Performance Analysis of Mix-Kernel

Convolutional Neural Network

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Yibin Huang

Dr. Zhihai He, Thesis Supervisor

MAY 20

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

Performance Analysis of Mix-Kernel

Convolution Neural Network

presented by Yibin Huang,

a candidate for the degree of Master of Science and hereby certify that, in their opinion,

it is worthy of acceptance.

Professor Zhihai He

Professor Dominic Ho

Professor Yuyi Lin

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to all those who provide me the possibility to complete the master topic. A special gratitude I give to my academic advisor, Professor He, who offered me continuous support of Master study and related research with his patience and immense knowledge.

Furthermore, I would also like to acknowledge with much appreciation to the crucial role of my experiments, the powerful workstation with excellent GPU in Video Processing and Communication Lab. It is financed by professor He, and can dramatically speed up the experiments on CAFFE. I can never finish my CAFFE experiments in time without this high-efficiency workstation.

Last but not least, I would like to thanks all my lab-mates in Video Processing and Communication Lab: Chen Huang, Zhi Zhang, Guanghan Ning, Hayder Y. Yousif, Zhiqun Zhao, Liang Zhao, Wenqiang Bo, Yan Tang, Qiyuan Gao for the genius suggestions they provided and the enjoyable time we spent together.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

ABSTRACT

Deep convolutional neural networks (DCNN) have achieved the state-of-the-art performance in a number of computer vision tasks in recent years, including object detection, classification and recognition. The DCNN is very computation-intensive, whose computational complexity can be controlled by a set of network configuration parameters. The relationship between the DCNN computational complexity and its classification accuracy has not been well understood. In this thesis, we aim to conduct a series of training-testing experiments with DCNN on benchmark datasets, such as MNIST and CIFAR-10, to characterize the complexity-accuracy behavior of DCNN. We demonstrate that, with proper configuration of the DCNN, we are able to significantly reduce the computational complexity of DCNN without much degradation on the classification accuracy. This provides important guidelines for practical implementation and use of DCNN.

Chapter 1

Introduction

1.1 Convolutional Neural Network

In machine learning, a convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial network where the individual neurons are tiled in a way that they respond to overlapping region in the visual field [5]. CNNs are biologically-inspired [14] variants of multiplayer perceptron (MLP) and very similar to ordinary Neural Networks. The typical architecture [2] of ConvNets consists of several kinds of layers, convolutional layers, ReLU layers, pooling layers and fully-connected layers. Optionally, normalization layers and other activation function layers may also appear in the convolutional neural network.

Stacking a CNN with a fully connected DNN or with one or more CNNs gives rise to a deep CNN [31]. In [32], Visual Geometry Group evaluated very deep convolutional networks (up to 19 weight layers) for largescale image classification, which was demonstrated that the representation depth is beneficial for the classification accuracy. Paper [8] trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-1010 contest into the 1000 different classes to achieve the state-of-art performance.

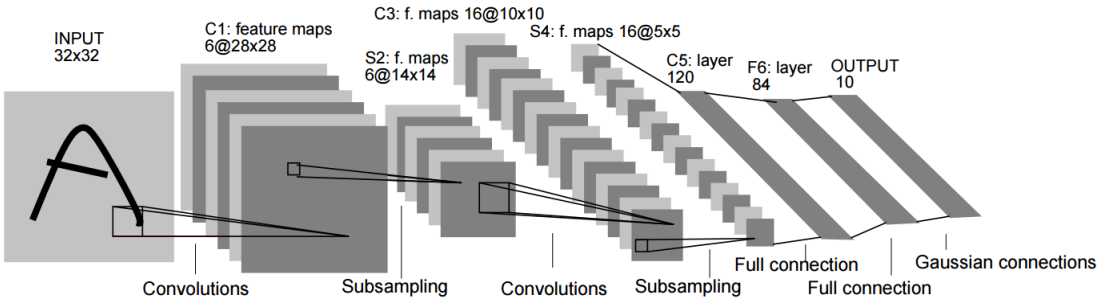

Figure1: A typical structure of CNN

1.1.1 Backpropagation algorithms

In CNN, Backpropagation, an abbreviation for "backward propagation of errors", is a supervised learning approach to train artificial neural networks. The algorithm can be divided into two phases, forward pass and backward pass. Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient.

- Forward pass: Input training samples to CNN and get the predicted outputs.
- Backward pass: Calculate the errors by predicted outputs and actual labels, then update the weights and bias from last layer to first layer.

1.1.2 DCNN Structure

Each convolutional layer [5] has several convolution kernels, also called neurons. Each neuron with a local receptive field is used to scan the input image, and the states of the neuron in corresponding location in a layer called a feature map [3]. In addition, in CNNs, each filter is replicated across the entire visual field. These replicated units

share the same parameterization (weight vector and bias) and form a feature map [2].The role of the convolutional layer is to detect local conjunctions of features from the previous layer [4].

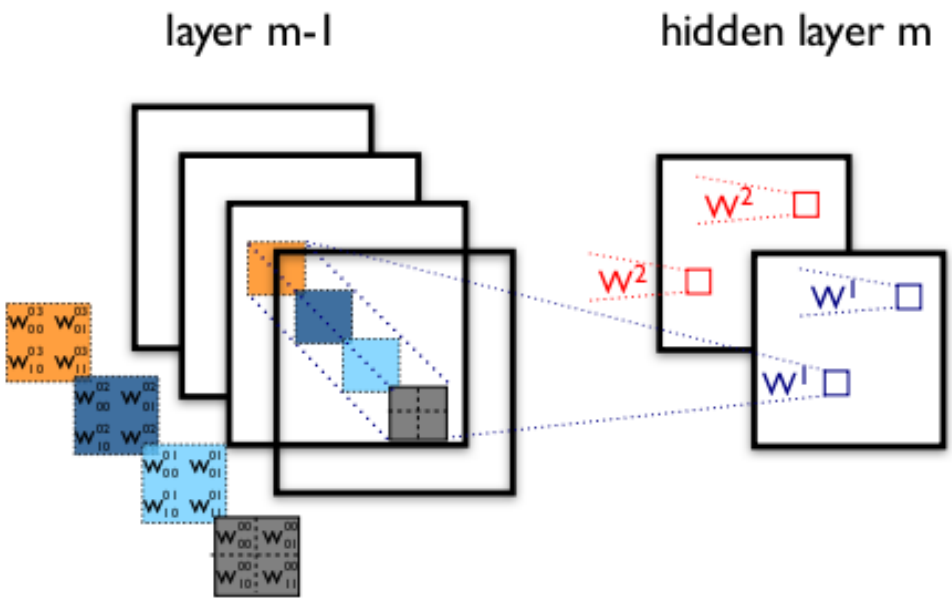

Figure2: Example of a convolutional layer.

ReLU layer is a rectified layer to increase the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolutional layer. A number of activation functions can be used in the layer, such as non-saturating activation function $f(x) = \max(0, x)$, saturating hyperbolic tangent $f(x) = \tanh(x)$, $f(x) = |\tanh(x)|$ and sigmoid function $f(x) = (1 + e^{-x})^{-1}$. With the help of nonlinear activation function, CNNs can perform nonlinear transform to achieve a better result than regular methods.

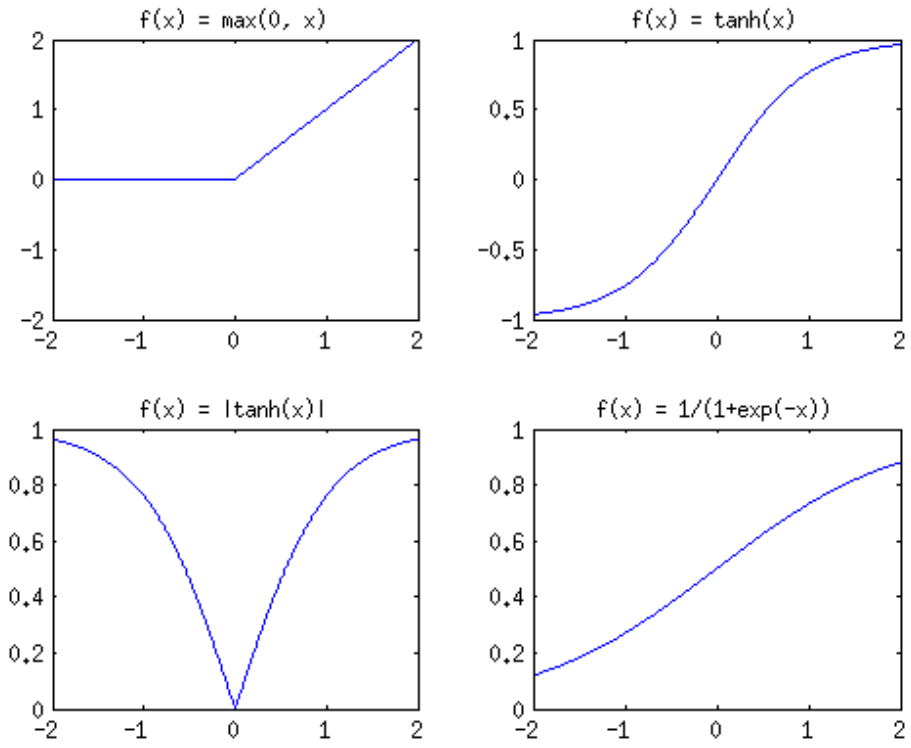

Figure3: Examples of activation functions

The pooling layer is a sub-sampling layer. Pooling reduces computation for upper layers and provides a form of translation invariance by computing a specific value of a particular feature over a region of the feature map. Max-pooling chooses the maximum value from the region while average-pooling calculates the average value over the local patch. The role of the pooling layer is to merge semantically similar features into one [4].

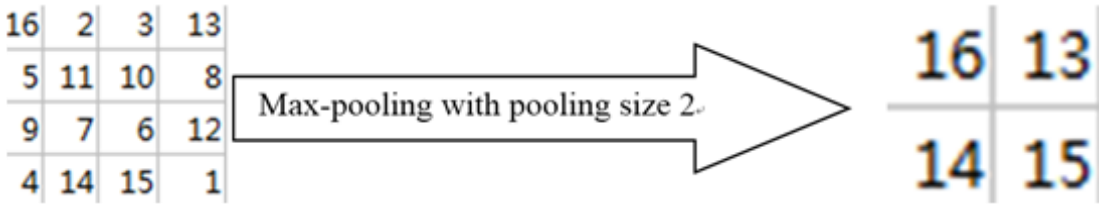

Figure4: An example of max-pooling

A fully-connected layer takes all neurons in the previous layer and connects it to every single neuron it has. Fully-connected layers transform 2-D data to one dimension features. Fully-connected layers are not spatially located anymore, so the methods if regular neural network can be applied.

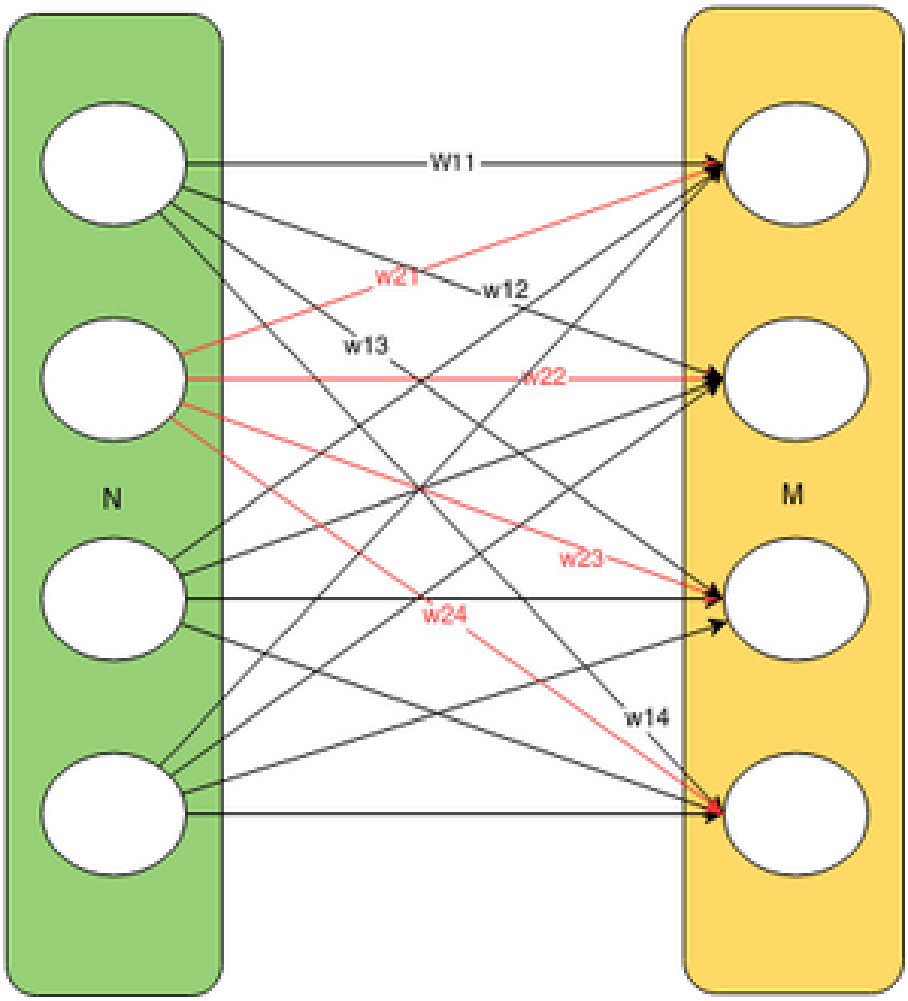

Figure5: Example of fully connected layer

According to a recent review paper on deep learning [4] recently, there are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling, and the use of many layers. All the explanations are summarized in the table.

Key ideas	Explanation
Local connections	In array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected.
Shared weights	If a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array.
Pooling	Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature.
Use of many layers	Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones.

Table1: Explanations of key ideas

1.2 Related Work

A number of techniques are proposed to improve the CNN performance. According to [1], the most straightforward way to improve the performance is by increasing their size. This includes both increasing the depth – the number of network levels – as well as its width: the number of units at each level. This method is an easy and safe way to achieve better CNN model. For example, the architecture of [6] consists of eight convolutional layers, one locally connected hidden layer and two densely connected layers, to process 32×32 images. The fully connected layers of [6] contain 3072 unit each.

However, the drawbacks are obvious. First, a large model means a lot of parameters and if the training samples are limited, causing the problem of overfitting. Another major disadvantage is that computational work will be dramatically increased. Thus, a number of other approaches are proposed to overcome the drawbacks. For instance, [8] takes over 5 days to train one dataset on two GTX 580 3GB GPUs.

1.2.1 Data Augmentation

The term data augmentation [16] refers to methods for constructing iterative optimization or sampling algorithms via the introduction of unobserved data or latent variables. In general, constructing data augmentation schemes that result in both simple and fast algorithms is a matter of art in that successful strategies vary greatly with the (observed-data) models being considered [16].

In computer vision, data augmentation is an approach to enlarge the training dataset, so the CNN model can be updated more accurate for the testing dataset. Usually, random noise and elastic distortion are popularly used to augment the training dataset. This is a very general approach and a lot of papers take advantage of data augmentation to improve their performance. Figure6 [18] is an example of data augmentation.

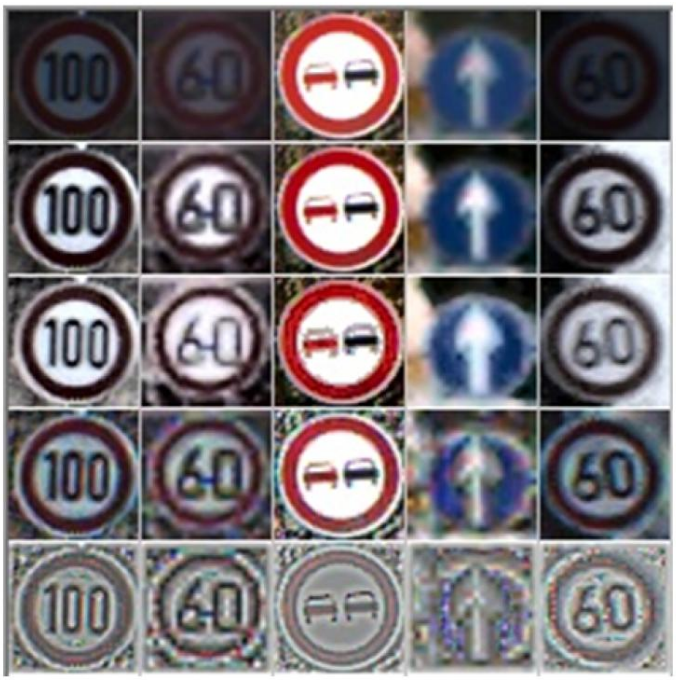

Figure6: Example of data augmentation

1.2.2 Data Validation

In CAFFE experiments, the original training dataset can be divided into real training set and validation set. The validation set is used to estimate how well the CNN model has been trained, so the model can tune the hyper-parameters and choose the early stop point. With the help of data validation, Yann LeCun [7] achieved state of art on MNIST (without distortions and preprocessing) in 2009. Validation is also a very

popular strategy to improve the CNN performance and it is always combined with data augmentation since the validation set will reduce the size of the final training set.

1.2.3 Data Preprocessing

Data preprocessing is a procedure to make the training data more effective for the algorithm, including data cleaning, normalization, transformation, feature extraction and selection, etc. The data preprocessing [17] can often have a significant impact on generalization performance of a supervised machine learning algorithm. In a broad sense, data augmentation and data validation may also regarded as a subset of data preprocessing, but in this master thesis, local contrast normalization and ZCA whitening are applied on CIFAR-10.

In small images, pixels show very strong correlations in the spatial domain, vertically and horizontally. When learning a statistical model of images, it might be nice to force the model to focus on higher-order correlations rather than get distracted by modeling two-way correlations [20]. Luckily, ZCA whitening (Zero-phase Component Analysis Whitening) can force the model to ignore second-order structure, and after transformation, it will be impossible to predict the value of one pixel given the value of only one other pixel.

Local contrast normalization (LCN) is an approach to normalize the contrast of one image in a non-linear way. Instead of performing global normalization based on the range of values of the entire image, the pixel value changes of LCN are only

corresponding to a local patch of the image. The method first removes the mean of a neighborhood of pixels from a particular pixel and then divides each pixel by their variance. Local contrast normalization is another general and simple way to preprocess the data.

1.2.4 Overlapping Pooling

Paper [8] shows that overlapping pooling can reduce the error rate compared with non-overlapping scheme, which produces output of equivalent dimensions. It also makes the models slightly more difficult to overfitting. Generally, the parameters of overlapping are kernel 3×3 with stride 2.

1.2.5 Stochastic Pooling

Stochastic pooling [9] is a simple yet effective method for regularizing large convolutional neural networks. The conventional deterministic pooling operations are replaced with a stochastic procedure, randomly picking the activation within each pooling region. This approach, according the paper [9], achieved state-of-the-art performance on four image datasets, relative to other methods that do not utilize data augmentation.

1.2.6 Dropout Method

Dropout [10] is a technique to overcome the problem of overfitting. The key idea is to randomly drop units (along with their connections) from the neural network during

training [10]. Standard backpropagation learning builds up brittle co-adaptations that work for the training data but do not generalize to unseen data, while random dropout breaks up these co-adaptations by making the presence of any particular hidden unit unreliable [10]. This technique works on supervised learning tasks in vision, speech recognition, document classification and computational biology. Figure7 is quoted from the dropout paper [10].

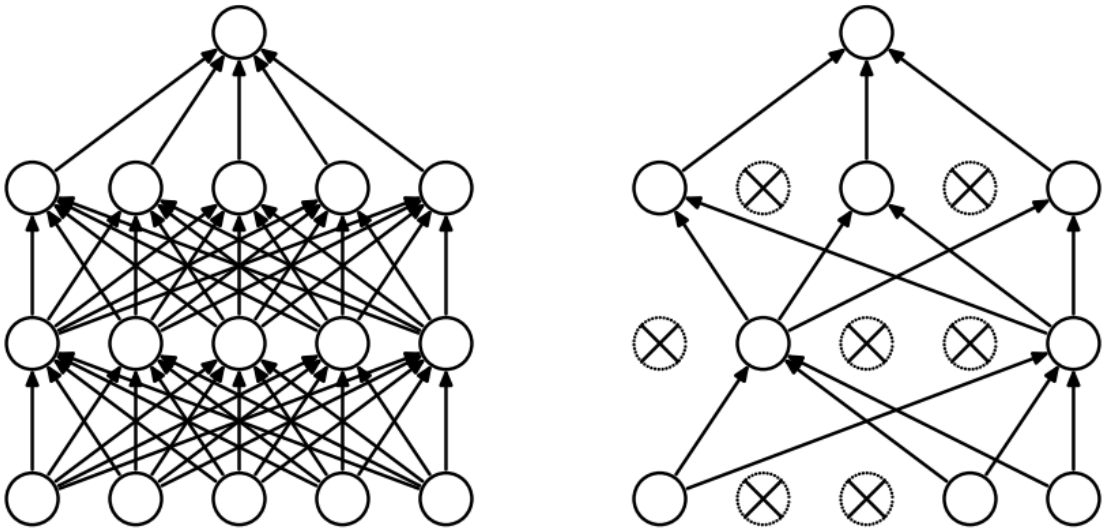

Figure7: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped

1.2.7 Maxout Networks

The maxout [11] model is simply a feed-forward architecture, such as a multiplayer perceptron or deep convolutional neural network, which uses a new type of activation function: the maxout unit. Combined with dropout method, maxout model achieved

state-of-the-art classification performance on MNIST, CIFAR-10, CIFAR-100, and SVHN.

Combining the methods in [11] and [23], [24] reports a novel deep architecture referred to as Maxout network In Network, which can enhance the model discriminability and facilitate the process of information abstraction within the receptive field.

1.2.8 DropConnect

DropConnect [13] is the generalization of dropout in which each connection, rather than each output unit, can be dropped with probability 1-p. The fully connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen at random during the training stage.

1.3 CAFFE

There are a number of experimental tools for deep learning, for example, CAFFE [15], cuda-convnet [25], Decaf [26], OverFeat [27], Pylearn2 [28] and Torch7 [29]. In my work, CAFFE is the only experimental tool to achieve the Mix-Kernel model.

CAFFE [15] provides multimedia scientist and practitioners with a clean and modifiable framework for state-of-art deep learning algorithms and a collection of reference models. It is the main experimental tool used in this master thesis. With the help of an active community of contributors on GitHub, Berkeley Vision and Learning Center (BVLC) is maintaining and developing CAFFE. A number of famous models can be achieved by CAFFE, such as GoogLeNet [1], AlexNet [8], etc.

According to [15], there're four reasons to choose CAFFE, expressive architecture, extensible code, speed and community. Flexible CAFFE model encourages application and innovation without hard-coding. A "solver_mode" flag is used to switch between CPU and GPU. CAFFE has been forked by over 1,000 developers and had many significant changes contributed back. The contributors and the framework tracks the state-of-the-art in both code and models. The BLVC believes that CAFFE is the fastest convent implementation available. It can process over 60M images per day with a single NVIDDIA K40 GPU*, which means 1ms/image for inference and 4ms/image for learning. A number of academic research projects, startup prototypes, and even large-scale industrial applications in vision, speech, and multimedia are powered by the fully open source framework CAFFE.

Framework	CAFFE [15]	Cuda-convnet [25]	Decaf[26]	OverFeat[27]	Pylearn2[28]	Torch7[29]
License	BSD		BSD		BSD	BSD
Core language	C++	C++	Python	Lua	Python	Lua
Binding(s)	Python/MATLAB	Python		C++/Python		
CPU	√		√	√	√	√
GPU	√	√			√	√
Open source	√	√	√		√	√
Training	√	√	√		√	√
Pre-train	√		√	√		
Development	distributed	discontinued	discontinued	centralized	distributed	distributed

Table2: Comparison of popular deep learning frame works. Core language is the main library language, while bindings have an officially supported library interface for feature extraction, training, etc.

1.3.1 Blobs

A blob [15] is a wrapper over the actual data being processed and passed along by CAFFE, and also under the hood provides synchronization capability between the CPU and the GPU. It is the unit of data in CAFFE, providing a unified memory interface, holding batches of images (or other data), parameters or parameter updates.

1.3.2 Layers

A CAFFE layer [15] is the essence of a neural network layer: it take one or more blobs as input, and yields one or more blobs as output. It is the essence of a model and the fundamental unit of computation. A number of kinds of layers are offered by CAFFE, including pooling, convolution, inner products, all kinds of activation functions, local response normalization, dropout, concatenation……

1.3.3 Nets

The CAFFE net [15] jointly defines a function and its gradient by composition and auto-differentiation. It is a set of layers connected in a computation graph – a directed acyclic graph (DAG) to be exact. A typical CAFFE network begins with a data layer that loads from disk and ends with a loss layer that computes the objective for a task such as classification or reconstruction [15].

1.4 Overview of Proposed Approach

The DCNN is very computation-intensive, whose computational complexity can be controlled by a set of network configuration parameters. The relationship between the DCNN computational complexity and its classification accuracy has not been well understood. In this thesis, we aim to conduct a series of training-testing experiments with DCNN on benchmark datasets, such as MNIST and CIFAR-10, to characterize the complexity-accuracy behavior of DCNN. First, five kinds of kernels (1×1, 3×3, 5×5, 7×7 and 9×9) are combined to produce twenty-six different types of Mix-Kernel layers. Then, thirty-one kinds of Mix-Kernel modules and Pure-Kernel modules are used to obtain the results through CAFFE experiments. For different datasets, the best Mix-Kernel module is usually different.

Chapter 2

Mix-Kernel model

2.1 Motivation

In convolutional layers, each convolutional kernel, (named neuron) is connected to a small region of the input feature maps. The certain region in the feature map is called local receptive filed. For example, a 3×3 size neuron, corresponding to 3×3 local receptive field, will extract 3×3 convolution features in each operation while a convolutional kernel with size 5×5, relating to 5×5 certain region, can extract 5×5 convolution features in each operation.

In most CNN models, each layer only has one single size neurons, mostly 3×3 or 5×5. Hence in one layer, only one size local receptive filed is existed to extract a single size convolution features to form a final feature map. However, in practice, different size of convolutional kernels are used for different datasets, which means the size of convolutional kernels do effect the performance of the CNN in different datasets.

Then the idea of the paper comes up naturally. Why don't we use different sizes of kernels in each layer? So we can extract different sizes of convolution features to form the final feature maps.

GoogLeNet [1] has similar Inception module, but my work tried all Mix-Kernel combinations. By comparing all the possibilities, I can find the best choice for different datasets.

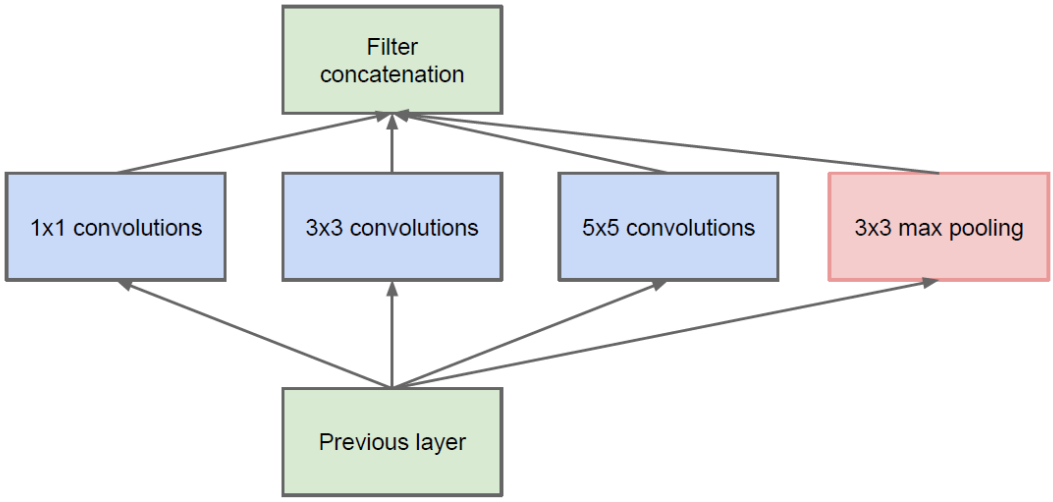

Figure8: Naive Inception model of GoogLeNet

2.2 Mix-Kernel Module

In a Mix-Kernel module, over one size of neurons are used to achieve the convolution operator. For example, a 3/5 Mix-Kernel module has 3×3 and 5×5 neurons. In this mater thesis, I choose 5 sizes of neurons (1×1, 3×3, 5×5, 7×7 and 9×9). Eventually, there are totally 31 combinations, while 26 are Mix-Kernel module and 5 are so-called Pure-Kernel module. Thus in one layer of CNN, several different size kernels connecting to different size local receptive fields, will extract different size convolution features. The feature maps through Mix-Kernel modules are naturally more variable than the regular ones.

Pure-Kernel module	
Number of size	Combination
1	1×1, 3×3, 5×5, 7×7, 9×9
Mix-Kernel module	
Number of Size	Combination
2	1/3, 1/5, 1/7, 1/9, 3/5, 3/7, 3/9, 5/7, 5/9, 7/9
3	5/7/9, 3/7/9, 3/5/9, 3/5/7, 1/7/9, 1/5/9, 1/5/7, 1/3/9, 1/3/7, 1/3/5
4	3/5/7/9, 1/5/7/9, 1/3/7/9, 1/3/5/9, 1/3/5/7
5	1/3/5/7/9

Table3: List of combinations

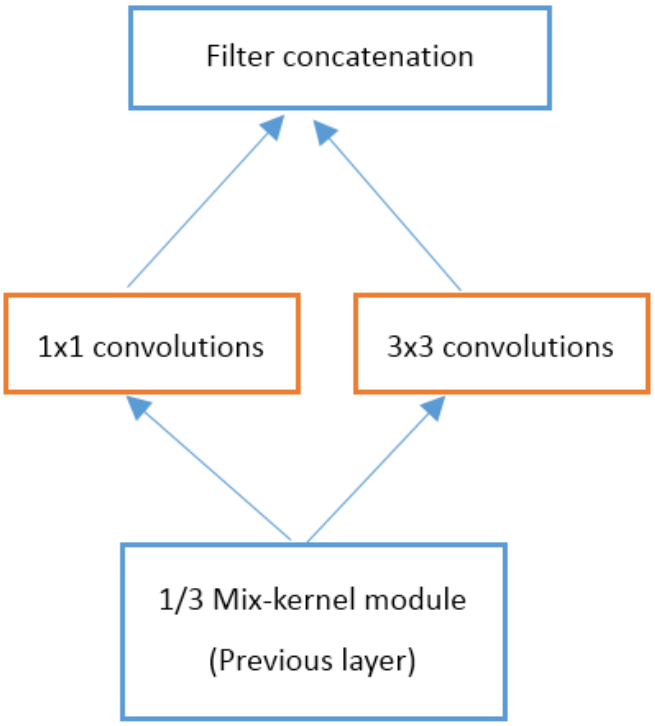

Figure9: Example of 1/3 Mix-Kernel module

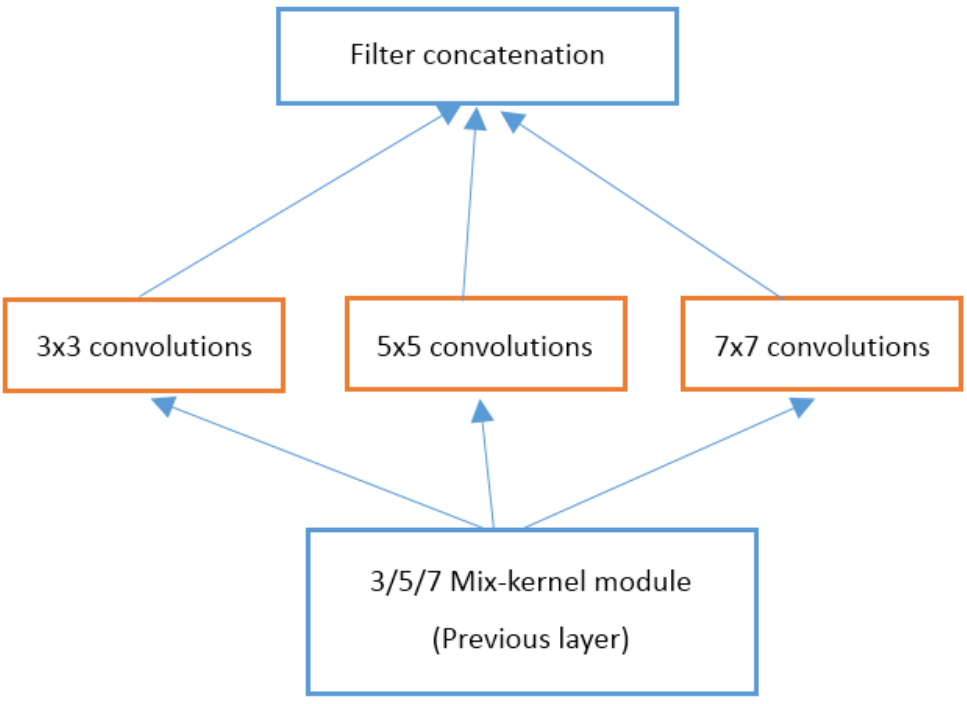

Figure10: Example of 3/5/7 Mix-Kernel module

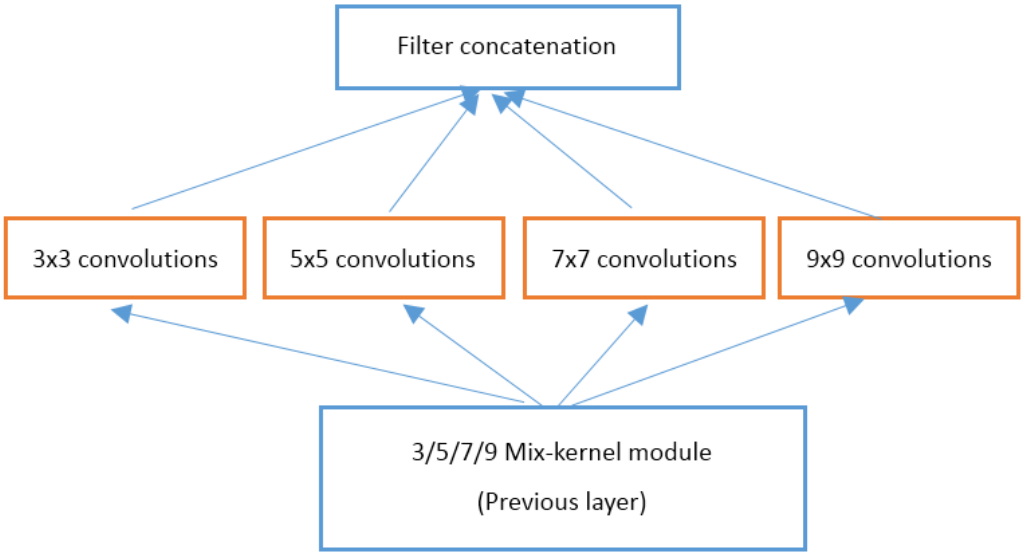

Figure11: Example of 3/5/7/9 Mix-Kernel module

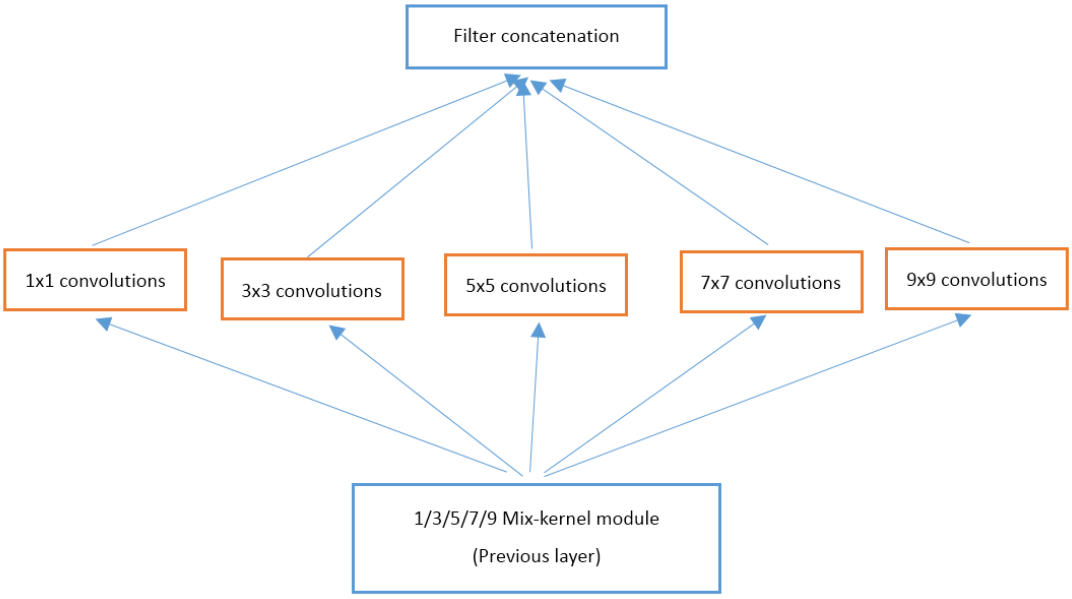

Figure12: Example of 1/3/5/7/9 Mix-Kernel module

2.2.1 Padding

A convolution operator usually shrinks the size of a feature map from last layer. For example, a 5×5 convolutional kernel doing convolutions on a 32×32 image, will get a 30x30 feature map. In the Mix-Kernel module, we have to get the same size of feature maps through different size of neurons. Therefore, padding is a very important operation.

Neuron size	1×1	3×3	5×5	7×7	9×9
Padding size	0	1	2	3	4

Table4: Neuron size with matched padding size

2.2.2 Filter Concatenation

There is a concatenation layer in CAFFE. The “Concat” layer [15] is a utility layer that concatenates its multiple input blobs to one single output blob. The input is several blobs with same size and the output is one single blob.

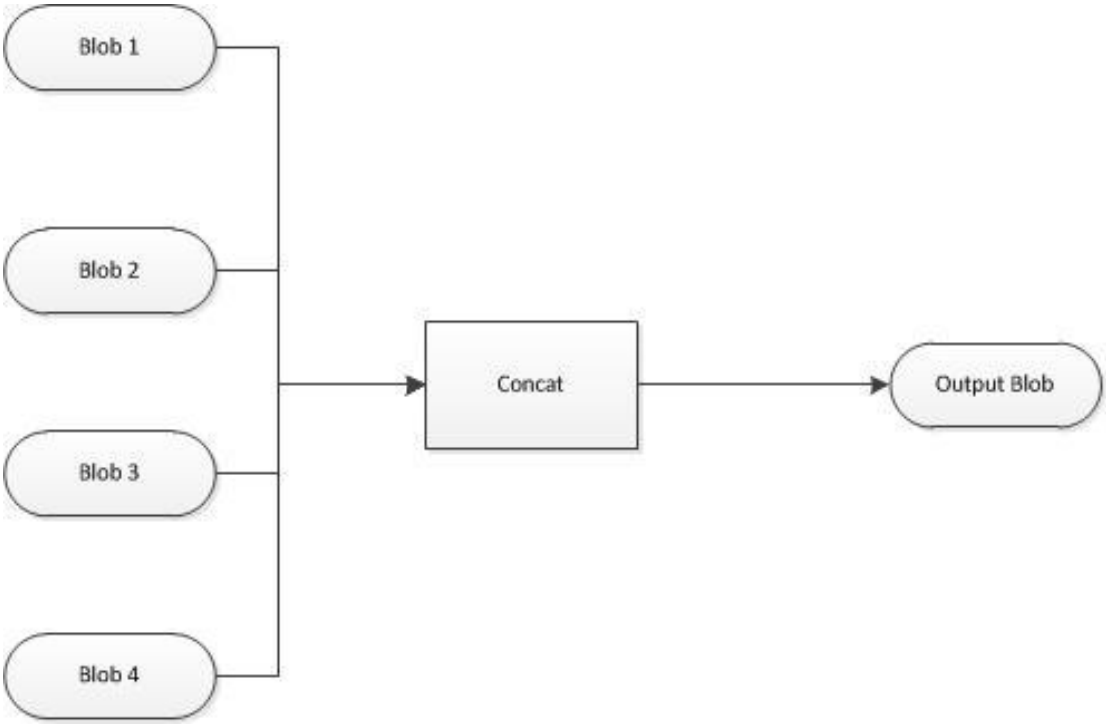

Figure13: Example of “Concat” layer in CAFFE

2.2.3 Number of Feature Maps

Decision of the feature map number is another important point. If we set the number too large, the experimental computation would be too large, and our lab do not have so much computational resources; if we set the number too small, the neuron may not extract enough features and the accuracies are not comparable. Therefore, an appropriate number is necessary to achieve the experiments.

Figure14 shows the CAFFE experiments on CIFAR-10. The architecture is three convolutions followed by a fully connected layer with soft-max outputs. The number of feature maps in 3 convolution layers are the same (8, 12, 16, 20, 24, 28, 32, 36, 40, 44, and 48). As the number of feature maps increasing, the accuracy of models rises at the first and then keeps stable within a certain range. In this experiment, all sizes of

convolutions achieve stable accuracy after 36 feature maps, thus the feature map number for each size neuron is 36 and the feature map number for each layer is 36×5 = 180.

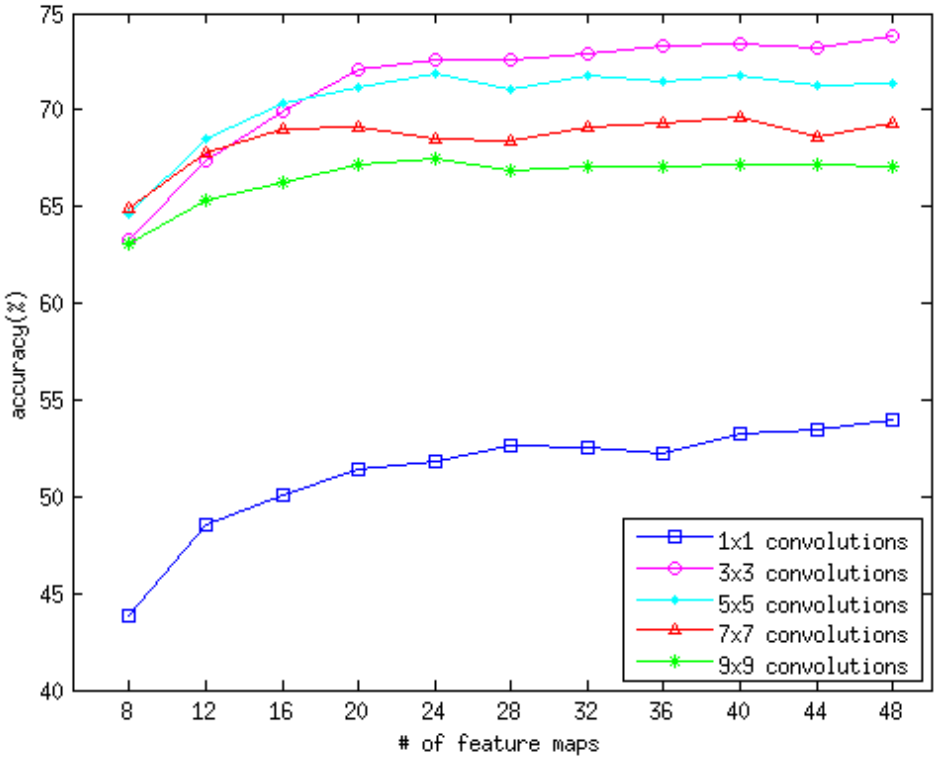

Figure14: Experiments on CIFAR-10

2.2.4 Stable Accuracy

In CAFFE experiments, even you use exactly the same model and same solver, the accuracy may be not the same from time to time. It's mainly because that the initialization and the optimization procedure of the CNN model are different in each time. But in my experiments, a stable result will be very important since I'm going to

compare thirty-one accuracies to choose the best module. Because of time limited, I am not allowed to run each model a lot of times to get the average accuracy.

In practice, one solution would be using as many iterations as possible with small learning rate. For example on MNIST, learning rate 0.01, is used to run 12,000 iterations, then another 12,000 iterations by reducing learning rate factor of 10.

Chapter 3

Experiments

3.1 Experimental Environment

All the CNN experiments run on the powerful workstation in Video Processing and Communication Lab. The workstation has 32 GB memory and an excellent GPU (GTX 980). The operating system is Ubuntu 14.04 LTS. Some figures are created by MATLAB and some flow charts are created by Microsoft Office Visio.

In this master thesis, two benchmark datasets are used to test this method, MNIST and CIFAR-10. The dataset will first run on easy experimental models to find the best Mix-Kernel module.

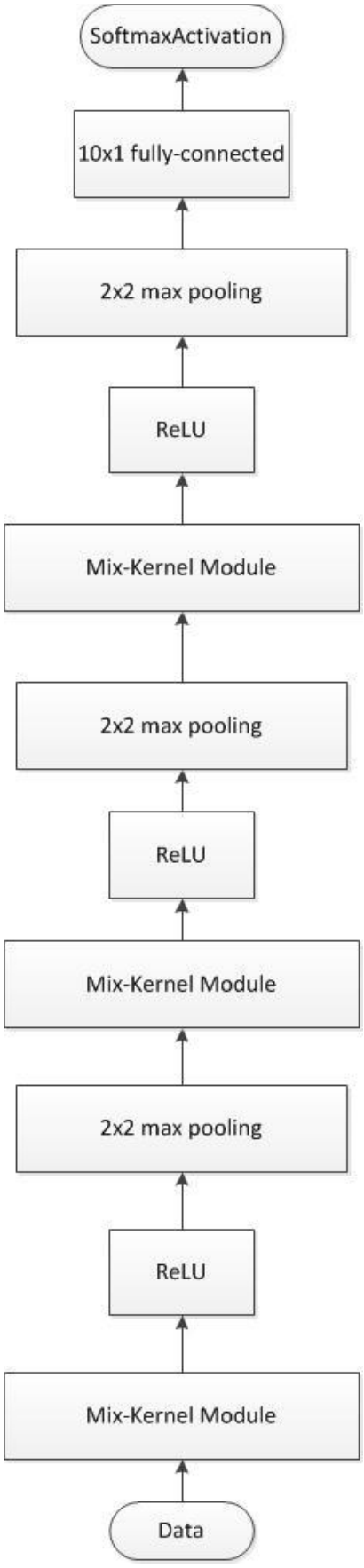

Figure15: Easy experimental model

3.2 Results

3.2.1 Performance Evaluations on the MNIST Dataset

The MNIST [19] database of handwritten digits, is a subset of a larger set available from NIST, having a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image [19]. The size of the image is 28 ×28, thus the 3-convolutions structure is enough to process the dataset. Figure16 [21] shows some examples from MNIST.

Figure16: Examples of MNIST

Mix/Pure-Kernel module	# of feature maps in each layer	# of feature maps in single size neuron	accuracy
1×1	160	160	82.39%
3×3	160	160	99.33%
5×5	160	160	99.32%
7×7	160	160	99.28%
9×9	160	160	99.30%
1/3	160	80	99.17%
1/5	160	80	99.25%
1/7	160	80	99.23%
1/9	160	80	99.16%
3/5	160	80	99.24%
3/7	160	80	99.34%
3/9	160	80	99.26%
5/7	160	80	99.25%
5/9	160	80	99.26%
7/9	160	80	99.26%
5/7/9	159	53	99.32%
3/7/9	159	53	99.26%
3/5/9	159	53	99.37%
3/5/7	159	53	99.34%
1/7/9	159	53	99.29%
1/5/9	159	53	99.28%
1/5/7	159	53	99.27%
1/3/9	159	53	99.23%
1/3/7	159	53	99.31%
1/3/5	159	53	99.31%
3/5/7/9	160	40	99.29%

1/5/7/9	160	40	99.25%
1/3/7/9	160	40	99.22%
1/3/5/9	160	40	**99.38%**
1/3/5/7	160	40	99.17%
1/3/5/7/9	160	32	99.30%

Table5: Results of easy experimental model on MNIST

According the method of 2.2.3, we use the same approach to get the best number of feature maps for single size neuron, which is 32. Therefore the number of feature maps in each layer is 32×5=160. All the results from table5 are trained by the easy experimental model with the exactly same solver. Through experiments, the best combination for the MNIST is 1/3/5/9 Mix-Kernel model, which achieve 99.38% in easy experimental model.

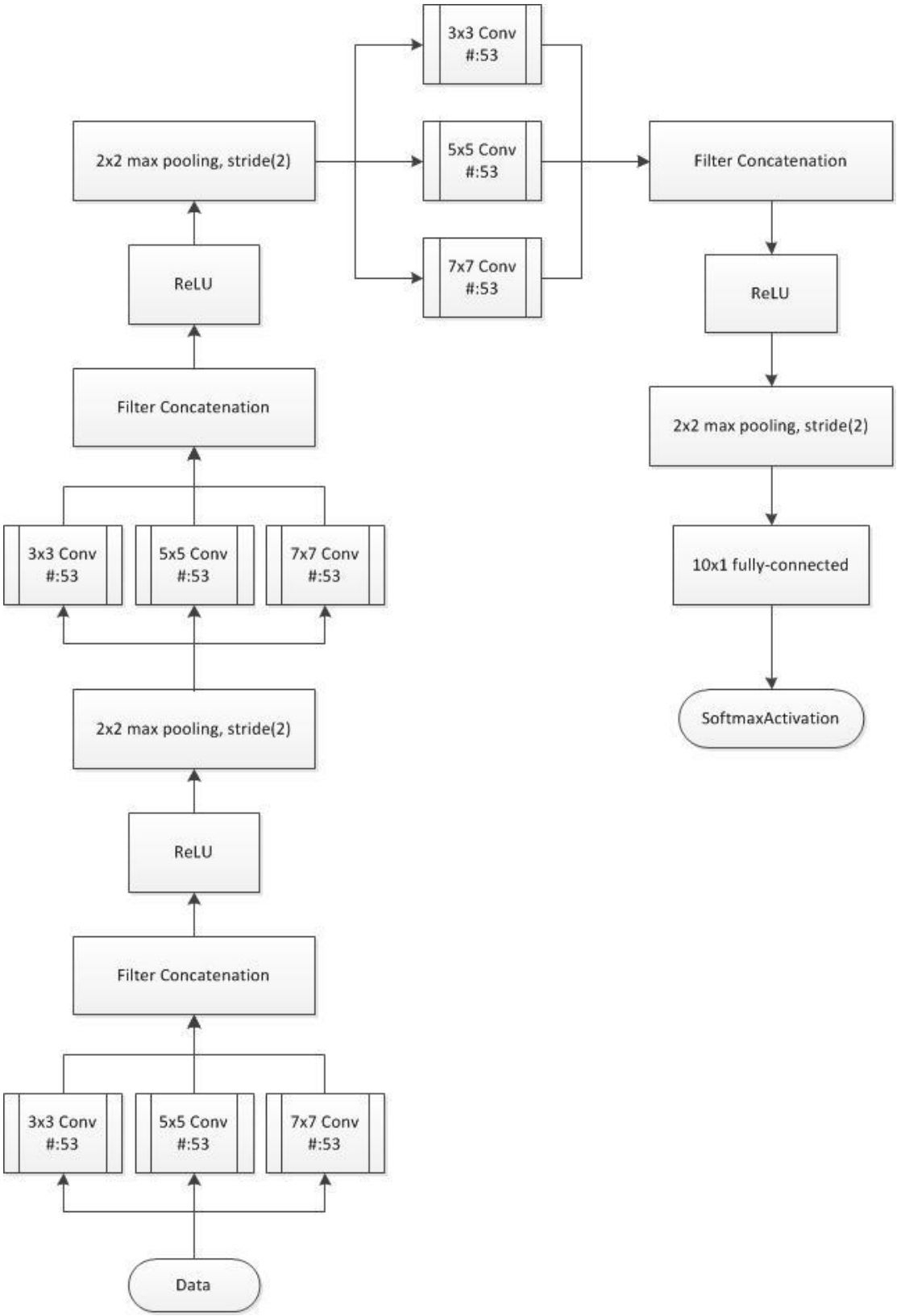

Figure17: Example of 3/5/7 Mix-Kernel easy experimental model on MNIST

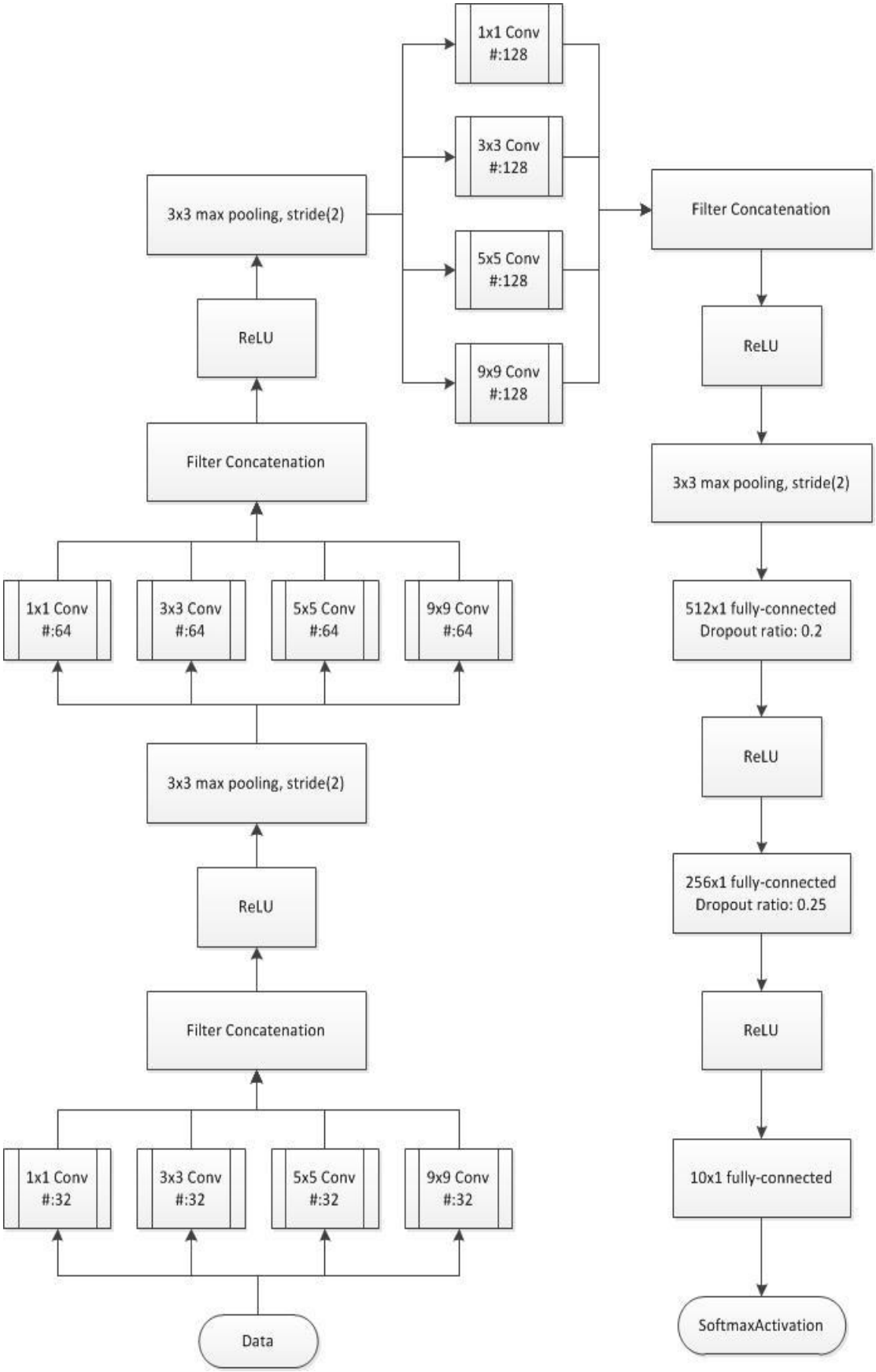

Figure18: Best Mix-Kernel model for MNIST

Finally in practice, I can obtain 0.43% test error on MNIST without data augmentation. The best Mix-Kernel model for MNIST consists 3 Mix-Kernel modules, followed by 3 fully-connected layers. The number of feature maps of single size neuron in convolutions layer are 32, 64, and 128 respectively. Overlapping max pooling is used to improve the performance with kernel 3×3 and stride 2. The dropout method only used in the first and second fully-connected layers.

As far as I know, the 0.39% test error is the state-of-the-art on MNIST without data augmentation. And this proposed method can achieve a very competitive result on MNIST. In my work, no other data preprocessing is used in this dataset.

Method	**Error**
3-Conv[7]	0.53%
Maxout[11]	0.45%
Stochastic Pooling[9]	0.47%
Deeply-Supervised Nets[22]	0.39%
NIN[23]	0.47%
Mix-Kernel + dropout	0.43%

Table6: Results on MNIST without data augmentation

3.2.2 Performance Evaluations on the CIFAR-10 Dataset

CIFAR-10 [20] is another benchmark dataset for object classification. It consists of 60,000 32×32 color images in 10 classes, with 6,000 image per class. There are 50,000 training images and 10,000 test images. Local contrast normalization and ZCA whitening are applied to preprocess the dataset.

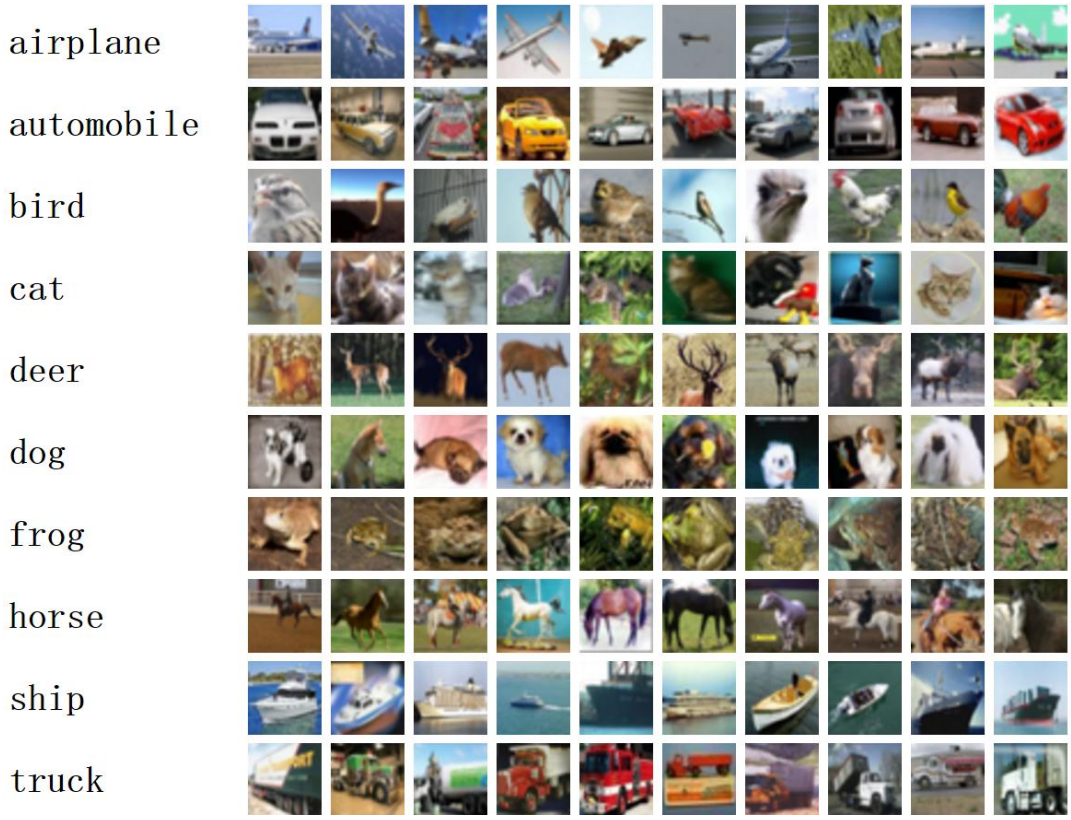

Figure19: Examples of CIFAR-10

Mix/Pure-Kernel module	# of feature maps in each layer	# of feature maps in single size neuron	accuracy
1×1	180	180	56.12%
3×3	180	180	75.46%
5×5	180	180	74.67%
7×7	180	180	68.88%
9×9	180	180	66.25%
1/3	180	90	75.37%
1/5	180	90	74.38%
1/7	180	90	72.95%
1/9	180	90	71.34%
3/5	180	90	**76.13%**
3/7	180	90	75.17%
3/9	180	90	75.53%
5/7	180	90	72.89%
5/9	180	90	71.73%
7/9	180	90	64.90%
5/7/9	180	60	70.88%
3/7/9	180	60	73.13%
3/5/9	180	60	74.17%
3/5/7	180	60	75.11%
1/7/9	180	60	71.02%
1/5/9	180	60	72.54%
1/5/7	180	60	74.26%
1/3/9	180	60	73.07%
1/3/7	180	60	75.20%
1/3/5	180	60	74.94%
3/5/7/9	180	45	73.18%

1/5/7/9	180	45	73.12%
1/3/7/9	180	45	73.29%
1/3/5/9	180	45	74.76%
1/3/5/7	180	45	75.18%
1/3/5/7/9	180	36	55.85%

Table7: Results of easy experimental model on CIFAR-10

According to 2.2.3, the number of feature maps for single size neuron is 36, therefore the number of feature maps in one layer is $36 \times 5 = 180$. All the results from table7 are trained by the easy experimental model with the exactly same solver. Through experiments, the best combination for the CIFAR-10 is 3/5 Mix-Kernel model, which achieve 76.13% in easy experimental model.

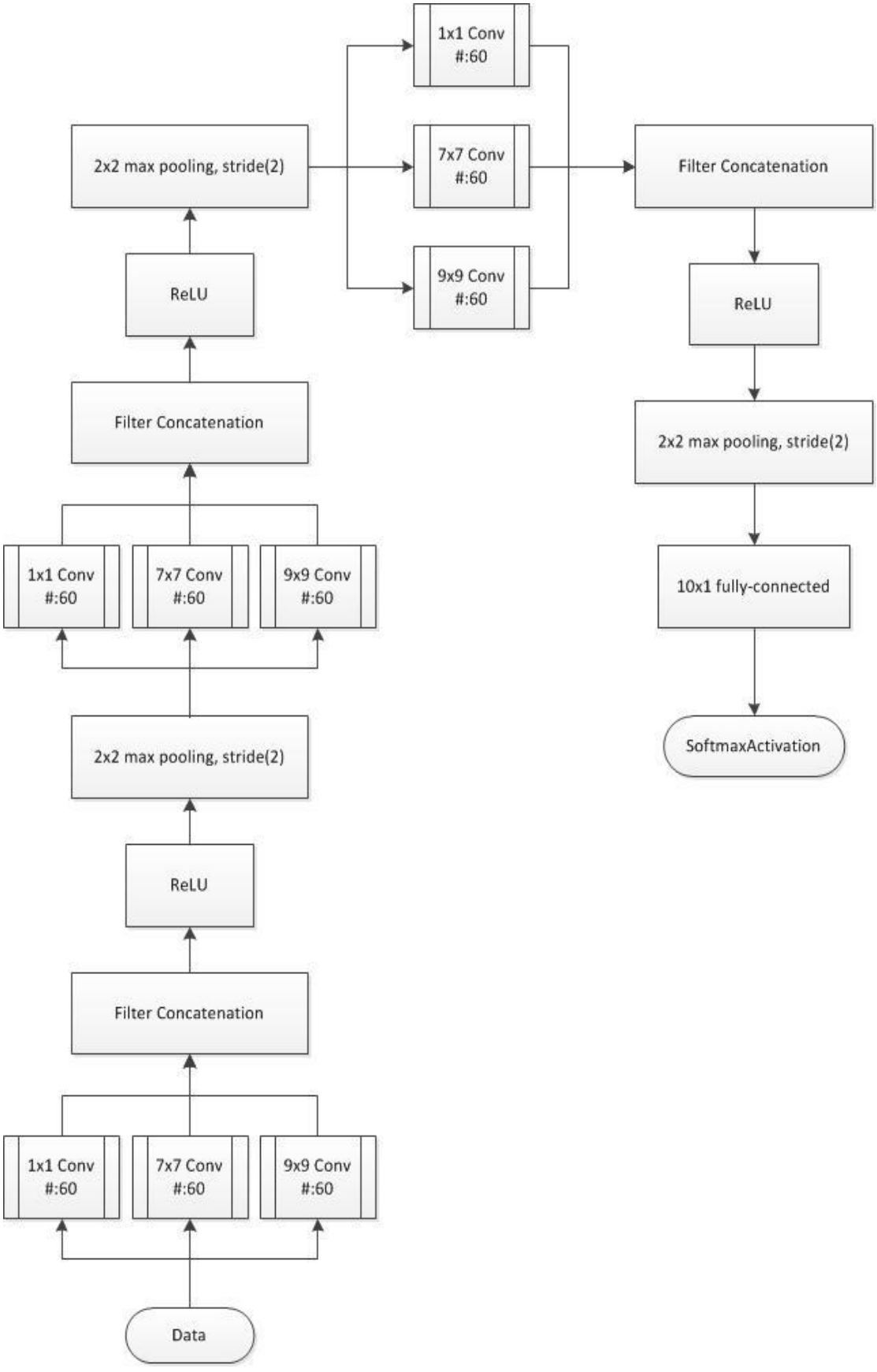

Figure20: Example of 1/7/9 Mix-Kernel easy experimental model on CIFAR-10

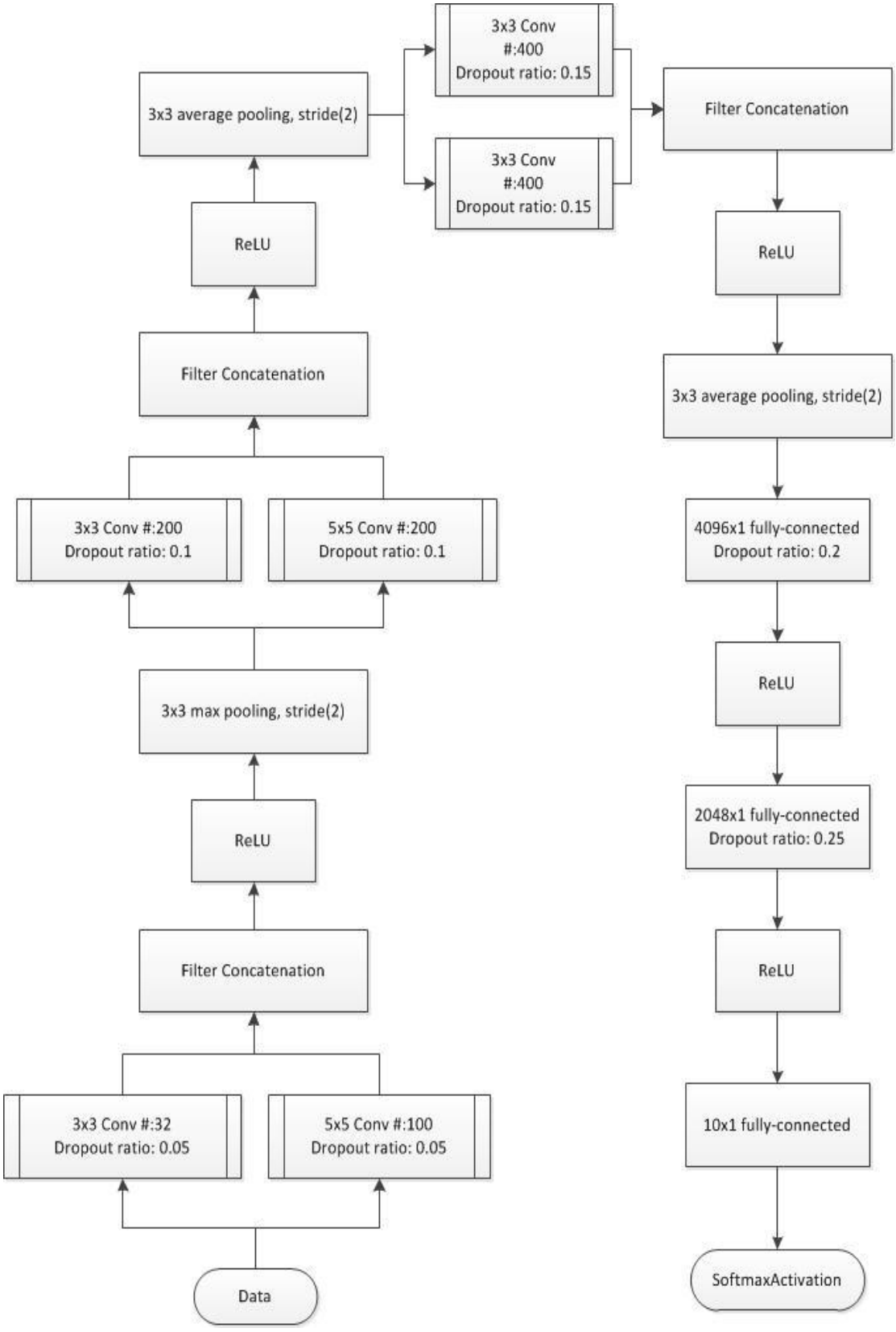

Figure21: Best Mix-Kernel model for CIFAR-10

The best Mix-Kernel model for CIFAR-10 has 3 Mix-Kernel modules and 3 fully-connect layers. The dropout method is applied in both convolution and fully-connected layers while the dropout ratio are 0.05, 0.1, 0.15, 0.2 and 0.25. The first overlapping pooling is max pooling and the rest two are average pooling. Finally, I can obtain 88.47% accuracy on this dataset.

Although the accuracy is not good enough to compete with the state-of-art result, but this method does improve the pure dropout method over 1% under the similar conditions (same preprocessing, no data augmentation and same net depth).

Method	**Error**
Dropout[10]	12.67%
Maxout[11]	11.68%
Stochastic Pooling[9]	15.13%
Deeply-Supervised Nets[22]	9.69%
NIN[23]	10.41%
Mix-Kernel + dropout	11.53%

Table8: Results on CIFAR-10 without data augmentation

Chapter 4

Conclusions and Future Work

4.1 Conclusions

In this master thesis, a simple method is proposed to improve CNN performance. I apply the method on two datasets, obtaining a competitive result on MNIST and improving the pure dropout method over 1% on CIFAR-10. However, not all the Mix-Kernel combinations can enhance the performance, some combinations do improve the accuracy, but some may be harmful to the results. Because of the limited datasets applied, Mix-Kernel module may not be certain to promote CNN performance in every other datasets. For different datasets, the best matched Mix-Kernel module may be different, for instance, the 1/3/5/9 is the best combinations for MNIST and the best matched module on CIFAR-10 is 3/5 Mix-Kernel Module.

4.2 Future Work

In the feature, all the data preprocessing methods can be applied in this experiments to improve the results, including data augmentation, data validation, and data normalization. What's more, not only dropout but also other techniques can combine with Mix-Kernel method to achieve more competitive accuracies. Last but not least, in my experiments, all the convolution layers in the same network have the same Mix-Kernel module, but different Mix-Kernel modules can be tried in a single network. For

example, you may use 3/5 Mix-Kernel Module in first convolutional layers, 5 Pure-Kernel Module in second convolutional layers and 3/5/7 Mix-Kernel Module in third convolutional layers.

Bibliography

[1] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." arXiv preprint arXiv:1409.4842(2014).

[2] LeCun, Yann, L éon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE86, no. 11 (1998): 2278-2324.

[3] LeCun, B. Boser, John S. Denker, D. Henderson, Richard E. Howard, W. Hubbard, and Lawrence D. Jackel. "Handwritten digit recognition with a back-propagation network." In Advances in neural information processing systems. 1990.

[4] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. “deep learning.” Nature 521(2015): 436-444.

[5] DeepLearning 0.1. LISA Lab. "Convolutional Neural Networks (LeNet) - DeepLearning 0.1 documentation". http://deeplearning.net/tutorial/lenet.html

[6] Goodfellow, Ian J., Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. "Multi-digit number recognition from street view imagery using deep convolutional neural networks." arXiv preprint arXiv:1312.6082 (2013).

[7] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato and Yann LeCun. "What is the Best Multi-Stage Architecture for Object Recognition?" ICCV 2009

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[9] Zeiler, Matthew D., and Rob Fergus. "Stochastic pooling for regularization of deep convolutional neural networks." arXiv preprint arXiv:1301.3557 (2013).

[10] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15, no. 1 (2014): 1929-1958.

[11] Goodfellow, Ian J., David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Maxout networks." arXiv preprint arXiv:1302.4389 (2013).

[12] Pinheiro, Pedro HO, and Ronan Collobert. "Recurrent convolutional neural networks for scene parsing." arXiv preprint arXiv:1306.2795 (2013).

[13] Wan, Li, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. "Regularization of neural networks using dropconnect." In Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 1058-1066. 2013.

[14] Matsugu, Masakazu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. "Subject independent facial expression recognition with robust face detection using a convolutional neural network." Neural Networks 16, no. 5 (2003): 555-559.

[15] Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "CAFFE: Convolutional architecture for fast feature embedding." In Proceedings of the ACM International Conference on Multimedia, pp. 675-678. ACM, 2014.

[16] Van Dyk, David A., and Xiao-Li Meng. "The art of data augmentation." Journal of Computational and Graphical Statistics 10, no. 1 (2001).

[17] Kotsiantis, S. B., D. Kanellopoulos, and P. E. Pintelas. "Data preprocessing for supervised leaning." International Journal of Computer Science 1, no. 2 (2006): 111-117.

[18] Ciresan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 3642-3649. IEEE, 2012.

[19] LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits." (1998).

[20] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

[21] OpenDeep Website. "OpenDeep tutorial on MNIST." http://www.opendeep.org/v0.0.5/docs/tutorial-classifying-handwritten-mnist-images

[22] Lee, Chen-Yu, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. "Deeply-supervised nets." arXiv preprint arXiv:1409.5185 (2014).

[23] Min Lin1, Qiang Chen and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).

[24] Chang, Jia-Ren, and Yong-Sheng Chen. "Batch-normalized Maxout Network in Network." arXiv preprint arXiv:1511.02583 (2015).

[25] Krizhevskey, A. "Cuda-convnet." (2014).

[26] Donahue, Jeff, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition." arXiv preprint arXiv:1310.1531 (2013).

[27] Sermanet, Pierre, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).

[28] Goodfellow, Ian J., David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. "Pylearn2: a machine learning research library." arXiv preprint arXiv:1308.4214 (2013).

[29] Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning." In BigLearn, NIPS Workshop, no. EPFL-CONF-192376. 2011.

[30] Google Inc. "Google Protocol Buffers." https://developers.google.com/protocol-buffers/

[31] Deng, Li, Ossama Abdel-Hamid, and Dong Yu. "A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

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