approach, as in Finn's article. The implement process is publicly available in [3]. 2-way 5-shot meta-training tasks are generated through Omniglot datasets. We set 60000 meta-train iterations and used 1000 validation images for each task. Finally, we selected the model that maximized the prediction accuracy on

[3] https://github.com/cbfinn/maml

SVRT SD validation images to test the 10000 different unseen SVRT SD images for each task for ten trials (1000 test samples for one trial).

4.4.3 Prototypical Network

Prototypical network is developed based on the idea that each class can be represented by the mean of its supporting examples in a representation space learned by a neural network on the meta-training tasks. We used the same architecture for this approach, as in Snell's article [29]. The implement process is publicly available in [4]. Omniglot datasets were used to do the 2-way 5-shot meta-learning task for the Prototypical network. Ten training samples in SVRT tasks were used as the supporting set. Finally, we predict the class for the 10000 SVRT SD test samples for ten trials (1000 test samples for one trial). The optimal number of epochs for each task was chosen from $\{8, 16, 32, 64\}$.

4.4.4 Relation Network

Relation network is designed to learn a deep distance metric through simulating the few-shot setting in each episode. For this model, we used the same CNN architecture and Relation Network architecture in Sung's article [31]. The implement process is publicly available in[5]. Omniglot datasets were used to do the 2-way 5-shot meta-learning task for the Relation Network. Ten training samples in SVRT tasks were used as the supporting set. Finally, the class was predicted for the 10000 test samples in SVRT SD tasks for ten trials (1000 test samples for one trial) by using the trained model. The batch number for each class is set to 5. The training episode and test episode are separately 10000 and 1000.

[4]https://github.com/jakesnell/prototypical-networks
[5]https://github.com/floodsung/LearningToCompare_FSL

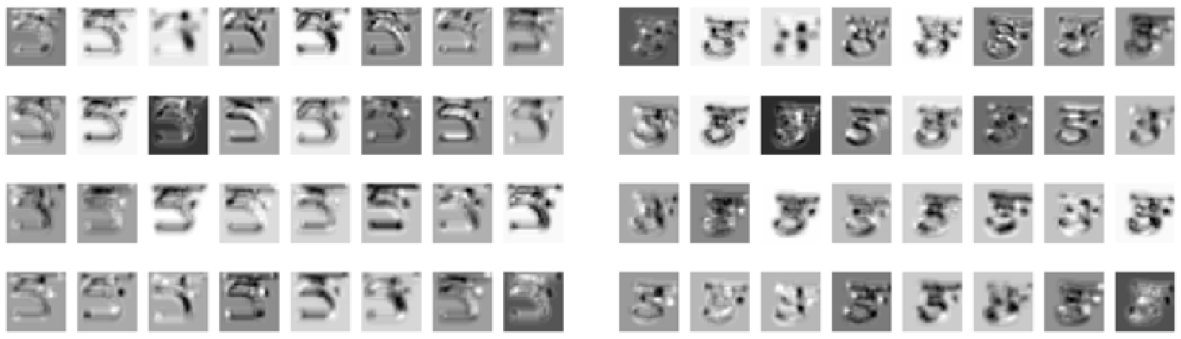


Figure 4.3: Examples of the fifth-layer convolutional filters learned by same-different twins network.

4.5 Experiment results

Our experimental results include two parts: one is for meta-training on Omniglot dataset, and another part is for meta-testing on SVRT same-different problems.

4.5.1 Experiment Results for Meta-training

As mentioned before, we used the meta-learning strategy to train the same-different twins network. Table 4.1 shows the meta-training experiment results for the same-different twins network on Omniglot dataset. Due to that our same-different twins network was developed based on the Siamese network [18], we also compared the 20-way one-shot performance of our network to the Siamese network, as showed in Table 4.1. It can be observed that our same-different twins network greatly improves the one-shot classification accuracy for nearly 5%, compared with the Siamese network.

Method	**Test**
Siamese network	92.0
Same-different twins network	96.7

Table 4.1: Comparing the one-shot accuracy between the Siamese network and Same-different twins network.

In Figure 4.3, we extracted 32 filters in the fifth convolution layer shown in Figure 3.4. It can be noticed that different filters have different effects. Some filters may look for very small point-wise features, while some filters may act as the larger scale edge detectors.

4.5.2 Experiment Results for SVRT SD Problems

Table 4.2 contains the performance for each method on the SVRT same-different problems. All results are averaged across ten trials except the results from Fleuret et al. [10], where the number of trials is unknown. All results, except GoogLeNet and Human performance, were obtained with only ten training samples. GoogLeNet used 20000 training examples. The human performance showed in Table 4.2 was estimated by Stabinger et al. [30] based on the original data reported by Fleuret et al. [10]. The average numbers of images required to learn the rules for humans for each SVRT SD task were reported by Fleuret et al. [10] and shown in the Figure 2.2 in Chapter 2.

The meanings of model acronyms in table 4.2 are as follows: Fleuret: The Adaboost and spectral features model from Fleuret et al. [10]. PN: Prototypical Network. RN: Relation Network. Vgg-16: Fine-tune deep CNNs, pre-trained Vgg-16 for feature extraction. MAML: Model-Agnostic Meta-Learning. GoogLeNet: Performance of GoogLeNet on the SVRT SD problems, published results from the paper [30]. Human: Estimated accuracy of participants, and human tests were done by Fleuret et al. [10].

4.6 Experiment Results Analysis

The most immediate observation from Table 4.2 is the performance difference between our approach and all other models we compare against. It can be observed that our model greatly outperforms other models on all SVRT same-different problems with an average advantage of above **35%** when only provided with ten training samples. On some SD problems such as SVRT #5, #15, and #16, our method even surpasses the reported performance of human beings [10]. For the performance of the baselines on SD tasks, such as Prototypical Networks,

Problem	**Fleuret**	**Vgg-16**	**PN**	**RN**	**MAML**	**GoogLeNet**	**Human**	**Ours**
Training#	10	10	10	10	10	20000	-	10
1	53.0 %	50.6 %	51.3%	51.2%	50.7%	50.0%	98.0%	97.3%
5	47.0 %	50.2 %	51.2%	51.3%	50.3%	50.0%	90.0%	96.2%
7	47.0 %	50.3 %	50.0%	50.5%	50.4%	50.0%	90.0%	90.3%
15	54.0 %	50.5 %	51.3%	50.4%	50.5%	50.0%	95.0%	99.1%
16	62.0 %	50.4 %	51.5%	50.8%	50.7%	50.0%	78.0%	92.1%
19	51.0 %	50.1 %	51.4%	50.7%	49.7%	50.0%	98.0%	85.1%
20	48.0 %	50.3 %	50.3%	50.3%	50.9%	50.0%	98.0%	55.2%
21	39.0 %	50.0 %	50.2%	50.1%	49.3%	51.0%	83.0%	62.3%
22	53.0 %	50.3 %	51.4%	50.6%	49.9%	50.0%	100.0%	98.6%
Average	50.4 %	50.3 %	51.0%	50.7%	50.1%	50.1%	92.2%	86.2%

Table 4.2: A summary of the test results on the SVRT SD problems for different methods.

MAML, Relation Network, and fine-tune deep CNNs, they are all nearly around 50%. This concurs with the performance of GoogLeNet from Stabinger et al [30]. Despite the large numbers of training samples, CNN has great difficulty learning the required "sameness" and "difference" feature.

SVRT #1 and SVRT #15 contain two and four random objects respectively in each class. The shapes of the objects are different in class 1, but same in class 2 for both tasks. Our approach performs extremely well on the two problems. The accuracy is 97.3% and 99.1% respectively. These results are explainable. When the numbers of objects in one image increase, there will be more identical features or different features in this image. Our method also achieves 98.6% classification accuracy on SVRT #22. For SVRT #22, class 2 contains three identical aligned objects, while class 1 contains three different objects, but the difference between the objects is minimal. All of these prove that our method is very good at classifying the visual tasks with abstract visual concepts "sameness" and "differences".

SVRT #5 and SVRT #7 require the attention to the highly abstract concept "grouping". Our method also performs very well on the two tasks, with an accuracy of 96.2% and 90.3% respectively. Another amazing result is for SVRT #16 that shows the highly abstract concepts "reflection". Human beings only achieve 78% classification accuracy with the 12.45 average number of images. However, our method can achieve up to 92.1% accuracy with only ten training examples. All of these prove that our method is very good at classifying the visual tasks with more abstract "sameness" and "differences" visual concepts "grouping".

SVRT #19, #20, and #21 require the machine to be able to identify shape similarity up to a number of invariants: scaling invariance, reflection invariance, and scaling and rotation invariance. For SVRT #19, each image contains two objects of different sizes. In class 2, the two shapes are equivalent up to scaling and translation. For SVRT #20, each image contains two objects. In class 2, one object can be obtained from the other by reflection and rotation. For SVRT #21, each image contains two objects. One of the objects in class 2 can be obtained from the other by scaling, translating, and rotating. Our approach achieved 85.2%, 55.2%, 62.3% performance, respectively, in these three tasks. SVRT #20 and #21 are the most difficult SD tasks for our method. In the future, we can work to solve these invariants.

Chapter 5

Applications

In this Chapter, we mainly discussed how our same-different twins network could be used to solve other same-different problems with one-shot learning and how our model for SVRT same-different visual reasoning tasks could be used to solve other same-different visual reasoning tasks. In addition, we also tested how well a network that has been trained on other tasks can generalize to new datasets.

5.1 MNIST

MNIST dataset has become a very popular database for researchers to conduct some research on machine learning algorithms. In the MNIST dataset, there are ten digit classes, and each class contains thousands of examples. The MNIST dataset is completely different from the Omniglot dataset that has been used to meta-train our same-different twins network. The Omniglot dataset contains a small number of handwritten letters in each class, and the numbers of classes far exceed the numbers of samples in each class. The original author said that this Omniglot dataset could be viewed as the "transpose" of the MNIST [20]. We thought it would be interesting to see that how well the same-different twins network that has been trained on the Omniglot dataset can generalize to the MNIST dataset. Therefore, we created 10-way one-shot classification tasks and MNIST SD visual reasoning tasks to test the adaptability of our

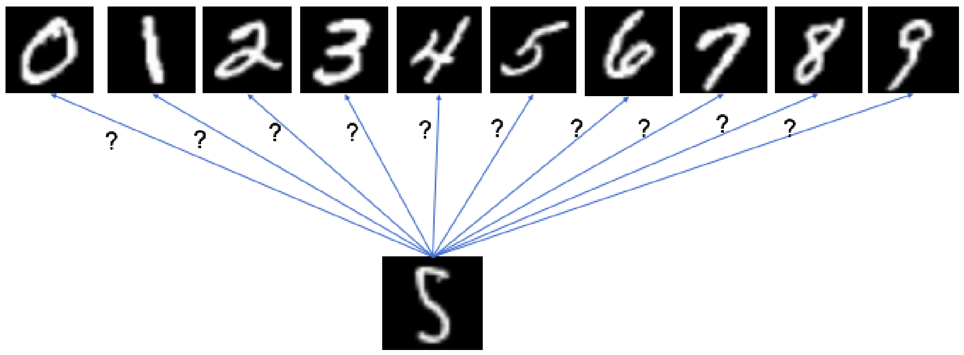


Figure 5.1: Generalize to evaluate 10-way one-shot task on MNIST based on learned features from Omniglot

same-different twins network and our model for SVRT SD problems separately. These results also help us to learn the generalization performance of Omniglot to MNIST.

5.1.1 10-way one-shot tasks

We generated 1000 10-way one-shot tasks through MNIST to test the adaptability of our same-different twins network and the generalization performance of Omniglot to MNIST. Figure 5.1 shows an example of the 10-way one-shot task in MNIST dataset. During the testing, we excluded any fine-tuning on the MNIST training set. Each image to be classified was paired with the other ten MNIST images. Then each pair was fed into a reduced version of our same-different twins network that had been pre-trained on the Omniglot datasets, which were downsampled to 28 x 28. The reduced version (input size:28 x28) is shown in appendixa.

We compared the performance of our same-different twins network on MNIST 10-way one-shot tasks to the Convolutional Siamese network [18] and 1 nearest-neighbor. The results for 1 nearest-neighbor and Convolutional Siamese network are borrowed from [18]. Table 5.1 shows the experimental results for the MNIST 10-way one-shot task. It can be observed that our same-different twins network is almost as good as the Convolutional Siamese network. Compared with the 1-Nearest Neighbor, it can be seen that we are still able to achieve reasonable generalization from the features learned on Ominiglot without training on MNIST.

Method	Test Accuracy
1-Nearest Neighbor	26.5
Convolutional Siamese Net	70.3
Same-Different Twins Network	68.2

Table 5.1: Experiment results for MNIST 10-way one-shot classification tasks.

5.1.2 MNIST same-different visual reasoning tasks

In addition, we also test the performance of our model on MNIST same-different visual reasoning classification tasks generated by using MNIST. Figure 5.2 shows several examples for the same-different visual reasoning classification tasks. The code to generate this dataset is publicly available at [1]. In RelationalMNIST-S, class 2 contains one exact duplicate, while objects in class 1 are totally different. RelationalMNIST-SI contains some invariants, which means that class 2 contains one duplicate, but this duplicate is obtained from one object by rotating and scaling. RelationMNIST-C and RelationMNIST-CI contain some fuzzy repeating, which means that "duplicates" are only from same classes in these two tasks. Compared with RelationMNIST-C, "duplicates" in RelationMNIST-CI have some rotation and scaling added.

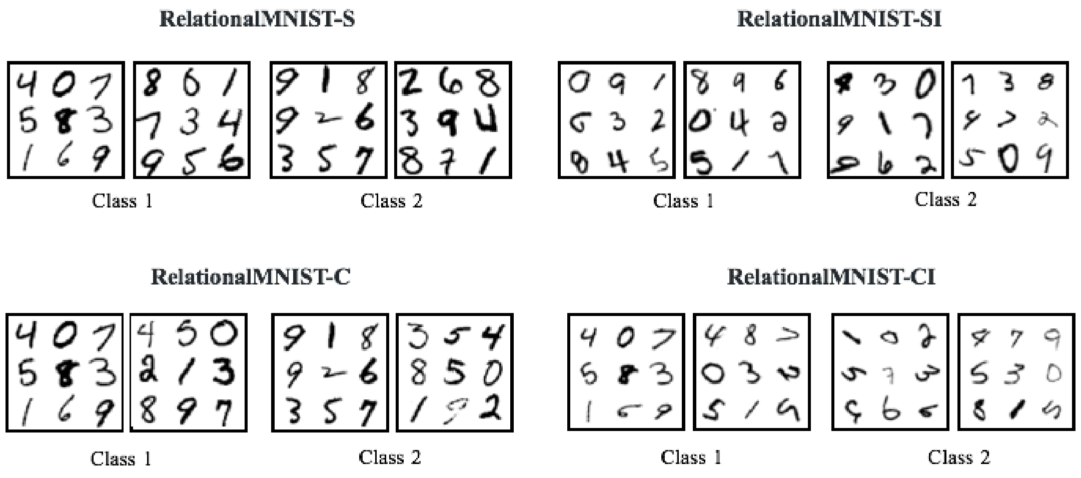


Figure 5.2: A few examples for MNIST same-different visual reasoning tasks

[1] https://github.com/tannerbohn/RelationalMNIST

We used the same model and the same experiment setup, which were used to solve the SVRT SD problems to solve MNIST same-different visual reasoning tasks. The same-different twins network was meta-trained on the Omniglot dataset. When predicting the class labels for new images, the same-different twins network did not have any fine-tuning. After relation scores among the Regions of Interest were obtained through same-different twins network, they were inputted into the pattern recognition module to predict the class label. We used 10 training set, 1000 validation sets, and 10000 test sets for each MNIST SD visual reasoning classification task. Same as the SVRT SD tasks, we also evaluated the performance for several strong baselines, and the performance was the average of classification accuracy for ten trials (1000 test sets for one trial). Table 5.2 shows the experiment results for baselines and our model on MNIST same-different visual reasoning tasks. It can be seen that our model significantly outperforms other strong baselines that are almost random guess with accuracy about 50%.

Method	S	SI	C	CI
Fine-tune deep CNNs	50.6	50.1	49.3	48.1
Prototypical Network	51.1	49.9	49.6	48.2
Relation Network	52.2	49.5	49.3	48.5
Model-Agnostic Meta-Learning	51.8	50.3	49.2	48.7
Ours	94.5	77.8	58.5	52.2

Table 5.2: Experiment results for MNIST same-different visual reasoning tasks.

5.2 Fashion-MNIST

Fashion-MNIST[2] is a dataset of Zalando's article images. In Fashion-MNIST, there are ten classes, which is same as MNIST. The ten classes are separately T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. Fashion-MNIST is usually used to replace the MNIST to benchmark machine learning algorithms. Each image in Fashion-MNIST is a 28x28 grayscale image.

[2]See https://www.kaggle.com/zalando-research/fashionmnist for the details.

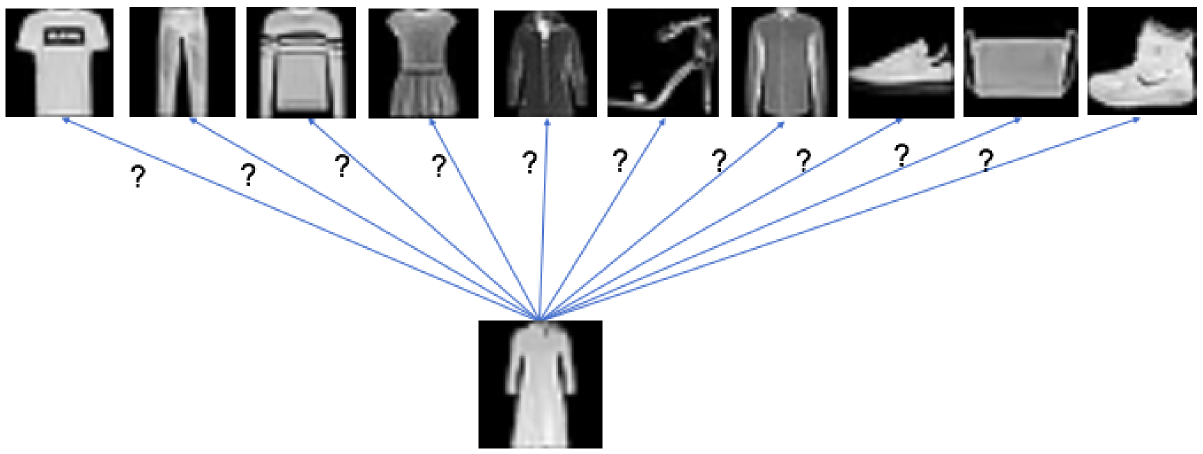


Figure 5.3: Generalize to evaluate 10-way one-shot task on Fashion-MNIST based on learned features

From the MNIST experiments, it has proven that our same-different twins network and our model for SD visual reasoning can be used to solve the MNIST problems. It also can be observed that we are able to achieve reasonable generalization from the features learned on Ominiglot without training on MNIST. Fashion-MNIST is a more complex dataset compared with the MNIST dataset. We would like to see whether our approach could be used to solve the more complex Fashion-MNIST dataset, and we also want to know that how well a model that has been trained on the Omniglot dataset can generalize to more complex problems such as Fashion-MNIST. Therefore, we create the 10-way one-shot classification tasks and Fashion-MNIST visual reasoning classification tasks to test the performance of generalization and adaptability of our method.

5.2.1 10-way one-shot tasks

We generated 1000 10-way one-shot tasks by using Fashion-MNIST to test the generalization performance. Figure 5.3 shows a few examples for the Fashion-MNIST 10-way one-shot tasks. During the testing, we excluded any fine-tuning on the fashion-MNIST training set. Each

Method	Test Accuracy
1-Nearest Neighbor	25.3
Same-Different Twins Network	47.2

Table 5.3: Results for Fashion-MNIST 10-way one-shot classification task.

image to be classified was paired with other ten images. Then each pair was fed into a reduced version of our same-different twins network that has been trained on the Omniglot datasets, which were downsampled to 28 x 28. Finally, select the category with the highest score as the predicted class.

We also evaluated the nearest-neighbor baseline on this Fashion-MNIST 10-way one-shot task. Table 5.3 shows the experiment results for the Fashion-MNIST 10-way one-shot experiment. From the experiment, it can be observed that we are still able to achieve generalization from the features learned on Ominiglot without training on Fashion-MNIST. Compared with the simple nearest-neighbor classifier, a more complex CNN network such as our same-different twins network can achieve better performance.

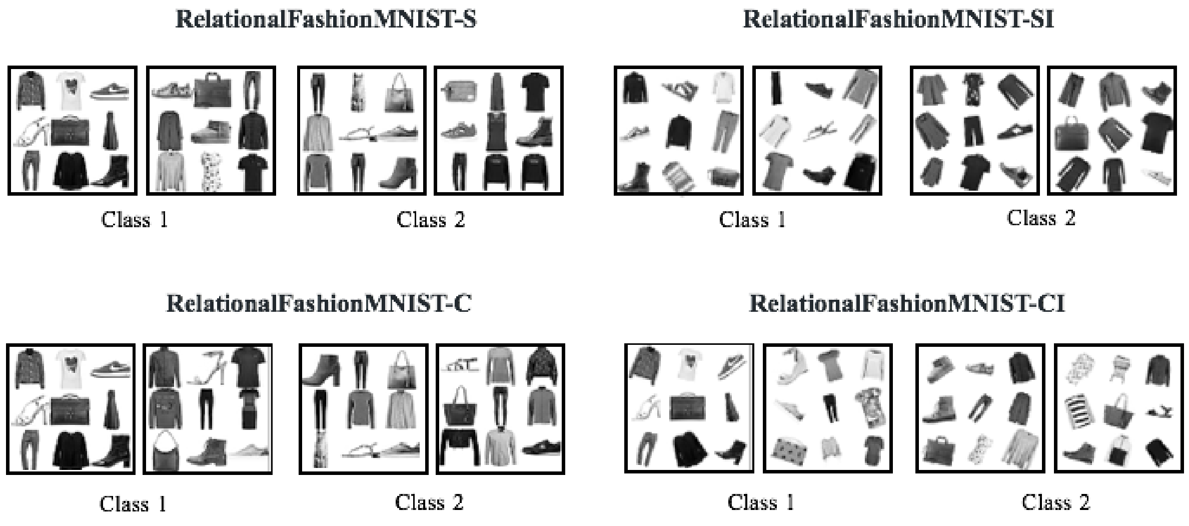


Figure 5.4: A few examples from Fashion-MNIST SD visual reasoning tasks

5.2.2 Fashion-MNIST SD visual reasoning tasks

We also tested the performance of our model on Fashion-MNIST same-different visual reasoning classification tasks generated by using Fashion-MNIST. Figure 5.4 shows several examples of the Fashion-MNIST SD visual reasoning tasks. The dataset was generated in the same way as the MNIST SD dataset and is publicly available at [3]. In RelationalFashionMNIST-S, class 2 contains one exact duplicate, while objects in class 1 are totally different. RelationalfashionMNIST-SI contains some invariants, which refers to that class 2 contains one duplicate, but this duplicate is obtained from one object by rotating and scaling. RelationFashionMNIST-C and RelationFashionMNIST-CI contain some fuzzy repeating, which means that "duplicates" are only from the same classes in these two tasks. Compared with RelationFashionMNIST-C, "duplicates" in RelationFashionMNIST-CI have some rotation and scaling added.

We used the same model and the same experiment setup, which were used to solve the SVRT SD problems to solve Fashion-MNIST SD visual reasoning tasks. The same-different twins network was meta-trained on Omniglot dataset without any fine-tuning on the Fashion-MNIST SD visual reasoning tasks. Same as the SVRT SD tasks, we also evaluated the performance for several strong baselines. Table 5.4 contains the experiment results for baselines and our model on FashionMNIST same-different visual reasoning tasks. It can be observed that our model greatly improved the performance on several SD tasks compared with other baselines.

Method	**S**	**SI**	**C**	**CI**
Fine-tune deep CNNs	50.3	49.1	48.5	48.3
Prototypical Network	50.1	49.5	49.2	48.7
RelationNetwork	49.5	49.3	49.0	48.5
Model-Agnostic Meta-Learning	50.2	49.4	48.7	48.8
Ours	78.1	60.1	56.5	51.2

Table 5.4: Experiment results for FashionMNIST SD visual reasoning tasks.

[3] https://github.com/tannerbohn/RelationalMNIST

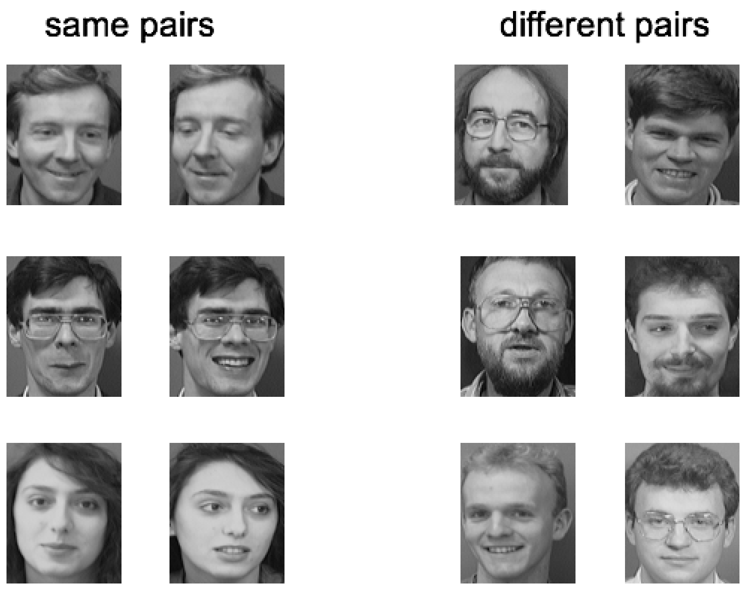


Figure 5.5: A sample for the generated pairs using AT&T Database of Faces

5.3 Facial Similarity

Face recognition has become a hot research area in recent years, and it has become one of the common features used in mobile applications. Convolutional Siamese network [18] is widely used in face recognition. Our same-different twins network is developed based on the Convolutional Siamese network. We are also interested in whether our same-different twins network can be applied to facial recognition. We use the AT&T Database of Faces, which can be downloaded from [4]. The dataset contains images of 40 subjects from various angles. In each subject, there are ten different images.

Same-different twins network requires input values as a pair along with the label, so we have to generate our data in such a way. We randomly took two images from the same subject and marked them as a genuine pair, and we took one image from two different subjects and marked them as a different pair. Figure 5.5 shows a sample for the generated dataset by using the AT&T Database of Faces. As you can see, a same pair has images of the same person, and

[4] https://www.kaggle.com/kasikrit/att-database-of-faces

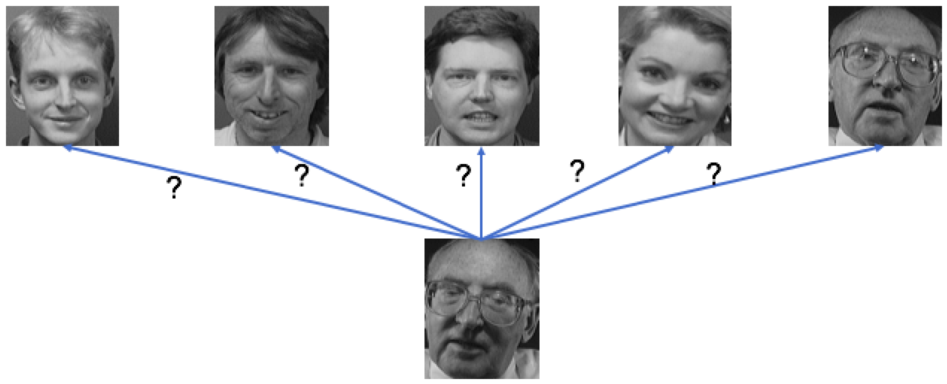


Figure 5.6: One-shot task on facial recognition

the different pair has images of different people.

Figure 5.6 shows an example of the one-shot task on facial recognition. Only one example is given in each subject, and we need to classify new images into its categories. In AT&T Database of Faces, there are images of 40 subjects from various angles. We randomly selected 35 subjects as the training set to meta-train our same-different twins network, and the remaining five subjects are used as the test sets. We generated 1000 5-way one-shot face recognition tasks to test the performance of our same-different twins network on facial recognition. We also evaluated the performance of the Convolutional Siamese network [18] on AT&T Database of Faces. The code for the Convolutional Siamese network is publicly available on[5]. Table 5.5 shows the test results for this experiment. It can be observed that our same-different twins network can be applied to facial recognition and improves the performance compare with the Convolutional Siamese network.

Method	**Test**
Same-Difference Twins Network	96.8
Convolutional Siamese Net	91.4

Table 5.5: Experiment results for one-shot facial recognition task.

[5]https://github.com/fangpin/siamese-pytorch

Chapter 6

Conclusion

In this chapter, we conclude our work presented in this thesis. Lastly, we discuss our future plans.

6.1 Conclusion

The ability to recognize highly abstract visual concepts is a ubiquitous human skill that has not seen significant progress. Same-different problems are a type of visual reasoning task with highly abstract visual concepts. Much research has proved that standard deep convolutional neural networks have failed to solve these kinds of tasks, and CNN has poor performance even when trained by plenty of training samples. SVRT same-different problems, as a representative of the SD visual reasoning tasks, include a series of examples that require a machine to be able to identify highly abstract "sameness " and "difference" visual concepts up to a number of invariants: scaling invariance, reflection invariance, and scaling and rotation invariance.

To solve the same-different visual reasoning problems with few-shot learning, in this thesis, we propose a new method to recognize the pattern through examining the similarities of objects in one image. When examining the similarities, it was defined as a one-shot learning in a broad sense, and we adopted a strategy of meta-learning to solve this problem such that transferable knowledge can be learned to help to perform better few-shot learning on the target tasks and classify the unseen images more effectively. For our model, firstly, we built a same-different twins network to compute the similarity score for any pair of images. Secondly, regions of Interest were extracted from the image through selective search method and then randomly grouping these regions of interest into pairs and feeding each pair into the same-different twins network. The same-different twins network can predict the relation score for each pair. After that, the pattern recognition module was used to compute the similarity for each image through setting the relation scores below the threshold to 0 and then adding all relation score among regions of interest. Finally, a k-nearest neighbor classifier was trained to map the similarities to the class labels.

From the experimental results, it can be observed that our approach dramatically outperforms published results for all SVRT SD problems, and also exceeds other strong baselines that achieve state-of-the-art performance at other few-shot image tasks. Most of these algorithms are random guesses, with about 50% accuracy. Our method can achieve accuracy above 95% on several SD tasks and above 85% on average, with only ten training samples. On some SVRT SD tasks such as SVRT #5, #15, and #16, our method even surpasses the reported human performance [30]. In the case that training a CNN using a large number of training samples cannot solve this SVRT same-different visual reasoning problems, our method shows a strong few-shot learning ability for same-different visual tasks with highly abstract concepts.

Besides, we also applied our same-different twins network and our model for SVRT SD tasks to solve more other same-different problems such as MNIST, FashionMNIST, and face recognition, achieving good performance on these tasks. It has been proven that our method could be applied to solve a wider variety of same-different problems. In the process of solving these tasks, we also learned that some knowledge could be transferred to solve other tasks.

In conclusion, we determined how to make the machine to learn visual tasks by one-shot learning, and we also explained how the machine solves the same-different visual reasoning tasks with highly abstract concepts such as “sameness”, “difference”, “grouping”, and “invariance under rotation, reflection, and scaling” by using few training samples. This opens a door for future research in this area. We believe this work will inspire future development in solving the same-different abstract visual reasoning tasks and one-shot learning questions.

6.2 Future Work

Our current work can be further improved in some ways. Firstly, one-shot recognition is done by the same-different twins network that has been trained on Ominiglot dataset. However, this can be improved by developing a more effective network of one-shot learning or using more datasets to meta-training the network. Secondly, our model achieved poor performance in same-different visual reasoning tasks that requires to solve the invariance on rotation, reflection, and scaling. In the future, we can focus on solving these problems.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[2] David GT Barrett, Felix Hill, Adam Santoro, Ari S Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. *arXiv preprint arXiv:1807.04225*, 2018.

[3] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in Neural Information Processing Systems*, pages 523–531, 2016.

[4] Mikhail Moiseevich Bongard. The recognition problem. Technical report, FOREIGN TECHNOLOGY DIV WRIGHT-PATTERSON AFB OHIO, 1968.

[5] Neelima Chavali, Harsh Agrawal, Aroma Mahendru, and Dhruv Batra. Object-proposal evaluation protocol is' gameable'. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 835–844, 2016.

[6] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016.

[7] Kevin Ellis, Armando Solar-Lezama, and Josh Tenenbaum. Unsupervised learning by program synthesis. In *Advances in neural information processing systems*, pages 973–981, 2015.

[8] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[10] François Fleuret, Ting Li, Charles Dubout, Emma K Wampler, Steven Yantis, and Donald Geman. Comparing machines and humans on a visual categorization test. *Proceedings of the National Academy of Sciences*, 108(43):17621–17625, 2011.

[11] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

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

[13] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2016.

[14] Drew A Hudson and Christopher D Manning. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*, 2018.

[15] Anil K Jain and Stan Z Li. *Handbook of face recognition*. Springer, 2011.

[16] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[17] Junkyung Kim, Matthew Ricci, and Thomas Serre. Not-so-clevr: visual relations strain feedforward neural networks. 2018.

[18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

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

[20] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

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

[22] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 464–471. IEEE, 2000.

[23] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[24] Jean Raven et al. Raven progressive matrices. In *Handbook of nonverbal assessment*, pages 223–237. Springer, 2003.

[25] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

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

[27] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

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

[29] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[30] Sebastian Stabinger, Antonio Rodríguez-Sánchez, and Justus Piater. 25 years of cnns: Can we compare to human abstraction capabilities? In *International Conference on Artificial Neural Networks*, pages 380–387. Springer, 2016.

[31] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

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

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

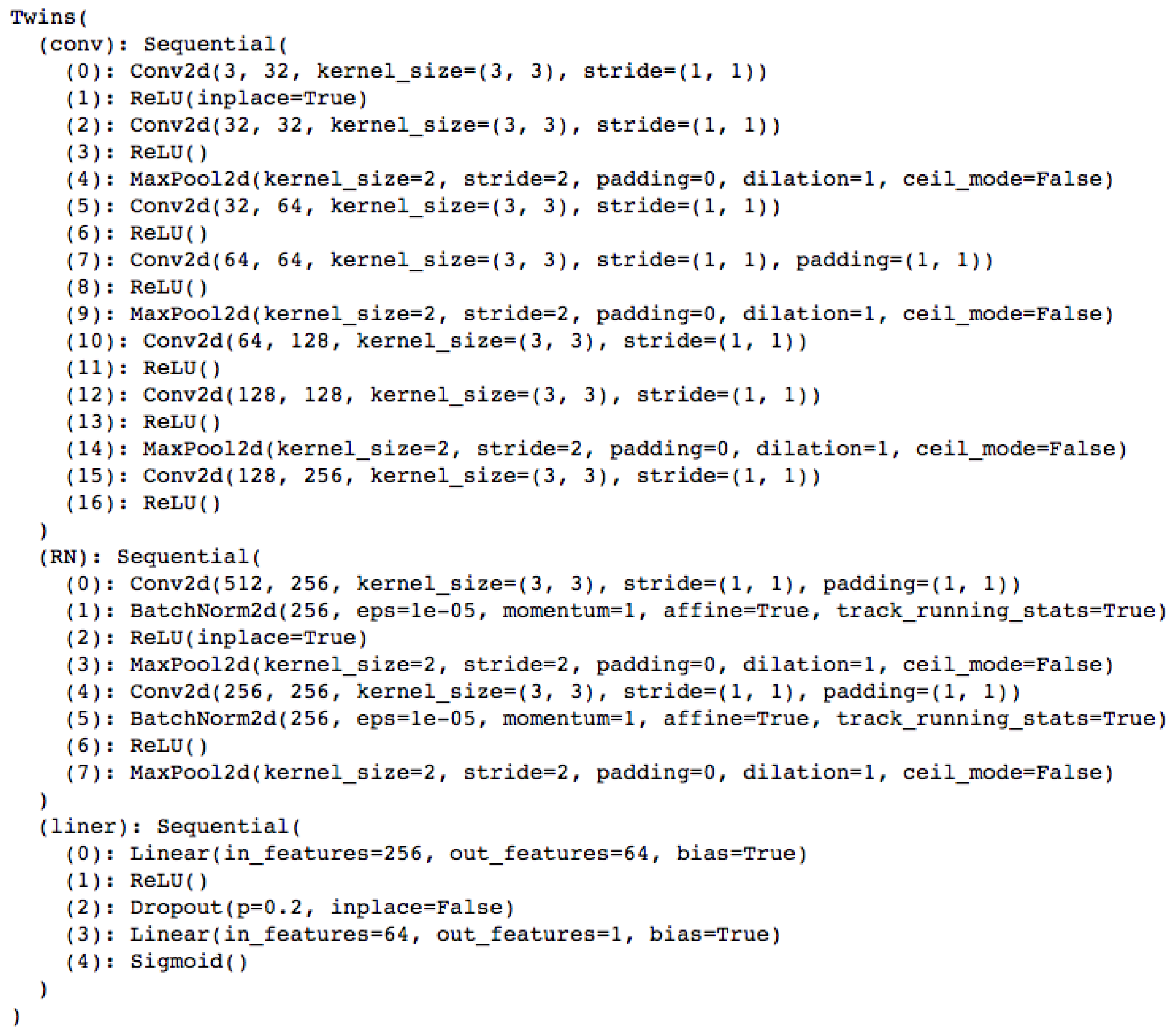
[34] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[35] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.

Appendix A

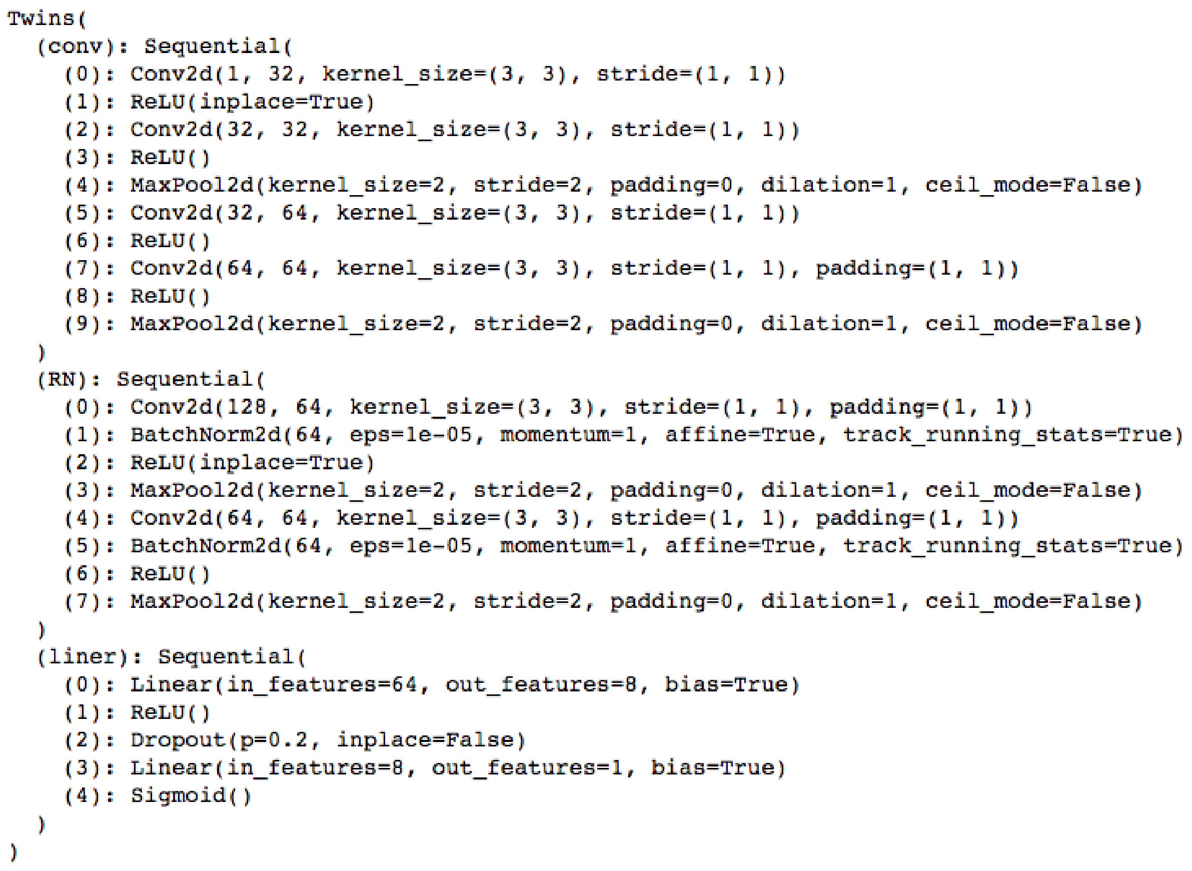
Summaries of the Same-different Twins Network

In the following, we provided the architecture summaries for same-different twins network in 84 x 84 Input size and 28 x 28 Input size, described in Chapter 3 and Chapter 5.



```
Twins(
  (conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (6): ReLU()
    (7): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
    (11): ReLU()
    (12): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1))
    (13): ReLU()
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (15): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
    (16): ReLU()
  )
  (RN): Sequential(
    (0): Conv2d(512, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(256, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (liner): Sequential(
    (0): Linear(in_features=256, out_features=64, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=64, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```

Figure A.1: Summary for Same-different Twins Network (input size: 84 x84)



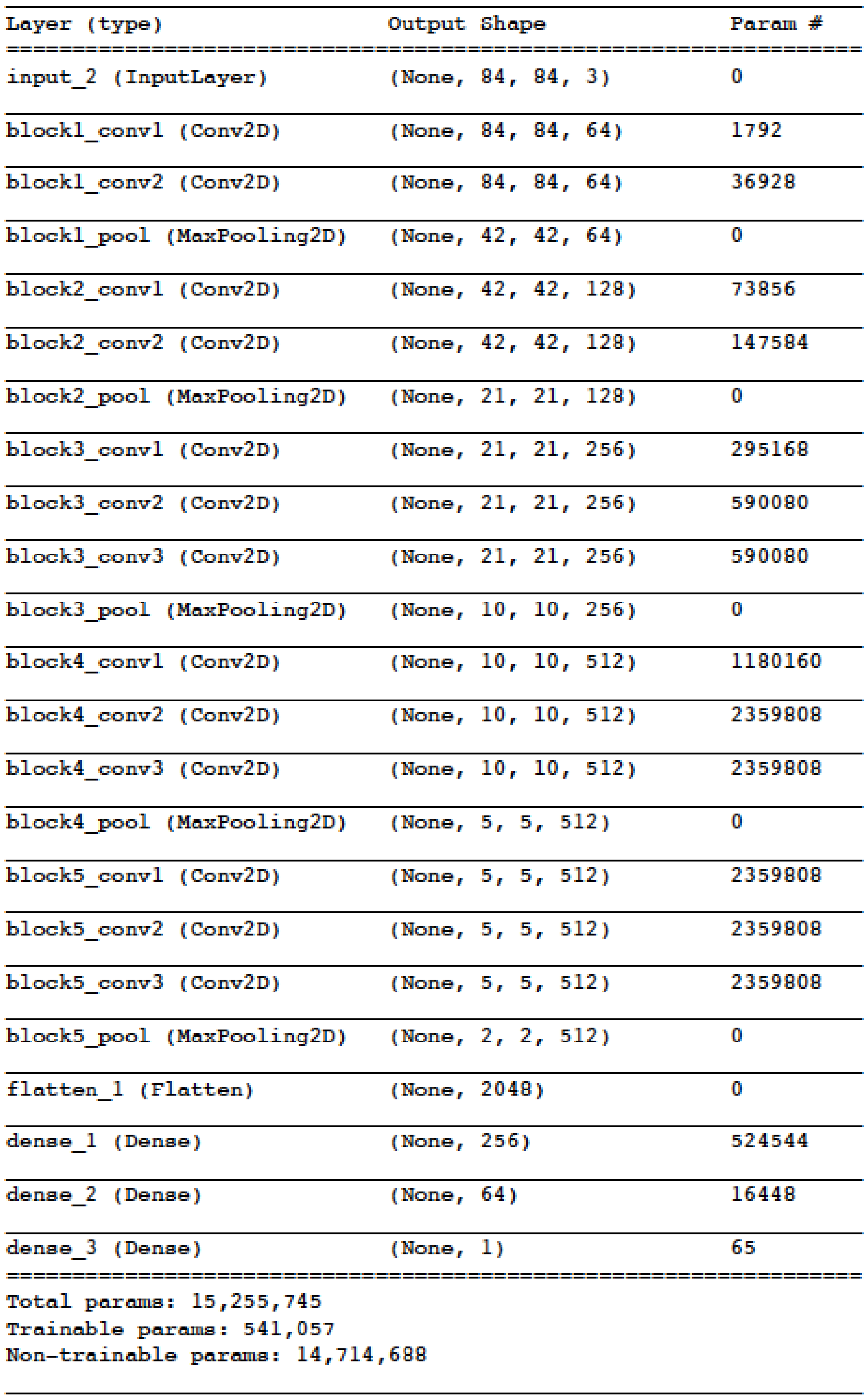
```
Twins(
  (conv): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (6): ReLU()
    (7): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (RN): Sequential(
    (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (liner): Sequential(
    (0): Linear(in_features=64, out_features=8, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=8, out_features=1, bias=True)
    (4): Sigmoid()
  )
)
```

Figure A.2: Summary for Same-different Twins Network (input size: 28 x28)

Appendix B

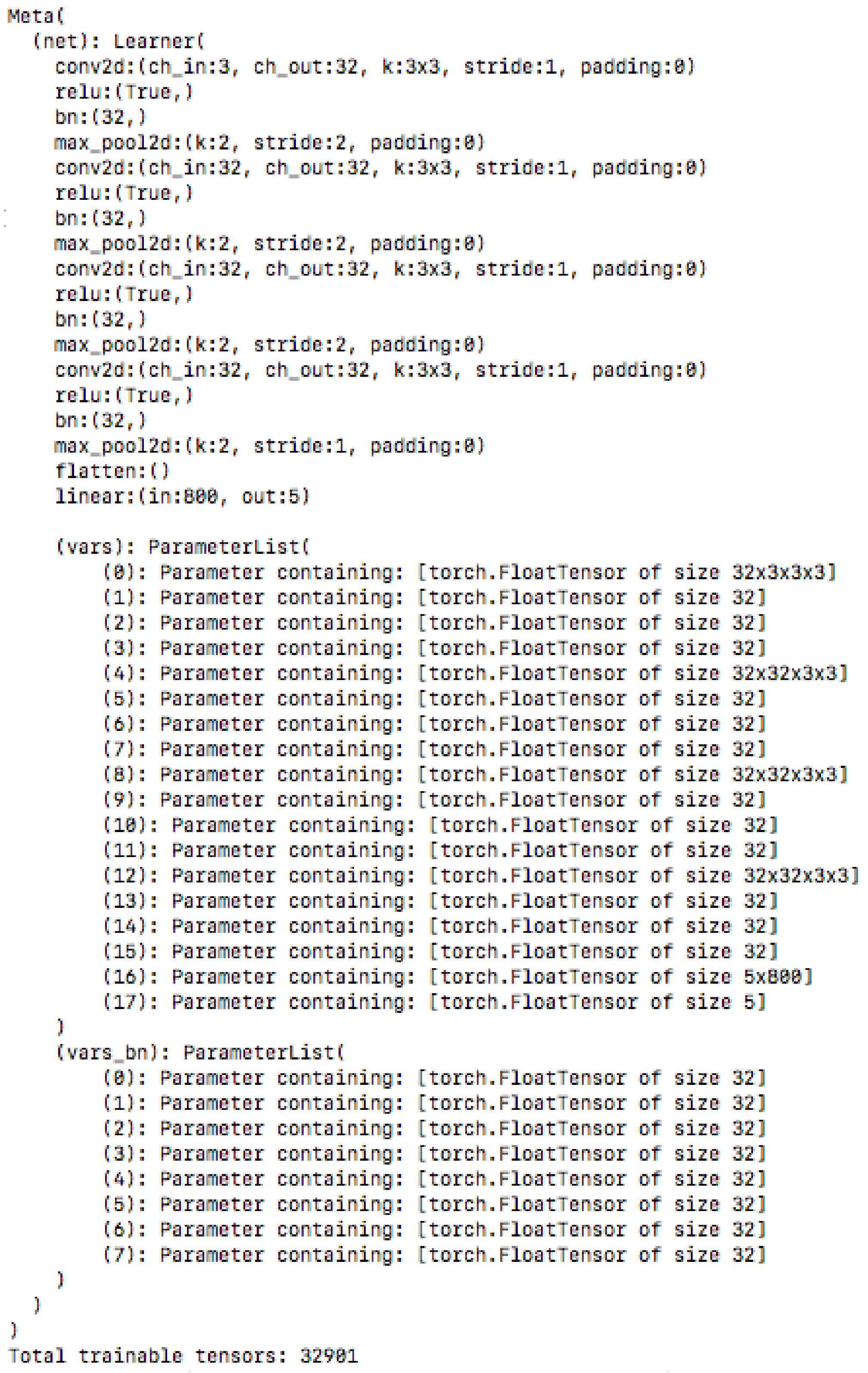
Summaries for the Baselines

In the following, we provided the architecture summaries for the baselines we benchmarked in Chapter 4 and Chapter 5.



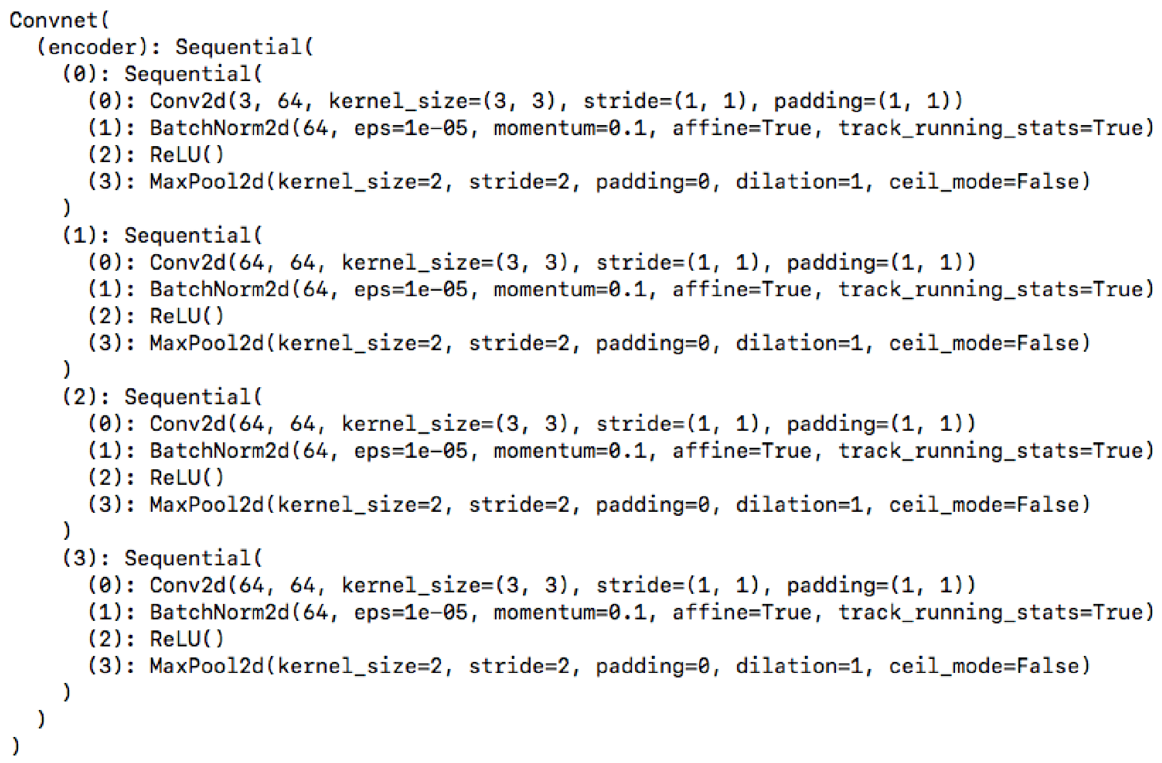
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 84, 84, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 84, 84, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 84, 84, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 42, 42, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 42, 42, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 42, 42, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 21, 21, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 21, 21, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 21, 21, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 21, 21, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 10, 10, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 10, 10, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 10, 10, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 5, 5, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 5, 5, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 5, 5, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 5, 5, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               524544
_________________________________________________________________
dense_2 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 15,255,745
Trainable params: 541,057
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure B.1: Summary for Fine-tune deep CNNs (Vgg-16)



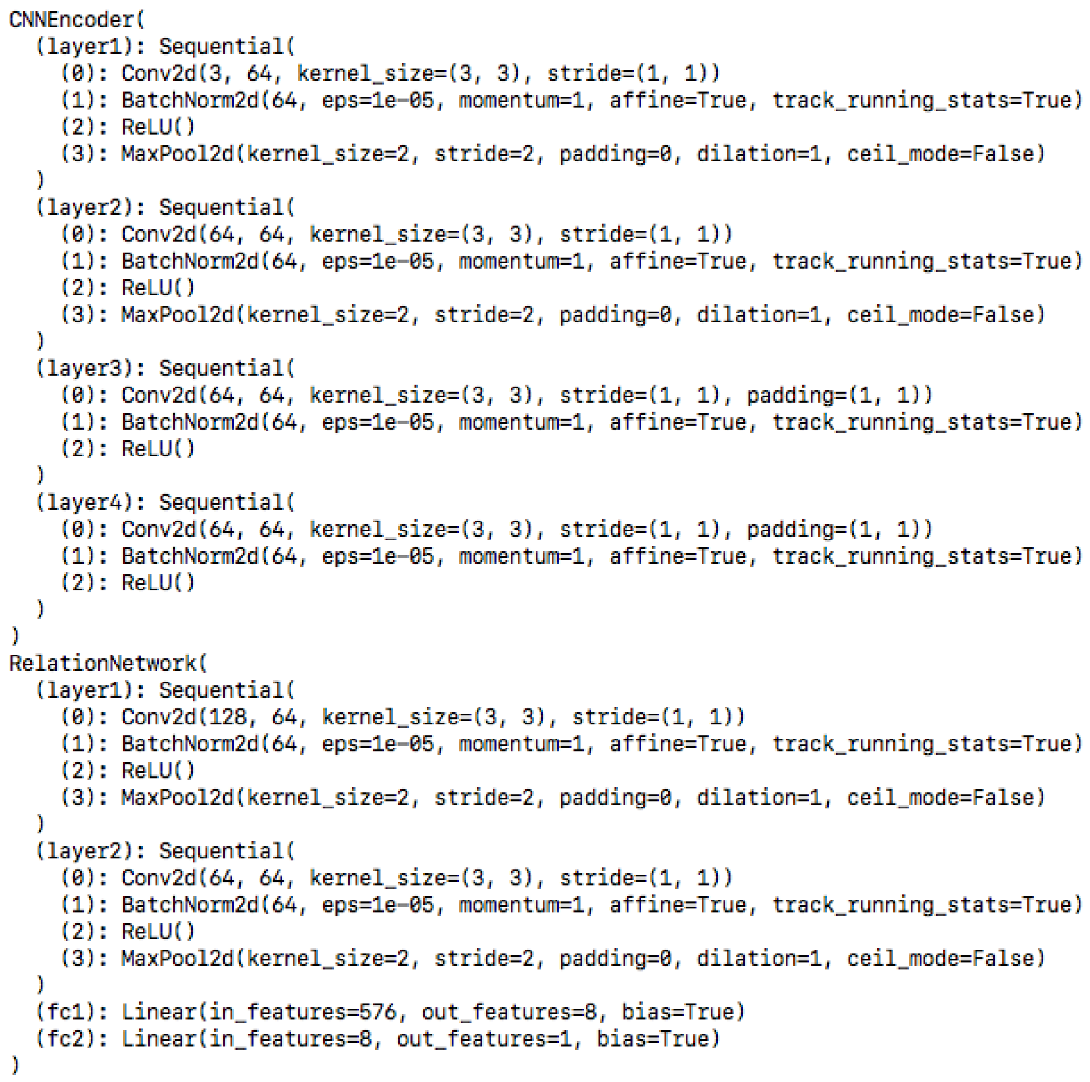
```
Meta(
  (net): Learner(
    conv2d:(ch_in:3, ch_out:32, k:3x3, stride:1, padding:0)
    relu:(True,)
    bn:(32,)
    max_pool2d:(k:2, stride:2, padding:0)
    conv2d:(ch_in:32, ch_out:32, k:3x3, stride:1, padding:0)
    relu:(True,)
    bn:(32,)
    max_pool2d:(k:2, stride:2, padding:0)
    conv2d:(ch_in:32, ch_out:32, k:3x3, stride:1, padding:0)
    relu:(True,)
    bn:(32,)
    max_pool2d:(k:2, stride:2, padding:0)
    conv2d:(ch_in:32, ch_out:32, k:3x3, stride:1, padding:0)
    relu:(True,)
    bn:(32,)
    max_pool2d:(k:2, stride:1, padding:0)
    flatten:()
    linear:(in:800, out:5)

    (vars): ParameterList(
        (0): Parameter containing: [torch.FloatTensor of size 32x3x3x3]
        (1): Parameter containing: [torch.FloatTensor of size 32]
        (2): Parameter containing: [torch.FloatTensor of size 32]
        (3): Parameter containing: [torch.FloatTensor of size 32]
        (4): Parameter containing: [torch.FloatTensor of size 32x32x3x3]
        (5): Parameter containing: [torch.FloatTensor of size 32]
        (6): Parameter containing: [torch.FloatTensor of size 32]
        (7): Parameter containing: [torch.FloatTensor of size 32]
        (8): Parameter containing: [torch.FloatTensor of size 32x32x3x3]
        (9): Parameter containing: [torch.FloatTensor of size 32]
        (10): Parameter containing: [torch.FloatTensor of size 32]
        (11): Parameter containing: [torch.FloatTensor of size 32]
        (12): Parameter containing: [torch.FloatTensor of size 32x32x3x3]
        (13): Parameter containing: [torch.FloatTensor of size 32]
        (14): Parameter containing: [torch.FloatTensor of size 32]
        (15): Parameter containing: [torch.FloatTensor of size 32]
        (16): Parameter containing: [torch.FloatTensor of size 5x800]
        (17): Parameter containing: [torch.FloatTensor of size 5]
    )
    (vars_bn): ParameterList(
        (0): Parameter containing: [torch.FloatTensor of size 32]
        (1): Parameter containing: [torch.FloatTensor of size 32]
        (2): Parameter containing: [torch.FloatTensor of size 32]
        (3): Parameter containing: [torch.FloatTensor of size 32]
        (4): Parameter containing: [torch.FloatTensor of size 32]
        (5): Parameter containing: [torch.FloatTensor of size 32]
        (6): Parameter containing: [torch.FloatTensor of size 32]
        (7): Parameter containing: [torch.FloatTensor of size 32]
    )
  )
)
Total trainable tensors: 32901
```

Figure B.2: Summary for Model-Agnostic Meta-Learning



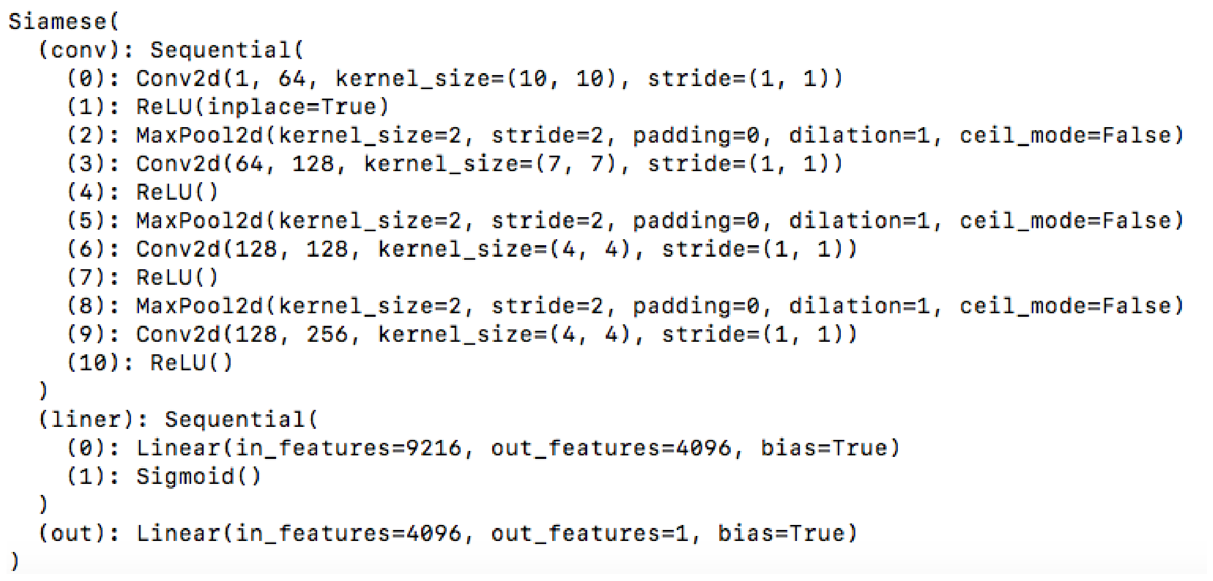
```
Convnet(
  (encoder): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (2): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (3): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
  )
)
```

Figure B.3: Summary for Prototypical Network



```
CNNEncoder(
  (layer1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer3): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (layer4): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
)
RelationNetwork(
  (layer1): Sequential(
    (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc1): Linear(in_features=576, out_features=8, bias=True)
  (fc2): Linear(in_features=8, out_features=1, bias=True)
)
```

Figure B.4: Summary for Relation Network

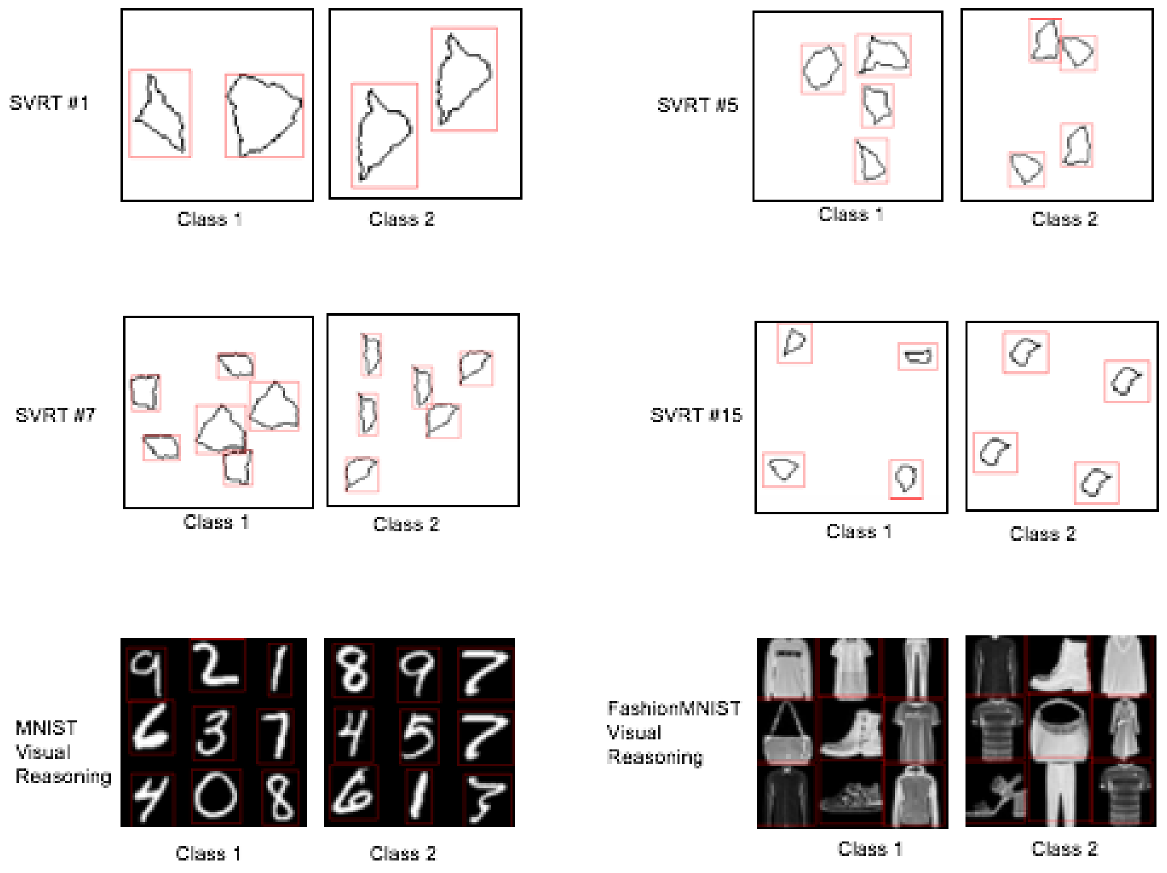


```
Siamese(
  (conv): Sequential(
    (0): Conv2d(1, 64, kernel_size=(10, 10), stride=(1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 128, kernel_size=(7, 7), stride=(1, 1))
    (4): ReLU()
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(128, 128, kernel_size=(4, 4), stride=(1, 1))
    (7): ReLU()
    (8): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (9): Conv2d(128, 256, kernel_size=(4, 4), stride=(1, 1))
    (10): ReLU()
  )
  (liner): Sequential(
    (0): Linear(in_features=9216, out_features=4096, bias=True)
    (1): Sigmoid()
  )
  (out): Linear(in_features=4096, out_features=1, bias=True)
)
```

Figure B.5: Summary for Convolutional Siamese Network

Appendix C

Examples for Regions of Interest Selection



Curriculum Vitae

Name:	Yuanyuan Han
Post-Secondary Education and Degrees:	Harbin Institute of Technology Habin, Heilongjiang, China 2010 - 2014 B.A.
	Harbin Institute of Technology Habin, Heilongjiang, China 2014- 2016 M.A.
Honours and Awards:	Western Graduate Research Scholarships(WGRS) 2018-2019
Related Work Experience:	Teaching Assistant and Research Assistant The University of Western Ontario 2018 - 2019