Using Capsule Networks

for Image and Speech Recognition Problems

by

Yan Xiong

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2018 by the
Graduate Supervisory Committee:

Chaitali Chakrabarti, Co-Chair
Visar Berisha, Co-Chair
Yang Weng

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

In recent years, conventional convolutional neural network (CNN) has achieved outstanding performance in image and speech processing applications. Unfortunately, the pooling operation in CNN ignores important spatial information which is an important attribute in many applications. The recently proposed capsule network retains spatial information and improves the capabilities of traditional CNN. It uses capsules to describe features in multiple dimensions and dynamic routing to increase the statistical stability of the network.

In this work, we first use capsule network for overlapping digit recognition problem. We evaluate the performance of the network with respect to recognition accuracy, convergence and training time per epoch. We show that capsule network achieves higher accuracy when training set size is small. When training set size is larger, capsule network and conventional CNN have comparable recognition accuracy. The training time per epoch for capsule network is longer than conventional CNN because of the dynamic routing algorithm. An analysis of the GPU timing shows that adjusting the capsule structure can help decrease the time complexity of the dynamic routing algorithm significantly.

Next, we design a capsule network for speech recognition, specifically, overlapping word recognition. We use both capsule network and conventional CNN to recognize 2 overlapping words in speech files created from 5 word classes. We show that capsule network achieves a considerably higher recognition accuracy (96.92%) compared to conventional CNN (85.19%). Our results show that capsule network recognizes overlapping word by recognizing each individual word in the speech. We also verify the

scalability of capsule network by increasing the number of word classes from 5 to 10. Capsule network still shows a high recognition accuracy of 95.42% in case of 10 words while the accuracy of conventional CNN decreases sharply to 73.18%.

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to my advisors, Dr. Chaitali Chakrabarti and Dr. Visar Berisha, for offering great support on this thesis research. I sincerely thank them for their invaluable guidance and patience throughout my thesis research. A special thanks to Dr. Visar Berisha for pointing out a perfect direction for this thesis.

I would like to thank my committee members, Dr. Yang Weng for taking the time to give valuable feedback. I am grateful to Dr. Lei Ying for introducing me to this valuable topic and important me with a platform to start the thesis.

I express my sincere thanks to Yiqiu Liu, who gave me guidance and suggestion on algorithm and coding. I would also like to thank Huan Song who helped on speech processing algorithm.

I would like to thank my parents for the continuous emotional and financial support they have given me throughout my time of MS research, and thank my girlfriend Jingyi Yuan for stay with me throughout the thesis research.

Finally, I show my highest respect to Dr. Geoffrey E. Hinton for devoting his life in deep learning area for more than thirty years and staying in the area even at the hardest time.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES viii

1 INTRODUCTION 1

1.1 Problem description 2

1.2 Contributions 3

1.2.1 Image recognition task 3

1.2.2 Speech recognition task 4

1.3 Thesis report organization 5

2 BACKGROUND: DEEP LEARNING AND CAPSULE NETWORK 6

2.1 Deep learning and neural network 6

2.2 Convolutional neural network 7

2.3 Capsule network 11

3 CAPSULE NETWORK FOR OVERLAPPING DIGIT RECOGNITION 17

3.1 Structure of capsule network 17

3.2 Baseline CNN 18

3.3 Training on small size training set 19

3.3.1 Experiment design 19

3.3.2 Result 20

3.4 Capsule size 26

3.4.1 Experiment design and result 26

3.4.2 NVVP profiler analysis 29

3.4.2 Summary and results 30

3.5 Conclusion 31

4 CAPSULE NETWORK FOR OVERLAPPING WORD RECOGNITION 32

4.1 Data set and pre-processing 32

4.2 Experiment 1: Design and Result 36

4.2.1 Capsule network 38

4.2.2 Effect of capsule size 39

4.3 Experiment 2 41

4.4 Recognizing individual word 42

5 CONCLUSION AND FUTURE WORK 46

5.1 Conclusion 46

5.2 Future work 48

REFERENCES 49

LIST OF FIGURES

Figure Page

Fig 2.1 From Perceptron to Deep Neural Network 7

Fig 2.2. Convolution between input and kernel array [8] 8

Fig 2.3. Features are extracted layer by layer 9

Fig 2.4. Three neurons judge the existence of digit ‘5’. Each rotates the input and makes prediction by comparing the rotated image to a known image of ‘5’. The max pooling ensures the prediction is correct when the input digit ‘5’ is rotated. 10

Fig 2.5. LeNet-5 structure [5] 11

Fig 2.6. AlexNet Structure [10] 11

Fig 2.7. ResNet Structure [9] 11

Fig 2.8. A face and a re-ordered ‘face’ 12

Fig 2.9. Use of capsules to describe parts of a face (top panel) and recorded face (bottom panel)..... 13

Fig 2.10. Dynamic routing scheme 14

Fig 2.11. Capsule network 15

Fig 2.12. Reconstruction as regularization 16

Fig3.1. Capsule network for digit recognition 18

Fig3.2. Baseline CNN structure 19

Fig3.3. Digits from MNIST and overlapping digits in MultiMNIST 20

Fig 3.4. Performance comparison when training on 50,000 images. 21

Fig 3.5. Performance comparison when training on 100,000 images 21

Fig 3.6. Performance comparison when training on 150,000 images 22

Fig 3.7. Performance comparison when training on 200,000 images.....22

Fig 3.8. Performance comparison when training on 250,000 images.....23

Fig 3.9. Training loss of baseline network for different sized training sets.....24

Fig 3.10. Training loss of capsule network for different sized training sets 24

Fig 3.11. Training loss of capsule network for different primary capsule size 27

Fig 3.12. Training loss of capsule network.....28

Fig 3.13. Time for dynamic routing and number of capsules.....31

Fig 4.1. Wave plot of word ‘backward’ and ‘follow’ 33

Fig 4.2. Wave plot of overlapping ‘backward’ and ‘follow’ 33

Fig 4.3 Filter bank on a Mel scale.....34

Fig 4.4 Block Diagram illustrating calculation of MFCC and filter bank coefficients35

Fig 4.5. MFCCs feature maps.....35

Fig 4.6. Filter bank feature maps 36

Fig 4.7. kernels configuration of convolutional layers 39

Fig 4.8. Original and reconstructed MFCCs feature map of word ‘backward’ 43

Fig 4.9. Original and reconstructed MFCCs feature map of word ‘follow’ 43

Fig 4.10. Original and reconstructed MFCCs feature map of overlapping speech consisting of ‘backward’ and ‘follow’ 44

Fig 4.11. Original and reconstructed Filter bank feature map of word ‘backward’ 44

Fig 4.12. Original and reconstructed Filter bank feature map of word ‘follow’ 44

Fig 4.13. Original and reconstructed Filter bank feature map of overlapping speech consisting of ‘backward’ and ‘follow’.....45

LIST OF TABLES

Table page

Table 3.1. Most time-consuming kernels in training process for baseline network 25

Table 3.2. Most time-consuming kernels in training process for capsule network 25

Table 3.3. Accuracy and training time for different primary capsule size 27

Table 3.4. Accuracy and training time of different class capsule size.....28

Table 3.5. Result of different primary capsule sizes.....29

Table 3.6. Computations in dynamic routing algorithm.....30

Table 4.1. Words in the dataset.....37

Table 4.2 words in speaker dependent dataset.....37

Table 4.3. Accuracy and training time for different configurations 39

Table 4.4. Accuracy and training time for different capsule sizes.....40

Table 4.5. Accuracy and training time for different capsule size 41

Table 4.6. Result of 10-word recognition 42

1. INTRODUCTION

The start of deep learning can be traced back to the invention of the abstract neuron computation (McCulloch-Pitts) model with fixed weight in 1943 [1]. Over the years, neural networks have developed from a simple perceptron model [2] to the complicated deep neural networks (DNN) of today. The number of layers in these networks have increased from tens to hundreds over the years. For instance, the network that won the most recent ImageNet challenge is a neural network with 152 layers. Different types of DNNs have been used successfully in several fields. For example, convolutional neural network (CNN) is widely used in computer vision and autopiloting, and recurrent neural networks (RNN) work well in speech processing and language translating.

In recent years, CNN has shown to have outstanding capability of dealing with information composed of multi-dimensional arrays such as images. In fact, the winners of the ImageNet challenges in recent years have all used large scale CNN [8]. A typical CNN consists of multiple convolutional layers for feature extraction and multiple fully connected layers for classification; the prototype is LeNet-5 proposed by LeCun [5]. CNN makes use of convolution operations to efficiently extract the features from the array input using different kernels and then applies pooling to exploit space invariance property and improve statistical efficiency. Recent ImageNet competitions have validated the outstanding performance of CNN for image recognition. More and more CNN structures are being designed to further improve the performance of tasks related to image and speech recognition. The availability of parallel computers like GPUs has enabled researchers to build and analyze such large scale CNNs with relative ease.

One drawback of traditional CNN is that it does not retain information about important spatial hierarchies between features [20]. So, in applications where spatial information is important, CNN is likely to not do as well. Capsule network [6] proposed in 2016 retains the spatial information and improves the capabilities of traditional CNN. It uses capsules, which is a group of neurons, to describe the features using multiple dimensions. For instance, in overlapping digit recognition, the values in the class capsule represent specific parameters like line weights and radius of a curve. Capsule network also makes use of dynamic routing algorithm, instead of pooling, to increase the statistical efficiency while taking spatial information into consideration. Weights between lower-level capsules and higher-level capsules are adjusted over several iterations. This procedure ensures that higher-level capsules receive more information from lower-level capsules that they "agree with". Capsule network achieved a 0.25% testing error on MNIST without any preprocessing, compared to 0.38% with conventional CNN [6].

1.1 Problem description

Existing work on capsule network [6] has demonstrated its advantage over conventional CNN with respect to digit recognition accuracy. Other aspects such as training time and convergence speed have not been studied. Furthermore, capsule networks have not been used for other applications, such as speech recognition. In fact, in any application where spatial information is of high significance, capsule network is very likely to have superior performance. In speech processing, features such as MFCC are arranged in time order and thus spatial information is available and should be exploited. So, in this thesis, we design and evaluate capsule networks for an image recognition task, namely, recognition of

overlapping digits and for a speech recognition task, namely, recognition of overlapping words.

1.2 Contributions

1.2.1 Image recognition task

In this work, we first consider capsule network for an image recognition problem, specifically, the problem of recognition of overlapping digits. We evaluate the performance of the network in terms of accuracy, training time convergence and computation complexity. For this evaluation, we train the capsule network and a traditional CNN using MultiMNIST, a dataset that includes images of overlapping digits. We record the training time, training and testing error of every epoch and also the best testing accuracy achieved during the training. We also train the networks with different training set sizes.

The results show that for all training set sizes, capsule network converges faster than traditional CNN. The capsule network achieves a higher test accuracy when the training set size is small. When the training set size is larger, the two networks achieve comparable accuracies. In terms of training time, the capsule network takes much longer than traditional CNN per epoch of training. However, the capsule network requires fewer epochs to achieve the same accuracy.

To analyze the computational complexity of the capsule network, we profile the training process using NVVP profiler from NVIDIA. We profile the capsule networks with different primary capsule sizes as well as a traditional CNN. The results from the profiler indicate that all networks spend similar time on convolutional layers and fully connected layers. A traditional CNN spends more time on nonlinear operations like ReLU and pooling

compared to multiplication computations in convolutional layers and fully connected layers. The capsule network spends significantly more time for dynamic routing. It calls extra kernels for matrix-matrix multiplication for calculation in dynamic routing. The analysis also shows that as the size of primary capsules increases, the time for dynamic routing decreases. Next, we change the capsule sizes and study their effect on the training time. The results show that when we increase the size of primary capsules and decrease the size of class capsules, the training time can be decreased by 50% while keep the testing accuracy almost unchanged.

1.2.2 Speech recognition task

Next, we design a capsule network for speech recognition, specially, recognition of two overlapping words from five different classes. The speech files are pre-processed using MFCC and filter bank and then fed into the neural network. To find the best configuration, we test several convolution kernel sizes and compare the accuracy. We find that convolutional layers with rectangular kernels (that spreads wider in temporal dimension) help achieve higher accuracy. We also train a traditional CNN with rectangular kernels and find that the capsule network achieves a much higher accuracy than traditional CNN. In case of speaker-dependent training set and using filter bank for pre-processing, capsule network achieves an accuracy of 96.88% compared to 84.88% of traditional CNN.

Next, we increase the number of classes from 5 to 10 and repeat the experiment. The results show that when the number of classes increase, the capsule network still has a high recognition accuracy (95.42% from 96.88%) while the traditional CNN suffers from a sharp decrease in recognition accuracy (73.18% from 84.88%). We also show that adjusting the capsule size can help decrease the training time of the capsule network.

Specifically, by increasing the primary capsule size and decreasing the class capsule size, the training time per epoch can be decreased from 170s to 113s.

1.3 Thesis report organization

This thesis is organized into the following chapters. Chapter 2 gives a brief introduction of deep learning and neural networks along with a detailed description of the capsule network. Chapter 3 focuses on use of capsule network for the image recognition problem. It evaluates performance and timing results for different configurations and provides comparison with baseline CNN. Chapter 4 focuses on use of capsule network for speech recognition tasks. It too compares the performance of the capsule networks for different configurations with traditional CNN. Chapter 5 concludes the thesis and lists future work in this area.

2. BACKGROUND: DEEP LEARNING AND CAPSULE NETWORK

2.1 Deep learning and neural network

Deep Learning architecture is a computation model which utilizes multiple processing layers to learn representation of data with multiple levels of abstraction [4]. Examples include deep neural network (DNN), convolutional neural network (CNN) and recurrent neural network (RNN). This model has been applied to computer vision, speech recognition and natural language processing and in each case has shown significant improvement in their state-of-the-art performances.

One of the earliest models of neural network is the perceptron [2] proposed in 1958. It is a simple neural network with one layer of input neurons and one layer of computation neurons as shown in Fig 2.1. The network is trained by adjusting the weights of the network. But such a network cannot solve non-linear classification problems like XOR. The two-layer perceptron with two computation layers was capable of solve the non-linear classification problem. But the computations were too complex and there were no efficient algorithms to train such a network at that time [19]. A three-layer network could be trained using the backpropagation algorithm [7] proposed in 1986. But the drawbacks include unacceptable long training time and training process stopping at local optimal points.

What started the new era of deep learning is introduction of Deep Belief Network [3] shown in Fig 2.1. More hidden layers were added into the neural network and a pre-training process was applied instead of randomly initializing the weights. In a three-layer neural network, the hidden layer transformed the input into a space that could be linearly classified, and then the output layer completed the classification task. More layers enabled the

network to transform the input more times and thereby handle more complex classification tasks. The pre-training process introduced in [3] was designed to solve the problem of local optimal points. The model would start from some point close to the global optimal instead of some random point. This method largely accelerated the training process. With the development of stochastic gradient descent and powerful computers designed for linear algebra operations, training time of DNNs decreased from days to hours.

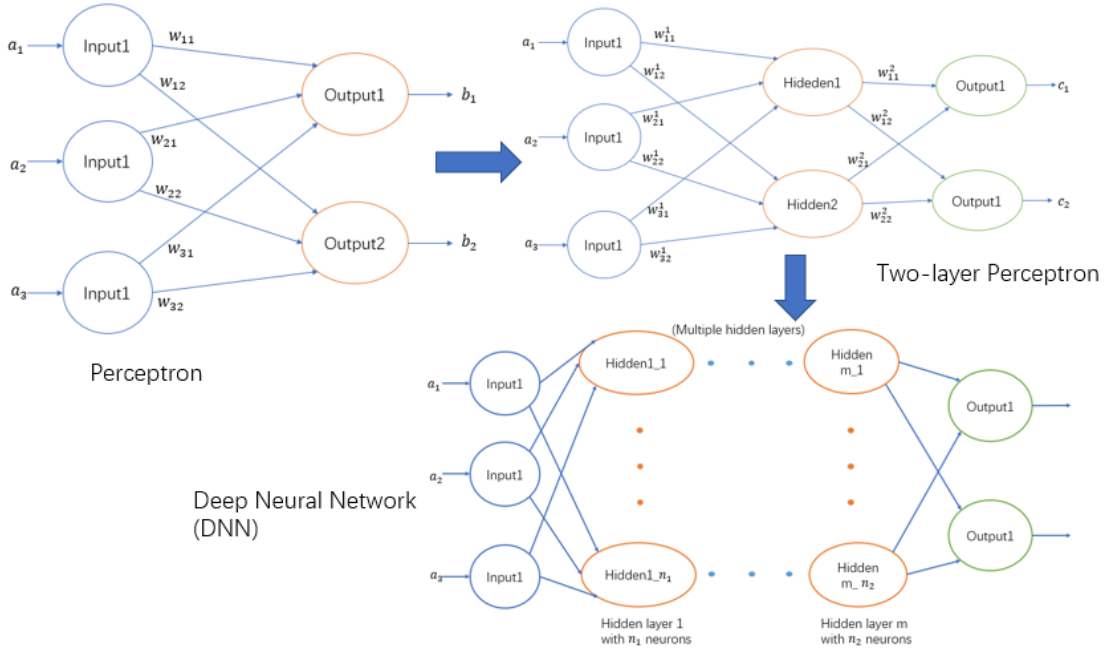


Fig 2.1 From Perceptron to Deep Neural Network

2.2 Convolutional neural network

Of the different types of deep learning architectures, CNNs have been very successful in processing data that can be presented by multiple arrays such as color images that are composed of three 2D arrays of pixel intensities [4]. These networks have been successfully applied to image processing, natural language understanding and vision system in self-driving cars [8]. The key operation in a CNN is convolution between the

input array and the convolution kernel, which is an array of weights. For example, convolution between two two-dimensional arrays is calculated as follows [8]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

where I is the input array, K is the kernel array and S is the output of convolution. Fig 2.2 describes the operation between two arrays pictorially.

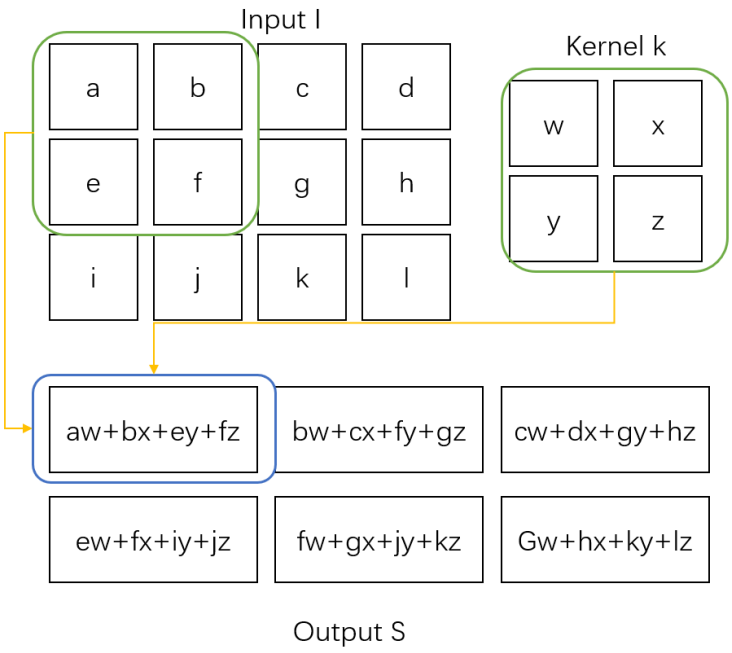


Fig 2.2. Convolution between input and kernel array [8]

A typical CNN consists of a few layers to extract the features from input data as described in Fig 2.3. Then a few classifier layers use the feature maps to form the final output of the neural network. In most cases, the classifier layers are several fully connected layers. The convolutional layers apply convolution operation to the input arrays to extract the features. In each convolutional layer, multiply convolution kernels or filters are applied to extract different types of features. With convolution kernels, one feature map contains

features extracted from a group of correlated values across all locations in the array. After features are detected, pooling layers merge semantically similar features into one to increase the reliability of feature detection. The most common-used pooling methods, like max pooling and average pooling enable the output to be unchanged when input is slightly changed. Fig 2.4 explains the operation of max pooling. After feature maps are generated from the convolutional layers, a few fully connected layers are applied to generate the output.

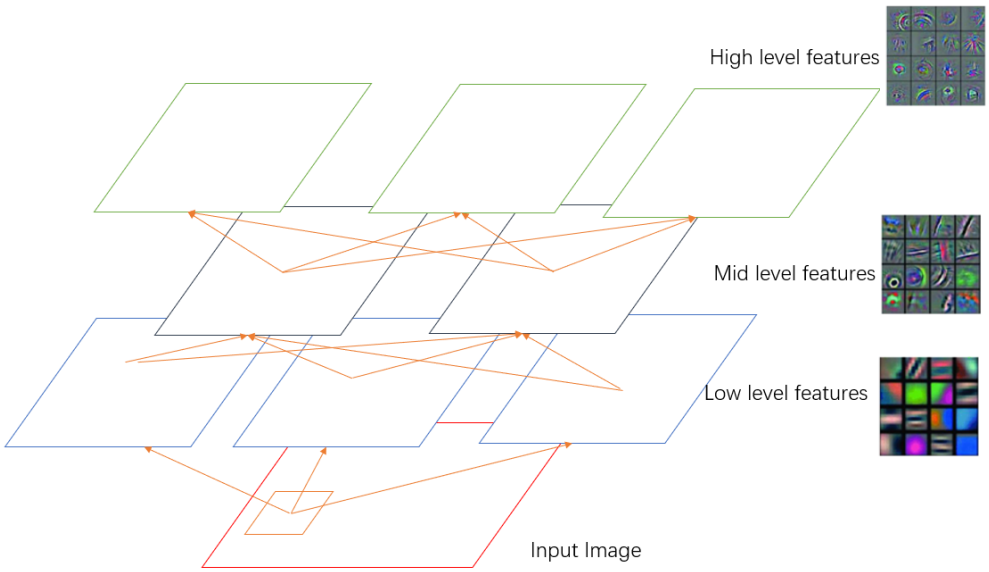


Fig 2.3. Features are extracted layer by layer

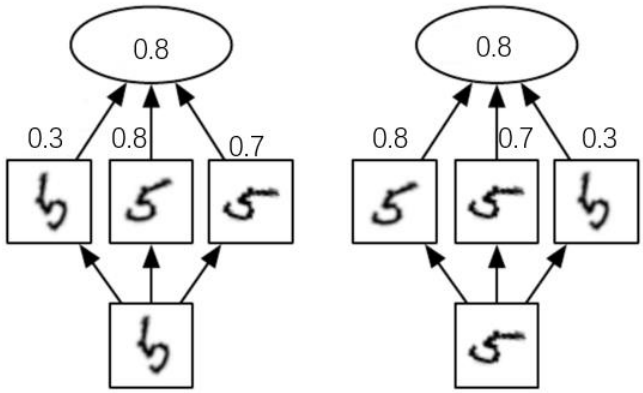


Fig 2.4. Three neurons judge the existence of digit ‘5’. Each rotates the input and makes prediction by comparing the rotated image to a known image of ‘5’. The max pooling ensures the prediction is correct when the input digit ‘5’ is rotated.

There are several popular CNN networks. One of the first networks is LeNet-5 [5], which consists of two convolutional layers to extract the pre-known features with fixed kernels and two fully connected layers to classify the features. AlexNet proposed by Krizhevsky in 2012 was one of the first convolutional neural networks whose kernels were trained to collect the features [10]. It has 5 convolutional layers and 3 fully connected layers. To solve the problem of gradient vanishing in deeper layers, ResNet was proposed by He in 2016 [9]. The key idea of ResNet is to create a residual block with bypass layer. ResNet-50, for example, has 49 convolutional layers and 1 fully connected layer and uses element-wise additions in residual blocks.

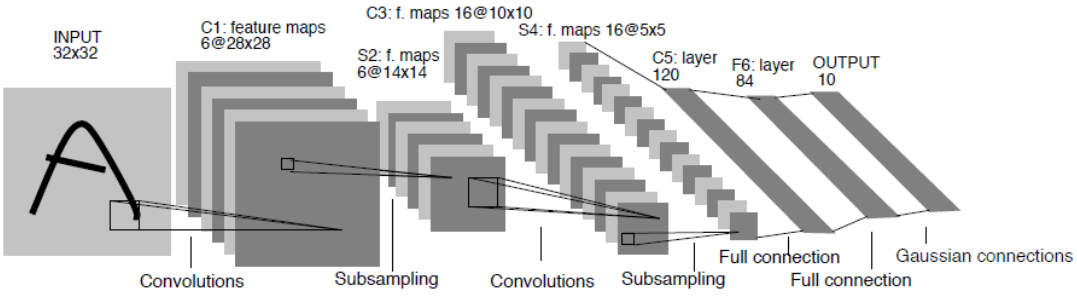


Fig 2.5. LeNet-5 structure [5]

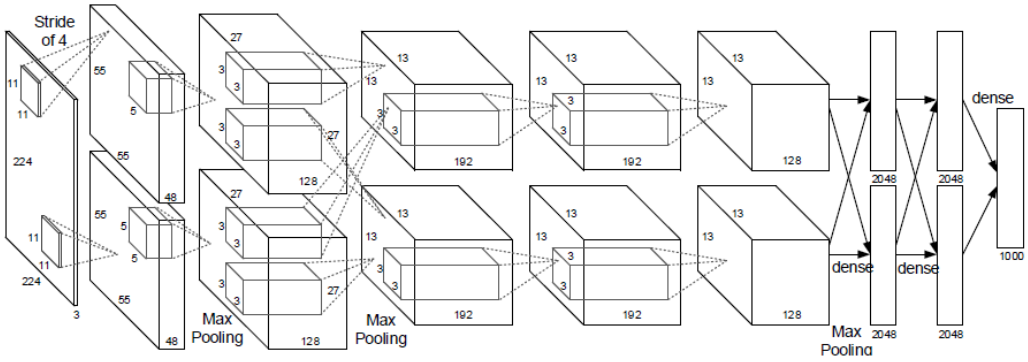


Fig 2.6. AlexNet Structure [10]

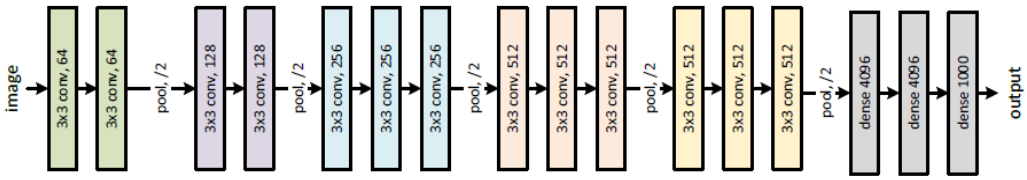


Fig 2.7. ResNet Structure [9]

2.3 Capsule network

Convolutional neural network (CNN) utilizes the property that features in an array are invariant to locations and uses pooling layers to ignore the location information about features. While this strategy forces the network to focus on features to make a decision, the lack of location information causes other problems that lead to wrong classification. In

some case, decisions that ignore location information are very likely to be wrong. For example, in the case of face sketch shown in Fig 2.8, a traditional CNN learns to judge the existence of face by checking if there are two eyes, one nose and one mouth. Since relative locations of these features are not considered, it is highly possible that the trained network will judge the Picasso style image on the right to also be a face. Capsule network proposed in [6] collects the features across the array while keeping the location information and thereby improves the performance.

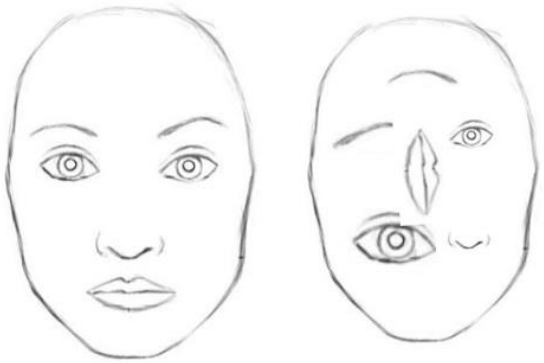


Fig 2.8. A face and a re-ordered 'face'

Capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part [6]. In a capsule network structure, features are described as a vector instead of a scalar. In Fig 2.9, every part of the face can be described by a three-dimensional vector which include the probability of existence, relative size and relative height. For instance, in the normal face, the left eye is represented by (0.95, 0.8, 5). In a normal face, the sizes are close to each other, and the heights obey specific rules. In the Picasso style face, two eyes are not at the same height, the mouth is above the nose and the size of the nose and one eyelash is too big or too small. This image is not likely to be classified to be a face by the capsule network.

Thus, by representing the information as a vector, a more informed decision can be made about whether it is a face or not.

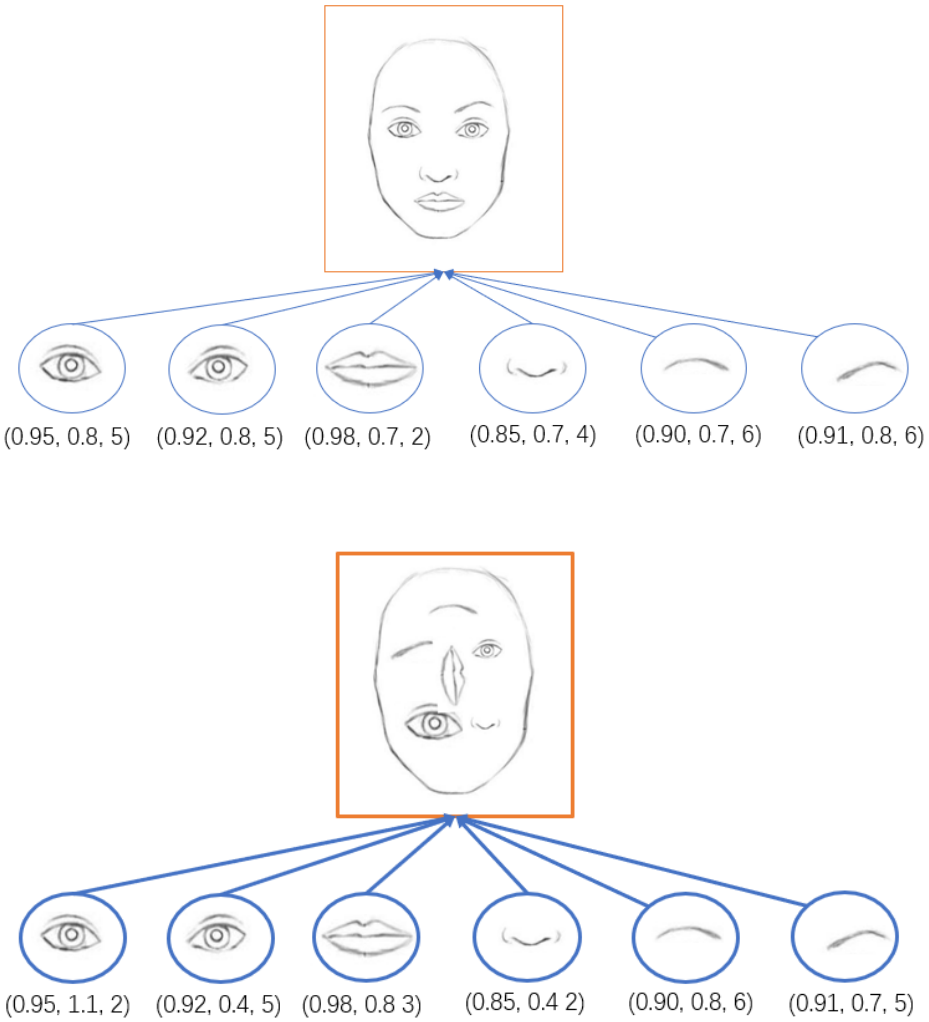


Fig 2.9. Use of capsules to describe parts of a face (top panel) and recorded face (bottom panel)

Information flow between lower-level capsules and higher-level capsules is done through the dynamic routing algorithm. This algorithm decides how to distribute information collected from the lower level to a proper capsule in the higher level. Instead of learning how to distribute during backpropagation, the dynamic routing algorithm enables higher-

level capsules to choose the input they “agree with” [6]. At the beginning of the routing process, all the output of lower-level capsules is distributed with equal weights. After a higher-level capsule receives the information from all inputs and forms an initial idea, it decides how much it “agrees with” a lower-level capsule. Then the weights are renewed to ensure that in the next iteration, higher-level capsules will receive more information from lower-level capsules that they “agree with”.

The procedure is described pictorially in Fig 2.10. The rectangles are high level capsules and the dots are predictions from lower level capsules. The green dots indicate that this cluster of predictions are close to each other and the blue dots indicate the predictions that are different from others. Higher-level capsules agree to a prediction when the prediction is in the cluster and in the following iteration, increase the weights of these predictions. Since higher-level capsules do not agree with the blue dots, then the weights are decreased in the next iteration.

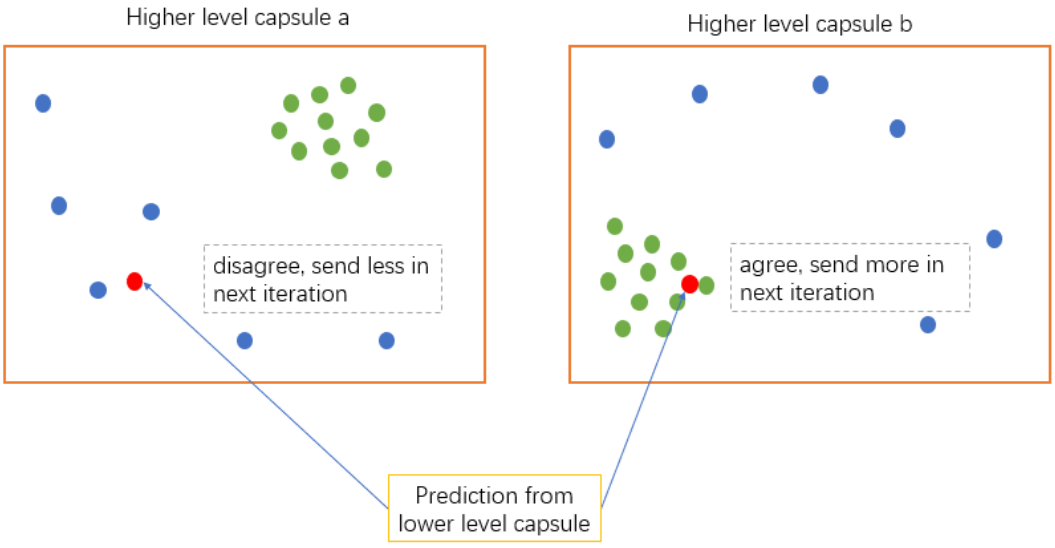


Fig 2.10. Dynamic routing scheme

In the capsule network used in this thesis, two convolutional layers are used to collect the features from original input. In convolutional layers, multiple kernels are used to extract the features and every kernel produces a feature map. The output of second convolutional layer are grouped into capsules called primary capsules. A primary capsule includes multiple neurons at the same location in different feature maps. The primary capsules feed data to class capsules, one per specific class, using dynamic routing. Fig 2.12 describes this network.

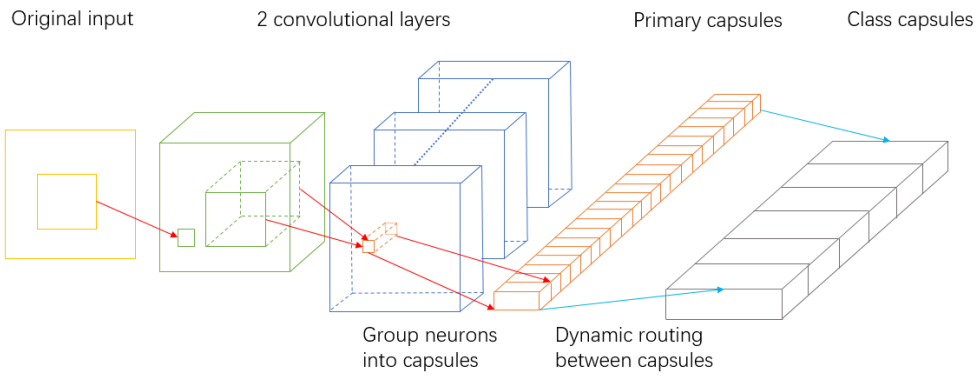


Fig 2.11. Capsule network

The loss function in capsule network consists of two parts: marginal loss for recognition of object existence and reconstruction loss used as regularization [6]. The values in a class capsule describe different characteristics of the object and the length of vector in class capsules denotes the probability of existence. A separate marginal loss allows multiple classes to exist simultaneously. In a capsule network, the loss function for an object of class k, L_k is given by:

$$L_k = T_k \ \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda \ (1 - T_k) \ \max(0, ||\mathbf{v}_k|| - m^-)^2$$

where v_k is the length of vector in class capsule k, $T_k = 1$ if class k is present, $m^- = 0.1$ and $m^+ = 0.9$, λ is for down-weighting the loss for absent classes, and λ is set to be 0.5. To regularize the model, the network reconstructs the input array using three fully connected layers [6]. Fig 2.12 describes the reconstruction network. During training, only output of capsules with correct label take part in the reconstruction. The reconstruction error is scaled Mean Square Error between reconstructed array and input array. The reconstruction error is added to the marginal loss to form the loss function [10].

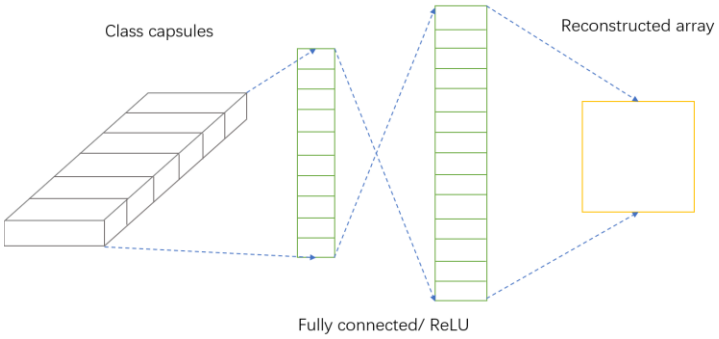


Fig 2.12. Reconstruction as regularization

3. CAPSULE NETWORK FOR OVERLAPPING DIGIT RECOGNITION

In this chapter, we evaluate the performance of the capsule network for an image recognition problem, specifically, overlapping digit recognition. We built MultiMNIST image dataset of overlapping digits and trained the network to recognize the digits in the images. To provide fair a comparison, we also build a traditional CNN and trained it with the same dataset.

3.1 Structure of capsule network

The capsule network used here has the following structure: The first convolutional layer has 256 channels, kernel size of 9×9, stride of 1, and no padding. The second layer has 256 layers, kernel size of 9×9, stride of 2, and no padding. The outputs of the second convolutional layer are grouped into 8-dimensional primary capsules. The second capsule layer consists of class capsules each with 16 dimensions. The two capsule layers are connected using the dynamic routing algorithm.

The reconstruction part uses three fully-connected layers: The first one has 512 neurons fully connected to the class capsules, the second has 1024 neurons and the last layer builds the reconstructed image with $32 \times 32 = 1024$ neurons. The configuration follows the capsule network proposed by Hinton [6] and makes slight adjustment to fit the input image size. Fig 3.1 describe the proposed capsule network.

The loss function consists of two parts: the first part is marginal loss calculated by labels and class capsules; the second part is Mean-Square-Error calculated using reconstructed image and input image. The reconstruction part is added to the marginal loss with a down-weighting parameter to ensure that the reconstruction loss is not the dominating part. The stochastic gradient descent is calculated with Adam optimizer [13].

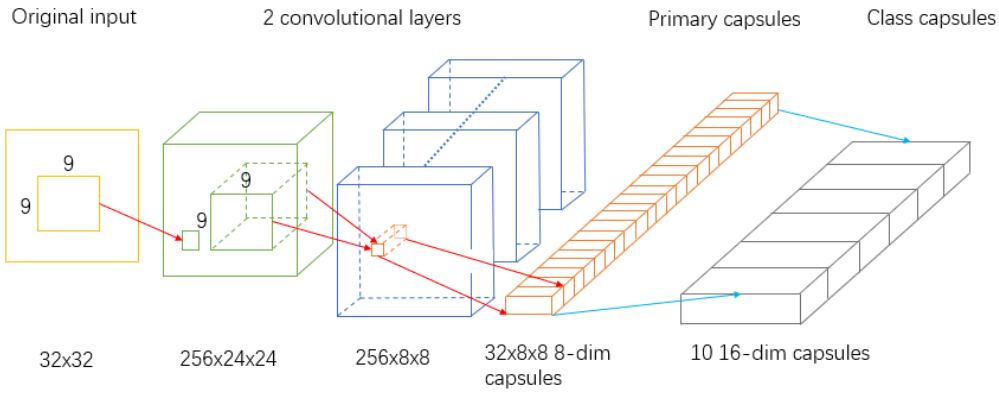


Fig3.1. Capsule network for digit recognition

3.2 Baseline CNN

We create a baseline CNN similar to that in [6]. It has similar number of weights as the capsule network. Baseline network has three convolutional layers with 256, 256 and 128 channels. The kernel sizes are 5×5 and stride is 1 for all three layers. After each convolutional layer, the results pass through a 2×2 max pooling layer and ReLU activation function. The classification is done by three fully connected layers with 328, 192 and 10 neurons. The last two layers are fully connected and use dropout as regularization method [6]. Fig 3.2 describes the baseline CNN structure.

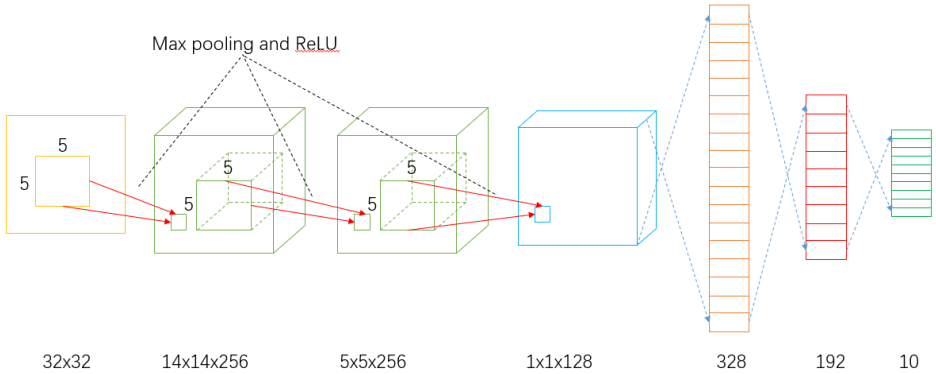


Fig3.2. Baseline CNN structure

3.3 Training on small size training set

In a capsule network with dynamic routing algorithm, the information in low-level capsule is sent to the high-level capsule which 'agrees with' the input. Thus, the capsule network is expected to converge faster than normal CNN, since normal CNN uses multiple fully connected layers to do the classification and requires more samples to reach the optimal value. In this section, we compare the performance of capsule network and the baseline CNN with respect to converging speed, training time and testing accuracy.

3.3.1 Experiment design

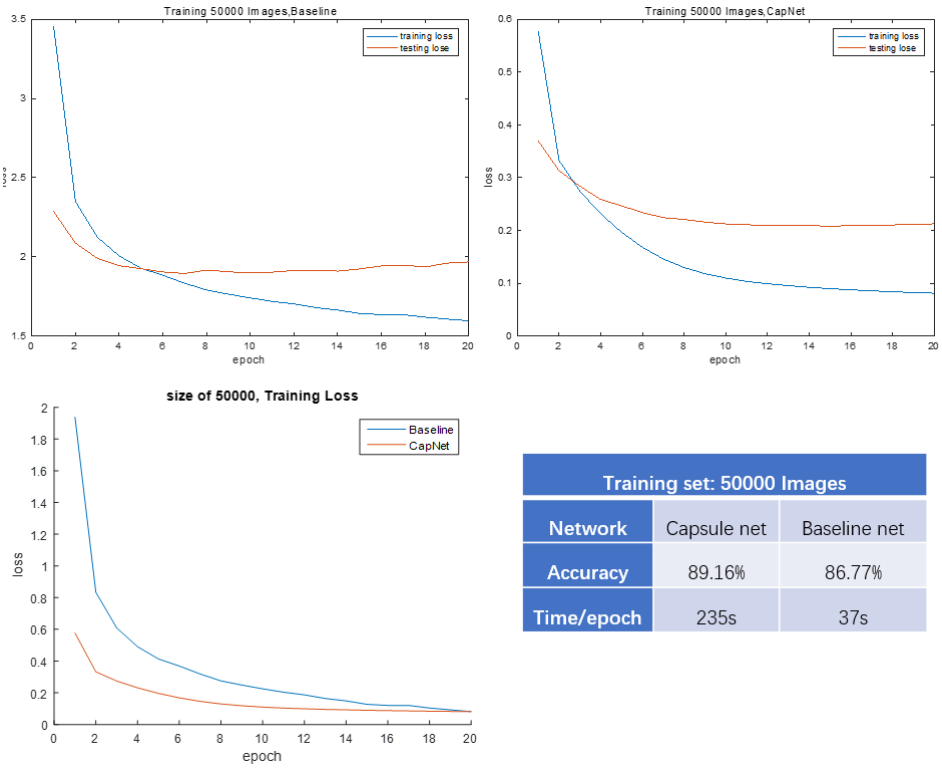
This experiment is based on a MultiMNIST dataset, which is built based on MNIST hand-written digit dataset [14]. In MNIST dataset, there are 50,000 images in the training set and 10,000 images in the testing set. Each image includes a hand-written digit from zero to nine. Along with the image is the label indicating the correct number in the image. The MultiMNIST dataset is built by overlaying two images with different labels to form one image including 2 digits. The size of images in the MNIST dataset is 28×28, and we shift the image randomly by up to 4 pixels in each direction. In this way we get an image of size 32×32. For instance, as shown in Fig.3.3, digits 0 and 7 are combined to form one image with label (0, 7). We can control the training set size by choosing how many images are mixed with one image in MNIST. In this experiment, we used training sets of size 50,000, 10,0000, 150,000, 200,000 and 250,000 images.



Fig3.3. Digits from MNIST and overlapping digits in MultiMNIST

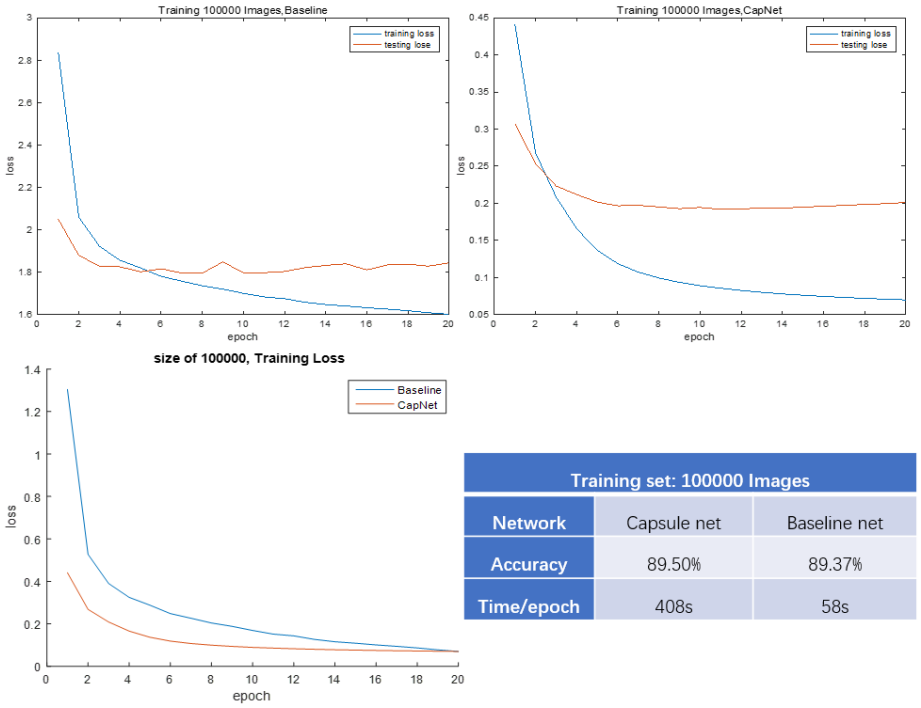
3.3.2 Result

Figure 3.4 -3.9 show the training and testing errors when training the capsule network and baseline network for 20 epochs. The training set size varies from 50,000 to 250,000. In each figure, the top panel shows the training loss and testing loss for baseline (left subplot) and capsule network (right subplot); the bottom panel compares the training loss of baseline and capsule network and the table shows the testing accuracies and average training time per epoch of the two networks. The implementation is in PyTorch with Python3.5 and Cuda9.0. The training time is the actual execution time of the algorithm when run on NVIDIA GTX1070 with 8G frame buffer.



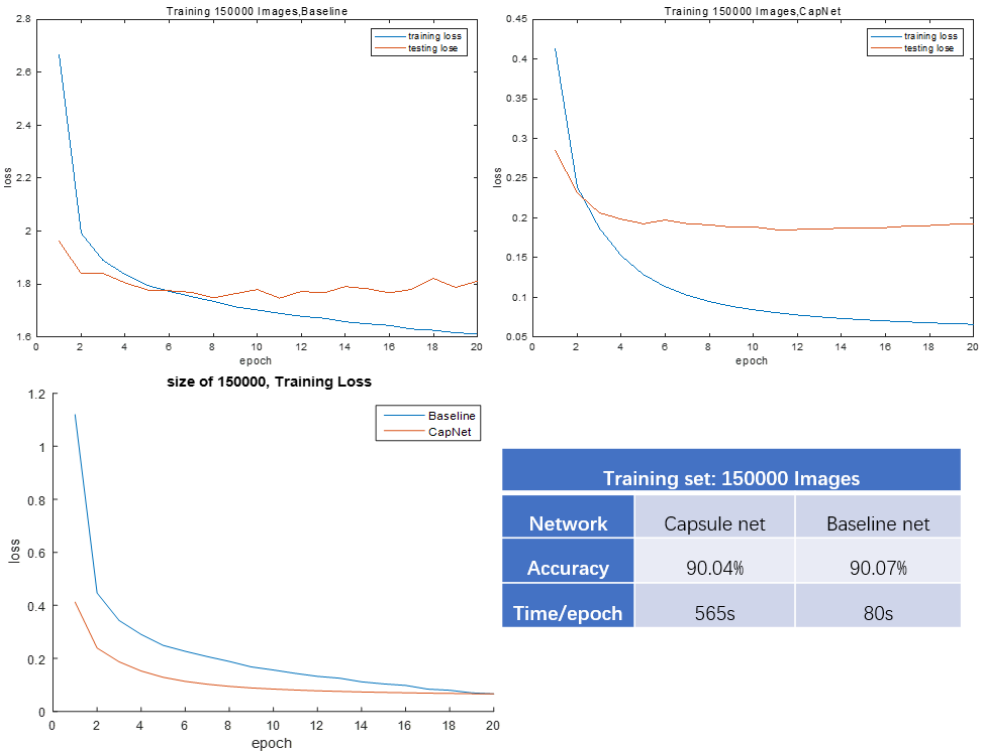
Training set: 50000 Images		
Network	Capsule net	Baseline net
Accuracy	89.16%	86.77%
Time/epoch	235s	37s

Fig 3.4. Performance comparison when training on 50,000 images.



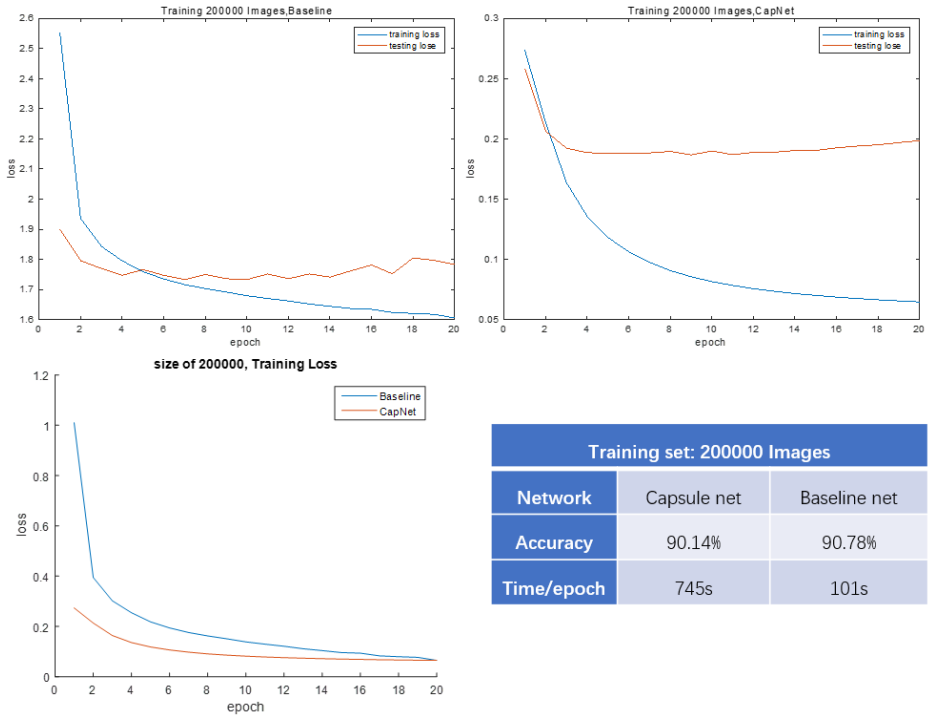
Training set: 100000 Images		
Network	Capsule net	Baseline net
Accuracy	89.50%	89.37%
Time/epoch	408s	58s

Fig 3.5. Performance comparison when training on 100,000 images



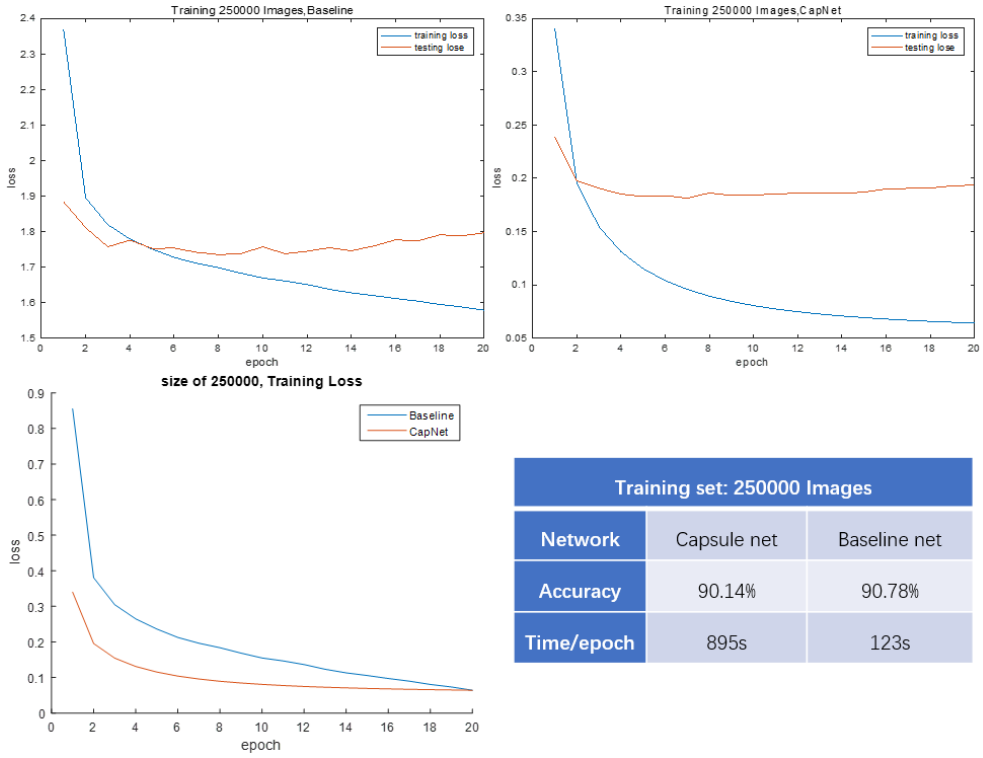
Training set: 150000 Images		
Network	Capsule net	Baseline net
Accuracy	90.04%	90.07%
Time/epoch	565s	80s

Fig 3.6. Performance comparison when training on 150,000 images



Training set: 200000 Images		
Network	Capsule net	Baseline net
Accuracy	90.14%	90.78%
Time/epoch	745s	101s

Fig 3.7. Performance comparison when training on 200,000 images



Training set: 250000 Images		
Network	Capsule net	Baseline net
Accuracy	90.14%	90.78%
Time/epoch	895s	123s

Fig 3.8. Performance comparison when training on 250,000 images

The curves of training loss show the convergence speeds of capsule network and baseline network. The curves of testing loss show whether the networks are overfitting. The subplot on the right of bottom panel compares the convergence of two networks. In the table is the best testing accuracy and training time per epoch. When training set size increases, convergence speeds of both networks increase. The training time per epoch and accuracy also increases as the training set size increases. Fig 3.9 and Fig 3.10 plot the training loss of the baseline network and capsule network respectively.

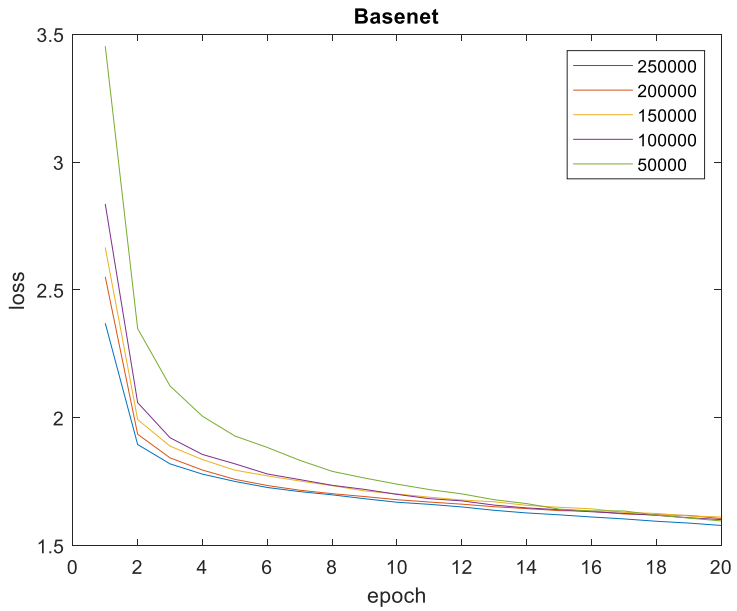


Fig 3.9. Training loss of baseline network for different sized training sets

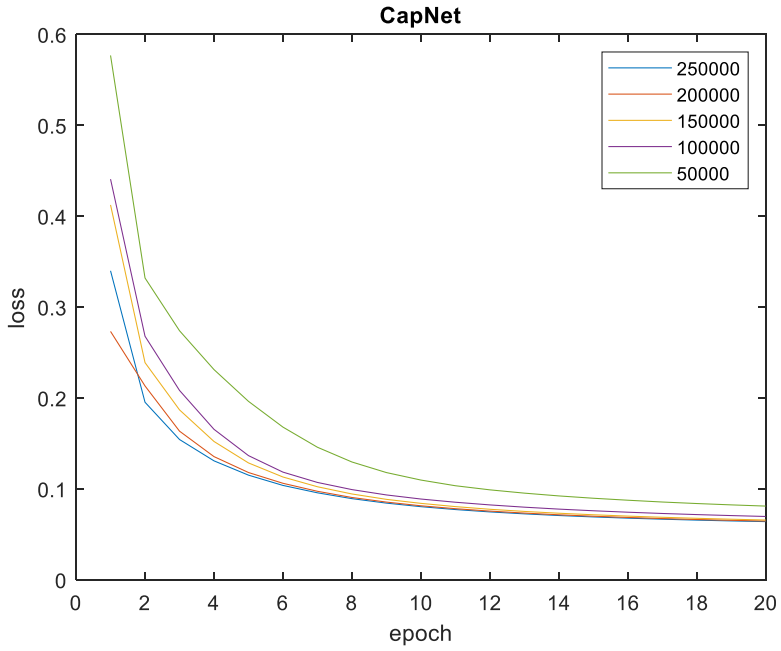


Fig 3.10. Training loss of capsule network for different sized training sets

This evaluation shows that, capsule network converges faster than the baseline network in all cases. When training set is small (50,000 images), the capsule network achieves a

higher accuracy. However, when training set is larger, both networks have comparable accuracy. In all cases, capsule network converges to a relatively low loss in about 12 epochs while baseline network requires more than 20 epochs. In all cases, the capsule network shows a faster convergence speed.

TABLE 3.1. MOST TIME-CONSUMING KERNELS IN TRAINING PROCESS FOR BASELINE NETWORK

Kernel in GPU	**Baseline CNN**	
	Time	**Percentage**
maxwell_scudnn_relu	10.52s	37.3%
Maxwell_cudnn_128x128	5.08s	18.0%
Maxwell_gcgemm_32x32	3.59s	12.7%
PoolingForward/Backward	3.41s	7.7%

TABLE 3.2. MOST TIME-CONSUMING KERNELS IN TRAINING PROCESS FOR CAPSULE NETWORK

Kernel in GPU	**Network with dynamic routing**	
	Time	**Percentage**
Batch_gemm_kernel	92.94s	41.5%
Cudnn::detail_dgrad_engine	39.83s	17.8%
Maxwell_cudnn_relu	16.62s	7.4%
Maxwell_cudnn_128x128	14.12s	6.3%

While the capsule network has higher convergence rate and higher accuracy, its training time is a lot higher which is 8 times more per epoch. To analyze the bottlenecks of the capsule network, we use NVVP profiler by NVIDIA to profile one epoch of training both capsule network and baseline network. Table 3.1 shows the profiling results of baseline network. We find that convolution operation and computation in fully connected layers can be efficiently computed with multiplication kernels in GPU (Maxwell_cudnn_128x128

and Maxwell_gcgemm_32x32). The most time-consuming part is nonlinear operations including ReLU and pooling.

Table 3.2 shows the profiling results of capsule network. The convolution operation and computation in fully connected layers take 16.62s which is only 7.4% of the training time per epoch. The dynamic routing algorithm calls GEMM kernels for matrix-matrix multiplication which is not as efficient as Cudnn kernels (for example, Maxwell_cudnn_128x128). This kernel spends 92.94s, which is 41.5% training time per epoch. In addition, Cudnn kernel for gradient calculation contributes 39.83s and ReLU operation spends 16.62s.

3.4. Capsule size

Next, we study the tradeoff between the number of capsules and the size of capsules. For primary capsules, a larger size means one capsule knows more about one specific feature and describes the features with more detail. It is expected that decision it makes for the next level will be more reasonable. On the other hand, using a larger number of primary capsules is likely to help detect more types of features, and there are also more capsules that take part in the 'voting' process. In terms of computation cost, more capsules mean more matrix-vector multiplications in the training process which results in increase of training time. For class capsules, a larger size also records more parameters to describe a specific digit, which is likely to force the primary capsule's vote to be more accurate while a small size for class capsule reduces the computation cost of dynamic routing algorithm.

3.4.1 Experiment design and result

We keep the capsule network to be the same as the last experiment and only change the capsule sizes in primary capsule layer and class capsule layer. For studying the effect of

size, primary capsule part, we consider 4, 8, 16, and 32-dimensional capsules and keep the class size to be 16. In each case, the training set is 50,000 images and testing set is 10,000 images. The training process lasts for 20 epochs. The results are shown in Fig 3.11 and Table 3.3. We see that the capsule size does not have much effect on accuracy. But networks with the larger size of primary capsule converges faster and has much short training time.

TABLE 3.3. ACCURACY AND TRAINING TIME FOR DIFFERENT PRIMARY CAPSULE SIZE

Capsule Size	**4**	**8**	**16**	**32**
Accuracy	88.28%	88.70%	88.98%	89.38%
Training time/epoch	357s	236s	177s	163s

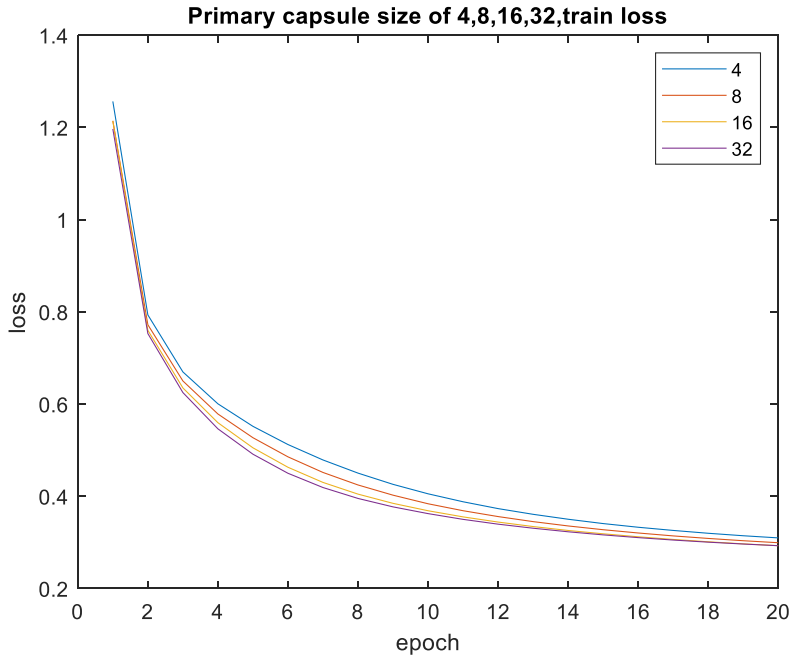


Fig 3.11. Training loss of capsule network when the class capsule is of size 4,8, 16 and 32; the class capsule size is 16.

For class capsule part, we use the primary capsule size of 8 and test the network with class capsule of size 16-dimensions and 8-dimensions. The training and testing sets are the same as the primary capsule part. The training also lasts for 20 epochs. The results are shown in Fig 3.12 and Table 3.4. The 16-dimensional network converges faster at the beginning, but after 10 epochs, the converging speed of the two networks are the same.

TABLE 3.4. ACCURACY AND TRAINING TIME OF DIFFERENT CLASS CAPSULE SIZE

Capsule Size	**8**	**16**
Accuracy	88.66%	88.70%
Training time/epoch	204s	236s

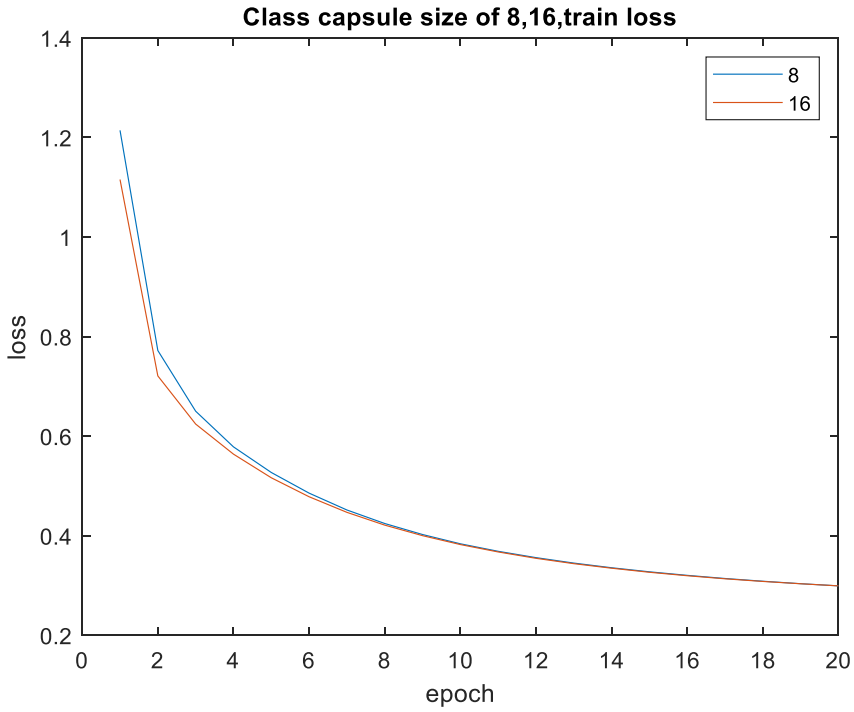


Fig 3.12. Training loss of capsule network when the class capsule is of size 8, 16; the primary capsule size is 8.

3.4.2 NVVP profiler analysis

Table 3.5 shows the result of profiling capsule network with different sizes of primary capsule. The result shows that the change of capsule size does not affect the calculation load in convolutional layers and fully-connected layers. But as the primary capsule size increases, the time for dynamic routing computation is reduced.

TABLE 3.5. RESULT OF DIFFERENT PRIMARY CAPSULE SIZES

Size of primary capsule / kernel in GPU	4	8	16	32
Batch_gemm_kernel	173.28s	92.94s	52.20s	48.19s
Cudnn::detail_dgrad_engine	39.72s	39.83s	40.00s	39.75s
Maxwell_cudnn_relu	16.31s	16.62s	16.82s	16.75s
Maxwell_cudnn_128x128	13.71s	14.12s	14.42s	14.46s

Table 3.6 describes the computations needed in the dynamic routing algorithm. Assume that there are N_1 primary capsules of size S_1, and N_2 class capsules of size S_2. In the network used here, $S_1 \times N_1 = 256 \times 21 \times 8 = 43008$.

The table shows that the number of primary capsules affects the number of computations in calculating the prediction, calculating the input of class capsule, and renewing the coupling coefficients. The size of the used to calculate prediction is determined by the size of the primary capsule. The total time can be expressed as:

$$T = N_1 \times N_2\big(t_1(S_1, S_2) + t_2(S_2) + t_4(S_2)\big) + N_2 t_3(S_2)$$

where $t_1(S_1, S_2)$ denotes the time for prediction, $t_2(S_2)$ denotes the time for computing input of class capsule, $t_3(S_2)$ denotes the time for applying squash function to the input in each class capsule, and $t_4(S_2)$ denotes the time for renewing the coefficients between primary capsules and class capsules.

TABLE 3.6. COMPUTATIONS IN DYNAMIC ROUTING ALGORITHM

Operations	Equations	Number of computations	Computation
Making Prediction	$\hat{u}_{ji} = W_{ij}u_i$	$\frac{43008}{S_1} \times N_2$	Multiplication between(S_2, S_1) and $(S_1, 1)$
Input of class capsule	$s_j = \sum_i c_{ij}\hat{u}_{ji}$	$\frac{43008}{S_1} \times N_2$	Multiplication between$(S_2, 1)$ and scaler c_{ij}, summation of two $(S_2, 1)$ vectors
Squash function	$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|}$	N_2	Calculating $\|s_j\|$
Renewing coefficients	$b_{ji} = b_{ij} + \hat{u}_{ji} \cdot v_j$	$\frac{43008}{S_1} \times N_2$	Inner product of two S_2-dimension vectors

Fig 3.11 shows the training time for dynamic routing algorithm as a function of numbers of primary capsules. It shows that for primary capsule size of 4, 8, and 16, the relation between training time and numbers of capsules is linear. This means for all computations with capsule size S_1 no larger than 16, the GPU implementation calls the same sized computation kernel. As a result, increasing the size of primary capsules efficiently decreases the training time by decreasing the number of computations. In case of capsule size of 32, the GPU implementation requires computation kernels of larger size, which results in more time per computation. So, though the number of computations is decreased, the training time decreases mildly.

3.4.3 Summary of results

For the MNIST recognition task, we can see that a larger primary capsule works better, which indicates that the digit recognition task does not require the network to collect large number of features. Fewer features, each with more details are enough for making correct decision. In this case, although there are fewer capsules voting, the vote from each capsule

contributes more. As for the class capsule, a larger size helps only in the beginning, which means that 8 values are enough to describe a digit for classification. The slight difference is not worth the increase in the computation. In terms of training time, we find that increasing the size of primary capsule and decreasing size of class capsule can cut down the training time of capsule network while keeping the accuracy nearly unchanged.

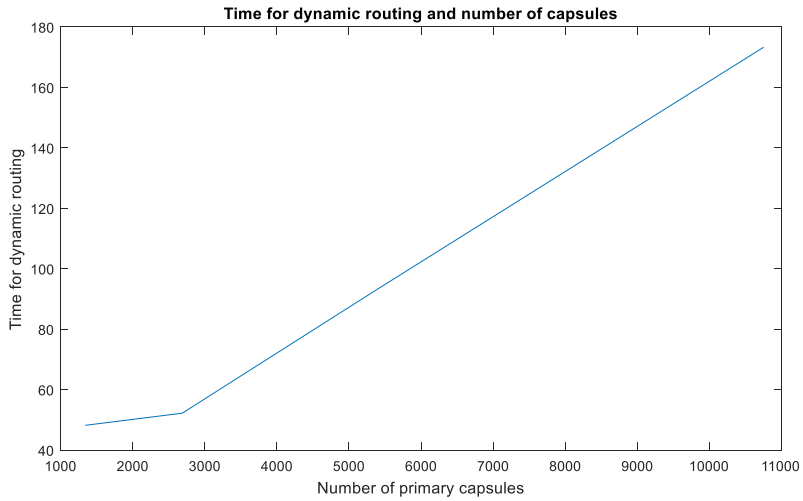


Fig 3.13. Time for dynamic routing and number of capsules

3.5 Conclusion

The largest training set used in this work was 250,000 images compared to 60M images in Hinton's papers [6]. So, in my experiments, the advantage on accuracy of capsule network may not be that obvious. But the convergence speed of capsule network stands out in case of small training set. Also, capsule network achieves a high accuracy when the training set is small (50,000 images). We found that the training time for capsule network is extremely long compared to baseline network. To mitigate this problem, we show that adjusting the capsule size can cut down training time by 50% (from Table3.5, training time decreased form 357 seconds to 163 seconds) while keeping the accuracy almost unchanged.

4. CAPSULE NETWORK FOR OVERLAPPING WORD RECOGNITION

In the last chapter, we analyzed the performance of capsule network for recognizing overlapping digits. We showed that capsule network can solve the complex feature extraction and recognition task well. We concluded that capsule network converges faster and has better performance compared to the baseline CNN when the training set is small. These results inspired us to apply capsule network to the speech recognition task since speech recognition problems have to deal with smaller training sets. Besides, in speech processing, features such as MFCC are arranged in time order and thus spatial information is available and could be exploited with capsules. In this chapter, we show use of capsule network for the speech recognition problem of identifying two overlapping words. We start with overlapping speech created from five different words, then increase the number of words to ten.

4.1 Data set and pre-processing

The data set we use for speech recognition is based on the Speech Commands dataset collected by Google [12]. It consists of over 105,000 WAVE audio files of people saying thirty different words. Every WAVE file includes one word from one speaker with a length of 2 seconds as shown in Fig 4.1. We built a mixed speech file by overlaying two WAVE files from two different words as shown in Fig 4.2. When mixing is finished, we have WAVE files, each of which contains 2 words with a total length of 2 seconds.

Before feeding these training samples into the neural network, we carry out speech processing on the speech file. Training network to learn features in the time domain is more difficult since the features are mixed across the two speakers. Thus, in speech recognition-

based applications, frequency domain features are commonly used and have proven to be successful. Among the various types of frequency domain features, Mel-Frequency Cepstral Coefficients (MFCC) and filter bank features are very popular and are often used in deep learning applications.

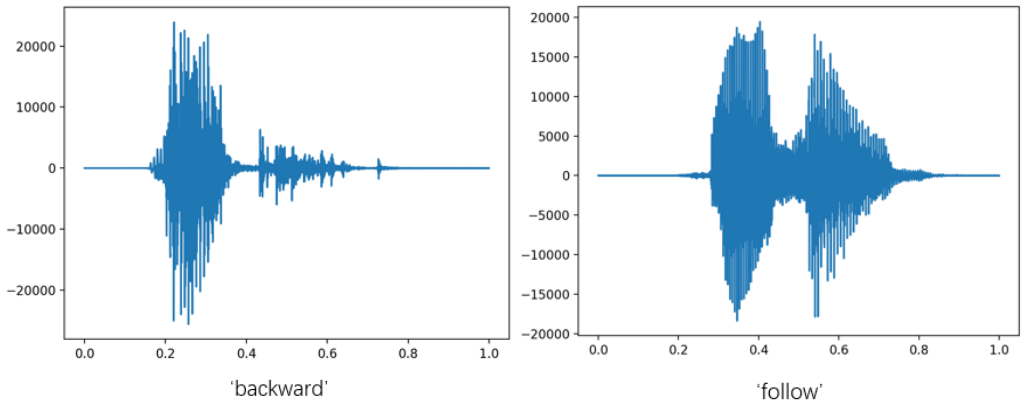


Fig 4.1. Wave plot of word ‘backward’ and ‘follow’

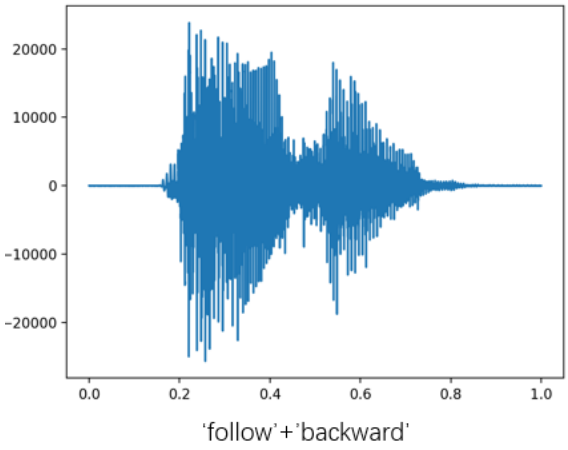


Fig 4.2. Wave plot of overlapping ‘backward’ and ‘follow’

Computing filter bank and MFCCs involve similar procedures, as in both cases filter bank is computed and MFCC can be obtained with a few additional steps. In a nutshell, a signal goes through a pre-emphasis filter, then gets sliced into (overlapping) frames and a

window function is applied to each frame. Then a Short-Time Fourier Transform (STFT) is done on each frame and then the power spectrum is calculated. The last step involves application of triangular filters on a Mel-scale (described in Fig 4.3) to the power spectrum to extract coefficients in different frequency bands. This procedure is referred to as filter banks. To obtain MFCCs, a Discrete Cosine Transform (DCT) is applied to decorrelate the filter bank coefficients. In both cases, the final step is mean normalization. In our experiment, we apply both MFCCs and Filter bank to get frequency-domain features as the input of our network. Fig 4.4 describes the procedures to calculate filter bank and MFCC based features.

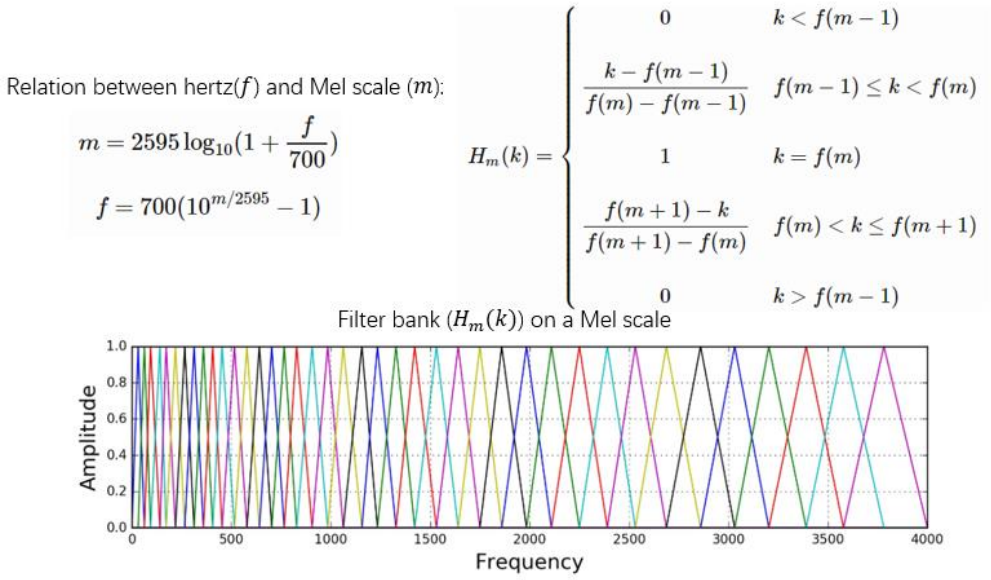


Fig 4.3 Filter bank on a Mel scale

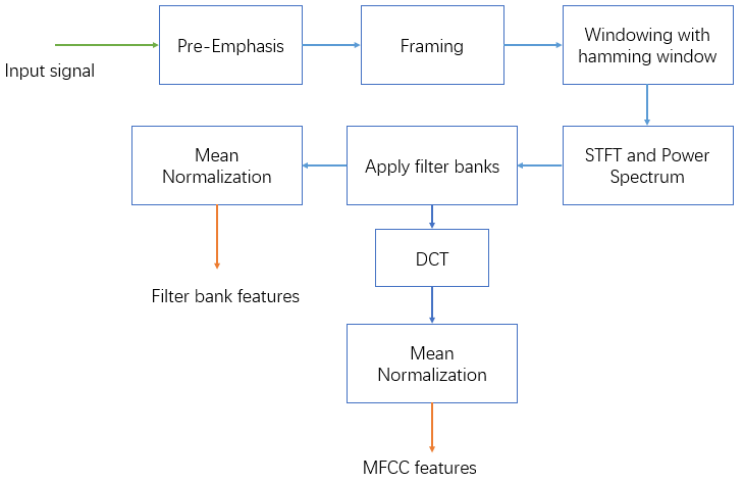


Fig 4.4 Block Diagram illustrating calculation of MFCC and filter bank coefficients

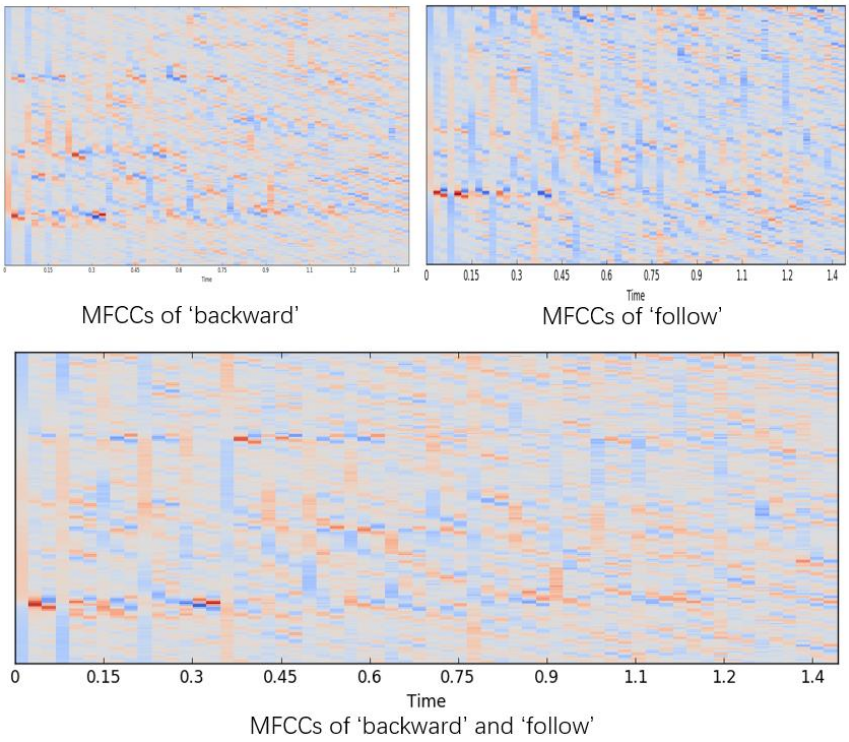


Fig 4.5. MFCCs feature maps

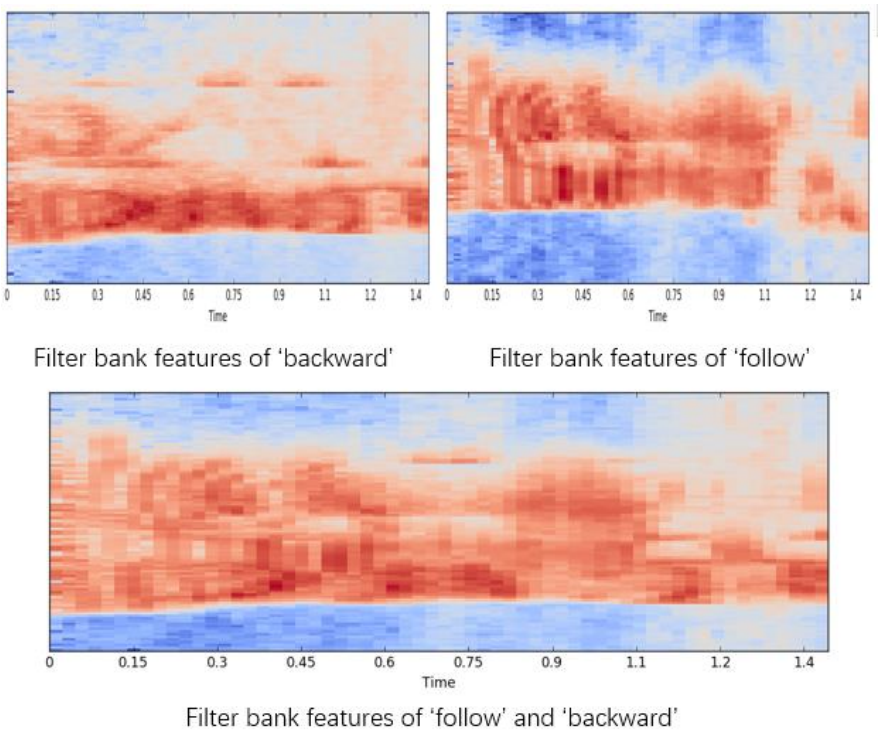


Fig 4.6. Filter bank feature maps

Fig 4.5 and Fig 4.6 show the feature maps obtained using MFCCs and Filter bank, respectively. It is clear from the figures that the feature maps of a mixed speech contain features that are specific to the individual speakers and words; this makes recognition from feature maps feasible.

4.2 Experiment 1: Design and Result

In the first experiment, we choose 5 words to build the dataset. The five words are 'backward', 'follow', 'marvin', 'sheila' and 'visual'. Every word has multiple speech files recorded by different speakers, the number of speakers for each word is shown in Table 4.1.

To derive the training set, for every speech file in all word categories, we mixed it with 20 speech files from other categories. Thus, there are 161,320 mixed speech files in the

training data set, each containing two words from two different word categories. The test set is created by a similar method, but the size is 1/5 of the training set.

TABLE 4.1 WORDS IN THE DATASET

Label	word	number
0	backward	1558
1	follow	1454
2	marvin	1831
3	sheila	1754
4	visual	1469
*	In total	8066

TABLE 4.2 WORDS IN SPEAKER DEPENDENT DATASET

label	word	Training samples	Testing samples
0	backward	1298	260
1	follow	1212	242
2	marvin	1528	303
3	sheila	1462	292
4	visual	1225	244
*	In total	6725	1341

We also create speaker independent training set by training the network using speech files from one group of people and then testing the network with another group of people. In this case, the wave file distribution is as shown in Table 4.2. The mixing procedure is the same as speaker independent training set.

After generating the training and testing wave files, preprocessing using MFCC and filter bank is done to create the training and testing feature map sets. Each feature map is of size 98x60. We choose training set size of 50,000 and testing set size of 10,000.

4.2.1 Capsule network

In this experiment, the capsule network has the same structure as in the last chapter. Specifically, there are two convolutional layers for feature extraction and two capsule layers with 8-dimensional primary capsules and 16-dimensional class capsules for classification. We tested different kernel and stride choices for the two convolutional layers. We choose kernels with rectangular shape instead of square shape. This is because features from MFCC and filter bank show more correlation in temporal dimension. So, a rectangular kernel makes better use of weights and collects more useful information. After the second convolutional layer, the outputs are grouped into capsules. For the reconstruction part, the output of reconstruction is now a 60x98 array; this is the same size as input feature map.

We tested for five different kernel size sets. The results are shown in Table 4.3. The two convolutional layers of the capsule network use kernels K1 and K2. Configuration of K1 and K2 is shown in Fig 4.7. We can see that a wider convolution kernel in temporal dimension has higher accuracy. For example, when trained with speaker-independent filter bank features using kernel K2 of size (6,6), the accuracy of network using kernel K1of size (18,2) is 78.75% and the accuracy of network using K1of size (6,18) is 82.24%. For baseline CNN network, we use K1 of size (6, 24) and K2 of size (6,12) for the first two convolutional layers. We choose these sizes for K1 and K2 since the capsule network has the best performance for these kernel sizes.

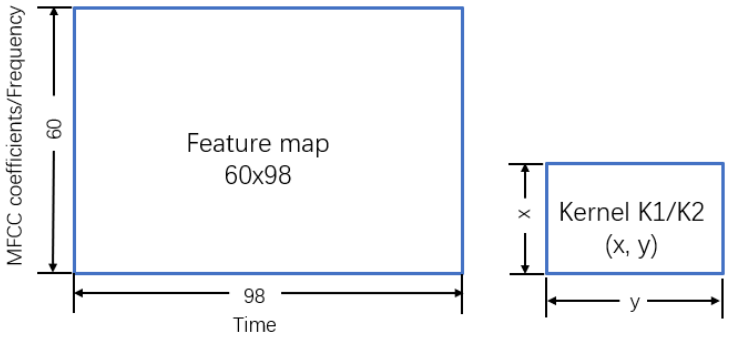


Fig 4.7. Kernel configuration in convolution layers

TABLE 4.3 ACCURACY AND TRAINING TIME FOR DIFFERENT CONFIGURATIONS

Accuracy / **Time**	**K1 (28,2) K2 (6,6)**	**K1 (18,2) K2 (6,6)**	**K1 (6,24) K2 (6,12)**	**K1 (6,18) K2 (6,12)**	**K1 (6,18) K2 (6,6)**	**Baseline CNN**
MFCC Speaker Ind.	75.91%	75.75s	78.76s	76.75%	81.67%	79.20%
	155s	171s	160s	210s	157s	38s
MFCC Speaker Dep.	89.64%	90.88%	93.94%	92.30%	93.56%	83.68%
	156s	186s	170s	218s	165s	38s
Filter Bank Speaker Ind.	79.60%	78.75%	83.68%	82.24%	81.52%	77.26%
	161s	174s	168s	208s	161s	38s
Filter Bank Speaker Dep.	95.64%	96.32%	96.72%	96.84%	96.92%	85.19%
	154s	170s	160s	205s	164s	38s

4.2.2 Effect of capsule size

The results in Table 4.3 also show that the training time of capsule network is long compared to baseline CNN. Results from last chapter indicate that the training time of capsule network can be decreased by changing the size of capsule. In this experiment, we train the capsule network with different capsule size configurations and compare the corresponding accuracy and training time with respect to a baseline network.

We use convolutional layers with K1 of size (6,24) and K2 of size (6,12) and primary capsule size to be 4, 8, 16 and 32. We also try one configuration with class capsule of size 16 to see if the training time can be further decreased. The results are shown in Table 4.4. The capsule size in the table is denoted in form of (primary capsule size)/ (class capsule size).

TABLE 4.4 ACCURACY AND TRAINING TIME FOR DIFFERENT CAPSULE SIZES

Accuracy / **Time**	**4/16**	**8/16**	**16/16**	**32/16**	**32/8**
MFCC Speaker Ind.	77.87%	78.76%	78.78%	79.52%	79.43%
	234s	160s	128s	121s	112s
MFCC Speaker Dep.	93.53%	93.94%	93.93%	93.05%	94.00%
	229s	170s	130s	122s	113s
Filter Bank Speaker Ind.	84.77%	83.68%	82.61%	83.00%	84.67%
	230s	168s	129s	122s	113s
Filter Bank Speaker Dep.	96.72%	96.72%	96.50%	96.28%	96.88%
	225s	170s	127s	122s	113s

From the result we can see that changing the capsule size does not have much effect on recognition accuracy, but a capsule with larger size requires shorter time for training. In case of using speaker-dependent filter bank features, comparing accuracy of networks that use primary capsule of size 8 and size 32 for primary, we see that the difference in accuracy is smaller than 1% but the training time for every epoch reduces from 170s to 122s which is about 28% lower. The results for config 32/16 and config 32/8 have similar accuracy, but the training time is further decreased by 9 seconds from 122s to 113s.

4.3 Experiment 2

To further investigate the performance of capsule network on overlapping speech, we trained the network on a training set that consists of more categories of words as shown in Table 4.5. In this experiment, we choose 5 more words in addition to the 5 words we used in the last experiment. In total, there are 10 categories of words to mix and there are 45 categories of mixed files.

The capsule network for this experiment uses convolutional layers with kernel size K1(6,24) and kernel size K2 (6,12). The primary capsule size is 32 and the class capsule size is 8.

TABLE 4.5 ACCURACY AND TRAINING TIME FOR DIFFERENT CAPSULE SIZE

label	**word**	**number**	**label**	**word**	**number**
0	backward	1558	**5**	bed	1686
1	follow	1454	**6**	forward	1452
2	marvin	1831	**7**	nine	3629
3	sheila	1754	**8**	six	3598
4	visual	1469	**9**	wow	1797

We build both speaker dependent and speaker independent training set by overlaying the word from different categories. Filter bank is used to build the feature maps for mixed files. The training set size is 54000, which means 1200 samples for each kind of mixed speech. The testing set size is 10800. The results are shown in Table 4.6.

In this experiment, the training sample per category decreased from 5000 to 1200. In case of training with speaker independent training sets, the accuracy of capsule network

shows slight decrease from 96.88% to 95.42%, while the baseline network's accuracy decreases sharply from 84.67 to 73.18%. This result validates the capability of capsule network to have high performance when dealing with small training set compared to traditional CNN. As for the speaker dependent case, both networks show sharp decrease. This result is predictable since speaker dependent training set shows fewer common features with testing set, which means the training set size has stronger effect on accuracy.

TABLE 4.6 RESULT OF 10-WORD RECOGNITION

Accuracy / Time per epoch	**Capsule Network**	**Baseline Network**
Speaker dependent	95.42%	73.18%
	157s	41s
Speaker independent	64.73%	61.45%
	157s	41s

4.4 Recognizing individual word

In this part, we show the reconstructed feature map from the capsule network to show that the capsule network can recognize overlapping speech recognizes individual speech components.

The network configuration used for this study is as follows: Convolutional layer 1 has kernel size (6, 24) and stride (2, 2); Convolutional layer 2 has kernel size (6, 12) and stride (2,1). The primary capsule size is 32 and class capsule size is 8. The network is trained with speaker independent training set.

We use network trained with speeches of overlapping words to recognize speech of single word and present the reconstructed feature maps. We take word 'backward' and

‘follow’ as example. The original and reconstructed feature maps are shown in Fig 4.8 to Fig 4.13.

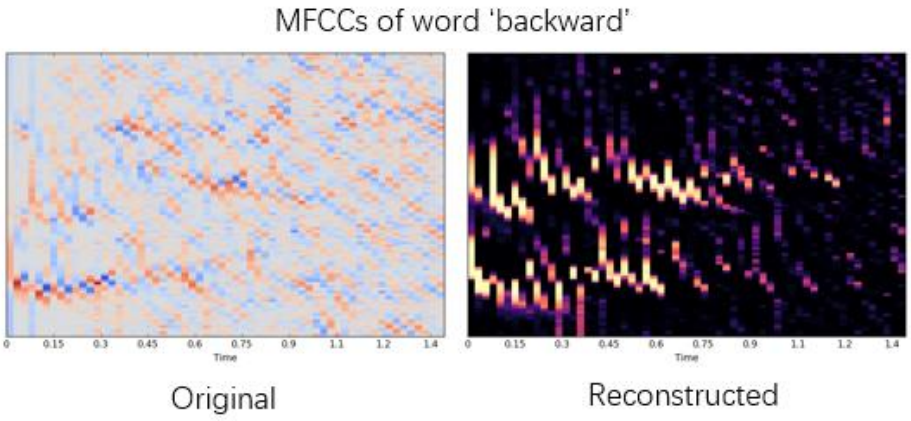


Fig 4.8. Original and reconstructed MFCCs feature map of word ‘backward’

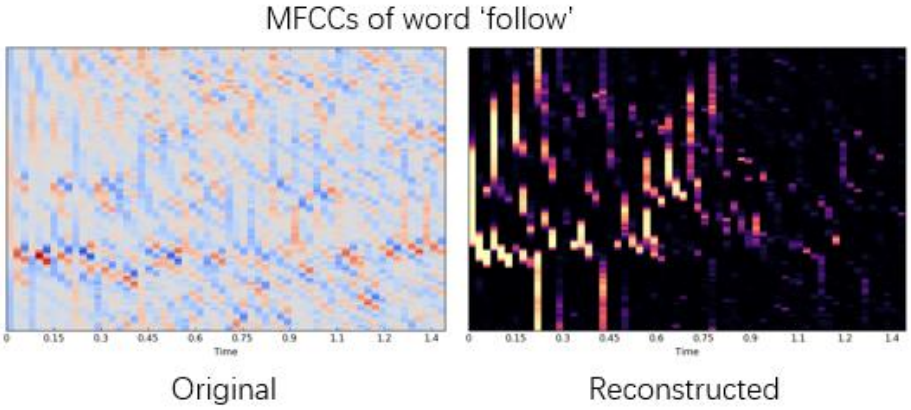


Fig 4.9. Original and reconstructed MFCCs feature map of word ‘follow’

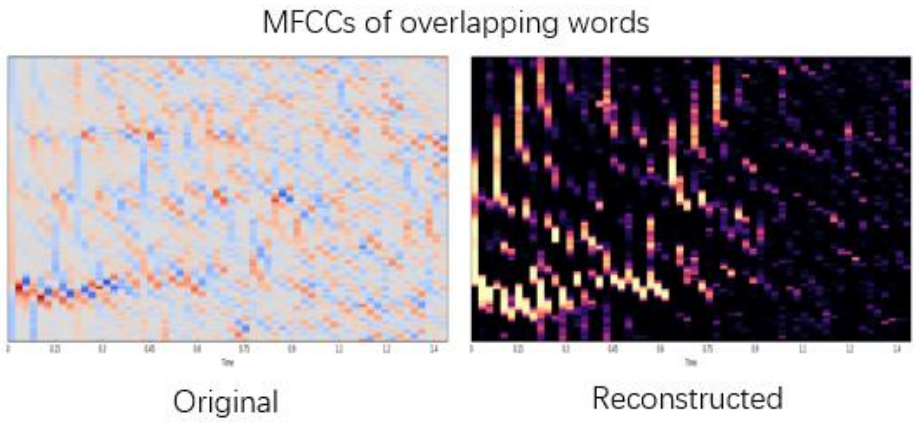


Fig 4.10. Original and reconstructed MFCCs feature map of overlapping speech consisting of ‘backward’ and ‘follow’

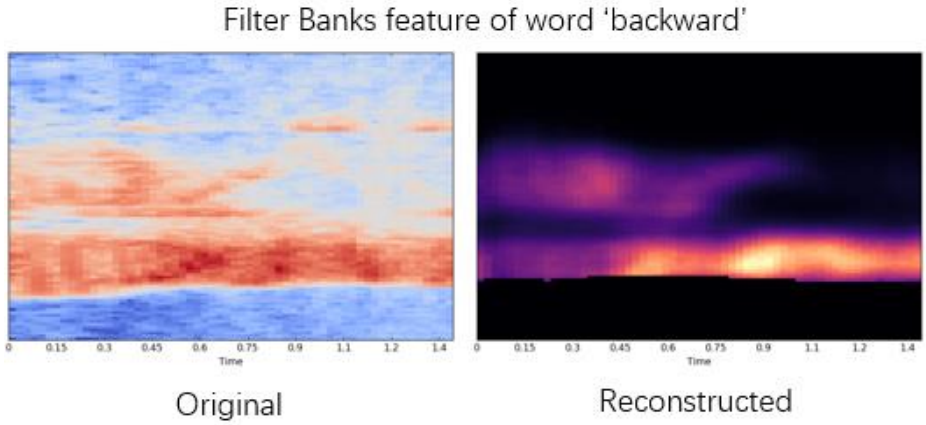


Fig 4.11. Original and reconstructed Filter bank feature map of word ‘backward’

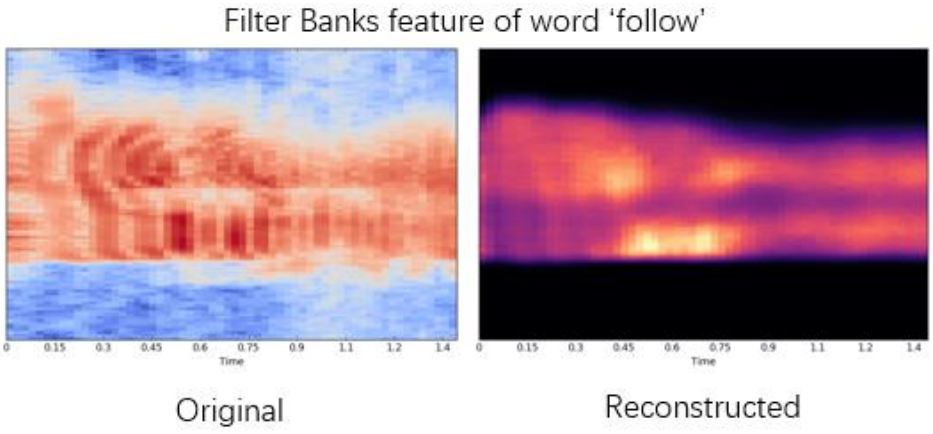


Fig 4.12. Original and reconstructed Filter bank feature map of word ‘follow’

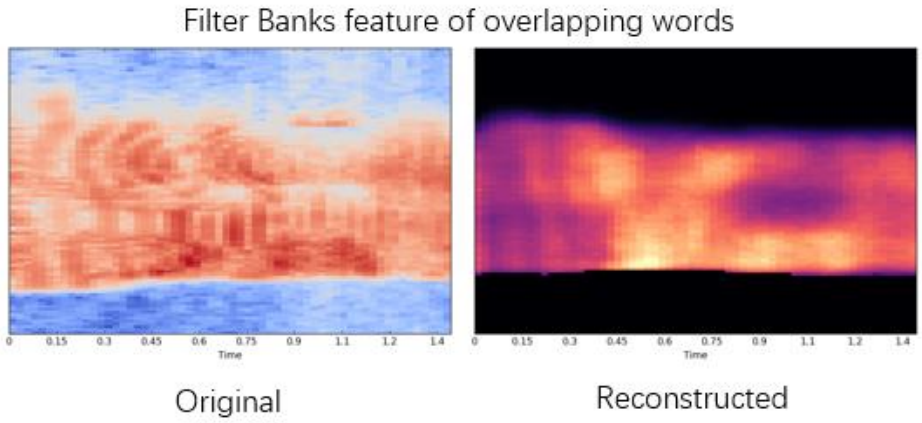


Fig 4.13. Original and reconstructed Filter bank feature map of overlapping speech consisting of ‘backward’ and ‘follow’

Capsule network trained by overlapping speech features can recognize individual speech features. The reconstructed feature map contains most of the features of the individual inputs. The reconstructed overlapping speech contains features from both inputs, which proves that capsule network is capable of recognizing every individual word in the overlapping speech.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis report described our work on evaluating the capsule network for image recognition and speech recognition tasks. For image recognition, we consider the problem of recognition of images with overlapping digits. We trained the capsule network and baseline CNN with training sets of different sizes and recorded for every epoch the training time, training loss and testing loss. We also recorded the best accuracy. A comparison of the performance of two networks showed that capsule network has faster convergence than baseline CNN network. For a small training set, the capsule network has higher testing accuracy. However, when training set became larger (more than 100,000 images), the accuracy of the two networks was comparable.

The results also showed that the training time of the capsule network for one epoch is much longer than that of baseline network (approximately 8 times of baseline network). So, we analyzed the training process using a NIVDIA profiling tool, NVVP. We found that in the baseline CNN, convolution operations and fully connected layers did not take up much time. The most time-consuming computations for baseline CNN are nonlinear operations like max pooling and ReLU. In the capsule network, the dynamic routing part takes up 41.5% of the total time. The dynamic routing algorithm requires several matrix-matrix multiplication kernels which were very time-consuming.

The computational complexity analysis also suggested that we could reduce training time per epoch by adjusting the capsule structure. So, we changed the size of primary capsules and class capsules and evaluated both accuracy performance and training time.

The results showed that the change in capsule sizes had no effect on accuracy but could cut down the training time by more than 40% (from 236s of 8-dim primary capsules to 163s of 32-dim capsules).

Next, we evaluated the performance of the capsule network for speech recognition. The specific problem that we considered was overlapping word recognition. We created wave files of two overlapping words for speaker-dependent and speaker-independent datasets. We used MFCC and filter bank to pre-process the speech signal. We started with 5 classes of words and trained the capsule network with different convolution kernel configurations. The results showed that rectangular kernels which covered more in time dimension worked better. We also trained a baseline CNN with rectangular kernels.

We showed that for both speaker-dependent and speaker-independent datasets, the capsule network achieved a much higher accuracy (96.92% with speaker-dependent and 83.68% with speaker-independent dataset) compared to baseline network (84.88% with speaker-dependent and 78.77% with speaker-independent dataset). However, the training time of the capsule network was approximately 4 to 5 times that of the baseline network. Next, we adjusted the capsule parameters and studied its effect on performance of the capsule network. The results showed increasing the primary capsule size to 32 and decreasing the class capsule size to 8 resulted in less than 1% change in accuracy. However, such a change decreased the training time significantly. It is now approximately 3 times that of the baseline CNN.

To further explore the potential of capsule network for overlapping words recognition, we added five more classes of word to the speech data set making the number of different words to mix with at ten. We used a training set of size 54,000 and testing set of size 10,800.

The testing accuracy of both networks decreased when training set is speaker-dependent. However, the capsule network kept a high accuracy of 95.42% when training set is speaker-independent while the baseline network's accuracy decreased from 84.88% to 73.18%. This experiment further proved that the capsule network has superior performance for overlapping word recognition task.

5.2 Future Work

There are several related problems that will be considered in the future:

1. The size of convolution kernel influences the network performance in terms of accuracy and training time. A set of experiments should be carried out to find the optimal kernel sizes.
2. Network with multiple capsule layers is worth investigating because in traditional CNN, multiple fully-connected layers are applied to improve the classification accuracy.
3. Mean-square-error is common but not the best choice for speech recognition. For reconstruction error-based regularization, better choice may be Itakura-Saito distance.
4. Dynamic routing makes use of inner product to judge whether a higher-level capsule 'agrees with' a lower-level capsule which is very time consuming. The timing problem can be mitigated by finding other techniques which have less computational load.

REFERENCES

[1] W.S. McCulloch and W. H. Pitts, “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical, Vol. 5, pp. 115-133, 1943

[2] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review, vol. 65, no. 6, 1958.

[3] G. E. Hinton, S. Osindero and Y. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, no. 18, pp. 1527-1554, 2006.

[4] Y. LeCun, Y. Bengio and G. E. Hinton, "Deep learning," Nature, vol. 521.7553, pp. 436-444, 2015.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based learning applied to document recognition," Processing of the IEEE, vol. 86(11), pp. 2278-2324, 1998.

[6] S. Sabour, N. Frosst, G. E. Hinton, “Dynamic routing between capsules,” Advances in Neural Information Processing Systems, 2017, pp. 3856-3866.

[7] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Representations by back-propagation errors," Nature, Vol. 323. 6088, pp. 533

[8] I. Goodfello, Y. Bengio and A. Courvile, "Deep learning," MIT press, 2016

[9] K. He, X. Zhang and S. Ren, "Deep residual learning for image recognition,"in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016

[10] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural network," Advances in neural information processing systems, 2012, pp. 1097-1105.

[11] C. Févotte, N. Bertin and L. J. Durrieu. "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis," Neural computation, 2009, vol. 21(3), pp. 793-830.

[12] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," Sixteenth Annual Conference of the International Speech Communication Association. 2015.

[13] D. P. Kingma and J. Ba "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[14] Y. LeCun, C. Cortes and C. JC. Burges, "The mnist database of handwritten digits,", 1998

[15] G. E. Hinton, Z. Ghahramani and Y. W. The, "Learning to parse images," Advances in neural information processing systems, 200, pp. 463-469.

[16] A. Krizhevsky, W. Wang and G. E. Hinton, "Transforming auto-encoders," International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2011: pp. 44-51.

[17] K. Greff, A. Rasmus, M. Berglund, et al. "Tagger: Deep unsupervised perceptual grouping," Advances in Neural Information Processing Systems. 2016, pp. 4484-4492.

[18] H. Song, M. Willi, J. J. Thiagarajan, et al. "Triplet Network with Attention for Speaker Diarization," arXiv preprint arXiv:1808.01535, 2018.

[19] M. Minsky, S. A. Papert, "Perceptrons: An introduction to computational geometry," MIT press, 2017.

[20] Xi E, Bing S, Jin Y. "Capsule network performance on complex data". arXiv preprint arXiv:1712.03480, 2017.