4-24-2020 3:15 PM

A Patch-as-Filter Method for Same-Different Problems with Few-Shot Learning

Yining Hu, *The University of Western Ontario*

Supervisor: Ling, Charles, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science
© Yining Hu 2020

Abstract

Convolutional Neural Network (CNN) has undergone tremendous advancements in recent years, but visual reasoning tasks are still a huge undertaking, particularly in few-shot learning cases. Little is known, especially in solving the Same-Different (SD) task, which is a type of visual reasoning task that requires seeking pattern repetitions in a single image. In this thesis, we propose a patch-as-filter method focusing on solving the SD tasks with few-shot learning. Firstly, a patch in an individual image is detected. Then, transformations are learned to create sample-specific convolutional filters. After applying these filters on the original input images, we, lastly, acquire feature maps indicating the duplicate segments. We show experimentally that our approach achieves the state-of-the-art few-shot performance on the Synthetic Visual Reasoning Test (SVRT) SD tasks by accuracy going up above 30% on average, with only ten training samples. Besides that, to further evaluate the effectiveness of our approach, SVRT-like tasks are generated with more difficult visual reasoning concepts. The results suggest that the average accuracy is increased by approximately 10% compared to several popular few-shot algorithms. The method we suggest here has shed new light upon new CNN approaches in solving the SD tasks with few-shot learning.

Keywords: Visual reasoning, Few-shot learning, Dynamic filters, Convolutional Neural Network

Summary for Lay Audience

With the rapid development of Convolutional Neural Networks, computers can achieve nearly the same performance as humans and even better in image recognition and classification. Learning highly abstract concepts, however, is still extremely challenging for computers with millions of training data. In particular, Same-Different (SD) tasks have been proven especially difficult for existing CNN approaches. These SD tasks require reasoning of the similarity between patterns located within the same image. The standard CNN fails on learning highly abstract visual concepts since its filters are optimized based on the training data and then applied on test samples. Therefore, the ability to learn abstract concepts within the same picture is lost. In this thesis, we are aiming to make machines learn highly abstract visual concepts by only observing a few labelled samples. Firstly, a patch in an individual image is detected. Then, transformations are learned to create sample-specific convolutional filters. After applying these filters on the original input images, we, lastly, acquire feature maps indicating the duplicate segments. By using our method, we can identify the highly abstract relations of shapes in each image. Since our filters are generated base on a patch from the input image, the filters of each image in our method are different in both training datasets and test datasets. These sample-specific filters enable our model to learn more abstract visual concepts, such as fuzzy same-different, specific rotation and specific scaling. We show experimentally that our approach achieves the state-of-the-art few-shot performance on the Synthetic Visual Reasoning Test (SVRT) SD tasks by accuracy going up above 30% on average, with only ten training samples. Besides that, to further evaluate the effectiveness of our approach, SVRT-like tasks are generated with more difficult visual reasoning concepts. The results suggest that the average accuracy is increased by approximately 10% compared to several well-performed few-shot methods.

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 helped me a lot since I came to Canada. I believe he considerably exceeded the expectations of his role as supervisor.

I would like to thank Yuanyuan and Tanner for their encouragement and advises during this research and the life in Canada.

Lastly I want to show special gratitude to my parents and Xi, for their constant support throughout my research and thesis writing.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BN	Batch Normalization
CNN	Convolutional Neural Networks
CLEVR	Compositional Language and Elementary Visual Reasoning
MAML	Model-Agnostic Meta-Learning
MANN	Memory-Augmented Neural Networks
N2NMNs	End-to-End Module Networks
NMN	Neural Module Network
PFL	Patch-as-Filter Layer
PN	Prototypical Networks
ReLU	Rectified Linear Unit
RN	Relational Network
SD	Same-Different
SR	Spatial-Relation
SVM	Support Vector Machine
SVRT	Synthetic Visual Reasoning Test

Chapter 1

Introduction

1.1 Motivation

With the rapid development of Convolutional Neural Networks, computers can achieve nearly the same performance as humans and even better in image recognition and classification [14]. For example, the two images (Figure 1.1 (a)) [6] with a bird and dog portrait can be correctly classified by a Convolutional Neural Network (CNN). The classification accuracy of the CNN even exceeds human observers on the ImageNet [6] classification challenge, after training with millions of images [14].

Learning highly abstract concepts, however, is still extremely challenging for computers with millions of training data [36]. In particular, Same-Different (SD) tasks have been proven especially difficult for standard CNN approaches [9, 7, 45, 12, 36]. These SD tasks require reasoning of the similarity between patterns located within the same image. For instance, the images [9] in Figure 1.1 (b) are SD tasks, which are quite simple compared to the image in panel (a). The relation between shapes in these graphs is quite obvious to human observers, i.e., Class 2 contains the same shapes while Class 1 not, but standard CNN methods fail on learning the concept of “sameness” in this simple scene even with

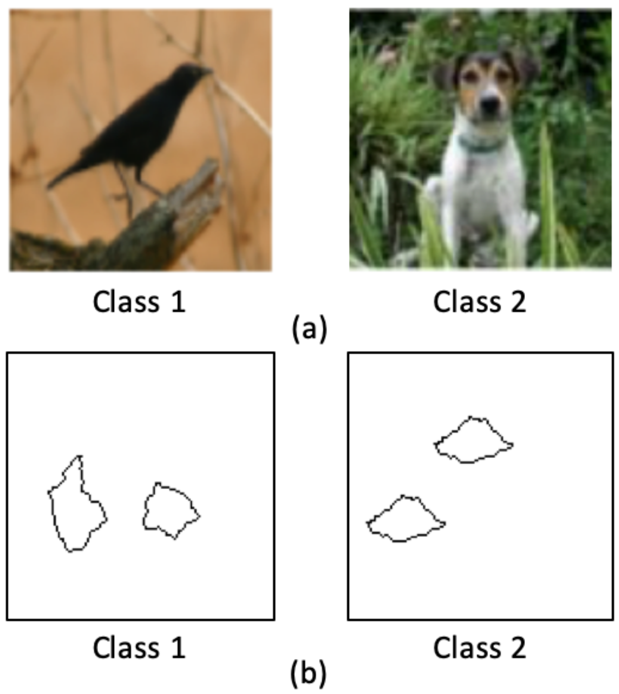


Figure 1.1: The two images [6] in panel (a) can be accurately classified as their categories by CNN algorithms. However, the same algorithms fail to learn the concept of "same" and "different" as images [9] shown in panel (b) with 20,000 training images [45].

20,000 training samples[45].

Learning highly abstract visual concepts is critical to the development of computer vision. The study of the Same-Different tasks also has many practical implications, such as finding transformation invariant same-different relations in texture analysis and locating a person from several people in face recognition. This thesis primarily focuses on solving SD tasks from the Synthetic Visual Reasoning Test (SVRT) dataset, which seems simple but contains more complicated abstract visual concepts than many real-world images. The SVRT tasks can be split into two groups, based on the type of patterns separating the two classes: Spatial-Relation (SR) tasks (ex. shapes in a line vs. not in a line) and Same-

Different (SD) tasks (ex. two pairs of unique shapes vs. two pairs of identical shapes)[45]. Previous work at these tasks has shown that CNNs are capable of solving SR tasks given enough training data (20K in [45] and 1 million in [36]), but the SD tasks have been proven to be more difficult. Even relational networks [40], which perform well on other reasoning tasks such as CLEVR [17], have demonstrated great difficulty in solving SD tasks, requiring millions of training samples to achieve merely above chance-level performance [36]. More recent work shows that SVRT tasks can be solved by very complex pre-trained deep neural networks such as ResNet-50 [15] with 28,000 training samples. However, human observers can learn most SVRT tasks rules by only viewing an average of 6.27 instances [9].

The reason why most CNN models fail on learning highly abstract visual concepts is that the filters are optimized based on the training data in the training process of CNN. Whereas, only a small number of the neurons in the network are activated for a test image, which indicates inefficient computation [52] and leads to failure on learning highly abstract concepts. When humans learn the same-different tasks, we observe a figure in the image first and then compare it with another figure in the same image to perceive the relationship between them. However, the traditional machine learning methods compare the shapes in the test images with the information learned from the training samples, so the ability to learn abstract concepts within a single picture is lost. Figure 1.2 presents images from the SVRT dataset [9] and their feature maps provided by prototypical network [44], which is a popular few-shot learning method based on CNN. The images on the left are from the SVRT Task 1, in which positive class contains two identical figures, while these images from the negative class contain two unique figures. As we can see, by the information learned from the training dataset, the feature maps can only indicate the position of the two shapes, but not the relationship between the shapes. That is, without the knowledge of shapes within each test image, the CNN embedding fails to learn the "same" or "different" concepts for the binary images with only two shapes. The filters of standard CNN can only extract information such as the positions and bounds of objects in the picture but are not able to learn highly abstract concepts.

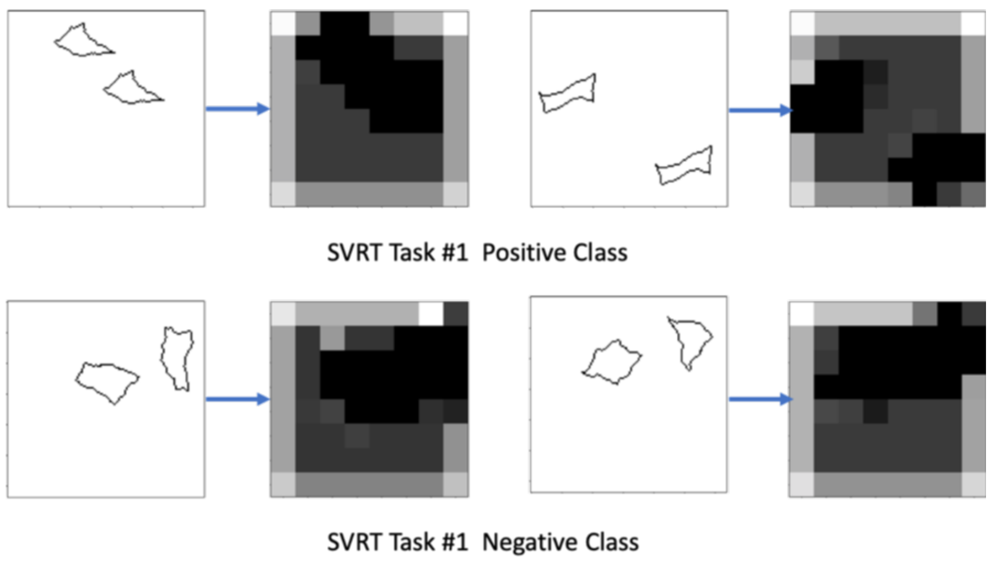


Figure 1.2: Images [9] and their feature maps after the embedding stage of prototypical network[44]. In input images, the positive class contains two shapes, which are the same, but the negative class does not. It is challenging to distinguish positive and negative classes since no concept about the “same” or “different” is provided by the feature maps.

In this thesis, we are aiming to make machines learn highly abstract visual concepts by only observing a few labelled samples. We solve it by establishing a patch-as-filter convolutional block. This method is divided into three parts: Firstly, a patch in an image was detected. Secondly, transformations are learned to generate sample-specific convolutional filters using the patch we detected. Finally, we obtain feature maps indicating the duplicate parts by applying these filters on the original input images through a convolutional layer. After that, feature maps are used as input for few-shot classifiers to obtain classification results.

By using our method, we can identify the highly abstract relations of shapes in each image. Since our filters are generated base on a patch from the input image, the filters of each image in our method are different in both training datasets and test datasets. This is distinct from standard CNNs, whose filters are updated by the training datasets, then applied to the test datasets. These sample-specific filters enable our model to learn more abstract visual concepts, such as fuzzy same-different, specific rotation and specific scaling.

We observe that by using feature maps provided by our patch-as-filter method in place of the raw images, we are able to improve the state-of-the-art on few-shot SVRT SD tasks by an average of nearly 30% with only ten training samples. By working towards solving these abstract visual reasoning tasks in an interpretable and straightforward way, we demonstrate how analogous problems may be approached.

1.2 Contributions

In this thesis, we propose a patch-as-filter method to solve the SD visual reasoning tasks. We also generate additional new SD tasks with more complicated visual reasoning concepts. The experimental results show that our method improves the average classification accuracy by more than 30% on the SVRT SD tasks and 10% on the new-generated SD tasks.

The main contributions of this thesis are as follows:

- We propose a patch-as-filter method for solving the same-different problems. Firstly, this approach dynamically generates sample-specific filters for each image. Then feature maps are acquired by applying these filters on the original input image, which contain the information of the duplicate segments in the images. Lastly, few-shot classifiers are trained using the feature maps we developed to get the classification results.

- We generate more complicated SD tasks based on the original SVRT SD datasets.

These tasks contain more highly abstract visual concepts such as “similar”, “same” up to a specific rotation or scaling, and even “same” up to the combination of specific rotations and scaling. More details of our new tasks are described in Chapter 4.

- We benchmark several popular few-shot learning algorithms on the SVRT SD tasks and our new datasets.
- We prove that by combining our method with different few-shot learning classifiers, better few-shot performance can be achieved on SVRT SD tasks and our new datasets with only ten training samples.
- Our experiments suggest that our algorithm is capable of achieving higher classification accuracy by increasing the number of filters we generated, which further supports the validity of our patch-as-filter approach.

1.3 Thesis Outline

The structure of this thesis is organized as follows. In Chapter 2, the background knowledge in Convolutional Neural Networks (CNN) and visual reasoning tasks was covered. Researches in the area of CNN in image classification, Few-Shot Learning, visual reasoning tasks and patch-as-filter method were also reviewed in this chapter. Chapter 3 presents the methodology we proposed to solve highly abstract visual reasoning tasks and the steps used in each stage. Chapter 4 introduces the datasets used in the experiments, experiments and baselines settings as well as the implementation of our method in detail. The experimental results are presented and analyzed in Chapter 5. The conclusion of the research and future work are discussed in Chapter 6.

Chapter 2

Background

In this chapter, we will briefly introduce the background knowledge in Convolutional Neural Networks (CNN), image classification and visual reasoning tasks. Besides, the previous work done on the Same-Different visual reasoning tasks, few-shot learning and patch-as-filter methods will also be reviewed.

2.1 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a kind of deep neural network that is widely adopted in Computer Vision and other forward position fields of computer science. CNN learns directly from the input data with the automatic generation of feature maps. Moreover, CNN has been proven to be very successful in image classification and object recognition applications.

CNN architectures consist of many layers. As shown in Figure 2.1, the input layer contains several neurons that accept a large number of non-linear input messages. The message is transmitted in the output layer for analysis and weighed in the neuron link to form an output. Hidden layers are the layers of many neurons and links between the input layer and the output layer. Hidden layers usually contain multiple convolutional layers, ReLU layers and pooling layers. Each layer has a specific target and performs specific operations, which are described below.

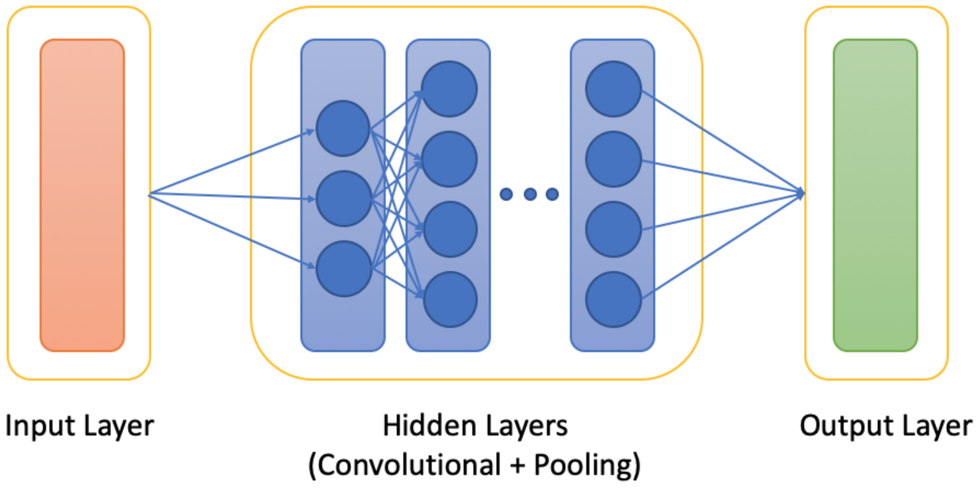


Figure 2.1: CNN architecture contains an input layer, hidden layers and an output layer. Hidden layers are composed of several convolutional layers and pooling layers.

Convolutional Layer

The convolutional layer is the core part of the Convolutional Neural Network, which constructs high-level features of the input image, applied to identify patterns for classification. In the standard neural network, the fully connected layer needs to train a large number of weights, which requires a large scale of calculation time and space, thus significantly affects the computational efficiency. The Convolutional Neural Network calculates the convolution of filters and a region of the input image (Figure 2.2), which is equivalent to training the same set of weights by different parts of the picture. CNN reduces the training parameters and includes the feature of the image in the resulting feature map.

Most CNN architectures include several convolutional blocks that consist of a convolutional layer, a ReLU layer, and a pooling layer. Usually, the first few layers are used to detect the edges and shapes of objects in the images, and the subsequent convolutional layer can obtain more abstract information. Thus, the abstraction level increases from layer to layer [27].

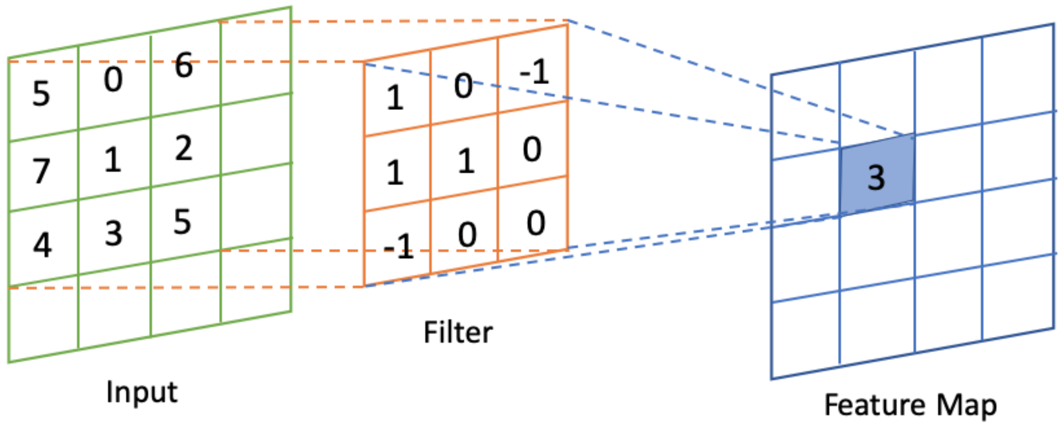


Figure 2.2: A convolutional layer calculates the convolution of an input image and its associated filters to get feature maps.

ReLU Layer

Following the convolutional layer, we need to utilize an activation function to increase the non-linearity of the neural network model. Otherwise, layers of the neural network are equivalent to the multiplication of several matrices. ReLU (equation 2.1) is one of the most used activation functions.

$$ReLU(x) = max(0, x) \tag{2.1}$$

As presented in equation 2.1, ReLU activation function produces more output to 0, which causes the sparseness of the network and reduces the interdependence of parameters and over-fitting problems.

Pooling Layer

The pooling layer improves computational efficiency and retains useful information in the meantime by reducing the size of the feature map. The pooling operations with higher application frequency are average-pooling and max-pooling. Figure 2.3[1] presents an example of max-pooling. The calculation method of max-pooling can launch the largest value in an area. In the matrix of 2×2 in the upper left corner of the figure, the maximum value is 6. As to the matrix of 2×2 in the upper right corner, 8 is the maximum value, so the results of the left image are 6, 8, 3, 4. For the average-pooling, the result feature maps are calculated by taking the average value in an area.

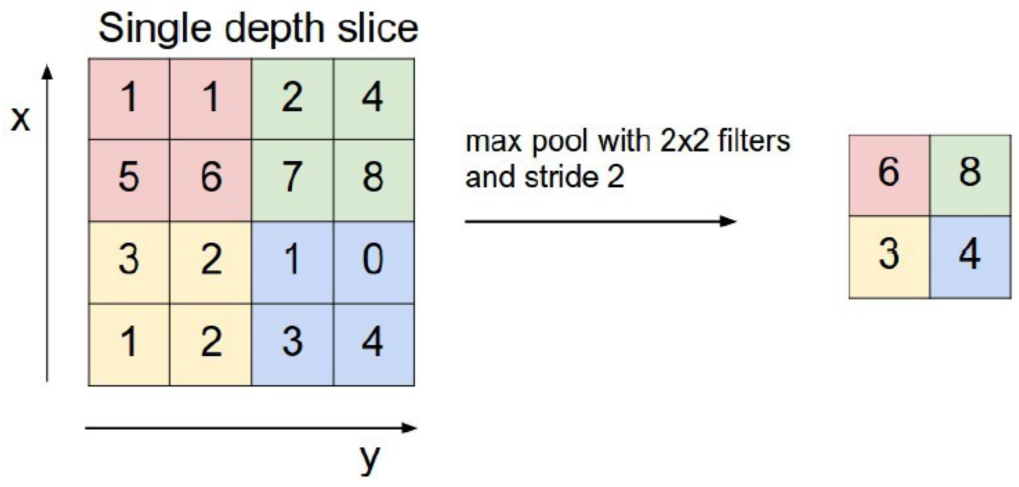


Figure 2.3: An example of how a max-pooling layer reduces the size of a feature map.

Based on the basic principles of CNN we mentioned above, we will discuss the re-

[1]Image from http://cs231n.github.io/convolutional-networks/

searches related to CNN below. The prototype of CNN was proposed by LeCun et al. [28] in 1989 for handwriting recognition. Some basic concepts of CNN were constructed, such as weight sharing and feature maps. After a few years of improvement, LeNet-5 [29], the first CNN, which can be universally applied, was born. LeNet-5 contains 7 layers: 2 convolutional layers, 2 pooling layers, 2 fully connected layers and an output layer. For the early LeNet-5, Tanh and mean-square error functions were selected as the activation function. Meanwhile, loss function, stochastic gradient descent and backpropagation method were performed to train the model.

Many researchers adopted LeNet-5 to solve different Computer Vision problems due to its good performance, including face detection [37, 38], face recognition [26] and visual document analysis [42]. However, some drawbacks of LeNet-5 were exposed in practice at the same time. Vanishing gradient, limit samples and computing abilities keep the LeNet-5 from becoming the mainstream of CNN. New effective methods were required. AlexNet [24] was proposed in 2012. As a substitute for LeNet-5, AlexNet has made many innovations. The AlexNet choose ReLU to be the activation function and apply the dropout method in training. These improvements not only solve the problem of gradient vanishing and gradient exploding but also increase the performance of the model. AlexNet became the top 1 in ILSVRC 2012. Since then, CNN was boomed, surpassing AlexNet in image classification become the research hotspot in recent years.

2.2 CNN in Image Classification

Image classification is a task that assigns a label to an image from a test set. The label comes from a pre-defined training set of possible categories. Due to the better process of image data, CNNs are widely adopted in image classification, especially after the AlexNet showed its best performance in ILSVRC 2012. In 2013, Zeiler proposed the ZFNet [54] to decrease the error rate of AlexNet by deconvolution, but the basic structure was not changed.

After that, the Visual Geometry Group of Oxford University proposed the VGG [43] to improve the AlexNet. There are two main modifications of VGG. First, VGG deletes the LRN layer, whose contribution to the whole network is not significant. Second, a smaller continuous 3x3 convolution kernel is applied to simulate a larger convolution kernel. Through these modifications, VGG obtained better expansion and generalization capabilities than other image datasets.

However, VGG was defeated by GoogleNet in the image classification. GoogleNet [46] was put forward in 2014 by Google company, and the inception mechanism was added to this CNN. The inception module combines convolutions and pooling in different scales together. In this way, the utilization of parameters is improved significantly. Besides, compares with the traditional AlexNet, GoogleNet removes the full connection layer, which accounts for almost 90% parameters of the whole network and may cause overfitting. Instead, the global average pooling is used. All the ideas come from the research work of "Network in Network" [30].

To finish the image classification task better, researchers updated the GoogleNet by absorbing the strengths of other CNNs. In the second version of GoogleNet, the 5x5 convolution kernel is replaced by two 3x3 kernels like VGG, which not only reduces the number of parameters but also speeds up the calculation. In addition, Batch Normalization (BN) layer is added to the GoogleNet, so that the output of each layer is normalized to a gaussian distribution of N(0,1). The second version reduces 2% error rate of the original model. Moreover, the improvement of conventional kernel factorization is added to the network in the third version, and the numbers of inception modules are increased to make the network deeper in the fourth version. However, one thing is not changed in the above CNNs. That is, they all require a vast number of samples in image classification tasks.

2.3 Few-Shot Learning

Few-shot learning is an area to explore how to make the machine learning or deep learning models perform great on the small-scale samples. For most machine learning/deep learning models, they are lack of generalization abilities. For instance, CNN needs at least thousands of images in each class to reach performance saturation. When facing a novel class, it is difficult for CNN to recognize it with a small number of labelled samples. Few-shot learning can overcome this problem by obtaining enough information from a small number of samples.

Modern few-shot learning methods can be mainly divided into two kinds, Meta-Learning based and Embedding-Metric learning based. The idea of meta-learning [41] suggests framing the learning problem at two levels. The first is a quick acquisition of knowledge within each separate task presented. This process is guided by the second, which involves a slower extraction of information learned across all the tasks. This idea is used in many recent few-shot learning approaches. For example, Model-Agnostic Meta-Learning (MAML) [8] is a general and model-agnostic method, which trains initial parameters of the model. Therefore, the model has the maximal performance on a new task. This method achieved high accuracy on few-shot learning tasks Omniglot [25] and miniImageNet [48].

The other kinds are based on embedding-metric learning approaches, such as the prototypical networks [44] and the siamese networks [21]. The siamese network [21] first pairs an input image with all other images to create pairs of images from the same class and different classes, then train a model to discriminate between the collection of pairs. Prototypical network [44] described a method that learns a neural network to map input images into an embedding space. Its main idea is to learn a metric space in which classification can be performed by computing distances to the prototype of each class. As we can see in Figure 2.4, the input images are divided into a support set and a query set, and then all the

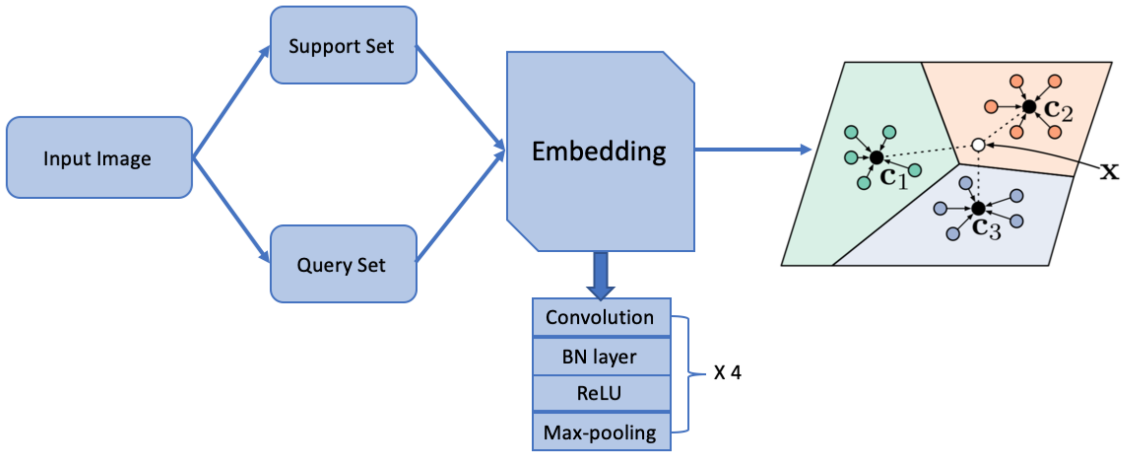


Figure 2.4: Architecture of the prototypical network [44].

images are embedded by a neural network which contains four convolutional blocks. In this embedding space, the prototype of a class is the mean of its support set. For a query point X from the query set, its classification is performed by finding the nearest class prototype.

2.4 Visual Reasoning Tasks

Visual reasoning task is a simulation task to learn human visual reasoning abilities. Humans can analyze visual information and be able to solve problems based on it. The visual reasoning task is more complicated than the traditional computer vision problem. For example, prediction on the next shape based on some shapes that have been given is a visual reasoning task. Some principles were put forward for visual reasoning, including datasets for image classification [5, 20], Turing challenge [32], image annotations [23] and object referring [19].

Based on these different types of datasets for visual reasoning tasks, various methods were proposed to deal with the tasks. Santoro et al. [40] constructed a simple Relation Networks (RN) for visual question answering, which performs well on the CLEVR dataset[17].

Another idea to finish the visual question answering tasks on the CLEVR dataset is End-to-End Module Networks (N2NMNs) [16]. Compared with the traditional Neural Module Network (NMN) architecture, the N2NMNs do not need the assistance of parser. The experiment result also indicates that N2NMNs can significantly reduce the error rate.

Similar to the CLEVR, FigureQA [18] is also a visual question answering dataset, which contains many scientific-style figures. A novel architecture FigureNet[35], was set up on this dataset. The FigureNet outperforms the traditional networks with shorter training time.

Besides the visual answering task of the CLEVR dataset or FigureQA, other datasets related to the tasks based on rule learning are also widely studied by researchers, such as Bongard problems[2], Ravens Progressive Matrices[33], Synthetic Visual Reasoning Test(SVRT)[9].

Nowadays, SVRT tasks are concentrated on by many researchers. SVRT dataset is a collection of 23 grayscale image classification tasks. In each task, two classes are differentiated according to the complete condition of a particular rule. For example, as shown in Figure 2.5, the positive class contains two same shapes in task 1, but the negative class is on the contrary. The SVRT tasks can be split into two groups: Spatial-Relation (SR) and Same-Different (SD) [45]. Figure 2.5 shows some examples of SVRT SR and SD tasks. In Spatial-Relation (SR) tasks, opposing classes differ in terms of positional relationships (like "left of", "inside of" or "touching") of the shapes in the images. Same-Different (SD) tasks have shapes that are the same up to some transformations. SD tasks have been proven to be extremely difficult for existing machine learning approaches[9], [7], [45], [12], [36].

Some methods were adopted to solve these problems in the previous study. Fleuret et al.[9] first introduced SVRT tasks. By comparing experiments of human and machine learning, they found these SVRT tasks are easy to solve for humans and extremely difficult to learn for generic machine learning systems. For the machine experiments, they combined

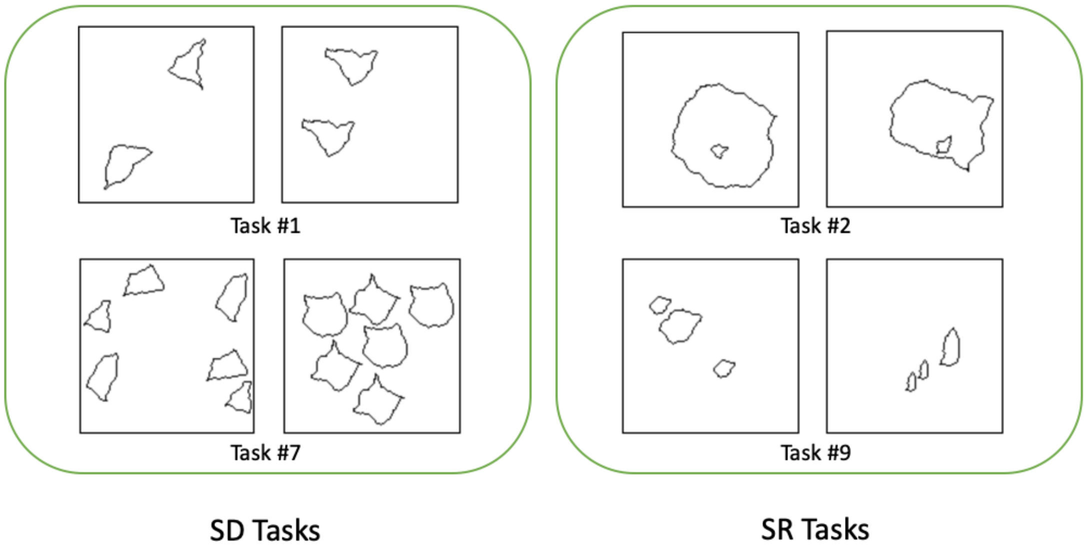


Figure 2.5: Examples from the SVRT dataset [9]. The SVRT dataset is split into two groups [45]: Spatial-Relation (right) and Same-Different (left).

two modules. The first one was a hand-designed feature extractor to compute the properties of the raw image data that may have been useful. The second was a machine learning algorithm, boosting of stumps (standard Adaboost [10] with feature sampling) and SVM [4] with a Gaussian kernel. Fleuret et al. [9] found that humans could solve the tasks much more efficiently than the machines and with higher accuracy. After seeing at most a few tens of examples, more than 90% of the participants solved 14 of the 23 problems. In contrast, the machines, with only 10 examples of each class for training, their performances are just like randomly guessed for every task. Even with 10,000 examples and complex image preprocessing, the machine learning algorithm was only able to solve a few tasks at an accuracy above 90%.

More recent work launched by Ellis et al. [7] proposed an unsupervised program synthesis approach. Their method first parsed input images into a symbolic form by locating shapes. Then, they applied grammar to synthesize a space of programs and found a program that could maximally compress these parses. Thus, the required common structure would be encoded in the program to reproduce the example images. When programs have been produced for training images, a classifier is trained to map these programs to class labels. However, this method could only solve a handful of the 23 SVRT tasks and was very slow in the program synthesis process. Lu et al. [53] improved the image parsings part of this program synthesis method and achieved higher accuracy on several tasks.

Stabinger et al. [45] trained LeNet [28] and GoogLeNet [46] CNNs to solve these tasks. They trained each network with 20,000 images, and found that both models could achieve perfect or near-perfect accuracy on the spatial relation problems but performed poorly on the Same-Different tasks. Similarly, the authors also showed how CNNs failed to learn a Same-Different task with simple binary items in [12].

Ricci et al. [36] also found that such tasks are complicated, and even sometimes impossible, for contemporary computer vision algorithms, including CNNs. In their work, they analyzed CNN architectures on each of 23 SVRT tasks and found Same-Different tasks much harder than spatial relation tasks. Furthermore, they showed that a well-known visual reasoning method, relation networks, failed to solve Same-Different tasks (on a similar dataset, PSVRT) even with millions of training samples.

However, the conclusion that all CNNs are unable to solve Same-Difficult SVRT problems was criticized by Borowski et al. [11]. They showed that feed-forward convolutional architectures are capable of abstract reasoning by applying a pre-trained ResNet-50 [15]. Their ResNet-50 could reach over 90% test accuracy on all the SVRT tasks with 28,000 training images. Their results show that very complex pre-trained neural networks can solve SVRT tasks with plenty of training data.

2.5 Patch-as-Filter Methods

Recently, scholars also proposed many methods based on the idea of finding discriminative filters have been proposed. Wang et al.[50] established a filter bank method to solve fine-grained recognition tasks. They achieved state-of-the-art results on CUB [49], Stanford Cars [22] and FGVC-Aircraft [31], by detecting a discriminative patch most related to classification results, then using this collection of convolutional filters.

Brendel et al.[3] proposed a bag-of-filter method, where the models extract features from small image patches and put them into a linear classifier to produce a heatmap for each class. These heatmaps are then passed through a softmax to derive the classification results.

Most of the previous work in patch-as-filter methods creates a pool of filters by extracting patches from training images, so a large training set is required to achieve high classification accuracy. However, we are aiming to solve SD tasks in few-shot cases. Therefore, dynamic filters are generated based on patches of each image in our method.

Chapter 3

Methodology

Considering the Same-Different (SD) tasks we discussed in Chapter 1, the most direct and effective method is to detect all repeated parts by scanning the whole image and comparing the shapes in it. Therefore, we can apply the concept of the convolutional layer (discussed in Chapter 2.1) to find duplicate parts within a single image by using shapes in this image as filters. As only ten labelled samples are used in our model, few-shot classifiers are adopted to learn the classification rules for each task instantly.

The architecture of our method is shown in Figure 3.1. We propose a patch-as-filter convolution block, which extracts a patch from an image and use this patch to create filters for a convolutional layer. The inputs of this convolutional block are original images, and the outputs are feature maps according to these sample-specific filters. Then we use the feature maps as the input of different well-performed few-shot classifiers. Our method can not only find the exact same part but also similar (not exactly the same) parts. Furthermore, our method also finds the same shape up to particular scaling, rotation and the combination of these transformations.

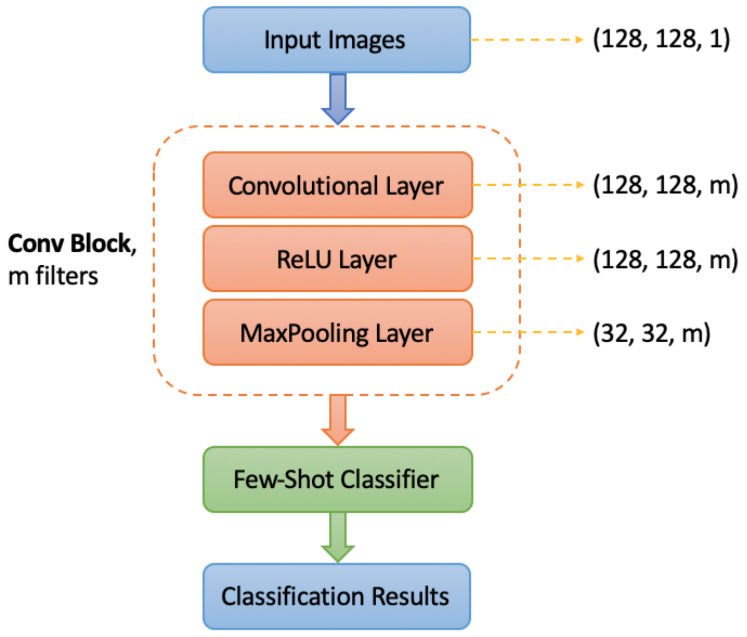


Figure 3.1: The architecture of our method.

3.1 Patch-as-Filter Convolutional Block

The primary purpose of our method is to detect repeated parts within each image. Figure 3.2 illustrates the stages of this process. The first stage (from (a) to (b)) is selecting a patch that contains a shape in it from the original image. The selective search [47] is adopted, which combines the strength of both exhaustive search and segmentation, and can detect object region with high accuracy.

After selecting a patch as shown in Figure 3.2 (b), we create filters int the next step (from (b) to (c)). To retain the shape information in this patch, we set all the pixels with black colour to 1, and all the white pixels to 0. When we apply this filter to the original image, it is easy to seek the parts with the same shape. The size of the filters we create is equal to the patch size. Thus the filter size is different according to different images. It depends on the object region detected by the selective search. Considering task 19, task 20

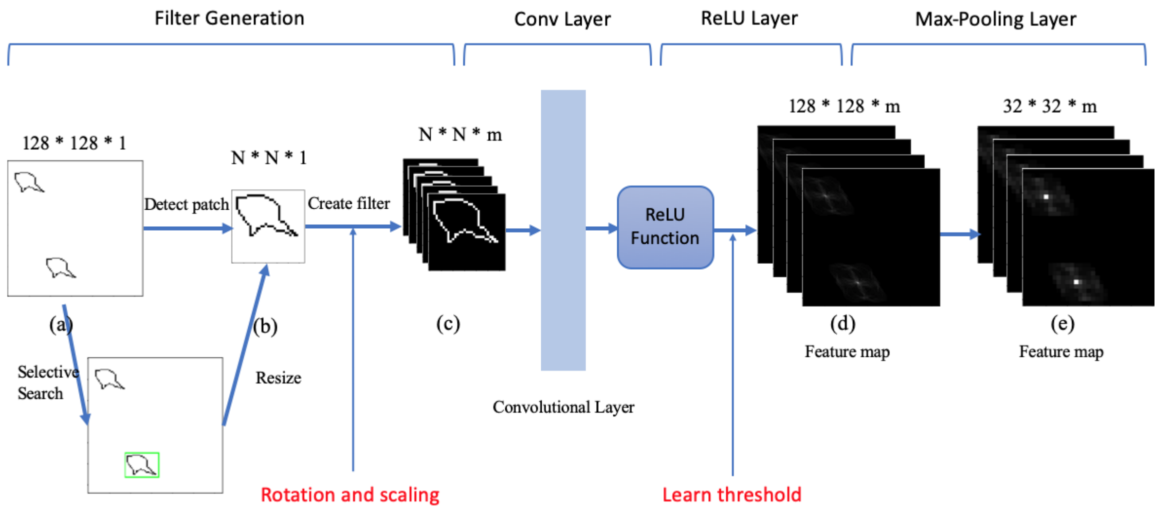


Figure 3.2: A demonstration of various steps in our patch-as-filter convolutional layer. Given an input image (a), the selective search is used to detect a patch with shapes to reach (b). From (b) to (c), we generate filters by rotation and scaling. To reach (d), we apply the filters in a convolutional layer. Finally, a max-pooling layer is applied to downsize the feature maps. After normalization, we obtain the feature maps, which indicate all duplicate parts in an input image.

and task 21 shown in Figure 4.1, whose positive samples contain shapes that are equivalent up to scaling and rotation. We need to generate various kinds of filters and learn multiple feature maps for each image. The filters are produced by scaling and rotating of the shapes we detected from images. A linear model was trained to determine what kinds of rotations and scaling will be adopted to generate filters.

To go from (c) to (d), we apply a convolution layer that uses filters we obtained in the previous stage to go through the input image. The parameter of stride is 1. After normalization, we obtain feature maps like Figure 3.2 (d). The white spots demonstrate the duplicate parts of the original image. This process provides us with a feature map of the same size as the input image.

We downsize the feature map with a 4×4 max-pooling layer (from (d) to (e)) for calculating efficiency. This patch-as-filter layer provides us 32×32 feature maps with white spots showing duplicate parts. For normalization, we set the minimum value of pixels across the feature maps to 0, and the maximum value to 1. This patch-as-filter convolutional layer is applied to both training and test images.

3.2 Generate Sample-Specific Filters

In the stage from (b) to (c) in Figure 3.2, we create filters by scaling and rotating the shapes we detected from images in the previous stage. To determine what kinds of transformations will be adopted, we train a linear model by the rotations and scaling detected from the training samples.

The input of this linear model is the value of rotation and scaling, and the output is a label of 0 or 1. Therefore, we need to know what kinds of transformation the positive and negative classes have in training samples. We use minAreaRect function in OpenCV to find the minimum bounding rectangles of shapes in the images. As shown in Figure 3.3, by comparing the longer edge of the rectangles, the scaling of shapes is calculated precisely. Furthermore, we create a vector from the midpoint of the longer edge of each minimum bounding rectangle to the center of each shape, in order to obtain the rotation angles of shapes by calculating the angle between the vectors. For example, in Figure 3.3, the rotation angle between the two shapes in the image on the right is 58.35.

After the training process, a set of random rotation and scaling are applied as the input of this linear model. Output label 1 means the input rotation and scaling will be applied to generate filters.

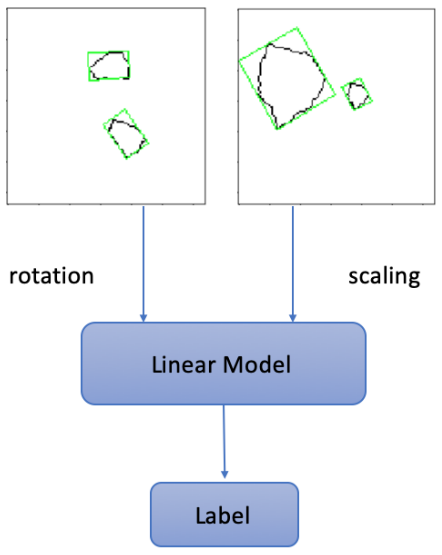


Figure 3.3: Examples of using minAreaRect function in OpenCV to measure the rotations and scaling between shapes in the images [9]. The rotations and scaling in training samples are used to train a linear model.

3.3 Normalization Threshold

In the real world, the repetitions in an image are not exactly the same. In order to enlarge the applications of our approach in more real scenes, it is essential to identify similar parts in an image. After our normalization step in the stage from (c) to (d) in Figure 3.2, we no longer simply set the maximum value in the feature maps to 1 and the minimum value to 0. Instead, we train a linear model using the training samples to learn a threshold, the values higher than this threshold is set to 1, and the values lesser is set to 0. Thus, similar but not exactly the same parts can be found in an image by applying this linear model on the original feature map. The parameters of the linear model are learned according to the same-different rules in the training samples. The threshold learned from training images are applied to the test sample for each task.

3.4 Classifier

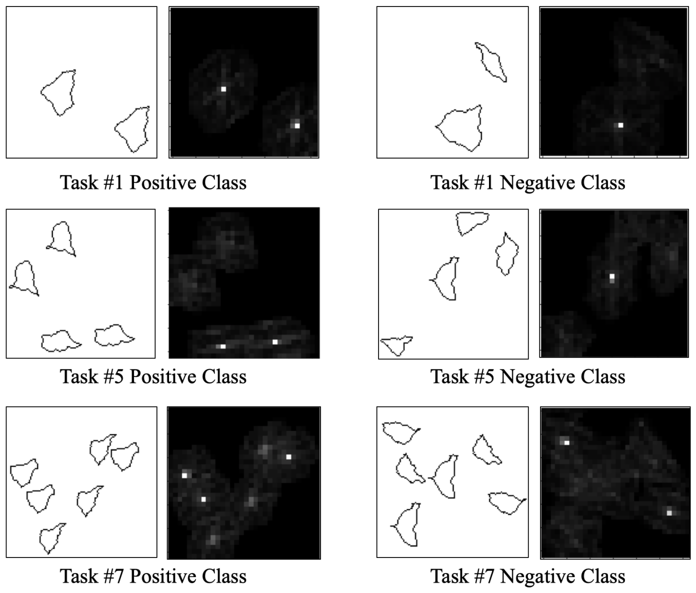


Figure 3.4: Examples of input images and feature maps produced by our method for SVRT Task 1, 5 and 7. In the feature maps of our method, lighter parts indicate duplication. The positive classes (the two columns on the left) always contain more repeated parts than the negative classes (the two columns on the right). Intuitively, classification can be performed by finding relations between the number of lighter parts and the label of input images.

Figure 3.4 shows some examples of several SVRT SD tasks and their feature maps provided by our patch-as-filter layer. As we can see, the differences between the positive and negative classes are obvious. So our patch-as-filter layer can effectively identify duplicate shapes within a single image.

Next, we need to correlate the feature maps with correct classification labels. Since we only use ten training samples, the classifiers are required to perform well on the few-shot classification. In this work, the few-shot classifiers we adopt are prototypical network[44], MANN[39] and MAML[8].

The pipeline of our model is straightforward. We first apply a patch-as-filter layer as described in section 3.1 to obtain feature maps for every image. After that, few-shot classifiers are trained to predict class labels given the feature maps.

The implementation details of the classifiers are discussed in Chapter 4.

Chapter 4

Experiments

In this chapter, we first describe the Synthetic Visual Reasoning Test (SVRT) Same-Different (SD) dataset and our newly generated SD tasks. Then, we briefly introduce the baseline models we adopt. Lastly, the experimental settings and implementation details of our method and several baselines are discussed.

4.1 Dataset

In this thesis, we mainly focused on the SD tasks from the SVRT dataset [9]. We also generated more difficult SVRT-like tasks to examine and analyze our model.

4.1.1 SVRT Same-Different Dataset

To evaluate our approach for abstract visual reasoning, we make use of the SD tasks from the SVRT challenge, which is a collection of 23 binary image classification tasks. Each task contains two classes. We generate data from publicly code[1]. As the task type grouping proposed by [45], the SVRT challenge is split into two groups: Spatial-Relation (SR) tasks

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

and Same-Different (SD) tasks. Solving SR problems requires attention to such features as relative positioning, sizes, alignments, and grouping of figures. The SD problems require comparing individual figures within a single image to identify if they are identical, often under transformation such as scaling or rotation.

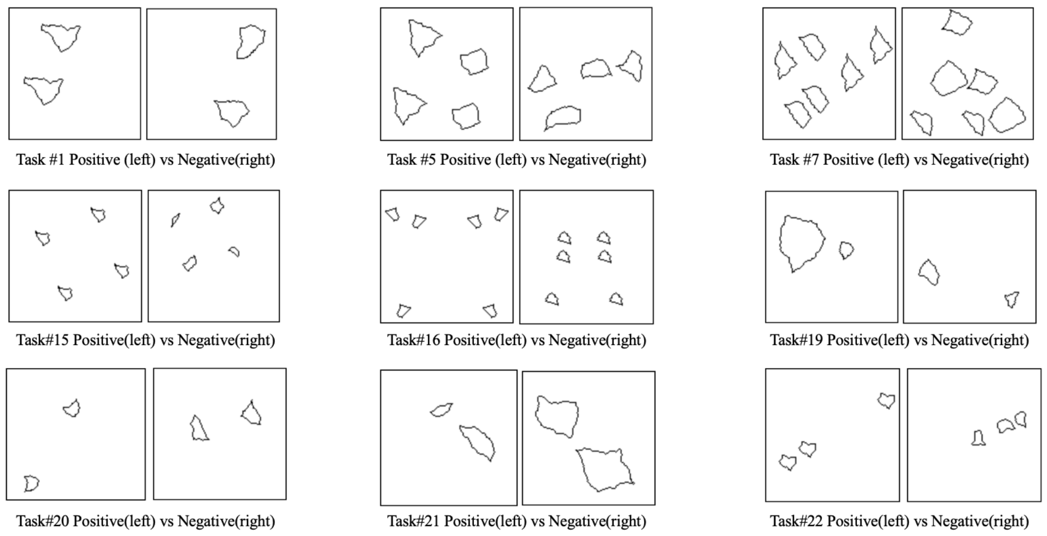

Task #1 Positive (left) vs Negative(right)
Task #5 Positive (left) vs Negative(right)
Task #7 Positive (left) vs Negative(right)
Task#15 Positive(left) vs Negative(right)
Task#16 Positive(left) vs Negative(right)
Task#19 Positive(left) vs Negative(right)
Task#20 Positive(left) vs Negative(right)
Task#21 Positive(left) vs Negative(right)
Task#22 Positive(left) vs Negative(right)

Figure 4.1: Examples from the SVRT Same-Different tasks. For most tasks, their positive classes contain the same shapes, while the negative classes do not. Task 7 is different, as its positive samples contain six shapes that can be organized into two groups of three identical shapes, but its negative samples also have repeated parts, which can be organized into three groups of two identical shapes. Tasks 19, 20 and 21 are considered more complicated tasks that require not only the Same-Different comparison but also the sameness of shapes after rotation and scaling. As reported by [9], tasks 7 and 21 were the most difficult tasks for humans.

In [36], SD tasks are proven to be more difficult, which cannot be solved by existing CNN-based methods nor visual reasoning approaches. Here we adopt the Same-Different and Spatial-Relation split from [36]. Task 1, 5, 7, 15, 16, 19, 20, 21 and 22 from SVRT challenge are considered as the SD tasks that we will train and evaluate on in this work.

As we can see in Figure 4.1, task 1, task 5, task 15 and task 22 are easier. In their negative classes, the shapes are all different, while the positive classes contain duplicate parts. For task 19, task 20 and task 21, the positive samples contain two shapes that are equivalent up to scaling and rotation. Task 7 is much more difficult, as its description is the duplicate degree of an image, and the positive class has two kinds of shapes (3 figures each). However, the negative class also contains duplicate parts and has three kinds of shapes(2 figures each).

Previous work has shown that existing machine learning approaches are capable of solving SR tasks with plenty of training data (20K training images used in [45] and 1 million used in [36]). However, the SD tasks have proven to be more difficult [45] [36]. In contrast, humans have the capability of solving SVRT tasks when seeing at most 20 examples for all the tasks [9].

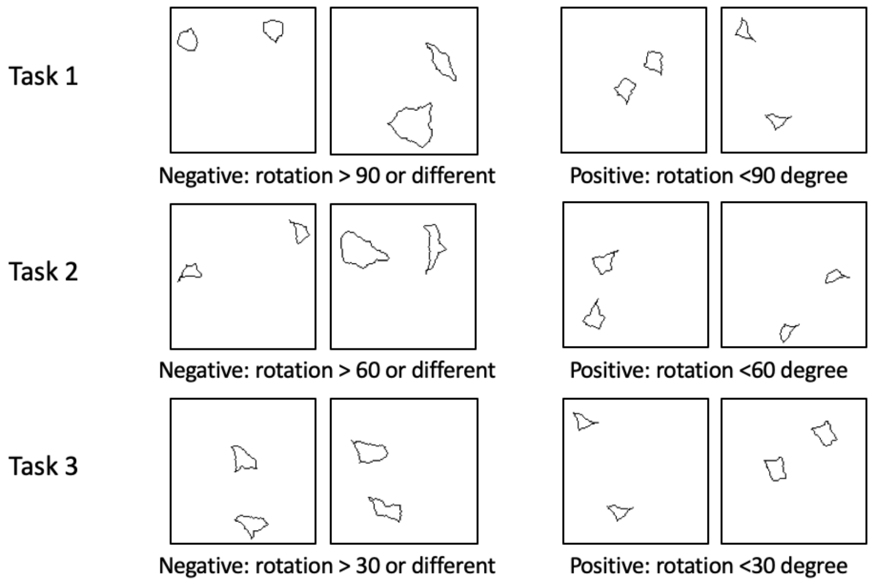

Task 1
Negative: rotation > 90 or different
Positive: rotation <90 degree
Task 2
Negative: rotation > 60 or different
Positive: rotation <60 degree
Task 3
Negative: rotation > 30 or different
Positive: rotation <30 degree

Figure 4.2: Same-Different tasks with two curves that are equivalent up to particular rotations in their positive classes.

4.1.2 More Complicate SVRT-like Data

In order to analyze and test our ability to learn highly abstract concepts in-depth, additional new visual reasoning problems are generated based on more complicated Same-Different concepts. In this section, we describe the details and show some examples of these problems.

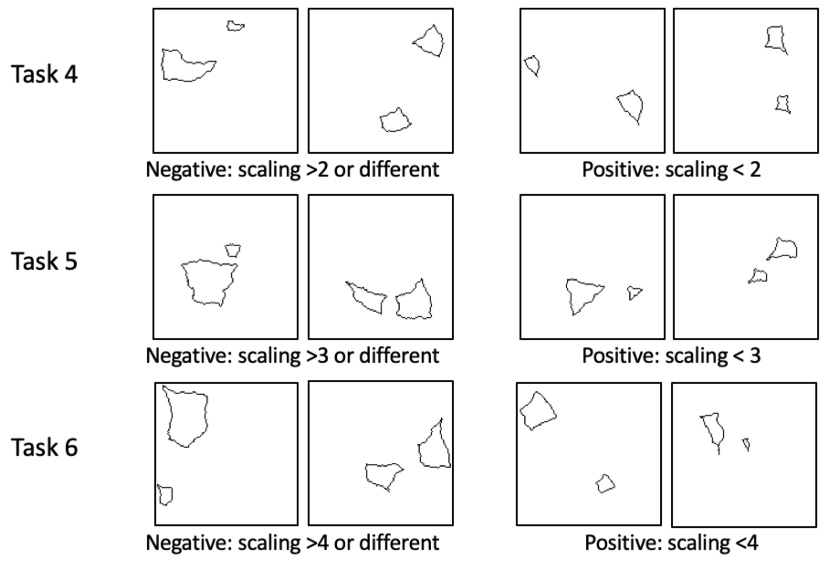

Task 4
Negative: scaling >2 or different
Positive: scaling < 2
Task 5
Negative: scaling >3 or different
Positive: scaling < 3
Task 6
Negative: scaling >4 or different
Positive: scaling <4

Figure 4.3: Same-Different tasks with two curves that are equivalent up to particular scaling in their positive classes.

As we can see, the positive samples of problems in Figure 4.2 and Figure 4.3 contain two same shapes after specific rotation and scaling. In the positive classes of Tasks 1, 2 and 3 (Figure 4.2), two curves are equal after less than 90 degrees, 60 degrees and 30 degrees of rotation, respectively. However, two curves in negative samples are based on other angles of rotations or are different. Similarly, in Tasks 4, 5 and 6 (Figure 4.3), the two curves of the positive samples are the same up to scaling less than two times, three times and four times, and the other relations or two different curves are considered as negative classes.

The following tasks contain the “and” or “or” relations of specific rotation and scaling.

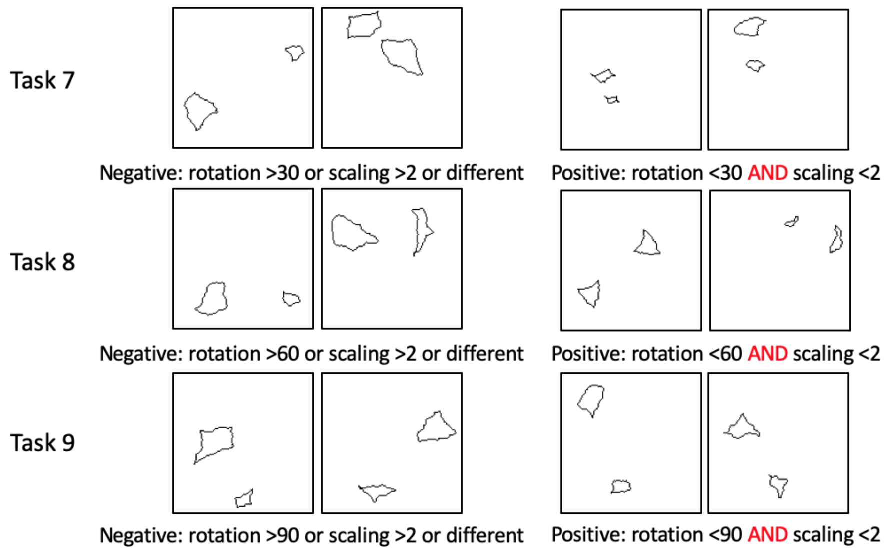

Task 7
Negative: rotation >30 or scaling >2 or different
Positive: rotation <30 AND scaling <2
Task 8
Negative: rotation >60 or scaling >2 or different
Positive: rotation <60 AND scaling <2
Task 9
Negative: rotation >90 or scaling >2 or different
Positive: rotation <90 AND scaling <2

Figure 4.4: Same-Different tasks with two curves that contain "and" logical concepts in their positive classes.

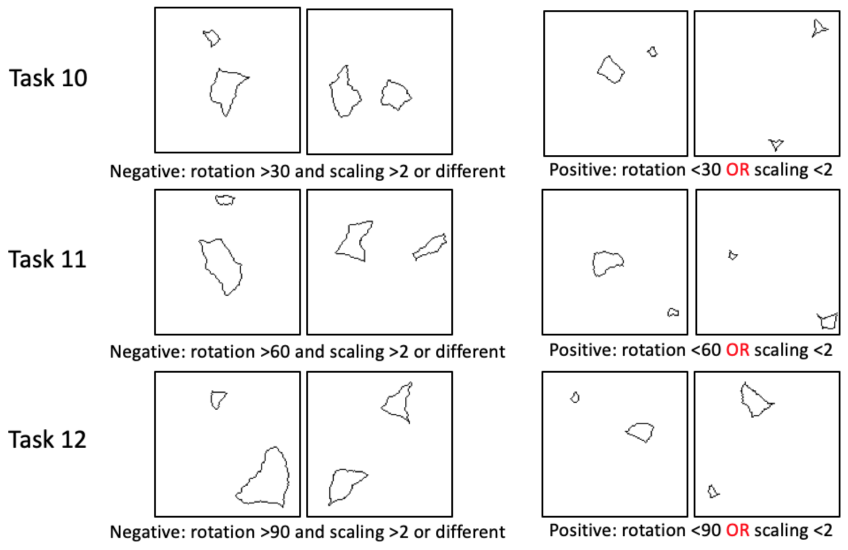

Task 10
Negative: rotation >30 and scaling >2 or different
Positive: rotation <30 OR scaling <2
Task 11
Negative: rotation >60 and scaling >2 or different
Positive: rotation <60 OR scaling <2
Task 12
Negative: rotation >90 and scaling >2 or different
Positive: rotation <90 OR scaling <2

Figure 4.5: Same-Different tasks with two curves that contain "or" logical concepts in their positive classes.

For example, in Task 7 (Figure 4.4), the two shapes of images on the right are equivalent up to less than 30 degrees of rotation and scaling less than two times. While the two shapes of each sample on the left are transferred based on other angles of rotations or scaling larger than two times.

Tasks 8 and 9 (Figure 4.4) are similar to Task 7, in their positive classes, the two shapes are equivalent up to less than 60 degrees and 90 degrees of rotation and scaling less than two times. In Figure 4.5, positive classes of Tasks 10, 11 and 12 have two curves that are equivalent up to less than 30 degrees, 60 degrees and 90 degrees of rotation or scaling less than three times.

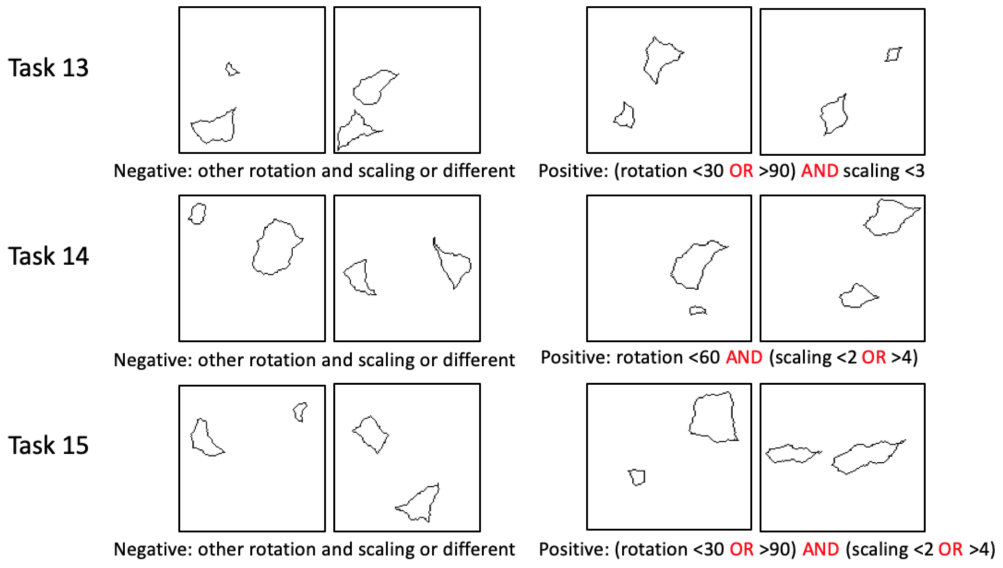

Task 13
Negative: other rotation and scaling or different
Positive: (rotation <30 OR >90) AND scaling <3
Task 14
Negative: other rotation and scaling or different
Positive: rotation <60 AND (scaling <2 OR >4)
Task 15
Negative: other rotation and scaling or different
Positive: (rotation <30 OR >90) AND (scaling <2 OR >4)

Figure 4.6: Tasks with two curves that contain “and” and “or” concepts in their positive classes.

Consequently, Tasks 13, 14 and 15 (Figure 4.6) contain more difficult “and” and “or” concepts between shapes inside each image. In Task 13, the positive class has two shapes, which are the same after rotation less than 30 degrees or greater than 90 degrees and scaling less than three times. While images with two shapes that are different or the same equivalent up to other rotation and scaling are considered are negative class. Similarly, Tasks 14 and 15 also contain combinations of “and” and “or” concepts as described in Figure 4.6.

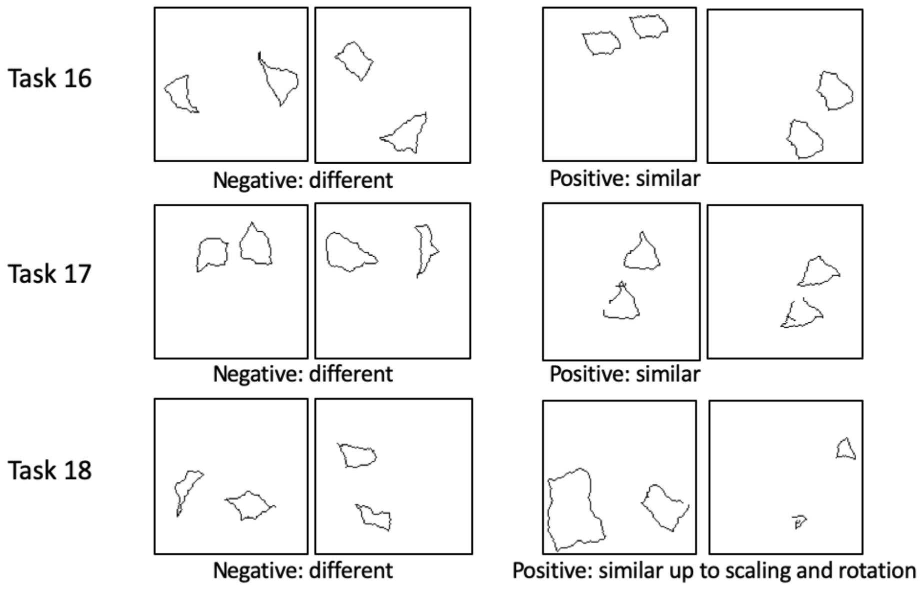


Figure 4.7: Same-Different tasks with two similar curves in their positive classes.

In Figure 4.7, the new Same-Different visual reasoning tasks are generated to be differentiated from the tasks we described above. The positive classes of Tasks 16 and 17 have similar shapes but not exactly the same. Likewise, the shapes in Task 18 are similar after scaling and rotation.

4.2 Baselines and Classifiers

We examine a variety of few-shot learning approaches and a visual reasoning approach in solving the Same-Different tasks. These approaches are proven to perform well on other few-shot learning or visual reasoning tasks. To evaluate the effectiveness of our patch-as-filter convolution block, we also combine it with these CNN-based few-shot classifiers.

4.2.1 Prototypical Networks

This method is proposed by Jake Snell et al. [44] and inspired by Matching Networks [25], which uses an attention mechanism over a learned embedding of labelled samples to predict classes for query points. A neural network is trained to map the labelled points into an embedding space and represent the prototype of each class as the mean of its examples in this learned embedding space. The classification of a new query point is corresponding to the embedding with the closest class prototype (measured with Euclidean distance). This method achieved state-of-the-art results on multiple few-shot tasks, including Omniglot [25], miniImageNet [48], and CUB [49] (a zero-shot learning task).

4.2.2 MANN

This method is proposed by Adam Santoro et al. [39] and inspired by learning to learn (Thrun and Pratt, 1998). In this paper, a high-capacity memory-enhancing neural network (MANN) is proposed to re-examine meta-learning problems and settings. Compared with previous methods, memory-augmented neural networks (MANN) represent a more widely applicable class of methods. This method was verified on Omniglot [25].

4.2.3 MAML

This method is proposed by C. Finn et al.[8] and inspired by Meta-Learner LSTM [34], which learns an initialization for learners that can solve new tasks. MAML is a generic framework for meta-learning across several aforementioned domains, which learns an update step that a learner can take to be successfully adapted to a new task. The purpose of the MAML algorithm is to learn the weight of a model so that standard gradient descent can make rapid progress on new tasks without over-fitting to a small number of examples.

Excellent results can be achieved on few-shot image recognition tasks by this method, including Omniglot [25], miniImageNet [48].

4.2.4 Relational Network

Relational Network was proposed by Santord et al. [40]. In this paper, a pairwise comparison over spatial locations method called RN is proposed. RN utilizes an MLP to carry out pairwise comparisons over each location of the convolution feature extracted from the image, including the question features extracted from LSTM as input to this MLP. The RN elements then sum up the resulting comparison vectors to form another vector for a final classifier to predict the answer. This approach is end-to-end differentiable, and trainable from scratch to high performance. This method achieved state-of-the-art, super-human performances on multiple visual reasoning tasks, including CLEVR [17], bAbI [51], and Sort-of-CLEVR [40].

4.3 Implementation Details

In this section, we describe the implementation and hyperparameters chosen details for each model in our experiments. For each Same-Different task, we divided the data into a training set, a validation set and a test set. The image size we use for all models is 128×128. For the training stage, we train all models with only 10 labelled samples. For hyperparameter tuning, we average across five trials for each measurement, with 1000 validation images for each task. For the test stage, we test on 1000 unseen images. The performance is measured with classification accuracy. The results in Chapter 5 are averaged across 10 trials. Next, we will discuss the details of the structure we used and the hyperparameters tuning settings of each classifier.

4.3.1 Our Model

In addition to the description in Chapter 3, we will include other experimental setting details about our model in this section. A patch-as-filter convolutional block is created to extract the Same-Different relations of shapes in input images and present the relations by the output feature maps. In the convolutional layer, when we generate sample-specific filters, the number of filters to be created is set to 64. In the max-pooling layer, the filter size is 4×4, and the stride is set to 4. Thus, the feature maps are downsized from 128×128 to 32×32.

4.3.2 Prototypical Networks

We use the same architecture for the embedding stage of this approach as in [44] and [48]. [2] In each training epoch, we randomly selected three training samples from each class as the support set of one class, and use the other two as query points. In the test stage, we also use three images per class from the training data set as its support set and 1000 test samples as a query set. The optimal number of epochs was chosen from $\{16, 32, 64\}$ to maximize the average score on the SD problems, where 32 epochs were found to perform best with raw images and 64 epochs for feature maps after patch-as-filter layer.

4.3.3 MANN

For this model, we use the public implementation [3] of the paper One-shot Learning with Memory-Augmented Neural Networks [39]. We use 200 as the number of hidden units in the LSTM controller network. The optimal number of epochs for the Same-Different SVRT task was chosen from $\{16, 32, 64\}$, where 64 epochs work for raw images and feature maps

[2] See `https://github.com/orobix/Prototypical-Networks-for-Few-shot-Learning-PyTorch`.
[3] See `https://github.com/tristandeleu/ntm-one-shot`.

after patch-as-filter layer.

4.3.4 MAML

We use a public implementation [4] of the supervised learning experiments from the paper Model-Agnostic Meta-Learning [8]. We use the same model as in the paper for Omniglot [25] tasks. The model was optimized with a cross-entropy loss function using the SGD optimizer with a learning rate of 0.001, weight decay 0.0005. The optimal number of epochs was chosen from $\{16, 32, 64\}$ to maximize the average score on the SD problems, where 32 epochs were found to perform best with raw images and 16 epochs for feature maps after patch-as-filter layer.

4.3.5 Relation Networks

For this model, we use the public implementation [5] of the model for sort-of-CLVER in the paper [40]. The convolutional network component of the model has for convolutional layers with kernel sizes of 3×3. There were 24 features per layer. The softmax output was optimized with a cross-entropy loss function using the Adam optimizer with a learning rate chosen from $\{0.0001, 0.00025, 0.0005\}$. For raw images, the best learning rate is 0.00025. Moreover, the optimal number of training epochs was chosen from $\{64, 128, 256\}$, where we found 128 epochs works best. This model is designed initially for sort-of-CLEVR tasks, which contains images, questions and answers. In order to better apply this model on the Same-Different tasks, we modified the model by taking off all the vectors for questions from networks. Since this method was designed for visual reasoning tasks but not few-shot learning, we only use this model as a baseline to compare with our approach, but not apply it as a classifier.

[4]See https://github.com/katerakelly/pytorch-maml.

[5]See https://github.com/kimhc6028/relational-networks.

Chapter 5

Results and Discussion

In this chapter, we present and analyze the accuracy of our model as well as several few-shot methods on the SVRT dataset and our newly generated dataset. How the number of filters affects the classification accuracy is also discussed in this chapter.

5.1 Results on SVRT Dataset

In this section, we present the results of several few-shot learning models and a visual reasoning model on SVRT SD tasks. To show the effectiveness of our proposed approach, we also demonstrate the results of our proposed method combined with these few-shot learning approaches. The meanings of model abbreviations are as follows. **MANN**: Memory-Augmented Neural Network. **MAML**: Model-Agnostic Meta-Learning. **PN**: Prototypical Network. **RN**: Relation Network. **MANN+PFL**: the combination of patch-as-filter convolutional layer and Memory-Augmented Neural Network. **MAML+PFL**: the combination of patch-as-filter convolutional layer and Model-Agnostic Meta-Learning. **PN+PFL**: the combination of patch-as-filter convolutional layer and Prototypical Network. **Fleuret et al.** refers to the Adaboost+spectral features model from [9]. The feature type abbreviations

Input	MAML	MANN	PN
raw	50.1%	50.2%	50.3%
PFL	78.5%	75.9%	80.2%

Table 5.1: An overview of all experiment results on SVRT SD tasks.

are as follows. **Raw**: original input images. **PFL**: feature maps provided by patch-as-filter convolutional layer as input.

Table 5.1 presents an overview of our results. The test results for each of the three few-shot classifiers we applied are averages across all SD tasks and evaluate with two feature sets (the original image and the feature maps provided by our patch-as-filter layer). Concerning the SVRT same-different tasks, our proposed method outperforms the baseline features over 25% with all classifiers. The prototypical network achieves the best performance, which is 80.2%. Compared to the average results with raw input images (50.3%), our results can achieve about 30% higher average accuracy.

With the raw images as input, the average results of all the few-shot classifiers are around 50%. These few-shot methods have achieved the state-of-the-art performance on other few-shot tasks, as we discussed in Chapter 4. The results we obtained show that solving the SD tasks with few-shot learning is challenging for standard CNN based methods, which occurs with the conclusion from Fleuret [9] and Stabinger [45]. With our method, these few-shot classifiers can achieve higher accuracy because we specially designed this method for solving SD tasks, and the feature maps provided by our patch-as-filter layer can correctly indicate the repeated objects in images.

More details of our results can be found in Table 5.2 and 5.3. All results are averaged across 10 trials except the results from [9], where the number of trials is unknown. The highest score on each task is in bold. As we can see, Task 15 possesses the best accuracy score, which is 95.5%. Task 15 contains four shapes in positive and negative classes.

SD Task	**Fleuret et al.**	**MANN**	**MAML**	**PN**	**RN**
1	53.0%	50.7%	50.1%	51.2%	50.0%
5	47.0%	50.3%	50.7%	50.6%	50.2%
7	47.0%	50.4%	49.9%	50.0%	49.8%
15	54.0%	50.5%	50.4%	50.1%	51.0%
16	62.0%	50.7%	50.7%	50.2%	50.4%
19	51.0%	49.7%	49.7%	50.5%	50.9%
20	48.0%	50.9%	50.9%	49.7%	50.3%
21	39.0%	49.3%	49.3%	49.7%	48.9%
22	53.0%	49.9%	50.2%	50.8%	50.7%
Average	50.4%	50.1%	50.2%	50.3%	50.2%

Table 5.2: Existing results (Fleuret et al. [9]) and our experiment results of three popular few-shot learning method and a well-performed visual reasoning method with raw input images. 10 labelled training samples are used in all the experiments.

The four shapes in positive class are all the same, while in negative class are all different from each other, so our method can effectively distinguish them. A significant increase of accuracy can be found in Task 1, 5, 7, 16 and 22 by applying our approach since their positive classes always contain more shapes that are precisely the same, while the negative samples are not.

SD Task	**MAML+PFL**	**MANN+PFL**	**PN+PFL**
1	89.7%	88.5%	**91.3%**
5	90.1%	88.1%	**92.5%**
7	81.0%	79.3%	**80.2%**
15	91.2%	86.4%	**95.5%**
16	87.8%	81.3%	**89.9%**
19	62.7%	63.2%	**68.4%**
20	57.1%	58.0%	**61.3%**
21	62.5%	64.1%	**65.2%**
22	91.2%	87.2%	**93.8%**
Average	78.5%	75.9%	**80.2%**

Table 5.3: By applying our patch-as-filter method with all three CNN-based few-shot classifiers, the results outperform the existing results and results of few-shot classifiers with raw images on every SD task.

It is noticeable that for task 19, task 20, and task 21, the rises of accuracy are less than other tasks (compare with [9] and all the few-shot learning methods). As presented in Figure 4.1, in task 19, each image contains two shapes, which are equivalent up to scaling in the positive class. As to the two shapes in the positive class of task 20, one shape can be obtained from the other by reflection. In task 21, one of the shapes in positive class can be obtained from the other by scaling and rotating. These three tasks contain "sameness" relations up to different kinds of transformations and are considered as a harder subset of the SVRT dataset.

We also compare the results of our method with another image preprocessing technique [1], which removes peaks in the amplitude spectrum of an image by percentile filtering to remove the non-unique parts of the image. Then by using these spectral features, this method can find the samples with unique parts in them. As shown in Table 5.4, the first two columns (MAML+A_p and PN+A_p) are the results of two CNN-based few-shot classifiers utilizing spectral features. The accuracy of the same few-shot classifiers using features provided by our method is presented in the last two columns in Table 5.4. With every few-shot classifier, the average accuracy over all SD tasks of our method surpasses the results with spectral features.

In Task 7, our method achieves over 20% higher accuracy than the spectral features method. In Task 7, each image contains six shapes. The shapes can be organized into three groups, each consisting of two identical shapes in negative class, while in positive class, they can be organized into two groups of three identical shapes [9]. The positive and negative classes in Task 7 both contain duplicate parts, so the shapes in both classes are non-unique parts and removed by the spectral features method. However, our approach can correctly distinguish two classes by finding three duplicate parts and two duplicate parts in positive and negative classes, respectively.

We observe that the spectral features method has better performance on Task 1, Task

SD Task	MAML+A_p	PN+A_p	MAML+PFL	PN+PFL
1	**99.8%**	99.6%	89.7%	91.3%
5	92.5%	**95.9%**	90.1%	92.5%
7	58.4%	58.6%	81.0%	**80.2%**
15	**100.0%**	99.6%	91.2%	95.5%
16	96.3%	**97.3%**	87.8%	89.9%
19	56.9%	56.1%	62.7%	**68.4%**
20	56.0%	55.5%	57.1%	**61.3%**
21	50.6%	49.6%	62.5%	**65.2%**
22	95.3%	**95.7%**	91.2%	93.8%
Average	78.4%	78.7%	78.5%	**80.2%**

Table 5.4: A comparison of an existing method utilizing percentile-filtered amplitude spectra features [1] to our method with two CNN-based few-shot classifiers. Our method achieve higher accuracy on average and several harder SD tasks. Both models are trained on 10 training samples and tested on 1000 images. All the results are averaged on 10 trails.

5, Task 15, Task 16 and Task 22. In positive classes for these tasks, some numbers of identical shapes are present, while in negative classes, the shapes are all unique. Their method performs particularly well on these purely same-different problems.

For the tasks contain shapes that are equivalent up to rotation or scaling transformations, such as Task 19, Task 20 and Task 21, the few-shot classifiers obtain better results with the features provided by our patch-as-filter method. Our method dynamically generated sample-specific filters by rotation and scaling learned from training samples. Therefore, our method is good at solving these tasks with harder same-different concepts.

5.2 Results on More Complicated SVRT-like Data

In this section, we analyze the performance of our model on new-generated SD tasks. Table 5.5 demonstrates a high-level overview of the results of new-generated tasks. The test results for each of the three few-shot classifiers are averages across all the tasks and evaluate with two feature sets: the original image and the feature maps provided by our patch-as-filter layer.

Input	**MAML**	**MANN**	**PN**
raw	49.5%	49.4%	50.4%
PFL	60.6%	59.2%	61.6%

Table 5.5: An overview of all experiment results of new generated tasks.

Prototypical Network approach achieves the highest average accuracy both with raw feature inputs and with our approach. The results demonstrate the superiority of PFL features over the raw images with the performances obtained by the two feature types, as the rate increased approximately by 10% across all classifiers from 50% to about 60%. Our method achieves higher accuracy since our patch-as-filter layer is specially designed for SD tasks, which generates sample-specific filters for each input image.

More details of our results can be found in Table 5.6 and Table 5.7. Task 1 to Task 6 contains two shapes in each image (examples of these tasks are shown in Figure 4.2 and Figure 4.3). In their positive classes, the two shapes are the same after a particular rotation or scaling rule, such as rotation smaller than 90 degrees or scaling less than 3 times. Their negative classes contain two kinds of samples. The two shapes in the first kind are totally different. The second kind of images in negative classes contain two shapes that are equivalent up to some transformations but not meet certain rules in the positive classes. The accuracy of these tasks increases by over 10%. From Task 1 to Task 6 are considered as easier tasks in our newly created dataset.

For Task 16, Task 17 and Task 18 (shown in Figure 4.7), the two shapes in positive classes are similar (not the same). Our method performs well on Task 16 and Task 17, whose classification accuracy increased above 35%. The positive classes of these tasks contain two shapes having 15 different pixels, while two shapes are entirely different in negative classes. The increase of the classification accuracy of Task 18 is less than Task

SD Task	MAML	MANN	PN	RN
1	47.9%	51.7%	48.3%	47.1%
2	44.6%	47.8%	51.2%	46.5%
3	51.6%	51.1%	51.9%	52.7%
4	49.8%	50.3%	48.1%	49.3%
5	49.6%	47.4%	51.7%	50.7%
6	49.1%	49.2%	48.8%	51.1%
7	48.6%	50.4%	49.6%	48.9%
8	51.0%	48.6%	51.4%	47.1%
9	48.1%	48.3%	50.3%	47.6%
10	49.9%	47.8%	50.3%	51.9%
11	51.9%	49.7%	51.0%	52.7%
12	50.7%	49.0%	51.0%	49.9%
13	48.2%	48.2%	51.8%	50.0%
14	52.1%	50.1%	51.1%	51.8%
15	49.7%	48.8%	50.7%	52.2%
16	51.1%	51.5%	49.2%	49.1%
17	49.3%	48.6%	51.4%	50.4%
18	47.3%	50.3%	49.7%	51.0%
Average	49.5%	49.4%	50.4%	50.0%

Table 5.6: A comparison of three few-shot methods and a visual reasoning method on new SD tasks. 10 labelled training samples are used in all the experiments.

16 and Task 17 since its positive class have two shapes that are similar up to rotation and scaling. Our method identifies the "sameness" based on transformations of two shapes in a picture by generating filters with rotation and scaling of the shape we select from the picture. The number of filters we generated for each task is 64. Therefore, for all tasks involving rotation and scaling transformations, our method is capable of achieving higher accuracy but not significant.

As shown in Table 5.6 and Table 5.7 that the rises of accuracy are less than 10% for Task 7 to Task 12 (compare with all the few-shot learning baselines). The positive samples of these tasks all contain two shapes, which are equivalent up to rotation and scaling. In Task 7, Task 8 and Task 9 (shown in Figure 4.4), the rotation and scaling transformation have

SD Task	MAML+PFL	MANN+PFL	PN+PFL
1	62.5%	60.6%	**63.8%**
2	62.9%	61.1%	**64.2%**
3	**64.4%**	60.1%	64.2%
4	60.5%	**61.3%**	61.2%
5	59.3%	58.9%	**61.0%**
6	60.2%	59.4%	**60.6%**
7	57.9%	56.3%	**58.6%**
8	55.8%	55.2%	**56.7%**
9	55.3%	54.1%	**57.2%**
10	54.7%	52.5%	**55.7%**
11	55.3%	53.8%	**56.1%**
12	53.7%	52.3%	**54.6%**
13	54.1%	52.7%	**55.0%**
14	**53.6%**	52.3%	53.4%
15	**52.2%**	51.6%	52.1%
16	90.1%	88.4%	**92.2%**
17	85.2%	84.3%	**89.3%**
18	52.5%	51.2%	**53.4%**
Average	60.6%	59.2%	61.6%

Table 5.7: Detail results for individual tasks. By using our patch-as-filter method with all three CNN-based few-shot classifiers, the results outperform the baselines on every task, with the Prototypical Network approach achieving the highest average accuracy.

"AND" logical relations; as to Task 10, Task 11 and Task 12 (shown in Figure 4.5), the rotation and scaling transformation have "OR" logical relations in their positive classes. The increases in the accuracy of these tasks are less than others because the abstract concepts in these six tasks are more complicated than the concepts in Task 1 to Task 6.

Task 13, Task 14 and Task 15 (shown in Figure 4.6) contain more complicated abstract concepts. The combinations of "AND" and "OR" relations can be found in the rotation and scaling transformations in their positive classes. There is no noticeable increase in the accuracy of these tasks. Task 13, Task 14 and Task 15 are considered as a harder subset of our new dataset that cannot be solved by our approach. To better solve these more challenging same-different tasks, we can increase the number of filters for each image. We will analyze the effects of the number of filters in the next section.

5.3 Robustness Analysis

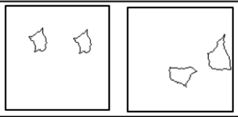
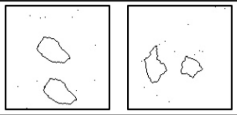
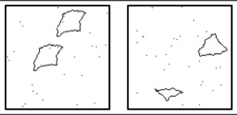
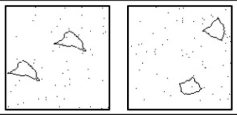
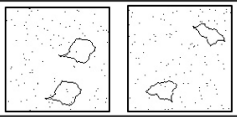
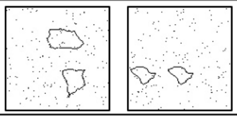
Description	Examples	Accuracy
Original SVRT Task #1		91.3%
Task #1 with salt and pepper noise		91.7%
		89.9%
		90.4%
		85.3%
		82.6%

Figure 5.1: Examples and classification results of original Task 1 and Task 1 with noises.

To test and analyze the robustness of our model, we add different degrees of noise to the SVRT Task 1 and compare their classification results. In most cases, robustness is inversely proportional to accuracy. So we select Task 1 in this analysis, which has achieved very high accuracy by our method with prototypical network classifier. As shown in Figure 5.1, the first row is the original Task 1, which obtains 91.3% classification accuracy. Other rows are examples of Task 1 added with different numbers of salt and pepper noises. The probabilities of the noises in each row are 0.001, 0.002, 0.004, 0.008, 0.0016. As we can see, with a small number of noises, the accuracy is still around 90%. However, when we apply more noises to each image, the classification accuracy decrease to 85.3% and 82.6%. The results indicate that our method is stable and robust to a small number of noises, but when the noise increases, it will affect the classification accuracy of our model.

5.4 Effects of the Amount of Filters

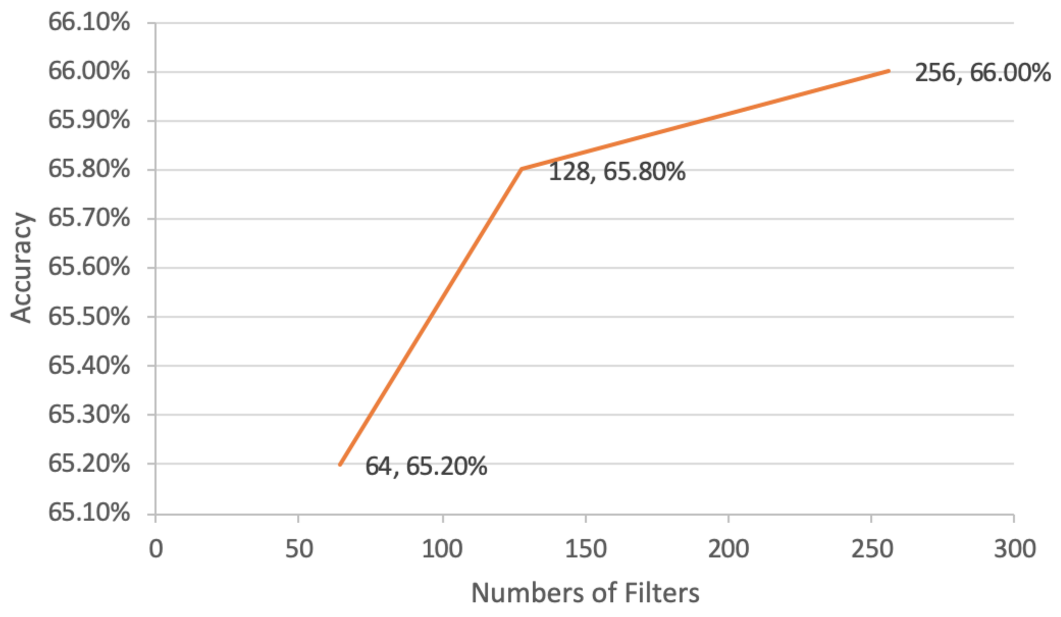


Figure 5.2: Accuracy score of Task 21 with different numbers of filters.

For all the experiments we described above, we set the number of filters to 64 for calculation efficiency, which means we only generate 64 filters in our patch-as-filter convolutional layer. In this section, we discuss and analyze the effect of the number of filters. Task 21 in the SVRT dataset is applied as an example since this task contains both rotation and scaling transformations in its positive class. The numbers of filters we choose to compare were 64, 128 and 256.

Obviously, with the increasing of the number of filters, the accuracy score goes up (Figure 5.2). The accuracy increases from 65.2% with 64 filters to 66.0% with 256 filters. The results show that our method is capable of achieving higher experimental accuracy with more filters. This result further proves the effectiveness of our approach. Moreover, future work can increase the classification accuracy by applying more filters.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We propose a patch-as-filter method to solve Same-Different (SD) tasks with few-shot learning. SD task is a kind of visual reasoning task, which requires machines to learn highly abstract visual reasoning concepts for each task. SD tasks are simple for humans but are proven to be challenging for general machine learning methods. Therefore, solving these tasks in few-shot cases is extremely challenging work.

Our patch-as-filter method is based on the idea that we can learn particular same-different rules by generating sample-specific filters. The selective search is applied to seek for shapes in each image, and then we create filters via learned transformations of this shape. By applying filters on the input image through a convolutional layer, feature maps are provided containing intuitive information of duplicate parts in this image. Classification is then performed by putting feature maps into well-performed few-shot learning classifiers.

In the experiments and results section, the performance of our approach and the baselines are demonstrated. More importantly, it has been proven that the state-of-the-art few-shot classifiers and a popular visual reasoning architecture relation networks exhibit poor performance on Same-Different (SD) tasks, which concurs with previous work suggesting that CNNs struggle with solving this kind of tasks. By combining our approach with these few-shot classifiers, our model exceeds the existing comparable state-of-the-art on the SVRT SD tasks in the few-shot cases. The affection of the number of filters we generated is also discussed in the results part, which will inspire future development in our patch-as-filter method.

Furthermore, additional new SVRT-like tasks are created with harder same-different rules, such as particular rotation and scaling. The baselines of our newly generated dataset are established by testing on several popular few-shot learning methods. Experimental results indicate that our method can significantly improve the accuracy of new tasks compared with these baselines.

In conclusion, we propose an efficient technique to learn highly abstract visual concepts such as “sameness”, “difference”, and “invariance under rotation and scaling” by few-shot learning. We compare our results with existing work and establish several few-shot learning methods on the SVRT dataset. The effectiveness of our method is also proved by the experimental results of our new dataset.

6.2 Future Work

We believe that this work is one step toward a simple technique that can be applied to the Same-Different visual reasoning task. However, more topics in this field could be further explored.

First of all, we use the selective search to find a patch in the original input images. Further research can be launched by using more advanced methods to extract features for the filter generating process in order to get higher classification accuracy.

Second, since we generate sample-specific filters for each training and test samples, our model takes a long time to run. Further research can focus on how to process the images more efficiently.

Lastly, we can also test our method on more real-world datasets. This study can have many practical implications, such as finding transformation invariant same-different relations in texture analysis and locating a person from several people in face recognition. When we apply our model on real-world problems, we need to adopt more advanced methods to select regions of the interest, such as Mask R-CNN [13], to extract the correct patch for comparison. Moreover, we can improve our model to extract all the objects in each image for better applications on real-world problems. The classification accuracy may not as high as what we achieved on the SVRT SD dataset since real-world images contain much more transformations of objects other than rotation and scaling. In future work, we will improve our model to explore more applications for this Same-Different visual reasoning research.

Bibliography

[1] Tanner Bohn, Yining Hu, and Charles X Ling. Few-shot abstract visual reasoning with spectral features. *arXiv preprint arXiv:1910.01833*, 2019.

[2] Mikhail M Bongard. The problem of recognition. *Fizmatgiz, Moscow*, 1967.

[3] Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet. In *International Conference on Learning Representations*, 2019.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[5] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European conference on computer vision*, pages 48–64. Springer, 2014.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[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] 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*, ICML'17, pages 1126–1135. JMLR.org, 2017.

[9] 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*, 2011.

[10] 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.

[11] Christina M Funke, Judy Borowski, Karolina Stosio, Wieland Brendel, Thomas SA Wallis, and Matthias Bethge. The notorious difficulty of comparing human and machine perception. *arXiv preprint arXiv:2004.09406*, 2020.

[12] Çaglar Gülçehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. *CoRR*, abs/1301.4083, 2013.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[15] 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.

[16] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 804–813, 2017.

[17] 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.

[18] Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Ákos Kádár, Adam Trischler, and Yoshua Bengio. Figureqa: An annotated figure dataset for visual reasoning. *CoRR*, abs/1710.07300, 2017.

[19] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[21] Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, June 2013.

[23] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, 2017.

[24] 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.

[25] B.M. Lake, R Salakhutdinov, J Gross, and J.B. Tenenbaum. One shot learning of simple visual concepts. *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, 01 2011.

[26] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

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

[28] 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.

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[31] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *CoRR*, abs/1306.5151, 2013.

[32] Mateusz Malinowski and Mario Fritz. Towards a visual turing challenge. *arXiv preprint arXiv:1410.8027*, 2014.

[33] John C Raven et al. *Raven's progressive matrices*. Western Psychological Services Los Angeles, CA, 1938.

[34] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[35] Revanth Reddy, Rahul Ramesh, Ameet Deshpande, and Mitesh M Khapra. Figurenet: A deep learning model for question-answering on scientific plots. *arXiv preprint arXiv:1806.04655*, 2018.

[36] Matthew Ricci, Junkyung Kim, and Thomas Serre. Not-so-clevr: Visual relations strain feedforward neural networks. *arXiv preprint arXiv:1802.03390*, 2018.

[37] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.

[38] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Rotation invariant neural network-based face detection. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*, pages 38–44. IEEE, 1998.

[39] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1842–1850, New York, New York, USA, 20–22 Jun 2016. PMLR.

[40] Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427, 2017.

[41] Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 May 1987.

[42] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

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

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

[45] 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.

[46] 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.

[47] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, Sep 2013.

[48] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016.

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

[50] Yaming Wang, Vlad I. Morariu, and Larry S. Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[51] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[52] Chao Xiong, Xiaowei Zhao, Danhang Tang, Karlekar Jayashree, Shuicheng Yan, and Tae-Kyun Kim. Conditional convolutional neural network for modality-aware face

recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3667–3675, 2015.

[53] Lu Yihe, Scott C Lowe, Penelope A Lewis, and Mark CW van Rossum. Program synthesis performance constrained by non-linear spatial relations in synthetic visual reasoning test. *arXiv preprint arXiv:1911.07721*, 2019.

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

Appendix A

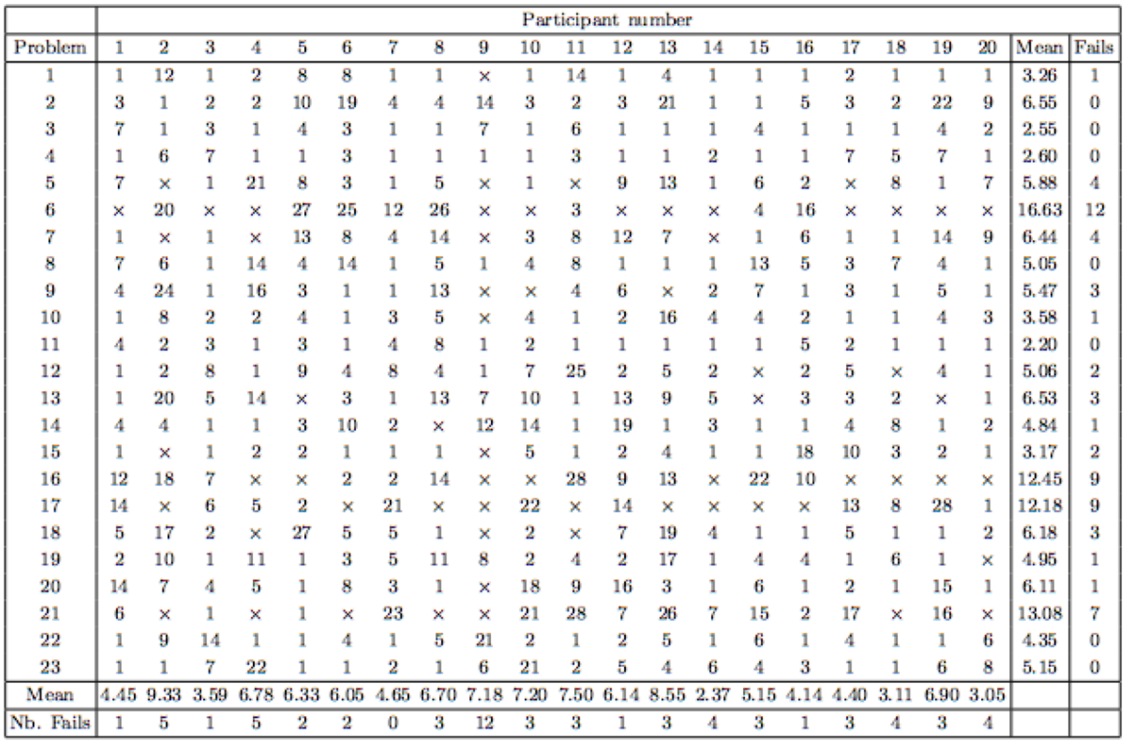
Other Results on the SVRT Dataset

A.1 Results on the SVRT SR Tasks

SD Task	Fleuret et al.	MAML	PN	RN	MAML+PFL	PN+PFL
2	55.0%	52.0%	53.5%	51.2%	50.6%	49.7%
3	54.0%	50.4%	52.9%	50.2%	50.7%	49.5%
4	56.0%	54.2%	50.4%	51.5%	49.0%	50.1%
6	50.0%	50.7%	49.7%	50.9%	49.9%	55.2%
8	57.0%	61.7%	50.8%	52.6%	68.8%	70.1%
9	52.0%	50.9%	51.0%	50.7%	49.8%	59.8%
10	50.0%	54.7%	51.7%	50.6%	57.8%	56.4%
11	52.0%	52.2%	51.1%	53.8%	53.7%	68.3%
12	46.0%	50.2%	51.8%	50.6%	50.4%	50.3%
13	50.0%	49.6%	50.3%	50.7%	57.6%	56.5%
14	51.0%	51.4%	50.5%	48.7%	51.0%	52.2%
17	59.0%	53.4%	47.2%	49.7%	50.5%	55.8%
18	50.0%	57.0%	49.8%	52.2%	58.1%	60.7%
23	53.0%	50.8%	47.9%	50.6%	50.0%	50.4%
Average	52.5%	52.8%	50.6%	51.1%	53.4%	56.1%

Table A.1: Comparing existing results (Fleuret et al. [9]) and four new baselines with the results of using our method with few-shot classifiers. 10 labelled training samples are used in all the experiments.

A.2 Human Performance on the SVRT Tasks



	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 A.1: Completed experimental results with humans(image from [9]). 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 [9].

Curriculum Vitae

Name:	Yining Hu
Post-Secondary Education and Degrees:	Northeastern University Shenyang, Liaoning, China 2013 - 2017 B.A.
Honours and Awards:	Western Graduate Research Scholarships(WGRS) 2018-2019
Related Work Experience:	Teaching Assistant The University of Western Ontario 2018 - 2019

Publications: