ABSTRACT

CHARACTERISTICS OF DIFFERENT DEEP NEURAL NETWORKS AND APPLICATION OF PRE-TRAINED MODEL WITHOUT TRANSFER LEARNING

by
Zhiqi Peng

Deep neural networks have been successful in many areas, some of them even surpass human performances. The goal of this thesis is using data simulations to present different characteristics of three deep neural networks: fully connected deep neural network, convolutional neural network, recurrent neural network, which will perform best when dealing with different feature patterns. By using these characteristics to design a deep neural network on top of an adopted pre-trained model with untrainable layers, achieved an averagely 11.1% improvement than a model with transfer learning method.

CHARACTERISTICS OF DIFFERENT DEEP NEURAL NETWORKS AND APPLICATION OF PRE-TRAINED MODEL WITHOUT TRANSFER LEARNING

by
Zhiqi Peng

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for Degree of
Master in Science in Computer Science

Department of Computer Science

December 2017

APPROVAL PAGE

CHARACTERISTICS OF DIFFERENT DEEP NEURAL NETWORKS AND APPLICATION OF PRE-TRAINED MODEL WITHOUT TRANSFER LEARNING

Zhiqi Peng

Dr. Zhi Wei, Thesis Advisor Date
Associate Professor of Department of Computer Science, NJIT

Dr. Usman Roshan, Committee Member Date
Associate Professor of Department of Computer Science, NJIT

Dr. Hai Phan, Committee Member Date
Associate Professor of Department of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Zhiqi Peng
Degree:	Master of Science
Date:	December 2017

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2017

- Bachelor of Science in Insurance,
  University of International Business and Economics, Beijing, P.R. China 2013

AKNOWLEDGEMENT

I would specifically like to thank Prof. Zhi Wei for supervising and advising my research on this topic and providing resources for my thesis.

Thanks to Prof. Usman W. Roshan and Prof. Hai Phan for being my Committee members.

Additionally, I thank Mr. Jie Zhang and Mr. Fei Tan for providing support during this research, and Miss Zi Wang for assisting with the chart drawing.

Lastly, I thank my family and friends for their love and support.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES

(Continue)

CHAPTER 1

BACKGROUND

1.1 Neuron

To understand how neural network behaves and deliver information, we should start with the simplest possible neural network: a single neuron (see Figure 1.1). Let's consider a data sample that has an input vector $x = (x_1, x_2, x_3, x_4)$, the output of neural is $o(x)$, we use the following equation to describe the relationship between input and output:

$$o(x) = W^T x + b \tag{1.1}$$

Where W is the weight matrix of neuron and b is the bias of neuron. After that, normally an activation function f will apply to the output to keep the non-linearity. The capability of one neural network to approximate any functions, hidden features, models is directly related to such non-linear transformation, otherwise, there is no difference between a neural network and linear regression model. Thus, the output of one neuron will become:

$$o(x) = f(W^T x + b) \tag{1.2}$$

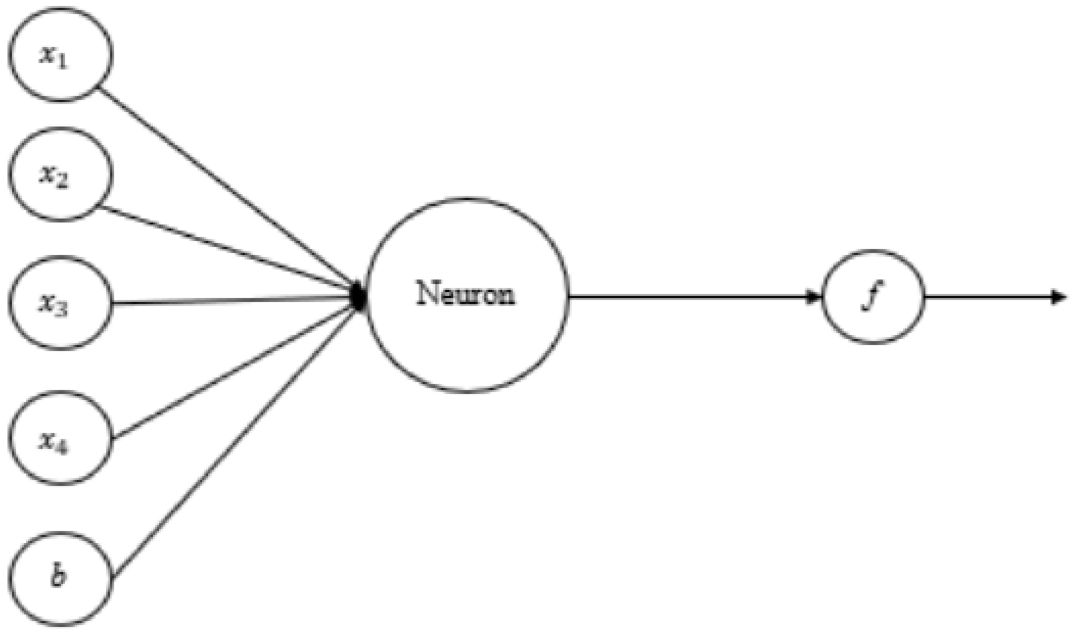


Figure 1.1 An example of one neuron architecture.

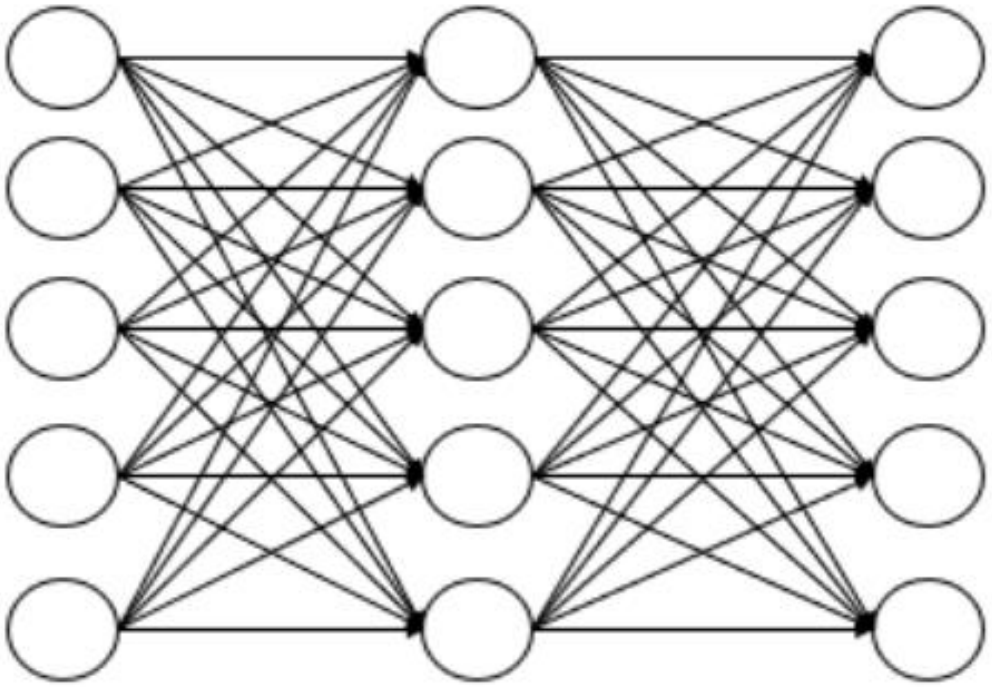


Figure 1.2 An example of fully connected neural network.

1.2 Activation Function

As we just discussed, activation function servers vital character in the neural network. There are many activation functions we use, but we will describe only some of them.

Sigmoid

The Sigmoid activation function transforms input into (0-1) interval. It’s commonly used for binary classification problem and the last activation for the model. We use the following equation to describe the Sigmoid activation function:

$$sig(x) = \frac{1}{1+exp^{-x}} \tag{1.3}$$

Hyperbolic Tangent

We commonly use *Tanh* to denote Hyperbolic Tangent activation function. The tanh function transform input into (-1,1) interval. We use the following equation to describe the *Tanh* activation function:

$$tanh(x) = \frac{exp^{x}-exp^{-x}}{exp^{x}+exp^{-x}} \tag{1.4}$$

Rectifier

The Rectifier activation function or *Relu* solved vanishing gradient and accelerate the backpropagation process by providing simple gradient derivation form. We use the following equation to describe the *Relu* activation function:

$$relu(x) = \max(0, x) \tag{1.5}$$

However, *Relu* function suffers from dying relu problem, which caused by no gradient flowing backward through the network when outputs within layer are all zero. We can use *Leaky Relu* activation function to mitigate such state. We use the following equation to describe the *Leaky Relu* activation function:

$$leakyrelu(x, alpha) = \begin{cases} alpha * x, \ x < 0 \\ x, \ x \geq 0 \end{cases} \tag{1.6}$$

1.3 Fully Connected Neural Network

This is the most common form of a neural network. Within such network, neurons between two adjacent layers are one by one densely connected. Figure 1.2 is an example layout of fully connected neural network.

1.4 Convolutional Neural Network

If information or features of data are connected adjacently, like an image or sequence, which suggests neural network's hidden units don't have to look all parts of data, instead, features can be learned by only look at part of the data which result in saving computational resources. A convolutional layer has *N* filters $F = \{f_1, f_2, f_3, \dots, f_N\}$, for each filter it will apply elementwise dot calculation to input feature map *x* then form into new a feature presentation map.

1.5 Recurrent Neural Network

Traditionally, we assume all inputs are independent of each other or not adjacent features are independent. But for some tasks, like voice recognition, language translation, sequence prediction, this assumption may not be valid at all. Thus, the recurrent neural network takes advantage of its internal memory mechanism to memorize arbitrary information for prediction purpose.

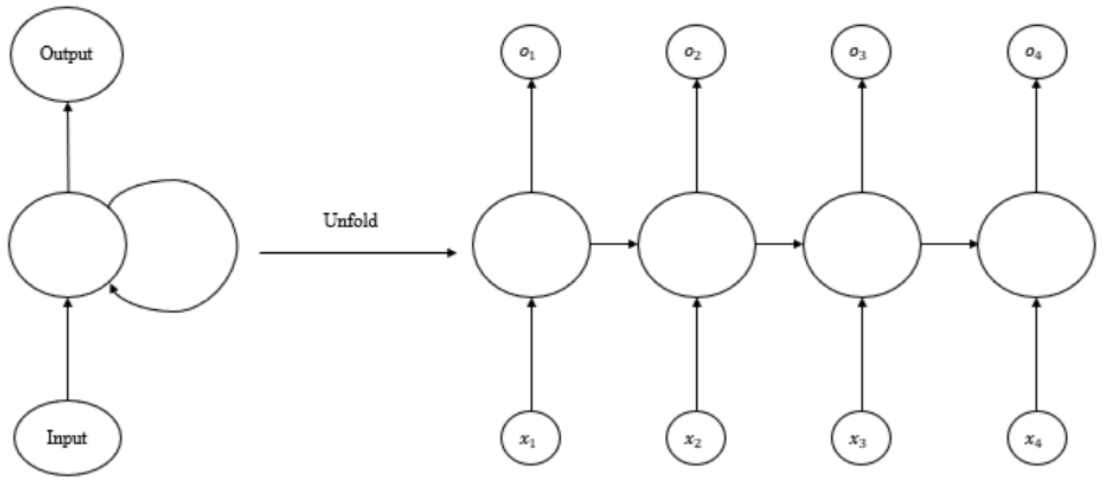


Figure 1.3 An example of recurrent neural network.

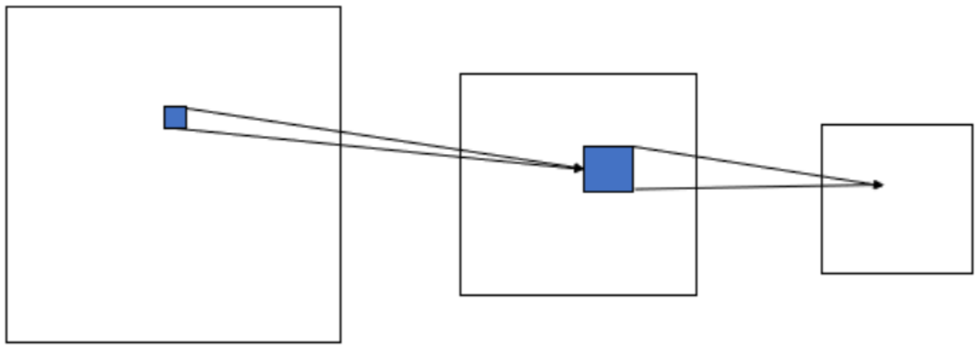


Figure 1.4 An example of convolutional neural network.

1.6 Residual Learning

Because very deep neural networks are very difficult to train but are essential for large dataset such ImageNet, Kaiming He [3] proposed Deep Residual Learning block for image recognition that achieves high improvement of accuracy on image classification tasks and has the ability of build very deep convolutional neural network architecture. We use the following mathematical formula to express residual learning building block:

$$o(x) = f(W^T x + b) + x \tag{1.7}$$

1.7 Dropout

A large number of parameters in the deep neural network makes it powerful to approximate any functions, but sometimes it can result in severe overfitting problem. Nitish Srivastava [16] proposed dropout mechanism to address such problem. This mechanism has already presented its ability to achieve top rank performance in many image classification tasks, such as Drop-connect [5] block. The idea is to randomly set output of the last layer to zero to prevent units from co-adapting too much during the training process and will disable during validation and testing process.

1.8 Optimization Function

Stochastic Gradient Descent

Stochastic Gradient Descent is also known as SGD, is the essential optimization algorithm used in deep learning models. If we use L(θ) refers as loss function, we can use following mathematical formula to express SGD:

$$\theta = \theta - \alpha \nabla_\theta L(\theta, x, y) \tag{1.8}$$

Where α is the learning rate of SGD algorithm.

Sometimes standard SGD can have a slow converging speed or stuck at local minima. We can use Momentum mechanism to alleviate such situation. We use m to denote momentum vector, thus, SGD can be presented as follows:

$$m = \gamma m + \alpha \nabla_\theta L(\theta, x, y) \tag{1.9}$$

$$\theta = \theta - m \tag{1.10}$$

Where γ is the momentum factor.

Adam

The Adam algorithm was proposed by Diederik P. Kingma [13]. It's an algorithm based on first-order gradient-based optimization function. Adam is capable of adaptive lower-order momentum and has combined advantages of AdaGrad and RMSProp. Thus, Adam can address sparse gradients and to deal with non-stationary objectives.

1.9 Batch Normalization

Different layers in the deep neural network may have a different distribution of inputs, this may slow down the training process due to vanishing gradient problem. Sergey Ioffe [6] address this problem by proposing batch normalization mechanism. The main idea is to shifting inputs of each layer to zero mean and unit variance.

CHAPTER 2

SIMULATION

My experiments intended to illustrate different characteristics of three different kinds of neural networks (fully connected neural network, convolutional neural network, recurrent neural network [7] [8] [9] [10]), and which network achieves the best performance under different scenarios. We designed three different simulations and with each, we repeat our experiment five times to eliminate affection of randomness of different initialization and split of the datasets. Our simulation data were in 2D shape, each sample with the shape of (step, features). We specifically use DNA type of data and encoded as [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1] corresponding to A(adenine), T(thymine), C(cytosine), G(guanine). We use “fullyconnected” to represent fully connected neural network, “cnn" to represent the convolutional neural network, "rnn" to represent the recurrent neural network. For each neural network, "_i” stands for the ith architecture of one model, for instance, "cnn_1" means the first model for the convolutional neural network. Unless specified in neural network’s detail and the last layer of each network, the default activation function for each layer is Relu. We save the best model based on loss value of validation dataset with the patience of 100 epochs.

2.1 Convolutional Neural Network

The sequence step length for this experiment is 50, thus dimension for each individual sample is 50×4. Designed 4 motif patterns which are [A, A, A, A, A], [T, T, T, T, T], [C, C, C, C, C], [G, G, G, G, G], allowed up to 80% mutation for each motif, which means for each motif, such as [A, A, A, A, A], only 1 step mutation is allowed. And randomly insert these patterns into an individual sequence from step 0 to 40 with no overlapping. If each motif pattern occurred only once in a sequence labeled as positive, otherwise labeled as negative. We generated 10 thousand positive and 10 thousand negative samples. The deep neural network must first recognize what are the four motifs and then learn to identify if each motif occurred once or not. This pattern is as identical as finding low-level feature then combined as a high-level feature.

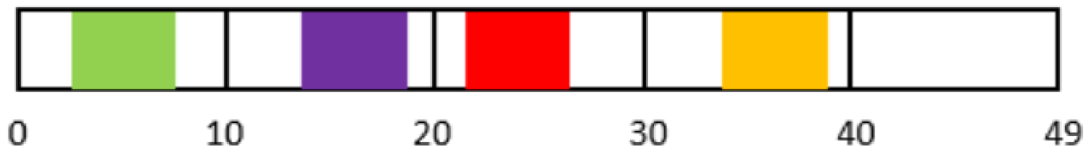


Figure 2.1 A positive sample motifs illustration, each colored rectangular represents one different motif with step length 5, and is randomly inserted into sequence with no overlapping.

2.1.1 First Experiment

We use different layouts of the deep neural network as presented in Table 2.1 in the first experiment. We applied three different dataset sizes (5000, 10000, 20000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.2 and Table 2.2. The convolutional neural network takes first place in all conditions with less trainable parameters.

Table 2.1 Architecture and Hyperparameters of each Neural Network for the First Experiment

Neural Network	Number of Parameters	Architecture Details

cnn_1	763	Convolution1D(6,6)
		LeakyReLU
		MaxPooling1D(10)
		Convolution1D(12,2)
		Convolution1D(12,3)
		Flatten
		Fullyconnected(1)

fullyconnected_1	835	Flatten
		Fullyconnected(4)
		Fullyconnected(5)
		Fullyconnected(1)

rnn_1	887	LSTM(12)
		Fullyconnected(5)
		Fullyconnected(1)

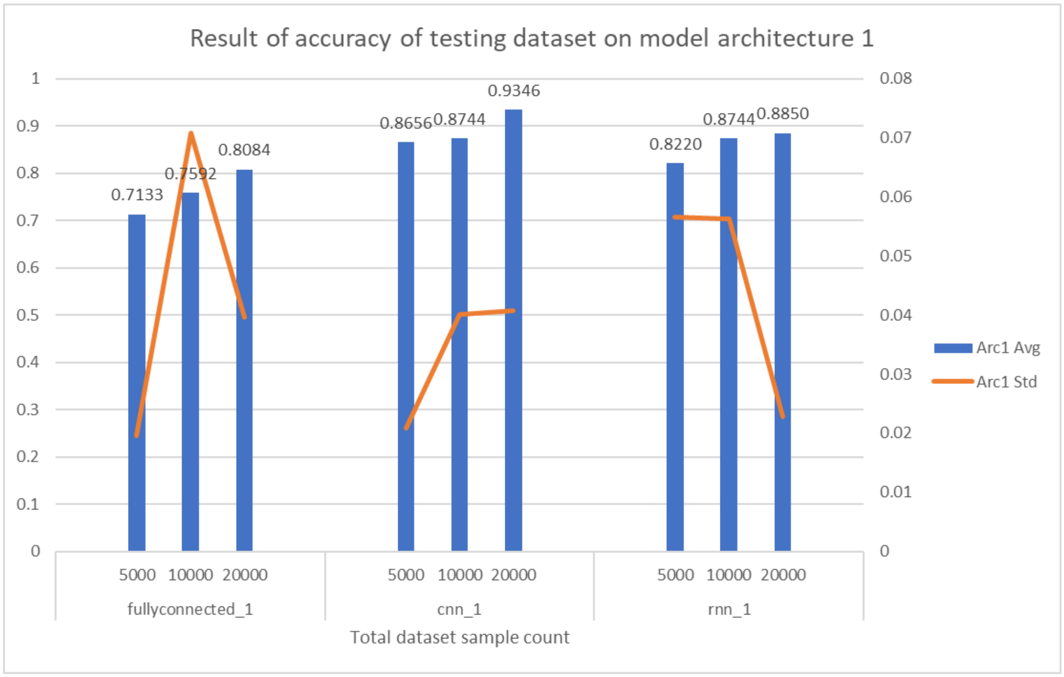


Figure 2.2 The model accuracy results of the first experiment.

Table 2.2 Detailed Results of the First Experiment

Neural Network	Dataset Sample Count	Average Accuracy on Testing Dataset	Sample Standard Deviation of Accuracy
fullyconnected_1	5000	0.7133	0.0196
	10000	0.7592	0.0708
	20000	0.8084	0.0397
cnn_1	5000	0.8656	0.0209
	10000	0.8744	0.0402
	20000	0.9346	0.0408
rnn_1	5000	0.8220	0.0566
	10000	0.8744	0.0563
	20000	0.8850	0.0228

2.1.2 Second Experiment

We use different layouts of the deep neural network as presented in Table 2.3 in the second experiment. We applied three different dataset sizes (10000, 15000, 20000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.3 and Table 2.4. The convolutional neural network takes first place in all conditions with less trainable parameters.

Table 2.3 Architecture and Hyperparameters of each Neural Network for the Second Experiment

Neural Network	Number of Parameters	Architecture Details

cnn_2	1512	Convolution1D(16,6)
		LeakyReLU
		MaxPooling1D(10)
		Convolution1D(12,2)
		Convolution1D(15,3)
		Flatten
		Fullyconnected(5)
		Fullyconnected(1)

fullyconnected_2	3341	Flatten
		Fullyconnected(15)
		Fullyconnected(15)
		Fullyconnected(5)
		Fullyconnected(1)

rnn_2	2915	LSTM(24)
		LeakyReLU
		Fullyconnected(5)
		LeakyReLU
		Fullyconnected(1)

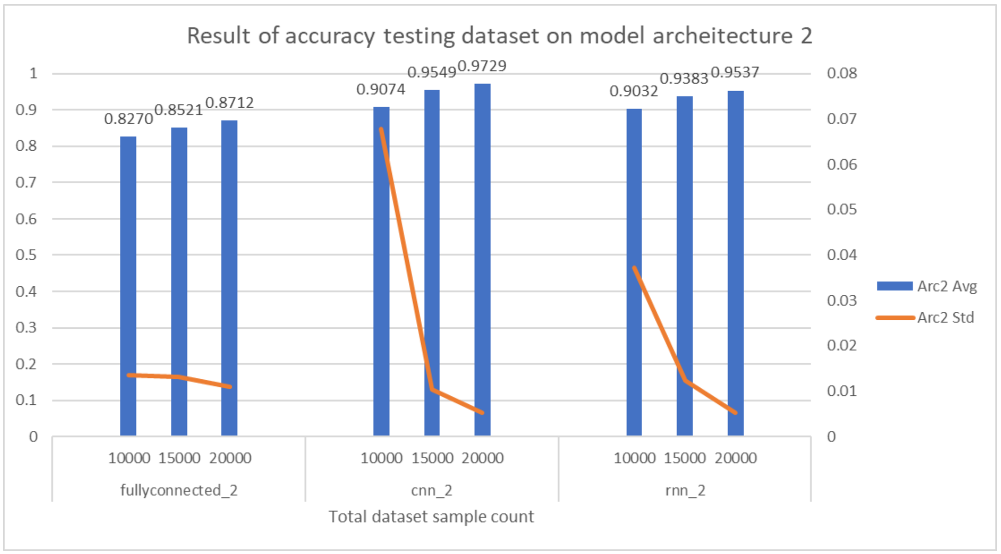


Figure 2.3 The model accuracy results of the second experiment.

Table 2.4 Detailed Results of the Second Experiment

Neural Network	**Dataset Sample Count**	**Average Accuracy on Testing Dataset**	**Sample Standard Deviation of Accuracy**
fullyconnected_2	10000	0.8270	0.0135
	15000	0.8521	0.0132
	20000	0.8712	0.0109
cnn_2	10000	0.9074	0.0678
	15000	0.9549	0.0104
	20000	0.9729	0.0052
rnn_2	10000	0.9032	0.0372
	15000	0.9383	0.0124
	20000	0.9537	0.0053

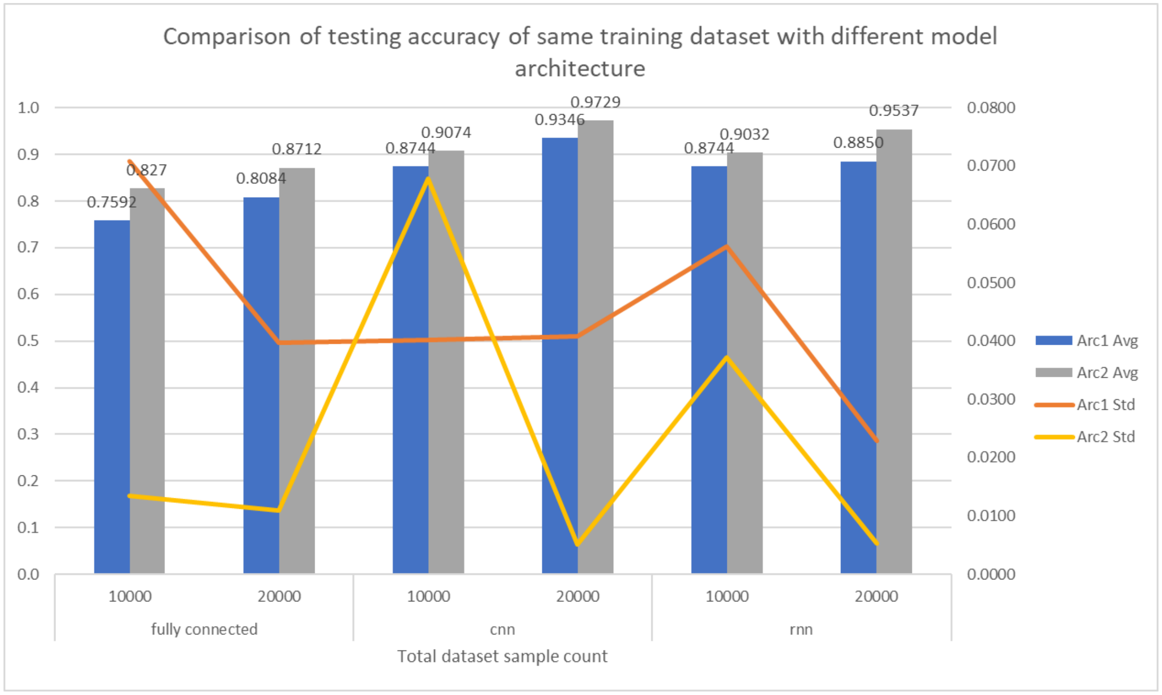


Figure 2.4 Comparison of testing accuracy with different model complexities under the same dataset.

2.1.3 Summary

As we summarized in Tables 2.2 and 2.4, during this part of the experiment, convolutional neural network exhibits high efficiency and high accuracy in identifying motif patterns. In the first experiment, cnn_1 using 763 parameters to achieve averagely 0.9346 accuracies compared to fullyconnected_1: 0.8084 with 835 parameters and rnn_1: 0.8850 with 887 parameters when using 20K samples. In the second experiment, cnn_2 using 1512 parameters to achieve averagely 0.9729 accuracies compared to fullyconnected_2: 0.8712 with 3341 parameters and rnn_2: 0.9537 with 2915 parameters when using 20K samples.

When comparing different model complexities under the same dataset as in Figure 2.4, not only average accuracy has increased in all three neural networks, but standard deviation also decreased as compared to Tables 2.2 and 2.4. Under the 20K sample size, fullyconnected_2 decreased standard deviation from 0.0397 of fullyconnected_1 to 0.0109, cnn_2 decreased standard deviation from 0.0408 of cnn_1 to 0.0052, rnn_2 decreased standard deviation from 0.0228 of rnn_1 to 0.0053.

2.2 Fully Connected Neural Network

The sequence step length for this experiment is 50, thus dimension for each individual sample is 50x4. The pattern is 5 [A]s for [0, 10, 20, 30, 40] step in each sample. We generated 10 thousand positive and 10 thousand negative samples. This pattern requires deep neural network not only able to capture what motifs are but also identify what position is.

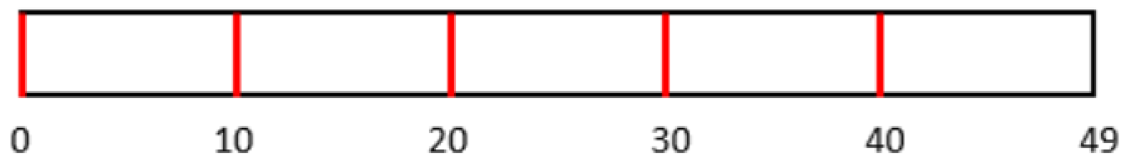


Figure 2.5 A positive sample motifs illustration, each colored line represents one different motif with step length one, and is inserted into sequence at steps: 0,10,20,30,40.

2.2.1 First Experiment

We use different layouts of the deep neural network as presented in Table 2.5 in the first experiment. We applied three different dataset sizes (200, 2000, 10000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.6 and Table 2.6. The fully connected network takes first place in all conditions with less trainable parameters.

Table 2.5 Architecture and Hyperparameters of each Neural Network for the First Experiment

Neural Network	**Number of Parameters**	**Architecture Details**
fullyconnected_1	203	Flatten Fullyconnected(1) Fullyconnected(1)

cnn_1	311	Convolution1D(10,2)
		Convolution1D(10,2)
		GlobalMaxPooling1D
		Fullyconnected(1)

rnn_1	206	LSTM(5)
		Fullyconnected(1)

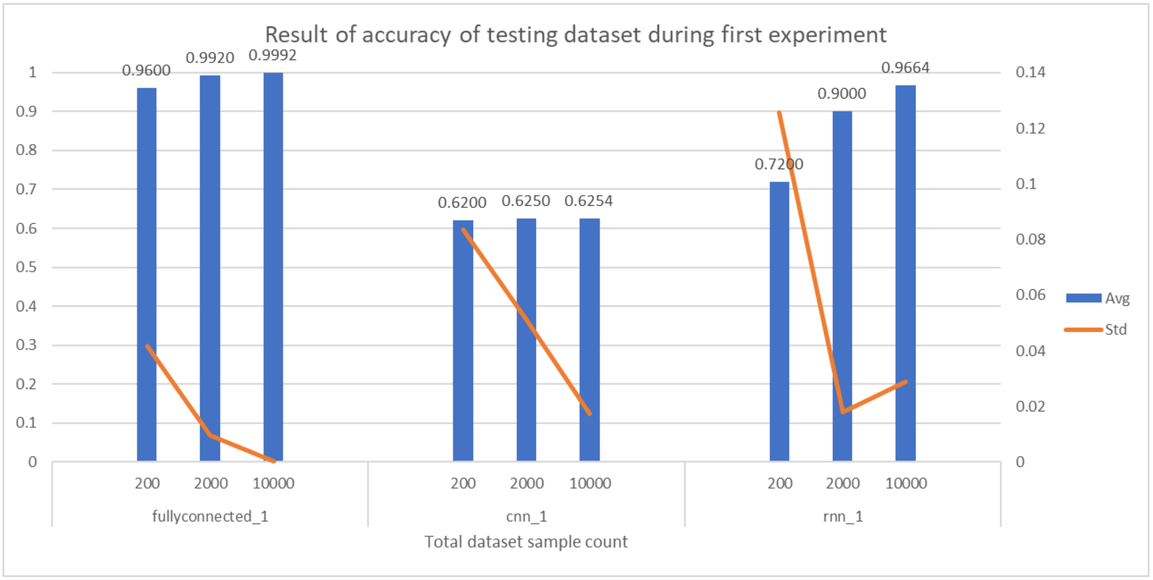


Figure 2.6 The model accuracy results of the first experiment.

Table 2.6 Detailed Results of the First Experiment

Neural Network	Dataset Sample Count	Average Accuracy on Testing Dataset	Sample Standard Deviation of Accuracy
fullyconnected_1	200	0.9600	0.0418
	2000	0.9920	0.0097
	10000	0.9992	0.0003
cnn_1	200	0.6200	0.0837
	2000	0.6250	0.0515
	10000	0.6254	0.0175
rnn_1	200	0.7200	0.1255
	2000	0.9000	0.0180
	10000	0.9664	0.0289

2.2.2 Second Experiment

We use different layouts of the deep neural network as presented in Table 2.7 in the second experiment. Since fullyconnected_1 performs exceptionally in the first experiment, there is no need to increase the model complexity of the fully connected network, we simply apply same architecture during the second experiment. We applied three different dataset sizes (2000, 10000, 20000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.7 and Table 2.8. The fully connected neural network still takes first place in all conditions with less trainable parameters.

Table 2.7 Architecture and Hyperparameters of each Neural Network for the Second Experiment

Neural Network	**Number of Parameters**	**Architecture Details**
fullyconnected_1	203	Flatten Fullyconnected(1) Fullyconnected(1)

cnn_2	1021	Convolution1D(20,2)
		Convolution1D(20,2)
		GlobalMaxPooling1D
		Fullyconnected(1)

rnn_2	611	LSTM(10)
		Fullyconnected(1)

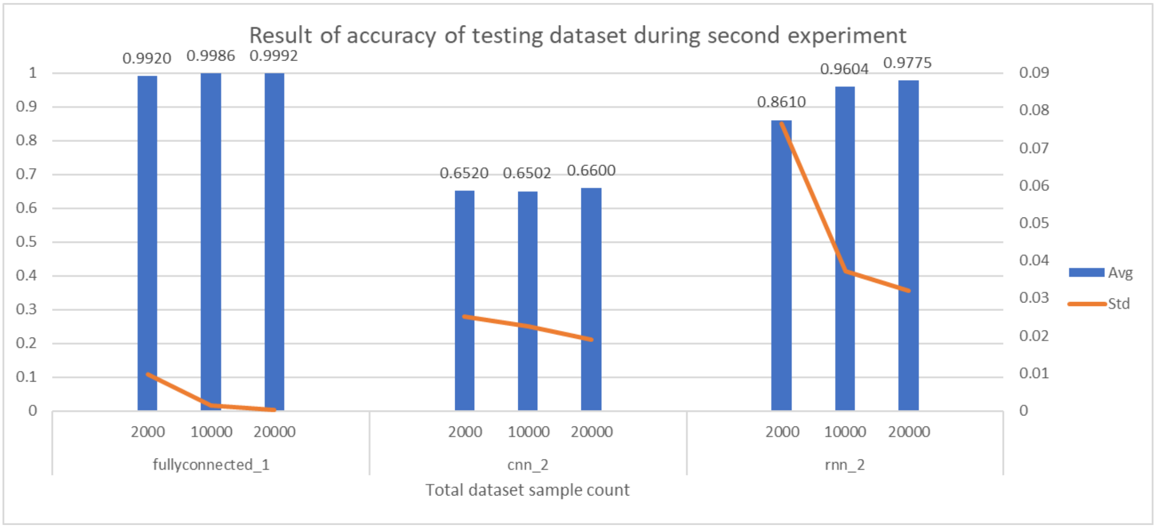


Figure 2.7 The model accuracy results of the second experiment.

Table 2.8 Detailed Results of the Second Experiment

Neural Network	**Dataset Sample Count**	**Average Accuracy on Testing Dataset**	**Sample Standard Deviation of Accuracy**
fullyconnected_1	2000	0.9920	0.0097
	10000	0.9986	0.0015
	20000	0.9992	0.0003
cnn_2	2000	0.6520	0.0251
	10000	0.6502	0.0226
	20000	0.6600	0.0189
rnn_2	2000	0.8610	0.0766
	10000	0.9604	0.0373
	20000	0.9775	0.0320

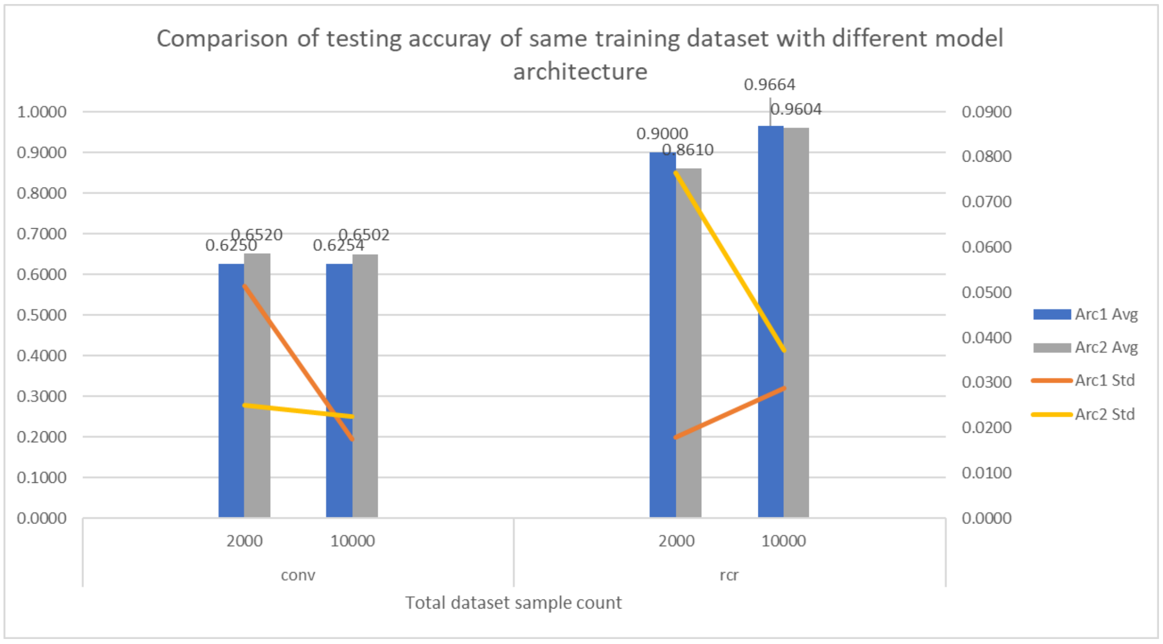


Figure 2.8 Comparison of testing accuracy with different model complexities under the same dataset.

2.2.3 Summary

As we summarized Table 2.6 and Table 2.8, during this part of the experiment, fully connected neural network exhibits high efficiency and high accuracy in identifying motif patterns. In the first experiment, fullyconnected_1 using 203 parameters to achieve averagely 0.9992 accuracies compared to cnn_1: 0.0.6254 with 311 parameters and rnn_1: 0.9664 with 206 parameters when using 10K samples. In the second experiment, fullyconnected_1 achieve averagely 0.9992 accuracies compared to cnn_2: 0.6600 with 1021 parameters and rnn_2: 0.9775 with 611 parameters when using 20K samples.

2.3 Recurrent Neural Network

The sequence step length for this experiment is 50, thus dimension for each individual sample is 50x4. The pattern is [G, A, G, T, C, C, T, A, G, C] with a total of 10 step features, and randomly inserted into sample's 50 steps with the preserved order. If the sample does not contain such sequence, labeled as negative, otherwise labeled as positive. We generated 10 thousand positive and 10 thousand negative samples. Identifying this pattern requires the deep neural network capable to memorize occurrence of motif sequence.

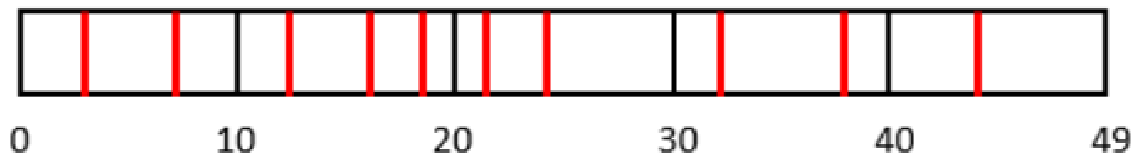


Figure 2.9 A positive sample motifs illustration, each colored line represents one different motif with step length one, and is inserted into sequence with random intervals.

2.3.1 First Experiment

We use different layouts of the deep neural network for as presented in Table 2.9 in the first experiment. We applied three different dataset sizes (5000, 10000, 20000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.10 and Table 2.10. The recurrent neural network takes first place in all conditions with less trainable parameters.

Table 2.9 Architecture and Hyperparameters of each Neural Network for the First Experiment

Neural Network	Number of Parameters	Architecture Details

rnn_1	206	LSTM(5)
		Fullyconnected(1)

cnn_1	651	Convolution1D(10,3)
		Convolution1D(10,5)
		GlobalMaxPooling1D
		Fullyconnected(1)

fullyconnected_1	629	Flatten
		Fullyconnected(3)
		Fullyconnected(5)
		Fullyconnected(1)

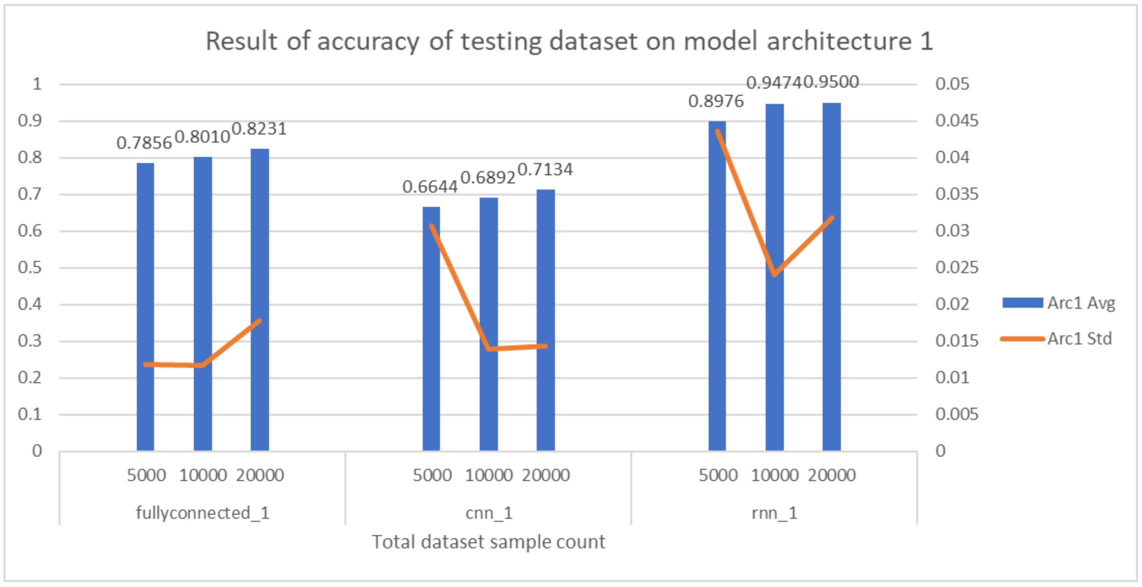


Figure 2.10 The model accuracy results of the first experiment.

Table 2.10 Detailed Results of the First Experiment

Neural Network	**Dataset Sample Count**	**Average Accuracy on Testing Dataset**	**Sample Standard Deviation of Accuracy**
fullyconnected_1	5000	0.7856	0.0118
	10000	0.8010	0.0117
	20000	0.8231	0.0178
cnn_1	5000	0.6644	0.0308
	10000	0.6892	0.0139
	20000	0.7134	0.0143
rnn_1	5000	0.8976	0.0436
	10000	0.9474	0.0240
	20000	0.9500	0.0318

2.3.2 Second Experiment

We use different layouts of the deep neural network for as presented in Table 2.11 in the second experiment. We applied three different dataset sizes (10000, 15000, 20000) to each model, and take the average and standard deviation of the testing dataset, the result is showed in Figure 2.11 and Table 2.12. The recurrent neural network takes first place in all conditions with less trainable parameters.

Table 2.11 Architecture and Hyperparameters of each Neural Network for the Second Experiment

Neural Network	Number of Parameters	Architecture Details

rnn_2	611	LSTM(10)
		Fullyconnected(1)

cnn_2	2301	Convolution1D(20,3)
		Convolution1D(20,5)
		GlobalMaxPooling1D
		Fullyconnected(1)

fullyconnected_2	2131	Flatten
		Fullyconnected(10)
		Fullyconnected(10)
		Fullyconnected(1)

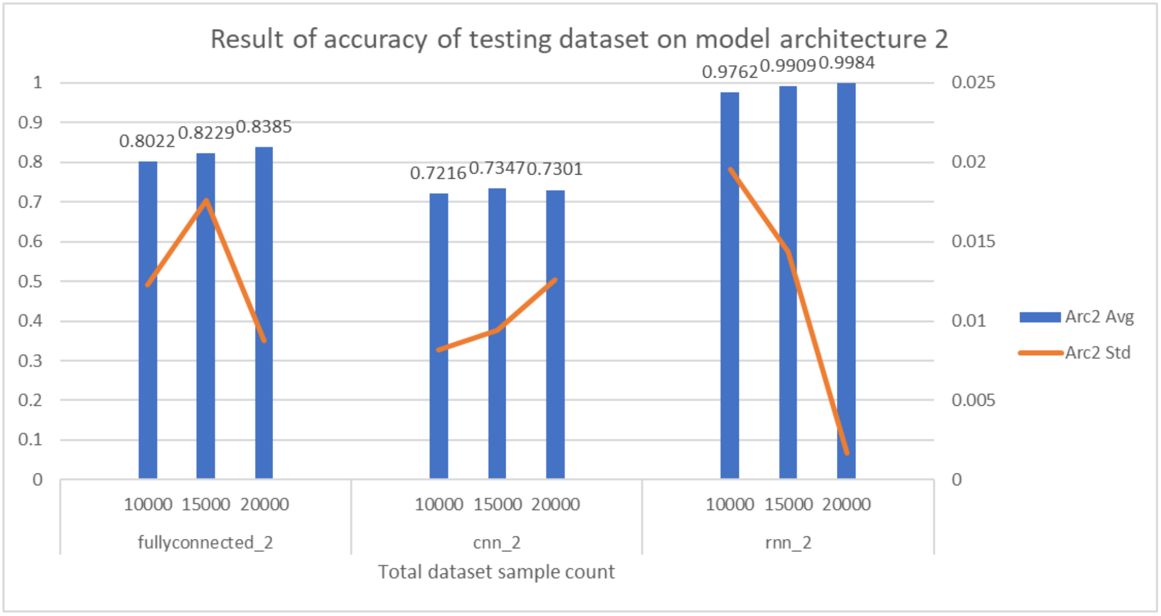


Figure 2.11 The model accuracy results of second experiment.

Table 2.12 Detailed Results of the Second Experiment

Neural Network	Dataset Sample Count	Average Accuracy on Testing Dataset	Sample Standard Deviation of Accuracy
fullyconnected_2	10000	0.8022	0.0123
	15000	0.8229	0.0176
	20000	0.8385	0.0087
cnn_2	10000	0.7216	0.0082
	15000	0.7374	0.0094
	20000	0.7301	0.0126
rnn_2	10000	0.9762	0.0195
	15000	0.9909	0.0143
	20000	0.9984	0.0017

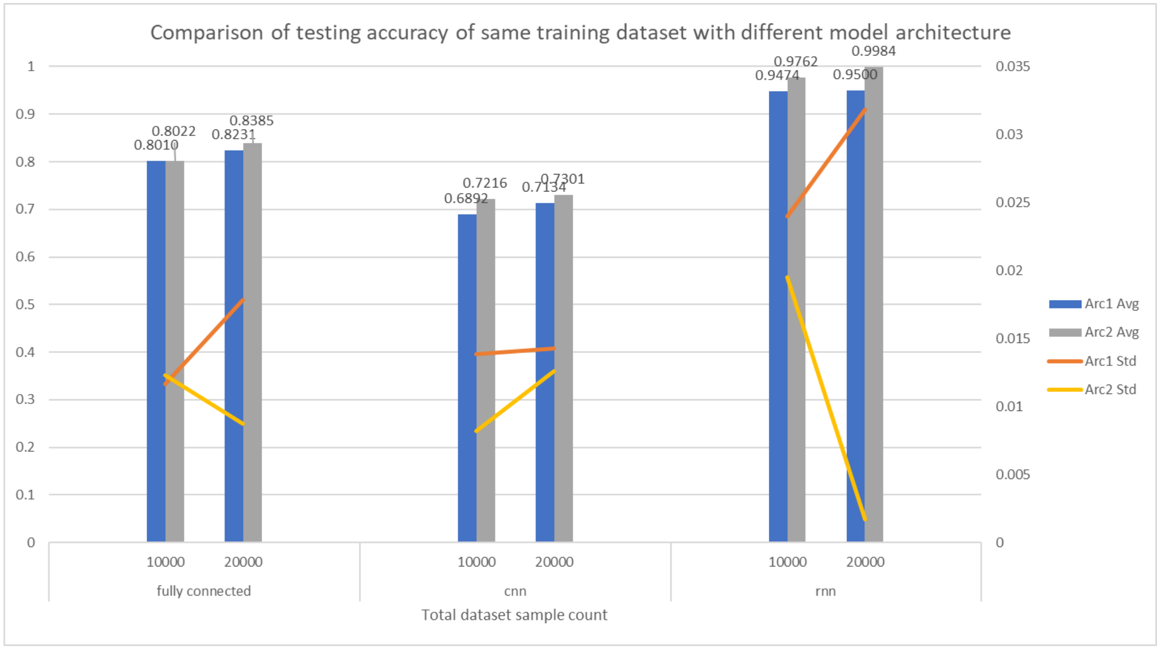


Figure 2.12 Comparison of testing accuracy with different model complexities under the same dataset.

2.3.3 Summary

As we summarized in Tables 2.10 and 2.12, during this part of the experiment, recurrent neural network exhibits high efficiency and high accuracy in identifying motif patterns. In the first experiment, rnn_1 using 206 parameters to achieve averagely 0.9500 accuracies compared to fullyconnected_1: 0.8231 with 651 parameters and cnn_1: 0.7134 with 629 parameters when using 20K samples. In the second experiment, rnn_2 using 611 parameters to achieve averagely 0.9984 accuracies compared to fullyconnected_2: 0.8385 with 2301 parameters and cnn_2: 0.7301 with 2131 parameters when using 20K samples.

When comparing different model complexity under the same dataset as in Figure 2.12, not only average accuracy has increased in all 3 neural networks, but standard deviation also decreased when comparing Table 2.10 and Table 2.12. Under 20K sample size, fullyconnected_2 decreased standard deviation from 0.0178 of fullyconnected_1 to 0.0087, cnn_2 decreased standard deviation from 0.0143 of cnn_1 to 0.0126, rnn_2 decreased standard deviation from 0.0318 of rnn_1 to 0.0017.

2.4 Kernel Size Affects Performance of Convolutional Neural Network

We want to determine what influences of kernel size may have when changing its kernel size. We use the same dataset in the previous experiment which convolutional neural network performs best. The dataset has four motifs each with a 5x4 shape. If we change the kernel size of the first convolutional layer, we change the shape of lowest level features that model captures. If the kernel's filter has less than five steps, we may assume all the motif information were split into each kernel. On the contrary, if the filter has more than five steps, we may deduce that each filter captures some noises.

We use same architecture of convolutional neural network except for the first layer and repeat our experiment five times for each architecture to eliminate affection of randomness of different initialization and split of the dataset. The results are shown in Table 2.13 and Figure 2.13.

Table 2.13 Hyperparameters of Convolutional Neural Network

Filter Number	**Step Length**	**Parameters**	**Average Accuracy of Testing Dataset**	**Standard Deviation**
3	8	652	0.9000	0.0120
4	5	661	0.8871	0.0465
6	5	751	0.9201	0.0348
6	6	775	0.9469	0.0395
6	8	823	0.9633	0.0237
6	10	871	0.9383	0.0336
12	3	925	0.8920	0.0328
12	6	1069	0.9685	0.0099

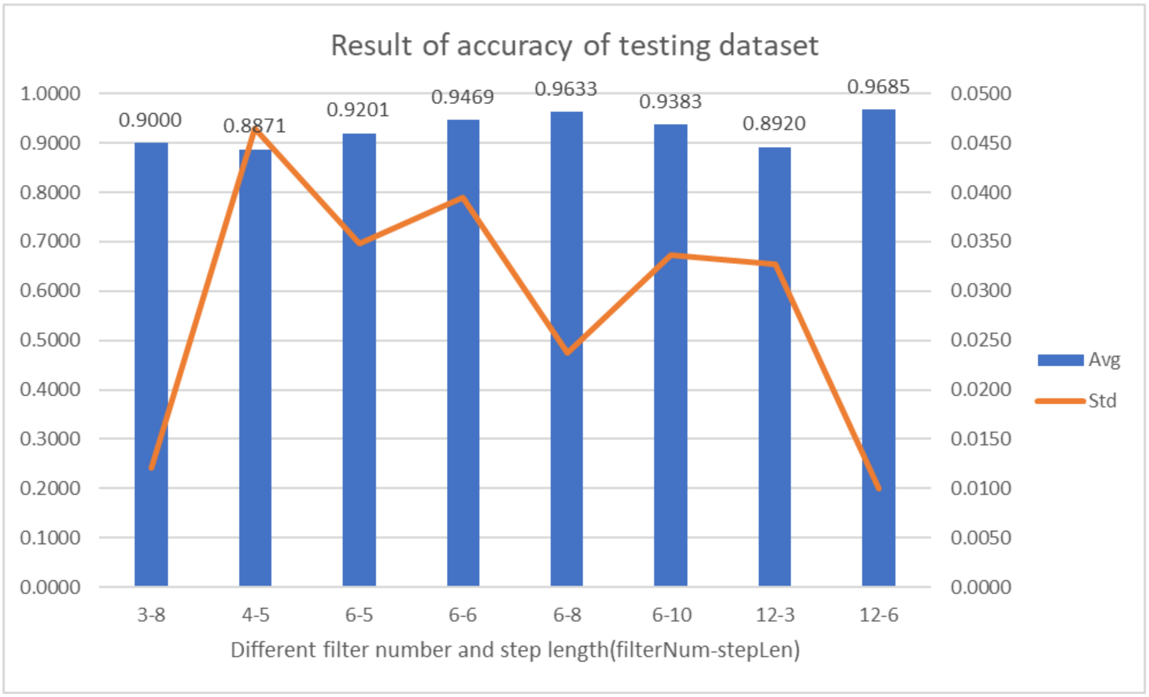


Figure 2.13 The model accuracy results of the experiment.

As seen in Figure 2.13, surprisingly the result of 4-5 (which suppose to be the optimum value), reached lowest average accuracy 0.8871 with highest standard deviation value 0.0465. Comparing results of 4-5 and 6-5, 6-6 and 12-6, indicates more filter number may result in higher accuracy with more stable performance. Comparing results of 6-6 and 12-3, indicates even with same total filter size, if filter's step length cannot cover the ground truth, will perform worse on average accuracy. Comparing results of 6-5, 6-6, 6-8 and 6-10, simply expanding single filter size will not necessarily increasing model's performance as larger filter size will contain more noise.

2.5 Further Experiment Using Position Related Pattern

As we discussed in Section 2.2, fully connected neural network performed best when the pattern is position related. However, such pattern is uncommon in the real world. But is it possible to solve such pattern using the convolutional layer to extract features followed by fully connected layer to preserve location information? We used the same dataset in Section 2.2, with a slightly differ convolutional neural network architecture as well as same fully connected neural network result. Table 2.14 describe details of each model, the second layer of cnn_fullyconnected model is actually a fully connected layer connected to a convolutional layer since the output of layer-Convolution1D(10,2) is (49,10) and output of layer-Convolution1D(1,49) is (1,1).

Table 2.14 Architecture and Hyperparameters of each Neural Network

Neural Network	**Number of Parameters**	**Architecture Details**
fullyconnected_1	203	Flatten Fullyconnected(1) Fullyconnected(1)

cnn_fullyconnected	583	Convolution1D(10,2)
		Convolution1D(1,49)
		GlobalMaxPooling1D
		Fullyconnected(1)

Table 2.15 Detailed Results of Two Model

Neural Network	**Dataset Sample Count**	**Average Accuracy on Testing Dataset**	**Sample Standard Deviation of Accuracy**
fullyconnected_1	2000	0.9920	0.0097
	10000	0.9986	0.0015
	20000	0.9992	0.0003
cnn_fullyconnected	2000	0.9925	0.0029
	10000	0.9988	0.0014
	20000	0.9991	0.0007

As the results in Table 2.15 indicate, it is possible to solve such pattern using the convolutional layer to extract features followed by fully connected layer to preserve location information. The difference between convolution neural network in Section 2.2 and network in this section is the lack of fully connected layer after convolutional layer to preserve such location information.

2.6 Discussion

For 2D data samples, if a pattern is a certain combination of lower level features, the convolutional neural network may be a better choice considering efficiency and performance. If a pattern is position related, a fully connected neural network is probably best since it preserves position information comparing with pooling layers wildly used in the convolutional neural network. If a pattern is order-related and with random steps, the recurrent neural network is the best choice. Additionally, recurrent performed averagely best in all three patterns we previously addressed, although not with the highest efficiency. With higher network's complexity and sample number, it can achieve the same level accuracy of other network architectures. The major setbacks of recurrent neural network susceptible to unstable training process and difficulty of interpreting parameters.

Filter numbers and filter size are two important hyperparameters we need to decide when designing convolutional neural network. As in Section 2.4, we should avoid setting filter size smaller than lowest level features, slightly larger filter size and more filter number should achieve a better result. However, we should balance model’s complexity and size of the dataset, as we should be considering the over-fitting scenario.

CHAPTER 3

NIH DISEASE DATASET CLASSIFICATION

NIH Clinical Center recently provided 112,120 chest x-ray scan images from more than 30,000 patients. This dataset includes many advanced lung diseases and each sample may contain multiple disease labels. The collection of diseases includes 14 categories: Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening, Hernia. Original image sample has the dimension of 1024×1024.

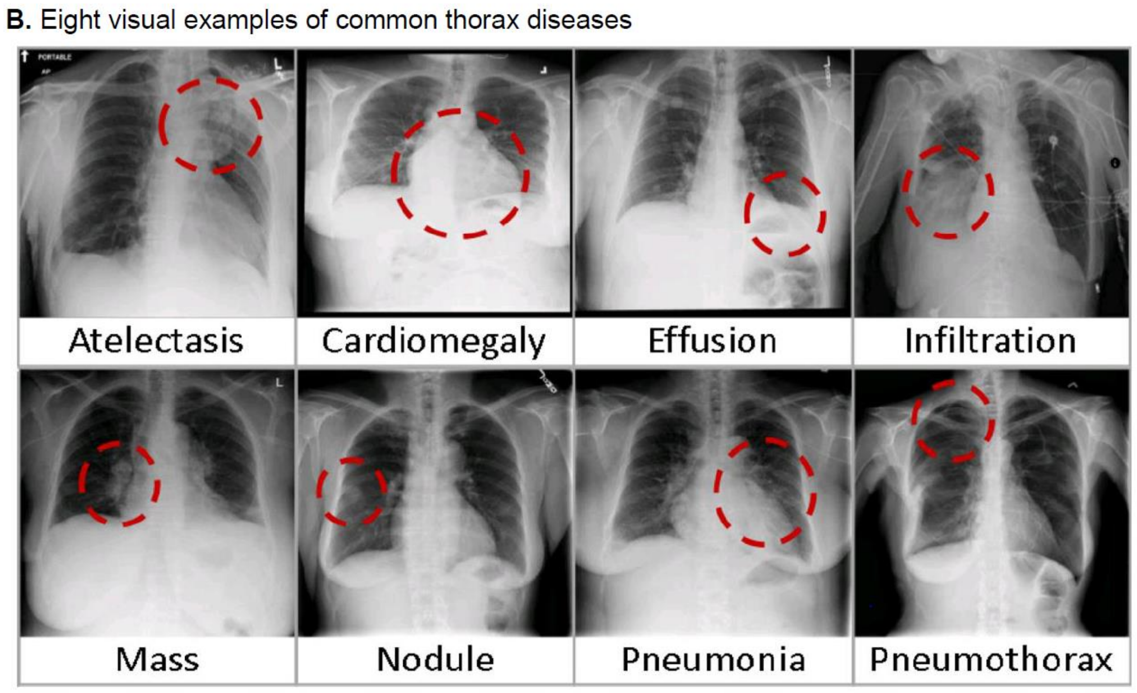
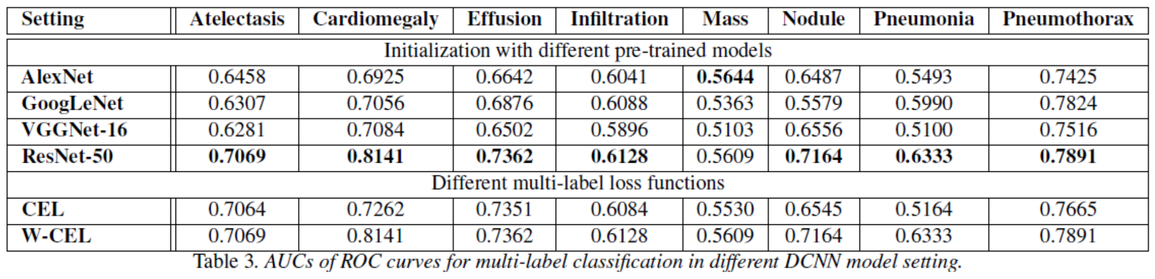


Figure 3.1 Eight visual examples of NIH diseases.

Sources: Wang, Xiaosong, et al. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases." *arXiv preprint arXiv:1705.02315* (2017).

3.1 Previous Work

Xiaosong Wang [1] performed transition training and fine tuning based on AlexNet, GoogLeNet, VGG-16, ResNet-50 of first 8 primary diseases. The detail of AUC values is in Figure 3.2.



Setting	Atelectasis	Cardiomegaly	Effusion	Infiltration	Mass	Nodule	Pneumonia	Pneumothorax
Initialization with different pre-trained models								
AlexNet	0.6458	0.6925	0.6642	0.6041	**0.5644**	0.6487	0.5493	0.7425
GoogLeNet	0.6307	0.7056	0.6876	0.6088	0.5363	0.5579	0.5990	0.7824
VGGNet-16	0.6281	0.7084	0.6502	0.5896	0.5103	0.6556	0.5100	0.7516
ResNet-50	**0.7069**	**0.8141**	**0.7362**	**0.6128**	0.5609	**0.7164**	**0.6333**	**0.7891**
Different multi-label loss functions								
CEL	0.7064	0.7262	0.7351	0.6084	0.5530	0.6545	0.5164	0.7665
W-CEL	0.7069	0.8141	0.7362	0.6128	0.5609	0.7164	0.6333	0.7891

Table 3. *AUCs of ROC curves for multi-label classification in different DCNN model setting.*

Figure 3.2 Previous AUC value of NIH disease classification result.

Source: Wang, Xiaosong, et al. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases." *arXiv preprint arXiv:1705.02315* (2017).

3.2 Preprocessing of Data

Since the memory requirement will be massive (219 GB) if we just loaded original dataset using float16 datatype, this kind of situation requires modification of dataset to minimize memory usage.

1. We observed for most of samples, the location of each disease is not located at edge of each sample, thus we perform 80% center cropping for each sample.
2. Resized each sample to 299×299 dimension.
3. Load each sample as only 1×299×299 dimension using int8 datatype
4. Balancing number of positive and negative samples for each individual disease.
5. Randomly shuffle the dataset.

3.3 Architecture

Many deep neural network models have shown the strong ability to classifying thousands of categories on ImageNet dataset. Fine tuning pre-trained model on smaller dataset has demonstrated the successful application of medical disease classification problem [1] [14]. The main goal of transfer learning method is to adapt those pre-trained features on the previous dataset to the new dataset. However, we are using pre-trained models to replace features from original dataset by freezing all layers in pre-trained models, instead of transfer learning model.

Since we load our dataset as one channel images, which is not compatible with models pre-trained on ImageNet dataset, we need to apply a channel expansion layer before we adopt pre-trained models. Then we applied pre-trained models and removed all fully-connected layers and kept the last convolution layer. We added a customized deep neural network block and output layer after pre-trained model.

Initially, we applied this dataset to both InceptionV3 [4] model and ResNet-50 [3] model, but InceptionV3 model beat by ResNet-50 model during most of the experiments. Thus, we only adopted ResNet-50 model instance pre-trained on ImageNet dataset.

Table 3.1 Hyperparameters and Output Shape of Customized Block

Description	**Hyper parameter**	**Padding**	**Output shape**
Spatial Dropping	90%	Not applicable	(2048,10,10)
c2dbn	48-1-1	Same	(48,10,10)
Convolutional Block	128	Same	(128,10,10)
Convolutional Block	128	Same	(128,10,10)
Global Max Pooling	Not applicable	Not applicable	(128,)
Dropout	50%	Not applicable	(128,)
Fully Connected	30	Not applicable	(30,)
Output Layer	1	Not applicable	(1,)

The main idea of Convolutional Block as shown in Figure 3.4 was borrowed from InceptionV3 module and the experiment we just conducted: using different kernel size trying to capture each feature with minimum noise included, and saving computational resources by letting each branch to capture part of extracted features then add them up.

3.4 Training Method

We separated 70% of data as the training dataset, 10% as validation dataset, 20% as the testing dataset. We set patience as 50 epochs and use Adam as default optimizer. Due to the massive requirement of computational resources, we freeze all layers of the pre-trained model.

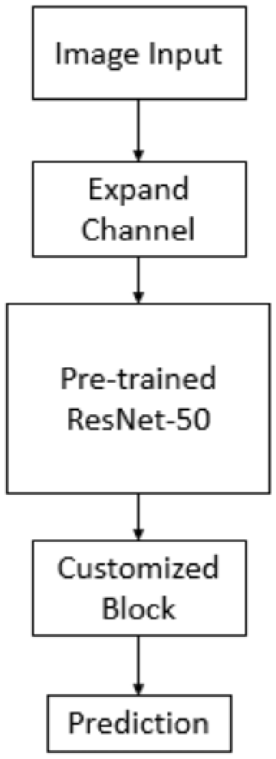


Figure 3.3 Main layout of deep learning model.

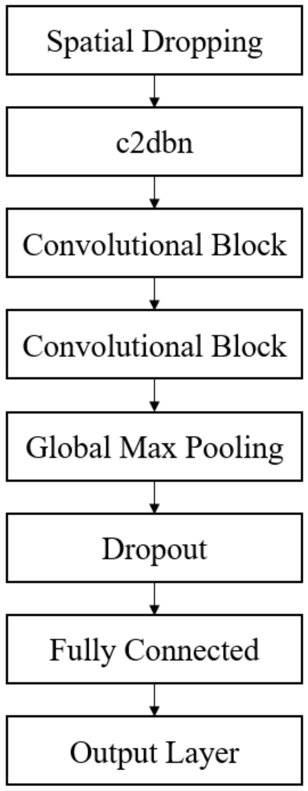


Figure 3.4 Layout of customized block.

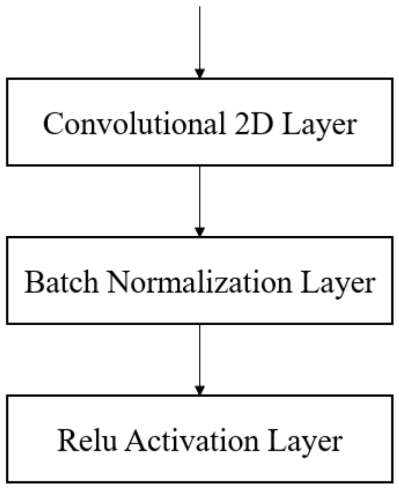


Figure 3.5 Layout of c2bn block.

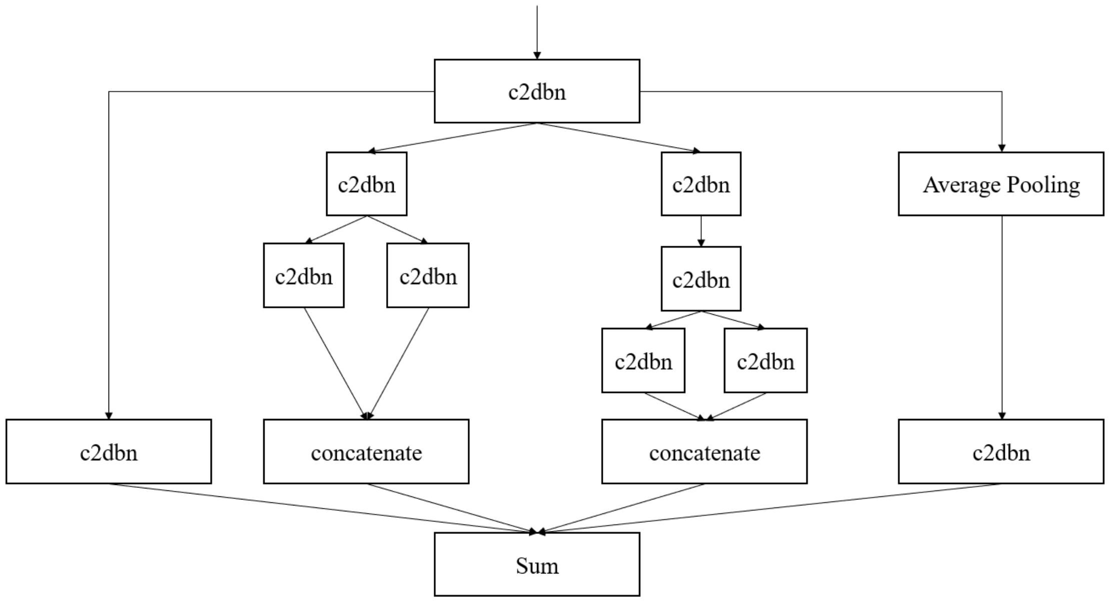


Figure 3.6 Layout of convolutional block.

We tuned the weighted-class parameter and applied multi-label classification training method during the training process, like W-CEL [1], but we cannot obtain the desired result. We had to follow the simplest rule: train one model for each disease with balanced dataset using same hyperparameters. Additionally, we applied fully-connected layer after second Convolutional Block in Figure 3.4, if extracted features from second Convolutional Block is position related, the result would be improved as in Chapter 2.5. The result is shown in the following table:

Table 3.2 Comparison of AUC Values Between Our Models and Previous Results in Figure 3.2 from Models Trained on NIH Dataset

Model	**Atelectasis**	**Cardiomegaly**	**Effusion**	**Infiltration**	**Mass**	**Nodule**	**Pneumonia**	**Pneumothorax**
Our Model	**0.7833**	**0.8721**	**0.8549**	**0.6907**	**0.7306**	0.6738	**0.7001**	**0.8503**
Add fully-connected layer	0.7665	0.8341	0.8398	0.6789	0.7247	0.6638	**0.7036**	0.8277
Wang, et.al	0.7069	0.8141	0.7362	0.6128	0.5644	**0.7164**	0.6333	0.7891

3.5 Conclusion

Due to only customized block is needed to be trained, our number of trainable parameter is 346,189, our model's total parameter count is 23,936,813, which means only 1.45% of total parameters were trained. Comparing with non-fully-connected models, there was no improvement, thus the extracted features from second Convolutional Block were not the position-related pattern. Our result is better in most of the diseases as in the previous table, average AUC score improvement is 11.1%. This indicates transfer learning of pre-trained model is not necessary if represented features from any pre-trained model are sufficient.

3.6 Discussion

Considering factors such as computational power and training time, our training method didn’t include image augmentation techniques. Random cropping, image flip, image random rotation, image normalizations etc. have shown their ability to alleviate overfitting problem and improve recognition accuracy by increasing data diversity in many image recognition tasks [2] [5]. In our model, we applied 90% of spatial dropping and 50% of fully connected dropping due to severe overfitting problem which may suggest the ImageNet pre-trained models are presenting too many features and data augmentation methods have potential to improve classification result.

REFERENCES

[1] Wang, Xiaosong, et al. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases." *arXiv preprint arXiv:1705.02315* (2017).

[2] Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

[3] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[4] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[5] Wan, Li, et al. "Regularization of neural networks using dropconnect." *Proceedings of the 30th international conference on machine learning (ICML-13)*. 2013.

[6] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International Conference on Machine Learning*. 2015.

[7] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

[8] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

[9] Graves, Alex. *Supervised sequence labelling with recurrent neural networks*. Vol. 385. Heidelberg: Springer, 2012.

[10] Gal, Yarin, and Zoubin Ghahramani. "A theoretically grounded application of dropout in recurrent neural networks." *Advances in neural information processing systems*. 2016.

[11] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks* 5.2 (1994): 157-166.

[12] Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." *Proc. ICML*. Vol. 30. No. 1. 2013.

[13] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[14] Esteva, Andre, et al. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542.7639 (2017): 115-118.

[15] Chollet, F. (2015). Keras: Deep learning for python

[16] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*15.1 (2014): 1929-1958