GCPS

ADAPTIVE NETWORK SELECTION TECHNIQUE IN CASCADED CONVOLUTIONAL NEURAL NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ATILIM UNIVERSITY

BY

EKİN SARP ÖNAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2023

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

Prof. Dr. Ender KESKİNKILIÇ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Computer Engineering Department, Atılım University**.

Assoc. Prof. Dr. Gökhan ŞENGÜL
Head of Department

This is to certify that we have read the thesis ADAPTIVE NETWORK SELECTION TECHNIQUE IN CASCADED CONVOLUTIONAL NEURAL NETWORKS submitted by EKİN SARP ÖNAL and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Erhan GÖKÇAY
Supervisor

Examining Committee Members:

Assoc. Prof. Dr. Emre SÜMER
Computer Science, Başkent University

Asst. Prof. Dr. Erhan GÖKÇAY
Software Engineering, Atılım University

Assoc. Prof. Dr. Çiğdem TURHAN
Software Engineering, Atılım University

Date: June 21, 2023

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : EKİN SARP ÖNAL

Signature :

ABSTRACT

ADAPTIVE NETWORK SELECTION TECHNIQUE IN CASCADED CONVOLUTIONAL NEURAL NETWORKS

Önal, Ekin Sarp

M.S., Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Erhan GÖKÇAY

June 2023, 52 pages

Dynamic neural network is an important research area in deep learning. The presented thesis focuses on cascaded neural network which is a sub-topic of dynamic neural network, that utilizes a router for connecting two or more neural networks with increasing depth in order to enhance the efficiency and adaptiveness of static models. In this thesis, we proposed a parameter-free technique for network selection in cascaded deep neural networks in order to reduce the computational time required for training and inference by taking advantage of the fact that shallow networks are also able to correctly classify many samples. Following a brief explanation of the cascaded neural network, softmax margin, and classical LeNet model; a novel cascaded neural network algorithm is introduced. The proposed model is compared to LeNet in terms of efficiency and performance on MNIST, EMNIST, and Fashion-MNIST datasets with various networks utilized as small modules of the cascaded model. Numerical results demonstrated that the proposed technique greatly improves the efficiency of the benchmark model without sacrificing accuracy.

Keywords: Dynamic networks, Cascaded neural networks, Convolutional neural networks, Adaptive network selection, Dynamic threshold

ÖZ

KADEMELİ EVRİŞİMLİ SİNİR AĞLARINDA UYARLANABİLİR AĞ SEÇİMİ TEKNİĞİ

Önal, Ekin Sarp

Yüksek Lisans, Bilgisayar Mühendisliği

Tez Yöneticisi : Dr. Ogr. Uyesi Erhan GÖKÇAY

Haziran 2023, 52 sayfa

Dinamik sinir ağı, derin öğrenmede önemli bir araştırma alanıdır. Sunulan tez, statik modellerin verimliliğini ve uyarlanabilirliğini artırmak için iki veya daha fazla sinir ağını artan derinlikte bağlamak için bir yönlendirici kullanan kademeli sinir ağına odaklanmaktadır. Bu tezde, kademeli derin sinir ağlarında ağ seçimi için parametresiz bir teknik önerdik. Bu teknik, sığ ağların da birçok örneği doğru bir şekilde sınıflandırabilmesi gerçeğinden yararlanarak, eğitim ve çıkarım için gereken hesaplama süresini azaltmayı amaçlamaktadır. Kademeli sinir ağı, softmax marjı ve klasik LeNet modelinin kısa bir açıklamasını takiben, yeni bir kademeli sinir ağı algoritmasının tanıtımı yapılmaktadır. Önerilen model; MNIST, EMNIST ve Fashion-MNIST veri kümelerinde etkinlik ve performans açısından LeNet ile karşılaştırılmaktadır. Sayısal sonuçlar, önerilen teknikle referans modelinin verimliliğinin büyük ölçüde arttığını ve doğruluktan ödün vermeden geliştirildiğini göstermektedir.

Anahtar Kelimeler: Dinamik sinir ağları, Kademeli sinir ağları, Evrişimsel sinir ağı, Uyarlanabilir sinir ağı seçimi, Dinamik eşik değeri

To my family.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Asst. Prof. Dr. Erhan Gökçay for his guidance and patience during this thesis. I could not have undertaken this journey without his invaluable support and feedback.

I shall also thank to my thesis committee members for kindly agreeing to review this thesis.

I am also grateful to my wonderful friends Berkan, Erdem, Doğan, Burak, and Yiğit for keeping my spirits high.

Finally, I would like to thank my family, especially my parents, brother, and sister for their support and belief in me during this process. In return, I would like to dedicate this thesis to you.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF SYMBOLS

n	:	Number of samples in a dataset
exp	:	Exponential function
$\sum_{j=1}^{n} x_j$	:	Summation of x values over all samples

CHAPTER 1

INTRODUCTION

Deep neural networks (DNN) have brought new perspectives to various areas including natural language processing (NLP) [1, 2], computer vision (CV) [3, 4, 5], data analysis, reinforcement learning (RL) and data mining. For the last few years, many successful neural network architectures have been proposed for different research areas such as LeNet [6], AlexNet [7], ResNet [8], GoogleNet [9]. Even though these deep neural network models have made it possible to train deeper networks and led to more accurate results; most of these networks are used in a static manner both in the training and prediction phases. In other words, the network structure, such as the number of layers and the number of neurons, and network parameters are kept the same after training. This causes static neural networks to have less efficiency, compatibility, adaptiveness, and representation power. However, dynamic neural networks have the ability to change network architecture or network parameters during both the training and testing phases. This leads to dynamic models having advantages over static models in the following properties:

- Efficiency: Most important advantage of dynamic networks over static networks is the ability to reduce computational complexity. Dynamic models can select different sub-networks [10] or layers [11] of a larger network in each iteration, instead of using the whole network for every sample. Therefore, computationally cheaper modules are used for relatively easy samples while computationally expensive modules are only used for hard-to-recognize samples. This causes a decrease in computation time and memory in the test phase, and even more if applied in the training phase.
- Compatibility: Dynamic models are compatible with the innovations achieved

in static deep neural networks which makes it possible to compete with static models by keeping up to date with the recent advances in the field. State-of-the-art techniques applied to static models in data preprocessing [12] and optimization [13] can also be applied to dynamic networks. They also benefit from the efficiency-increasing methods used in static neural networks such as network pruning [14] and weight quantization [15]. Also, dynamic models can be generalized to a wide range of research areas. Dynamic models implemented for object detection can be easily transferred to image classification.

- Adaptiveness: Dynamic neural networks supply a trade-off between efficiency and accuracy which makes it possible for the users to take advantage of this trade-off. Therefore, dynamic models are adaptable to hardware with varying computational power, since it is possible to reduce the computational complexity of larger algorithms dynamically. On the other hand, static neural networks have fixed computational complexity.

- Representation power: Because of the dynamic architecture of the models, the size of the deep neural networks depends on the input sample, which leads to a larger parameter space and hence better representation power [16]

Another advantage of using dynamic neural networks is that they are easy to interpret. The working mechanism of dynamic models is very similar to the human brain since the human brain also has dynamically altering active modules for different tasks [17]. This can make it possible to overcome the problem of recognizing deep neural networks as black box models because of their inexplicit mechanism of decision-making. It can be observed which modules of the dynamic network are active for a certain sample; therefore, which parts of the dynamic network are responsible for certain classifications can be identified.

Dynamic neural networks have a wide range of application areas and different implementation techniques. Therefore it is divided into subtopics in [18] according to input sample type, dynamic on training or test phase, changing architecture or parameter. These subcategories are presented in Figure 1.1 . Firstly, it is divided according to input types such as sample-wise (red boxes), spatial-wise (blue boxes), or temporal-wise (green boxes). Sample-wise methods change architecture or parameters according to

the hardness of the input sample, spatial-wise methods focus on reducing computational complexity or increasing performance by ignoring the unimportant parts of an image and focusing on the parts of the image which include more information, and temporal-wise methods focus on sequential data. Secondly, dynamic models are sub-categorized according to which phase is dynamic: training or prediction. Yellow represents dynamic prediction and pink represents dynamic training. In this paper, we focus on sample-wise dynamic neural networks with changing depth both in test and training phases with early exiting in cascaded convolutional neural networks.

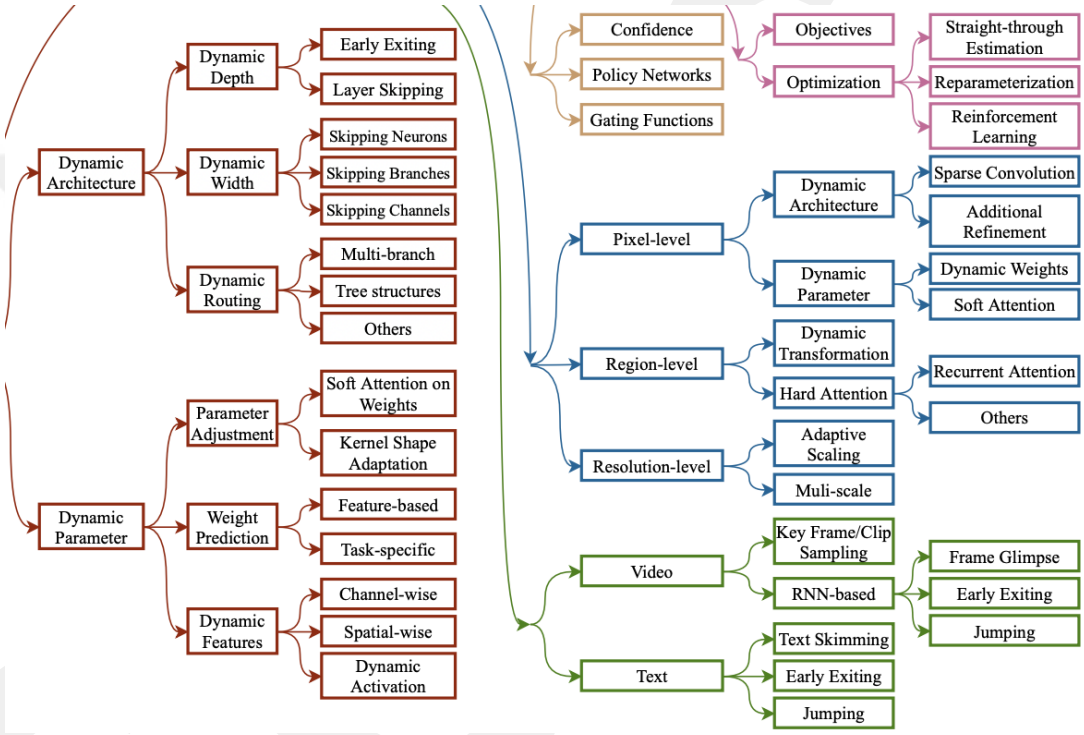

Figure 1.1 Overview of sub-classes of dynamic neural networks according to [18]

In [19], a method is proposed to use dynamic deep neural networks in order to decrease the required time for evaluation of new samples with losing accuracy as little as possible. Instead of redesigning or approximating current networks, two adaptive schemes are suggested to utilize networks. First, a dynamic neural network model is implemented which chooses the required components of the network for each sample. Computational complexity occurring because of the evaluation of the full network is reduced, by allowing quick classification using early layers of the deep neural network. Then this proposed solution is expanded to cascaded deep neural network models, which chooses the network for each sample. It is demonstrated that it is pos-

sible to reduce the computational time required for inference by taking advantage of the fact that many samples can also be successfully classified using smaller networks. In this paper, a cascaded deep neural network model is implemented only for the inference phase by using the entropy for network selection for the ImageNet dataset.

The paper [20] introduces the concept of the big-little deep neural network (BL-DNN) which aims to reduce the energy consumption required for carrying out deep neural networks while maintaining test accuracy. The BL-DNN consists of a low-energy consuming, small DNN and an energy-consuming, big DNN. To reduce energy consumption, the BL-DNN executes the small DNN first and only runs the big DNN if necessary. This approach results in reduced energy consumption while maintaining similar levels of inference accuracy. The paper presents design-time and runtime methods to control the execution of the big DNN, which are used to strike a balance between energy consumption and inference accuracy. Experiments are carried out on ImageNet and MNIST datasets for a cascaded network only for the inference phase where they use static thresholding with grid search.

The paper [21] explores the idea of using model cascades to save on cost by exploiting class asymmetry. The authors introduce a new framework called the "I Don't Know" (IDK) prediction cascades, which allows pre-trained models to be combined for faster inference without sacrificing accuracy. They propose two methods for building cascades and a cost-aware objective for this framework. The IDK cascade framework can be easily implemented into existing model serving systems without needing to retrain models. The authors present the results of the proposed techniques for ImageNet and CIFAR-10 datasets. Class probability and entropy are used for network selection with a static threshold for the inference phase.

The remainder of the thesis is organized as follows. Chapter 2 starts with the introduction of the cascaded convolutional neural networks and the connection of two networks according to the proposed selection algorithm. Later, the well-known softmax activation function is explained and how it is adapted to cascaded convolutional neural networks is described. This adaptation also introduces cross entropy as a loss function for the training of convolutional neural networks. Afterward, softmax margin is derived and a dynamic update of the threshold for solving the network selection

problem is introduced. Later, MNIST, EMNIST, and Fashion-MNIST datasets used for evaluating the proposed cascaded model are explained. Different network architectures which are utilized as small networks and benchmark networks as large networks for each dataset are introduced.

Chapter 3 includes three sets of numerical simulations. In the first set, the performance and efficiency of the proposed cascaded model are compared to small networks and LeNet [20] in the MNIST dataset. In the second set, the superior efficiency of the proposed cascaded model over the benchmark LeNet-5 model [6] without sacrificing accuracy in the EMNIST dataset is demonstrated. The last experiment compares the introduced cascaded model to the LeNet-5 model on the Fashion-MNIST dataset. Chapter 3 is concluded with a summary of experimental results and an investigation of the proposed technique's performance under various conditions.

The final remarks and the prospective future studies are presented in Chapter 4. The main contributions of the thesis are listed below:

- A novel technique for cascading deep neural networks using softmax margin is presented. (Chapter 2)
- Proposed algorithm derives the threshold dynamically and it is parameter-free. (Chapter 2)
- Dynamic deep neural network model is utilized for both training and inference. (Chapter 2)
- Proposed cascaded convolutional neural network model is evaluated in three real-world benchmark datasets: MNIST, EMNIST, and Fashion-MNIST. (Chapter 3)

CHAPTER 2

SOFTMAX MARGIN BASED DYNAMIC THRESHOLD

2.1 Introduction

The main interest of the presented thesis is the parameter-free cascaded dynamic deep neural networks both in training and inference based on class probability estimates of a small network. Therefore, first dynamic neural networks and cascaded models are explained. Hence, the softmax activation function used for estimating class probabilities is explained first, and then the derivation of the softmax margin used for calculating the confidence of a small network for each input sample is presented. After that, how these techniques, used for the inference phase, are adapted to the training phase and transformation to parameter-free dynamic threshold calculation technique is introduced. Different-sized small neural networks are used in order to observe the effect of little networks on the performance and efficiency of the cascaded model and proposed network selection algorithm. Architectures of the previously mentioned small networks and large benchmark networks are described for different datasets. The chapter is concluded with simulation-based experiments in order to compare the proposed technique to benchmark CNN models according to both accuracy and time consumption.

Figure 2.1 Cascaded Deep Neural Networks as a subtopic of Dynamic Neural Networks

Focus of the presented thesis is proposing a novel cascaded deep neural network

model with network selection both in the training and inference phases. Figure 2.1 demonstrates that dynamic neural network consists of a broad range of research areas. This thesis aims to contribute to the dynamic neural network research area with sample-wise input representation by dynamically adjusting the depth of the model with an early exiting strategy called cascaded deep neural networks. Cascading of two neural network models is achieved as shown in Figure 2.2. This is a general representation of a cascaded deep neural network model with two sub-modules. The working principle of such a system is that the input sample is fed to a small network (Model1) and the output of this small network is transferred to a router. This router decides to use the output of the model1 as final prediction or use a larger network (Model2) to make a better prediction using the same input; according to the output of the first model. This principle originates from the idea that in a dataset, some samples are relatively easier and a small network can accurately classify them, but there are also harder samples that need to be classified by a more complex and deeper network.

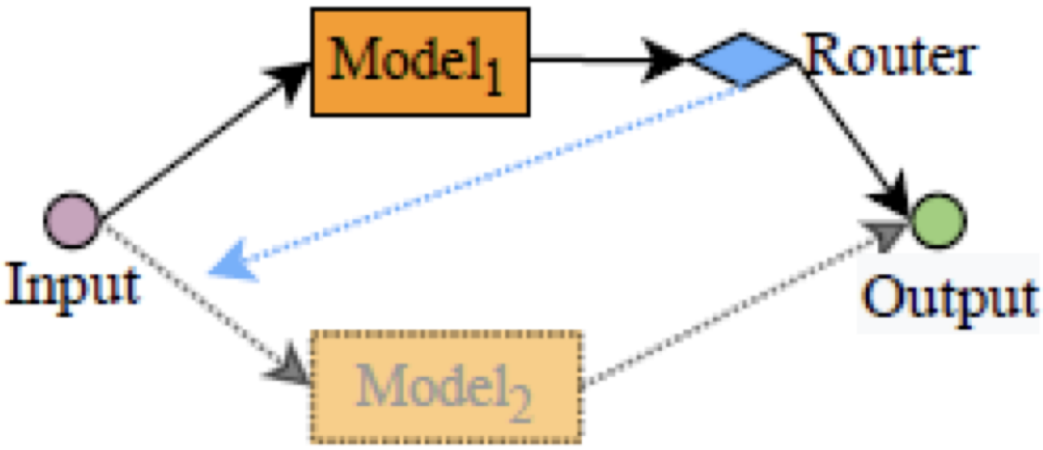

Figure 2.2 Cascading of models

2.2 Proposed Model

The proposed novel model introduced in this thesis is demonstrated in Figure 2.3 for the training phase, and in Figure 2.4 for the test phase. It is possible to observe that the cascaded model acts very similarly in both phases, with an exception in dynamic threshold calculation. The threshold is calculated according to training data and a decision is determined considering the inequality between softmax margin derived for

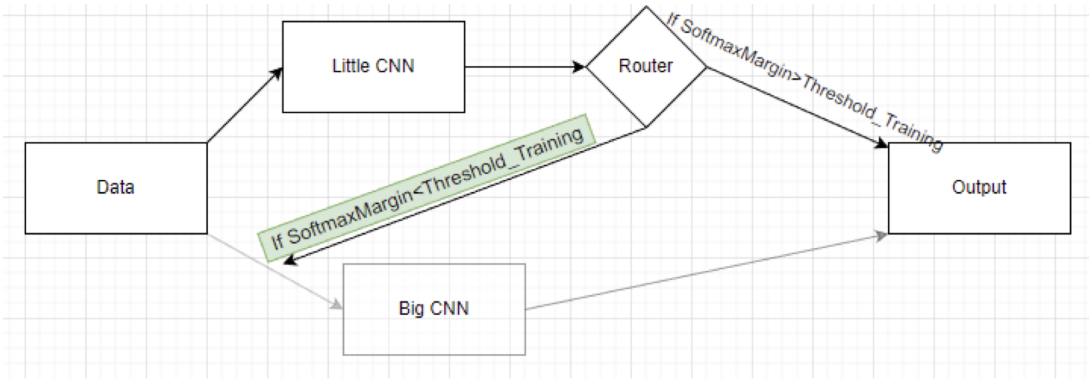

Figure 2.3 Proposed model for training phase

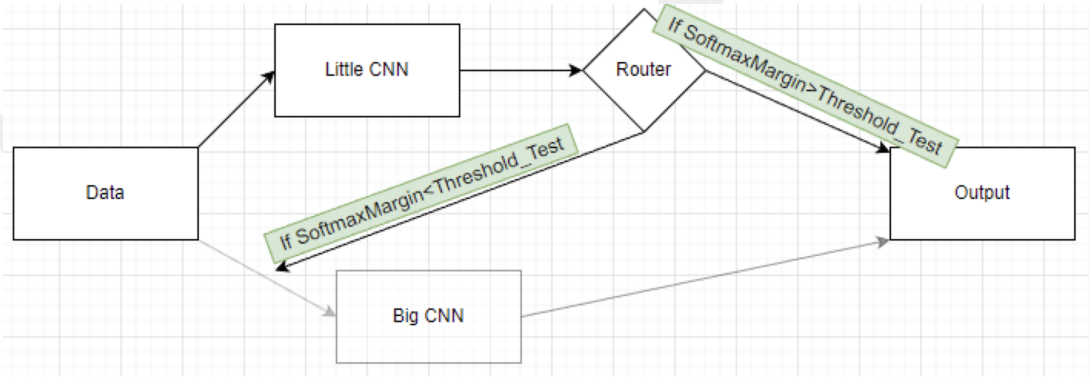

Figure 2.4 Proposed model for test phase

each individual sample and the threshold is calculated as the average of softmax margins. This calculation allows the cascaded model to dynamically update the threshold, hence leading to a better network selection without any parameters to tune. In the inference phase, threshold derivation is conducted again, but according to the test set. It is observed during the implementation that using the threshold calculated in training for the inference phase is not viable since the complexity of samples varies in training and test sets. This results in a decrease in the adaptability of the cascaded model and hence reduces the performance.

Softmax activation function is added at the end of the little convolutional neural network in order to calculate the predicted class probabilities for each sample. The softmax function is utilized to transform a set of actual numbers into a likelihood distribution of potential outcomes as first described in [22]. This technique extends the logistic function to multiple dimensions and is employed in multinomial logistic regression. The softmax function is frequently employed as the final activation function

in a neural network to standardize the output of the network to a probability distribution across anticipated output categories. The equation of softmax activation function is presented in 2.1.

$$softmax(x)_i = \frac{\exp x_i}{\sum_{j=1}^{K} \exp x_j} \tag{2.1}$$

Then softmax margin is derived from class probabilities according to the equation demonstrated in equation 2.2. Softmax margin can be used as an evaluation metric for the confidence and performance of the small network since it represents the probability difference between the most likely estimated class and the second most likely estimated class. Therefore, the larger the softmax margin value, the small network is more confident with its estimation and more likely to classify accurately. On the other hand, a lower softmax margin means the small network is less confident with its predictions and more likely to falsely classify the given sample.

$$SoftmaxMargin = \text{ Largest Softmax Value} - \text{Second Largest Softmax Value} \tag{2.2}$$

$$Threshold = \frac{\sum_{j=1}^{n} Softmax\_Margin_j}{n} \tag{2.3}$$

$$\text{selected network} = \begin{cases} \text{small network} & \textbf{if } Softmax\_Margin \geq threshold \\ \text{large network} & \textbf{if } Softmax\_Margin < threshold \end{cases} \tag{2.4}$$

Network selection in the router is conducted according to softmax margin and dynamically calculated threshold. The threshold is calculated as the average of softmax margin values which is demonstrated in equation 2.3. And decision to benefit from the larger network or accept the classification of the small network as output is made according to equation 2.4. If softmax margin for a sample is larger than the threshold, then small network is used and the large network does not observe that sample. Since

larger softmax margin results in better prediction and high confidence. On the other hand, if softmax margin is small for an input sample it means the largest softmax value is close to the second largest softmax value. Meaning that the input sample is hard to classify and little convolutional neural network is not confident about its prediction, hence router decides to use the larger network for the classification of the corresponding sample image. Proposed algorithm is demonstrated below (2.2):

Small Network ← Input sample
Input of the router ← Softmax (Output of small network)
Softmax Margin = Largest Softmax Value - Second Largest Softmax Value
Threshold = $\frac{\sum_{j=1}^{n} Softmax\_Margin_j}{n}$
if Softmax Margin ≥ Threshold **then**
    selected network = small network
    output of cascaded model = argmax(Softmax(output of small network))
else
    if Softmax Margin < Threshold **then**
        selected network = large network
        Large Network ← Input sample
        Output of Cascaded Model ← argmax(Softmax(output of large network))
    end if
end if

2.3 Network Architectures

The aim of this thesis is to propose a novel dynamic deep neural network model for reducing time consumption while the decrease in accuracy is negligible. Specifically, I have conducted experiments on three different datasets and used two state-of-the-art models as benchmarks to compare the performance of the proposed cascaded deep neural network model. Conducted experiments focus on classification tasks with benchmark image datasets.

The three datasets used in the experiments are (i) the MNIST dataset of handwritten digits, (ii) the EMNIST dataset of handwritten letters, and (iii) the Fashion-MNIST dataset of labeled fashion images. The MNIST dataset consists of 60,000 images of

handwritten digits (0-9) with 10 classes and a size of 28x28 pixels per image. The EMNIST dataset contains 124,800 images of 28 different classes, where each image is of size 28x28 pixels. Finally, the Fashion-MNIST dataset is a collection of clothes that are categorized into 10 different classes.

For experiments, two state-of-the-art models are selected as benchmarks: (i) LeNet-1 [20], and (ii) LeNet-5 [6]. These models are widely used in the field of image classification and have been shown to perform well on benchmark datasets used in this thesis.

To evaluate the performance of the proposed cascaded convolutional neural network model, extensive experiments are conducted on each of the three datasets using the two benchmark models for comparison. The proposed model utilizes a novel network selection technique that aims to reduce the time complexity of a relatively large deep neural network, at the same time without sacrificing the accuracy of classifications. Also, the effect of varying small networks on the performance of the proposed cascaded convolutional neural network model is explored.

The results of the experiments demonstrate that the proposed network selection algorithm for cascaded deep neural networks remarkably reduces the time consumption of benchmark models on all three datasets, achieving very similar accuracy scores. These findings suggest that the proposed model in this thesis is a promising approach for lessening the time complexity of large networks in image recognition tasks without sacrificing performance.

2.3.1 MNIST Dataset

The MNIST dataset is a popular benchmark dataset in the area of deep learning and computer vision. It consists of 60,000 images of handwritten digits (0-9) with a size of 28x28 pixels. The dataset is split into 54,000 images for training and 6,000 images for testing. The MNIST dataset has been broadly used for developing and evaluating deep neural network and convolutional neural network models for image classification [6, 23, 24, 25, 26]. MNIST dataset is selected as one of the benchmark datasets for this thesis because it is accepted as a standard for developing and evaluating novel models.

The MNIST dataset has played a significant role in the development of deep learning algorithms, especially for convolutional neural networks. Figure 2.5 demonstrates sample images from the MNIST dataset. Table 2.1 demonstrates information about the size of the MNIST dataset.

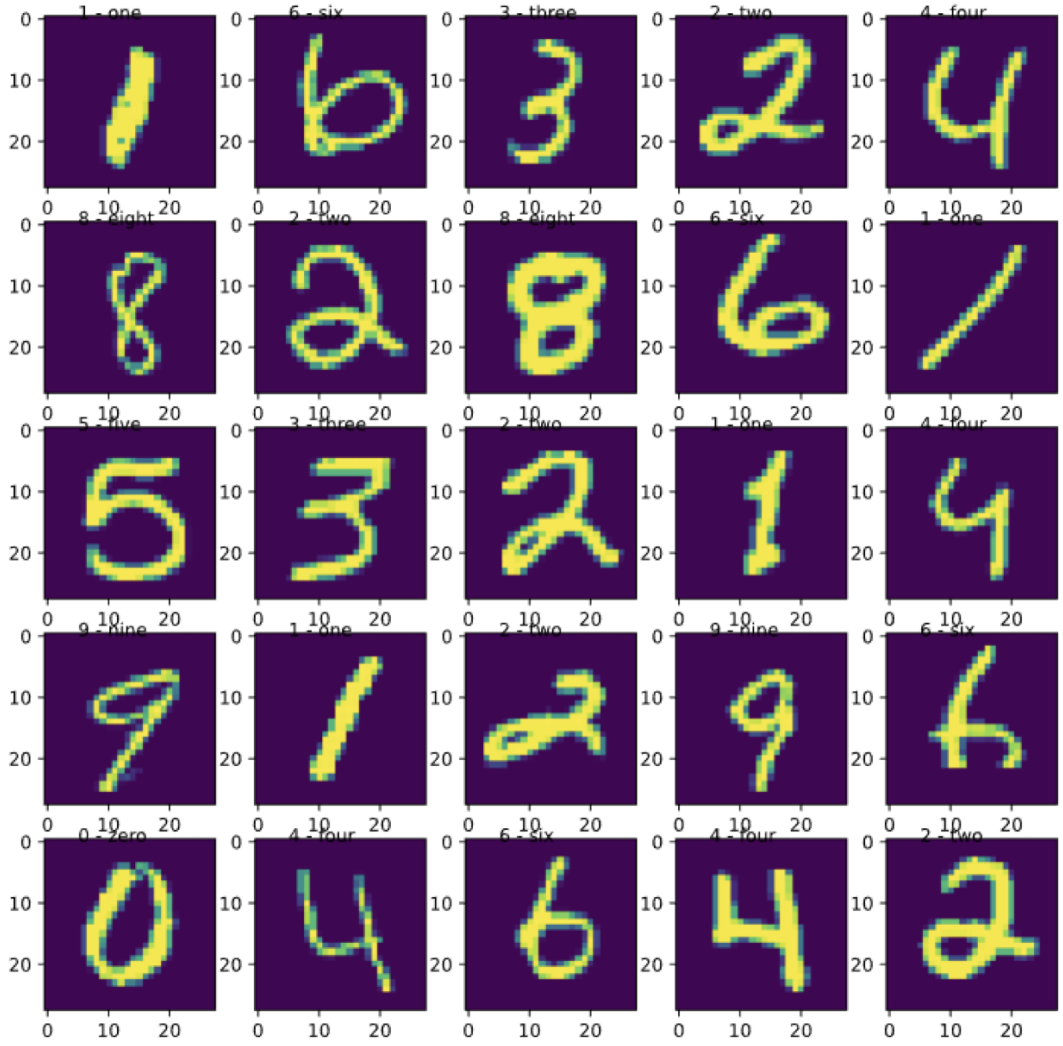

Figure 2.5 Samples images from MNIST dataset

LeNet [6] is used as a benchmark model in order to compare the performance and time complexity reduction provided by the introduced cascaded deep neural network model. The reasons for choosing LeNet as the benchmark model are that it is one of the earliest and most successful convolutional neural network models for classifying handwritten digits, commonly used as a benchmark model to evaluate novel neural network models [27, 28] and it has a broad range of application areas such as computer vision, image recognition, and natural language processing.

Table 2.1 Information about MNIST dataset

Element of Dataset	Size
Sample Size	60.000
Training Set Size	54.000
Test Set Size	6.000
Image Size	28x28
Number of Classes	10

Table 2.2 Architecture of modified LeNet used in MNIST dataset

Layer name	Size
Input Layer	28x28
Convolutional Layer	5x5x20
Maxpool Layer	2x2
Convolutional Layer	5x5x50
Maxpool Layer	2x2
Fully connected Layer	500
Fully connected Layer	10

A modified version of the original LeNet model, inspired from the [20] is used in MNIST dataset as a benchmark. It consists of 6 layers, including two convolutional layers, two subsampling layers, and two fully connected layers. The input to the network is a grayscale image of size 28x28 pixels, which is passed through the two convolutional layers to extract features. The first convolutional layer has twenty filters of size 5x5, while the second layer has fifty filters of size 5x5. The outputs of the convolutional layers are then passed through the subsampling layers, which reduce the spatial dimensions of the feature maps. The subsampling layers use a technique called max pooling, which divides the feature maps into non-overlapping regions and computes the maximum value for each region. The output of the subsampling layers is then passed through two fully connected layers, which perform the classification task. The first fully connected layer has 500 units, and the second has 10 units (one for each class from 0 to 9). Softmax function, which is a non-linear activation function, is used to map the score for each class to a probability between 0 and 1. The neural network is trained with backpropagation using stochastic gradient descent, by using subsets of training set at each iteration. The architecture of the LeNet model is presented in Table 2.2.

Three different small networks are designed for constructing three different cascaded

Table 2.3 Architectures of small networks used in MNIST dataset

Layer name	Small1	Small2	Small3(10 Epochs)
Convolutional Layer	5x5x10	5x5x2	5x5x2 with stride=2
Maxpool Layer	2x2	2x2	2x2
Convolutional Layer	5x5x20	5x5x4	5x5x4 with stride=2
Maxpool Layer	2x2	2x2	2x2
Fully connected Layer	50	50	50
Fully connected Layer	10	10	10

networks all including the LeNet model as the large network, in order to observe the effect of different-sized small networks on the performance and efficiency of the introduced cascaded convolutional neural network model. Architectures of three small convolutional neural network models are presented in Table 2.3. While the number of layers is the same for all three small networks, the size of the layers and stride values are different in order to change the number of connections in each network, hence altering the number of parameters to be trained. Therefore, these previously presented small networks diverge in accuracy and time complexity.

2.3.2 EMNIST Dataset

The EMNIST dataset is a broadly used benchmark dataset in the field of deep learning and computer vision. It consists of 124.800 images of handwritten letters of the English alphabet with a size of 28x28 pixels. The dataset is split into training and test sets, which include 112.320 and 12.480 samples respectively. The EMNIST dataset has been widely used for developing and evaluating deep neural network and convolutional neural network models for image classification. EMNIST dataset is preferred as one of the benchmark datasets in this thesis since it is acknowledged as a standard for advances and assessment of novel models. The EMNIST dataset has played a vital role in the advancement of deep learning algorithms, especially for convolutional neural networks. Figure 2.6 demonstrates sample images from the EMNIST dataset. Table 2.4 demonstrates statistical information about the EMNIST dataset.

LeNet-5 [6] model is employed as the large network in the cascaded neural network model and used as a benchmark convolutional model in the EMNIST dataset. An illustration of LeNet-5 model architecture is demonstrated in Figure 2.7 and a com-

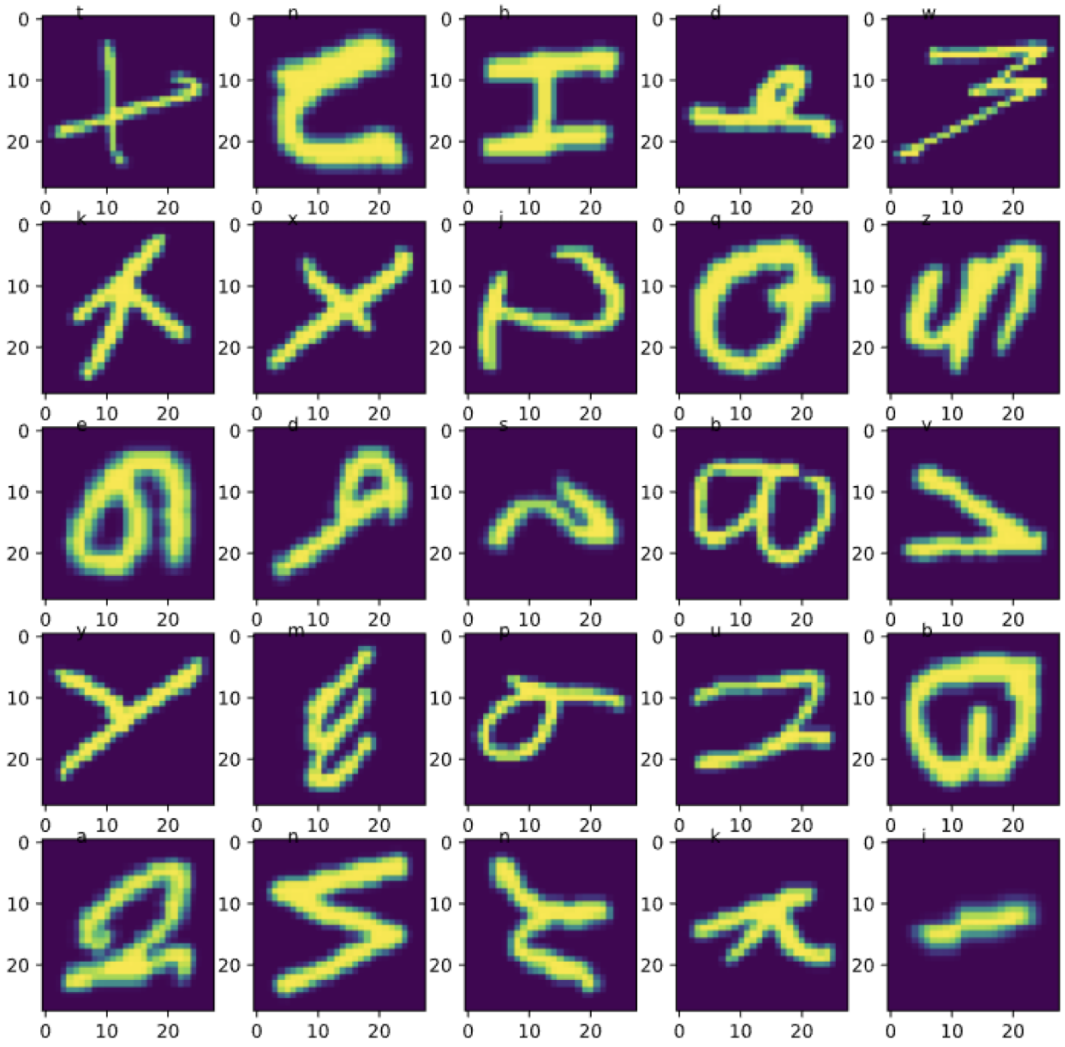

Figure 2.6 Sample images from EMNIST dataset

prehensive explanation of each layer is presented in Table 2.5. LeNet5 consists of 2 convolutional layers, 2 subsampling layers which are utilized in order to reduce the spatial dimension of feature maps, and 3 fully connected layers, with a softmax activation function in the last fully connected layer. The first convolutional layer includes 6 filters with a size of 5x5, and the second convolutional layer consists of 16 filters with a size of 5x5. There are subsampling layers after each convolutional layer, and both of them have a size of 2x2. These subsampling layers use a technique called max pooling, which divides the feature maps into non-overlapping regions and computes the maximum value of each region. Then, the output of the second max-pooling layer is passed into three fully connected layers, with decreasing sizes, for the classification task. The first fully connected layer has 120 neurons, the second one has 84 neurons

Table 2.4 Information about EMNIST dataset

Element of Dataset	Size
Sample Size	124.800
Training Set Size	112.320
Test Set Size	12.480
Image Size	28x28
Number of Classes	26

Table 2.5 Architecture of LeNet-5 used in EMNIST dataset

Layer name	Size
Convolutional Layer	5x5x6
Maxpool Layer	2x2
Convolutional Layer	5x5x16
Maxpool Layer	2x2
Fully connected Layer	120
Fully connected Layer	84
Fully connected Layer	26

and the last fully connected layer consists of 26 neurons (one for each letter). Lastly, the output of the third fully connected layer is fed into the softmax activation function in order to calculate class probabilities.

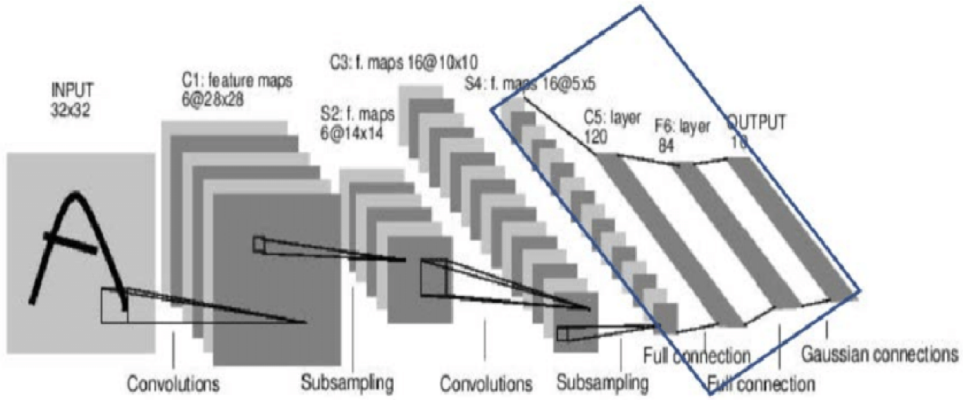

Figure 2.7 Figure of LeNet-5 model architecture

To examine how the size of small networks affects the performance and efficiency of cascaded convolutional neural network models, four different small networks are designed that all incorporate the LeNet-5 model as the large network. Each of these small convolutional neural network models has a difference in the number of layers, the size of their layers, or stride values, which altered the number of connections in

Table 2.6 Architecture of First small network utilized in EMNIST dataset

Layer name	Size
Convolutional Layer	5x5x10
Maxpool Layer	2x2
Convolutional Layer	5x5x20
Maxpool Layer	2x2
Fully connected Layer	50
Fully connected Layer	26

Table 2.7 Architectures of three small networks utilized in EMNIST dataset

Layer name	Small2(Epoch=5)	Small3(Epoch=1)	Small4
Convolutional Layer	5x5x10	5x5x10	5x5x10 with stride=3
Maxpool Layer	2x2	2x2	2x2
Fully connected Layer	50	50	50
Fully connected Layer	26	26	26

each network and the number of parameters to be trained. As a result, presented small networks demonstrate varying levels of accuracy and time complexity. Detailed descriptions of the architectures for these small networks are provided in Tables 2.6,2.7. The first small network presented in Table 2.6 has the most number of layers compared to other little networks utilized in the EMNIST dataset. Three different convolutional neural networks are acquired by omitting a convolutional layer and max-pooling layer from the first small neural network. Even though these three networks have the same depths, they vary in layer width and stride of the convolutional layer. Hence, each network has a diverse number of connections between layers and a different number of parameters. Which results in different networks, with varying performances and efficiencies.

2.3.3 Fashion MNIST Dataset

The Fashion MNIST dataset is a collection of images used as a benchmark for machine learning algorithms, particularly those designed for image classification and image recognition, example images of the Fashion MNIST are provided in Figure 2.8. It consists of 70.000 images divided into training and test sets of 60.000 and 10.000 respectively, with size 28x28 pixels. Statistical information on Fashion MNIST is

presented in Table 2.8. The dataset is uniformly categorized into 10 different clothing classes, including shirts, sneakers, sandals, bags, and trousers. The main purpose of creating the Fashion MNIST dataset is to replace the most widely used MNIST dataset of handwritten digits with a more complex dataset without changing size and format. Because of the fast advancements in deep learning field, the MNIST dataset has become too simple for modern machine learning algorithms to effectively demonstrate their capabilities [29]. Due to its similarity to the MNIST dataset in terms of size and format, the Fashion MNIST dataset is a convenient substitute for researchers who wish to evaluate their algorithms using a more challenging image dataset. Hence, Fashion MNIST dataset is widely used in order to assess the performance of many novel convolutional neural network algorithms, such as [30, 31, 32, 33, 34].

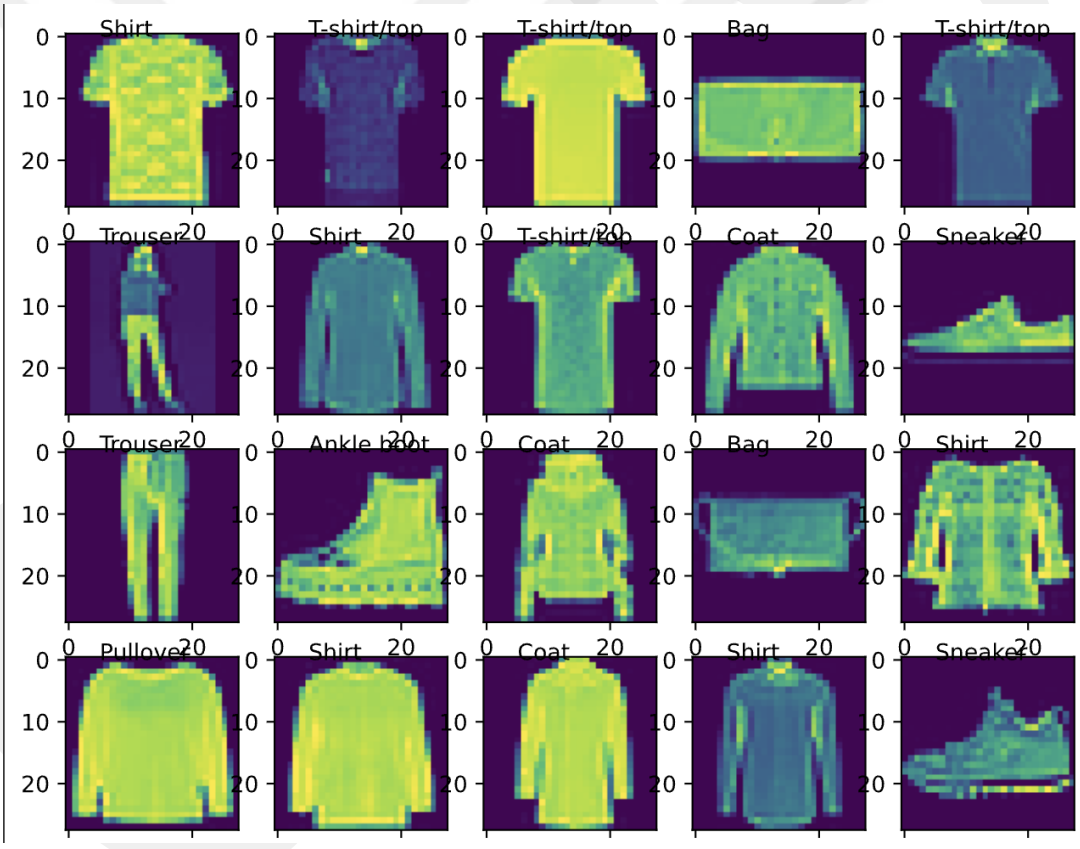

Figure 2.8 Sample images from Fashion-MNIST dataset

The leNet-5 model used in the EMNIST dataset is adapted to the Fashion MNIST classification problem, by altering the last fully connected layer. Since there are 10 categories in the F-MNIST dataset, the last fully connected layer is decreased to 10 neurons. The architecture of the slightly changed LeNet-5 model for the Fashion

Table 2.8 Information about Fashion-MNIST dataset

Element of Dataset	Size
Sample Size	70.000
Training Set Size	60.000
Test Set Size	10.000
Image Size	28x28
Number of Classes	10

Table 2.9 Architecture of LeNet-5 used in FMNIST dataset

Layer name	Size
Convolutional Layer	5x5x6
Maxpool Layer	2x2
Convolutional Layer	5x5x16
Maxpool Layer	2x2
Fully connected Layer	120
Fully connected Layer	84
Fully connected Layer	10

MNIST dataset is presented in Table 2.9. Three relatively small neural network models with varying depths and number of parameters are implemented and separately cascaded with the LeNet-5 model in order to evaluate the performance and efficiency of the proposed network selection technique over the Fashion-MNIST dataset. The depth of network architecture and number of neurons per layer of three small convolutional neural networks are demonstrated in Table 2.10.

Table 2.10 Architectures of three small networks utilized in Fashion MNIST dataset

Layer name	Small1	Small2	Small3
Convolutional Layer	5x5x6	5x5x2	5x5x2 with stride=3
Maxpool Layer	2x2	2x2	2x2
Fully connected Layer	50	50	50
Fully connected Layer	10	10	10

CHAPTER 3

EXPERIMENTAL RESULTS

In the first set of simulations, the accuracy and execution time of cascaded neural networks utilized with the network selection technique proposed in Chapter 2 compared to benchmark model LeNet [20] for the MNIST dataset. Three varying-sized small neural networks are cascaded with the LeNet model separately in order to observe the effect of the small network module over the introduced cascaded convolutional neural network model. For simulation purposes, a big convolutional neural network (CNN) is fixed to LeNet while little CNN is altered in the sense of depth and width. Parameters of the LeNet model are obtained from the paper [20] in order to have a well-performing benchmark model, hence hyperparameter tuning and cross-validation are not required.

In the second and third sets of simulations, the big CNN model and benchmark model are changed to LeNet-5, since experiments are conducted on different datasets. EMNIST and Fashion-MNIST are more complex and harder to classify compared to the MNIST dataset, therefore more complex and deeper CNN model is utilized both as a big network module of the cascaded system and a benchmark model. Parameters of the LeNet-5 model are obtained from the paper [34] for the Fashion-MNIST dataset and from the paper [6] for the EMNIST dataset in order to have a well-performing benchmark model, hence hyperparameter tuning and cross-validation is not required.

The numerical results of three sets of simulations acquired from three different datasets are compared in order to observe the effect of the complexity of the classification task over the performance and efficiency of the proposed parameter-free network selection algorithm designed for cascaded convolutional neural networks.

3.1 MNIST Dataset

LeNet model is utilized with hyperparameters optimized in paper [20] as 0.001 learning rate for 10 epochs, and results in 98.50% test accuracy and 99.19% training accuracy with 421 seconds of total run-time (training and test phase run-times are combined). Test accuracy and run-time of LeNet are employed as benchmark for the assessment of cascaded deep neural network models in the next sections. The performance of the benchmark model is presented in Figure 3.1.

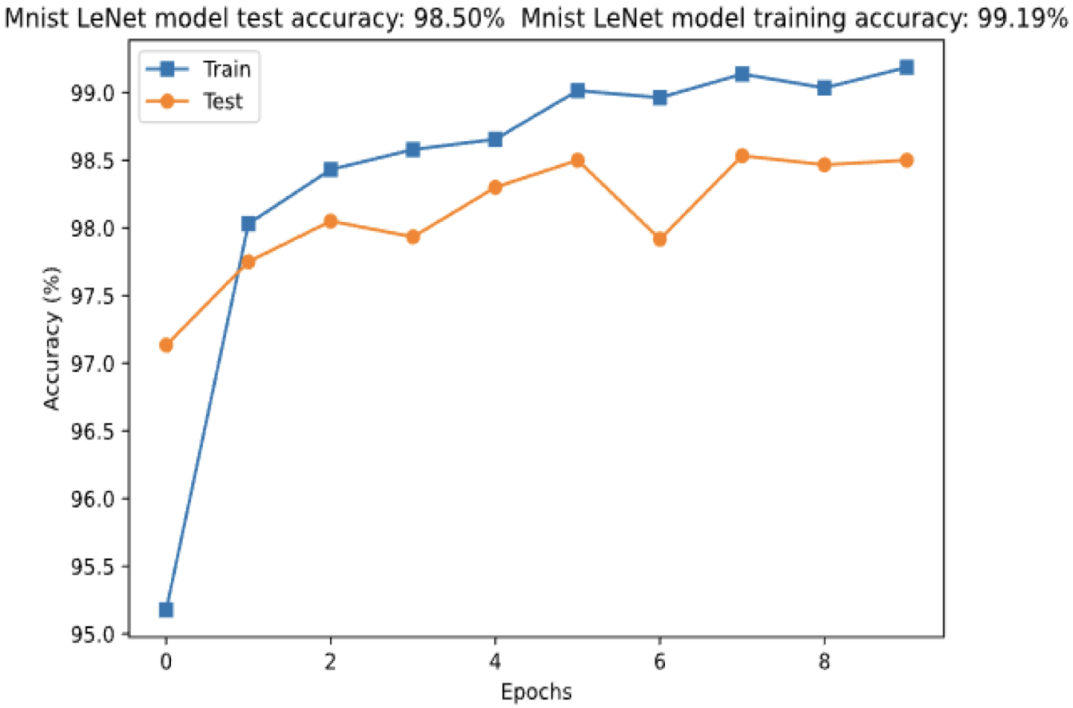

Figure 3.1 Performance of benchmark model in MNIST dataset

3.1.1 Small1 + LeNet

This section involves the experimentation on the first cascaded network using the small network described in Table 2.3 and the big network described in Table 2.2 in Chapter 2. Test accuracies of small CNN, large CNN, and cascaded model acquired by combining small and large networks with the introduced network selection algorithm are presented in Figure 3.2. In addition to evaluating the performance of the models in terms of accuracy, the runtimes of each model were also measured, which are

demonstrated in Figure 3.3. The runtimes of the models were found to be 38 seconds, 421 seconds, and 67 seconds, respectively. These results suggest that the model with the highest accuracy was not necessarily the most efficient in terms of runtime. It is possible to observe from Figure 3.2 and Figure 3.3 that, the loss in performance is very small with the cascaded model while the decrease in runtime is substantial. The ratio of samples routed from little to big network during training is 4.9% and 12% during test, which are shown in Figure 3.4. It can be observed from these Figures, even though the accuracy of the cascaded model is very similar to the benchmark network, performance of the cascaded model is only slightly better than the small network when the ratio of training samples passed from little CNN to large CNN is very low. In the case of a small network that has a very good performance, it is mostly confident in classifications, hence passing only a small amount of samples to the large network. This causes most of the classifications to be made by the little CNN, in this case, 88% and leads to very similar accuracy between small and cascaded models. Another reason for the close performances of the small and cascaded model is, the low ratio of samples passed during the training phase, hence big CNN module of the cascaded model is under-trained. In order to assess the performance of the introduced network selection algorithm more clearly, more experiments are conducted with higher little to big ratios by either utilizing more simpler small networks or using more complex datasets in the next sections.

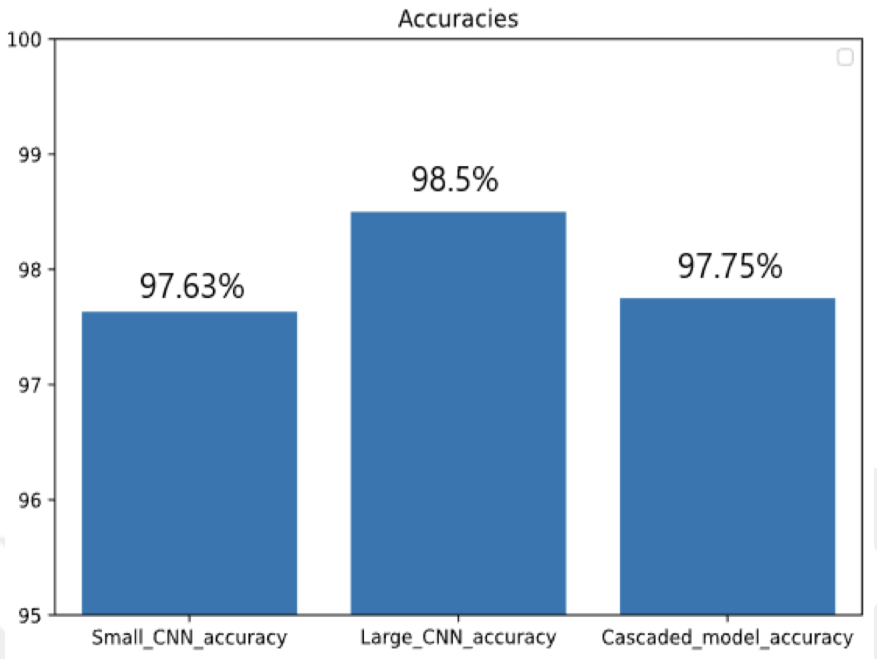

Figure 3.2 Performance of Small1, LeNet and cascaded model in MNIST dataset

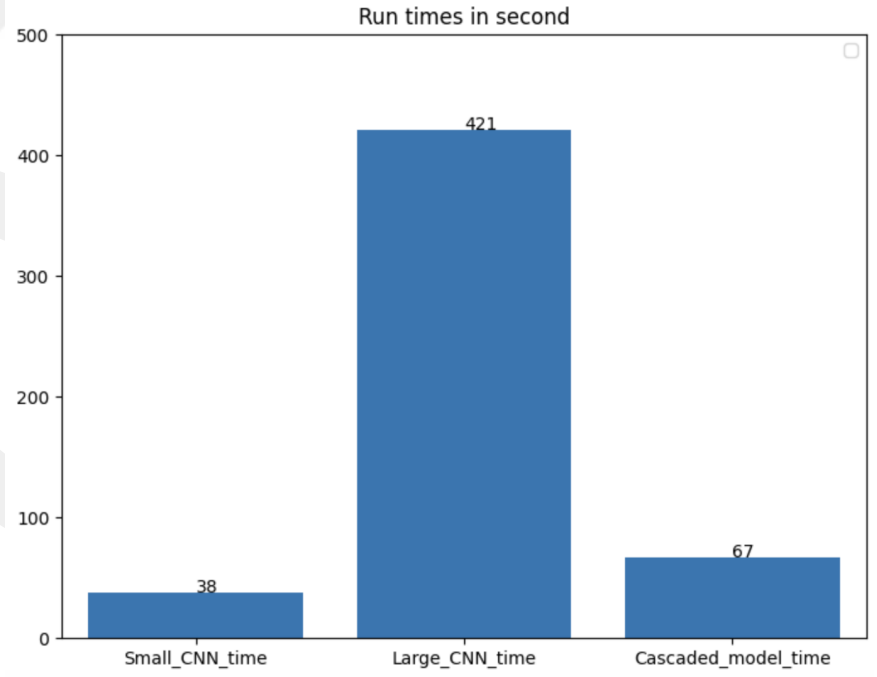

Figure 3.3 Time consumption of Small1, LeNet and cascaded model in MNIST dataset

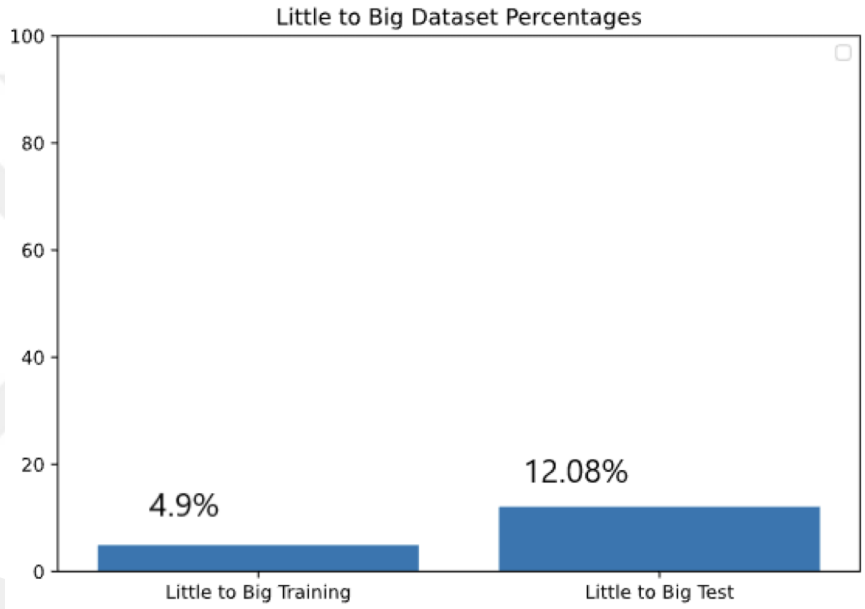

Figure 3.4 Ratio of samples passed from Small1 CNN to big CNN in the cascaded model

3.1.2 Small2 + LeNet

This section describes the experimentation carried out on a cascaded network, which uses both small and big networks as described in Tables 2.3 and 2.2 in Chapter 2. Test accuracies for the small CNN, large CNN, and cascaded model obtained using the proposed network selection algorithm are presented in Figure 3.5, and their runtimes are shown in Figure 3.6. The runtimes were found to be 13 seconds, 421 seconds, and 31 seconds, respectively. These results indicate that the most accurate model is not always the most efficient in terms of runtime. Interestingly, the cascaded model demonstrates only a small loss in accuracy compared to the benchmark network, but a substantial decrease in runtime. Figures 3.5, 3.6, and 3.7 show that only a small percentage of samples were routed from the small to large network during training and testing, suggesting that when the small network performs well, it is confident in its classifications and only passes a small number of samples to the large network. Overall, these findings demonstrate the potential of the proposed network selection algorithm to improve the efficiency of cascaded models without sacrificing much accuracy. It is possible to notice the performance difference between small and cascaded networks, as the accuracy of small network is less in this experiment. However, there is a larger performance gap between benchmark and cascaded models, because the ratio of samples routed to big CNN during training is too low (2.4%).

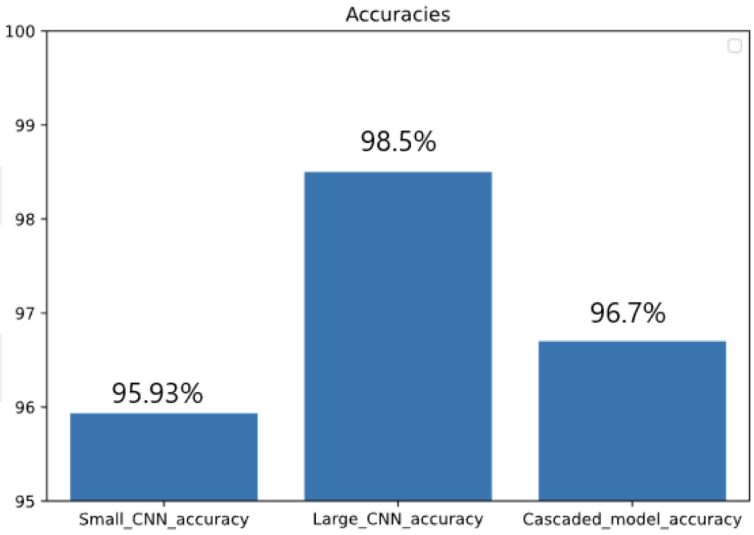

Figure 3.5 Performance of Small2, LeNet and cascaded model in MNIST dataset

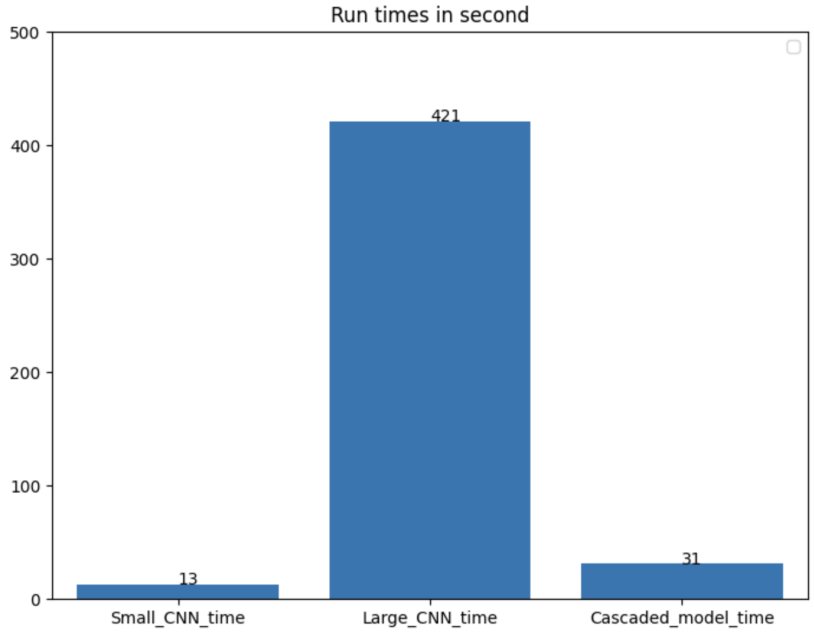

Figure 3.6 Time consumption of Small2, LeNet and cascaded model in MNIST dataset

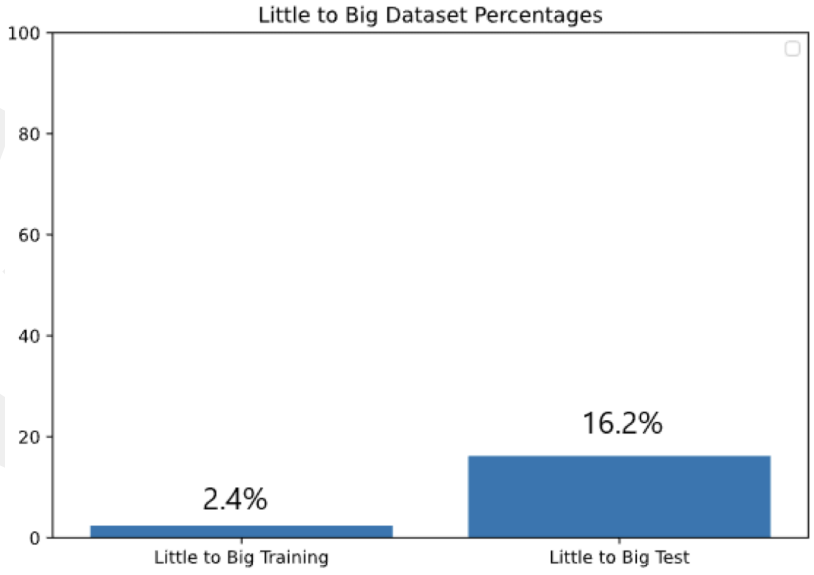

Figure 3.7 Ratio of samples passed from Small2 CNN to big CNN in the cascaded model

3.1.3 Small3 + LeNet

This section reports on an experimental evaluation of the cascaded network, which utilizes both small and big networks described in Tables 2.3 and 2.2 from Chapter 2. Figure 3.8 presents test accuracies for the small CNN, large CNN, and cascaded model obtained using the introduced network selection algorithm, and Figure 3.9 shows their corresponding runtimes. The results reveal that, although the model with the highest accuracy achieved a score of 98.50%, it had a much longer runtime of 808 seconds compared to the other models. Interestingly, the cascaded model exhibits only a small reduction in accuracy when compared to the benchmark network, yet achieved a significant decrease in runtime. Figures 3.8, 3.9, and 3.10 reveal that higher percentage of samples were routed from the small to large network during training and testing, indicating that the small network is not confident in its classifications and passes more samples to the large network; lower test accuracy of small network also supports this argument. Hence, it can be observed that lower performance of little module causes more training of large module; therefore a significant performance gap between small and cascaded networks and less runtime reduction. These results highlight the potential of the proposed network selection algorithm to enhance the efficiency of cascaded models while maintaining high accuracy.

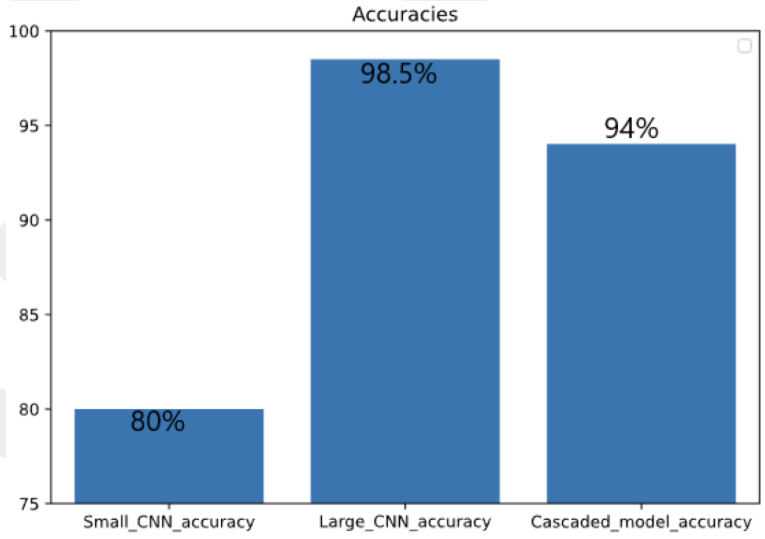

Figure 3.8 Performance of Small3, LeNet and cascaded model in MNIST dataset

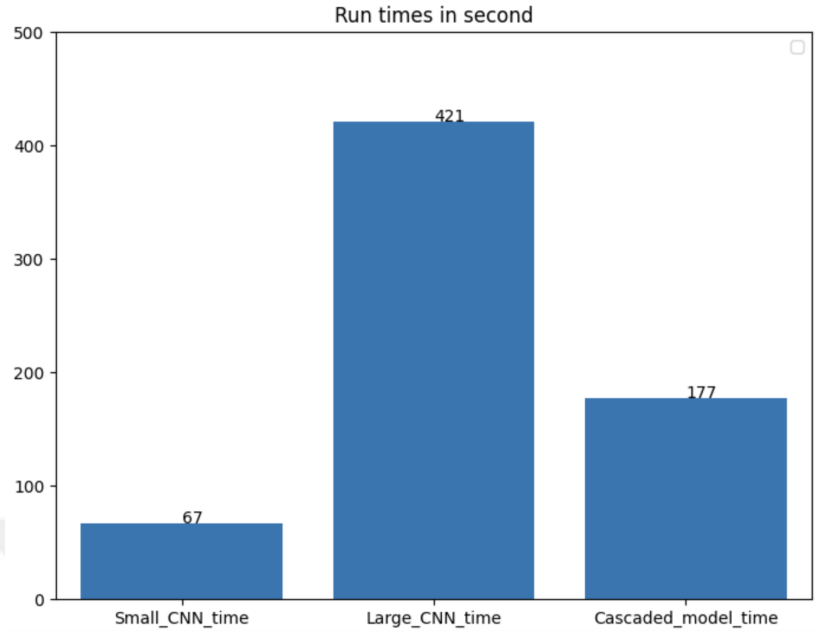

Figure 3.9 Time consumption of Small3, LeNet and cascaded model in MNIST dataset

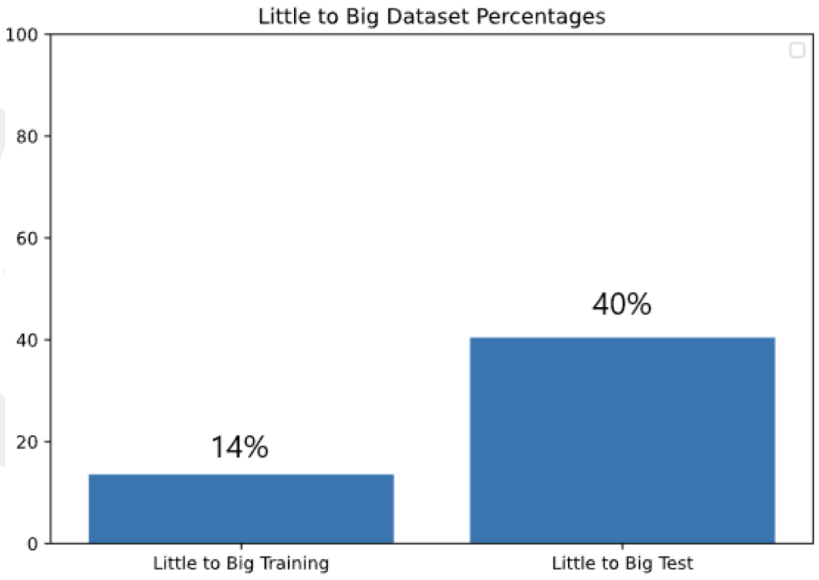

Figure 3.10 Ratio of samples passed from Small3 CNN to big CNN in the cascaded model

3.2 EMNIST Dataset

LeNet-5 model is utilized with hyperparameters optimized in paper [6] and results in 91.46% test accuracy and 94.01% training accuracy with 386 seconds of total run-time (training and test phase run-times are combined). Test accuracy and run-time of LeNet-5 are employed as benchmark for the assessment of cascaded deep neural network models in the next sections. Performance of the benchmark model is presented in Figure 3.11.

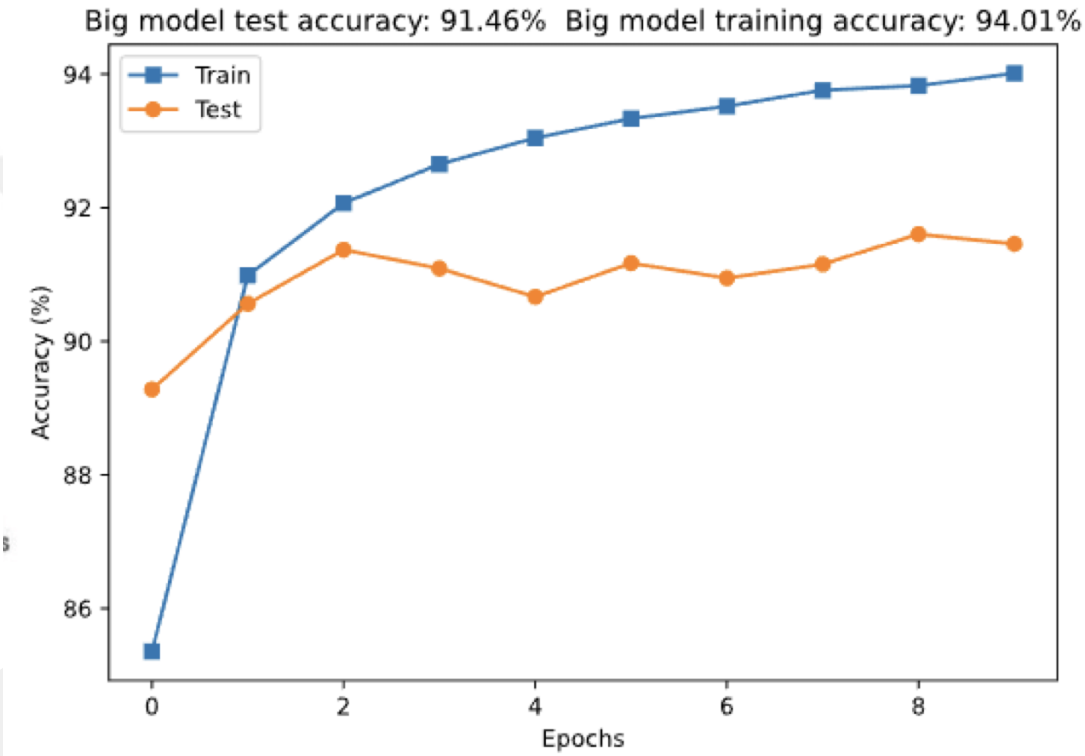

Figure 3.11 Performance of benchmark model in EMNIST dataset

3.2.1 Small1 + LeNet5

This section involves the experimentation on the first cascaded network using a small network described in Table 2.6 and a big network described in Table 2.5 in Chapter 2. Small1 CNN has less depth and width compared to LeNet-5, with fewer fully connected layers and fewer neurons in each fully connected layer. Test accuracies of small CNN, large CNN, and cascaded model acquired by combining small and large networks with the introduced network selection algorithm are presented in Figure

3.12. In addition to evaluating the performance of the models in terms of accuracy, the runtimes of each model were also measured, which are demonstrated in Figure 3.13. The runtimes of the models were found to be 91 seconds, 386 seconds, and 156 seconds, respectively. It is observed that the high accuracy of the small CNN module causes a low ratio of samples to be routed from little CNN to big CNN as in Figure 3.14. Therefore, performance of the cascaded model is more similar to the small neural network, although the accuracy gap between the cascaded model and benchmark LeNet-5 is less than 2%. Even though the cascaded model increases efficiency by more than 50% while losing only 2% accuracy in this experiment; the advantage of the proposed network selection algorithm is more noticeable when there is more difference between small and large networks regarding performance.

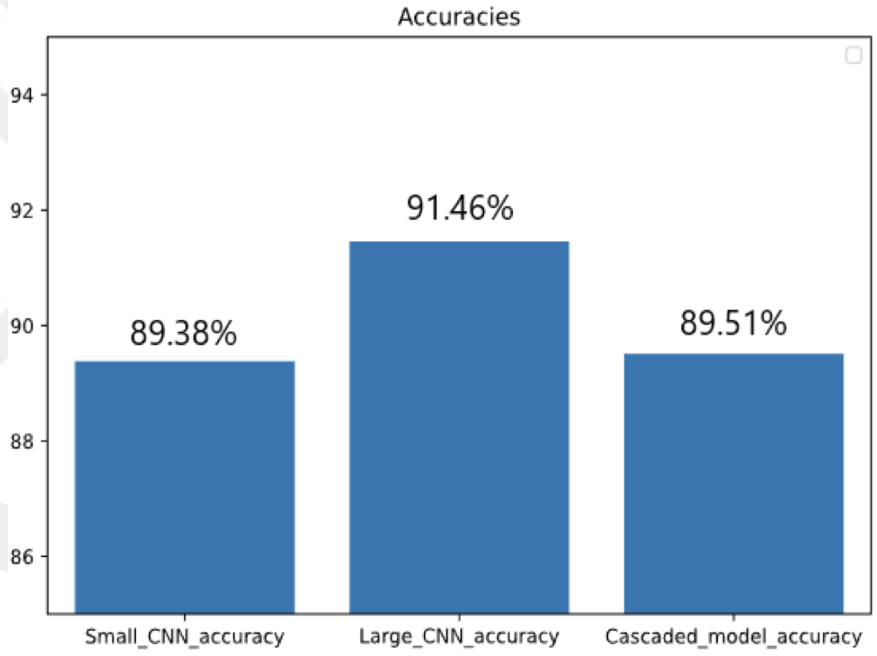

Figure 3.12 Performance of Small1, LeNet-5 and cascaded model in EMNIST dataset

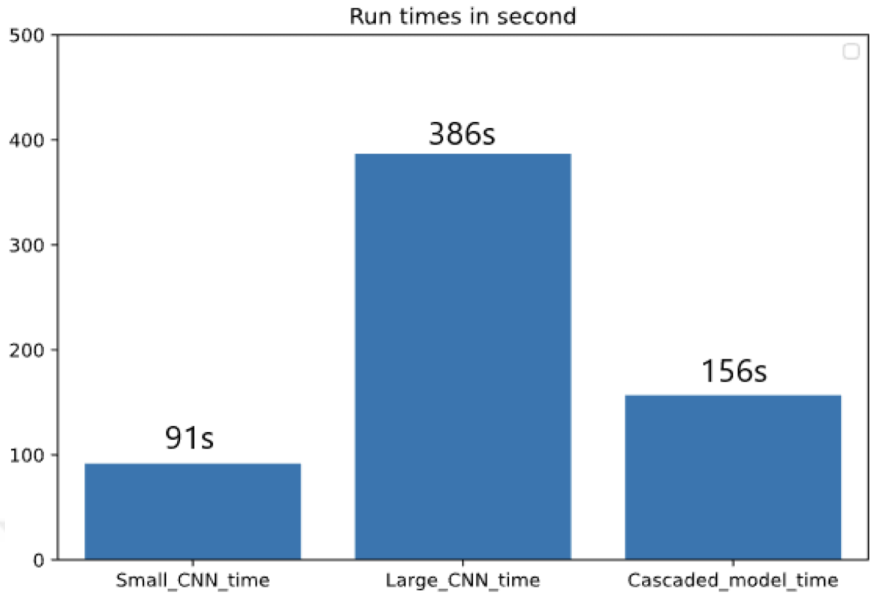

Figure 3.13 Time consumption of Small1, LeNet-5 and cascaded model in EMNIST dataset

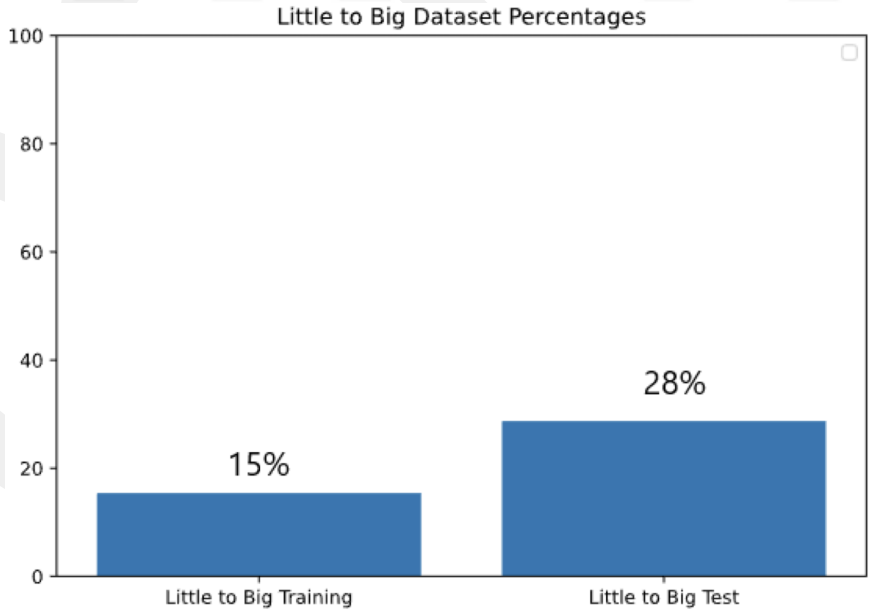

Figure 3.14 Ratio of samples passed from Small1 CNN to big CNN in the cascaded model for EMNIST dataset

3.2.2 Small2 + LeNet5

This section involves the experimentation on the cascaded network using the small network described in Table 2.7 and the big network described in Table 2.5 in Chapter 2. Small2 network has reduced depth compared to Small1 CNN utilized for the EMNIST dataset, with fewer convolutional layers; while possessing the same network depth. Small2 is trained for a higher number of epochs compared to Small1; hence has greater runtime in spite of its shallower architecture. The number of epochs is not optimized for small networks utilized in cascades in order to prove that the proposed network selection technique performs well even with non-optimized small networks. Test accuracies of small CNN, large CNN, and cascaded model acquired by combining small and large networks with the introduced network selection algorithm are presented in Figure 3.15. In addition to evaluating the performance of the models in terms of accuracy, the runtimes of each model were also measured, which are demonstrated in Figure 3.16. The runtimes of the models were found to be 235 seconds, 386 seconds, and 279 seconds, respectively. It is noticed that the cascaded model raises efficiency by 27.7% with a decline in performance by 4.5%. These results demonstrate that a small network with a relatively large runtime can cause a harsh decrement in efficiency without any improvement in accuracy.

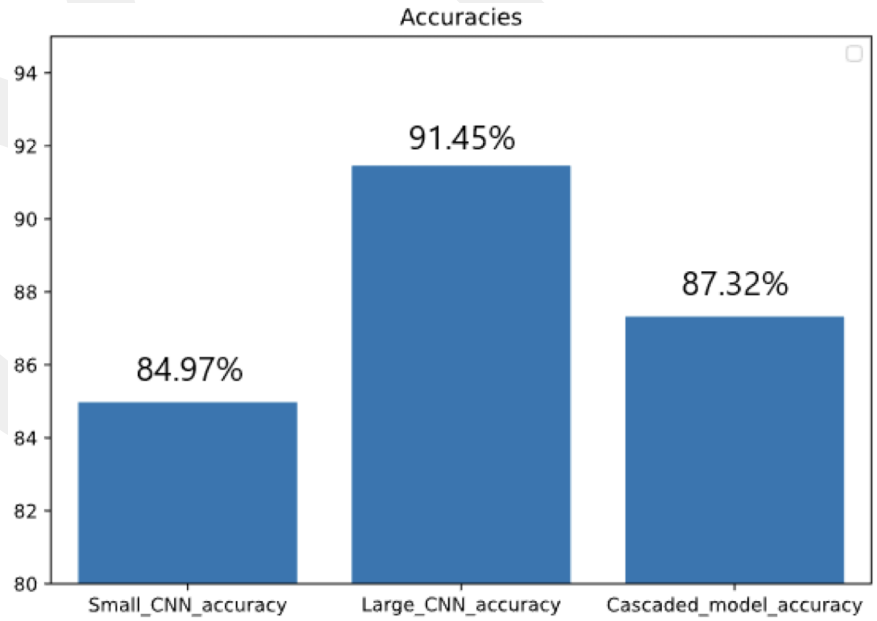

Figure 3.15 Performance of Small2, LeNet-5 and cascaded model in EMNIST dataset

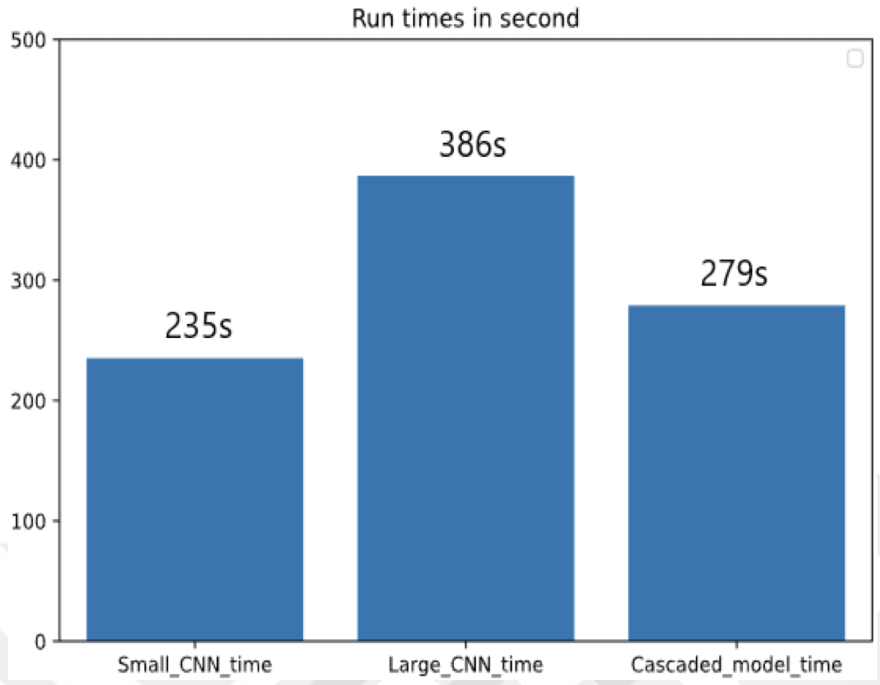

Figure 3.16 Time consumption of Small2, LeNet-5 and cascaded model in EMNIST dataset

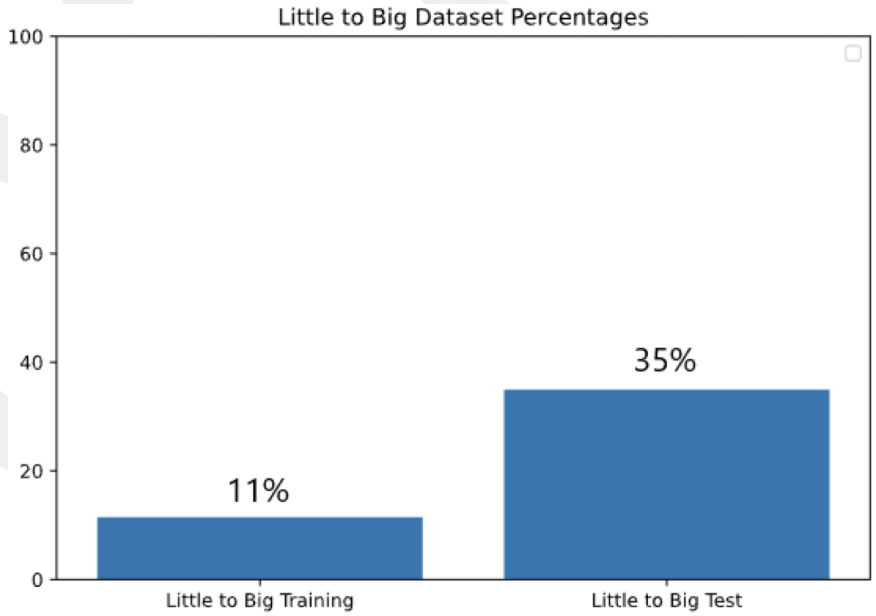

Figure 3.17 Ratio of samples passed from Small2 CNN to big CNN in the cascaded model for EMNIST dataset

3.2.3 Small3 + LeNet5

This section involves the experimentation on the cascaded network using the small3 network described in Table 2.7 and the big network described in Table 2.5 in Chapter 2. Test accuracies of small CNN, large CNN, and cascaded model acquired by combining small and large networks with the introduced network selection algorithm are presented in Figure 3.18. In addition to evaluating the performance of the models in terms of accuracy, the runtimes of each model were also measured, which are demonstrated in Figure 3.19. The runtimes of the models were found to be 38 seconds, 386 seconds, and 95 seconds, respectively. Small3 network has the same depth and width compared to Small2, hence the same number of neurons and connections. However, it employs 1 training epoch, which reduces runtime drastically while also causing a decrease in the performance of the small module. However, it is observed that the cascaded model performs better and more efficiently even though employed little CNN is worse in both properties. This situation can be explained by the larger rate of samples passed from little CNN to big CNN, which is demonstrated in Figure 3.20. Since a small module has lower accuracy, it also has lower confidence in both training and test samples. Therefore, more samples are routed to the big module in both phases and enhanced performance is obtained.

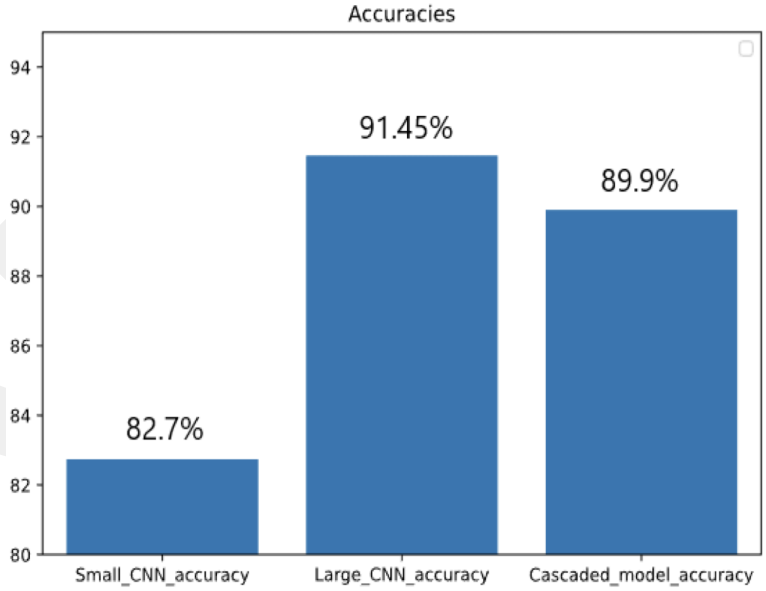

Figure 3.18 Performance of Small3, LeNet-5 and cascaded model in EMNIST dataset

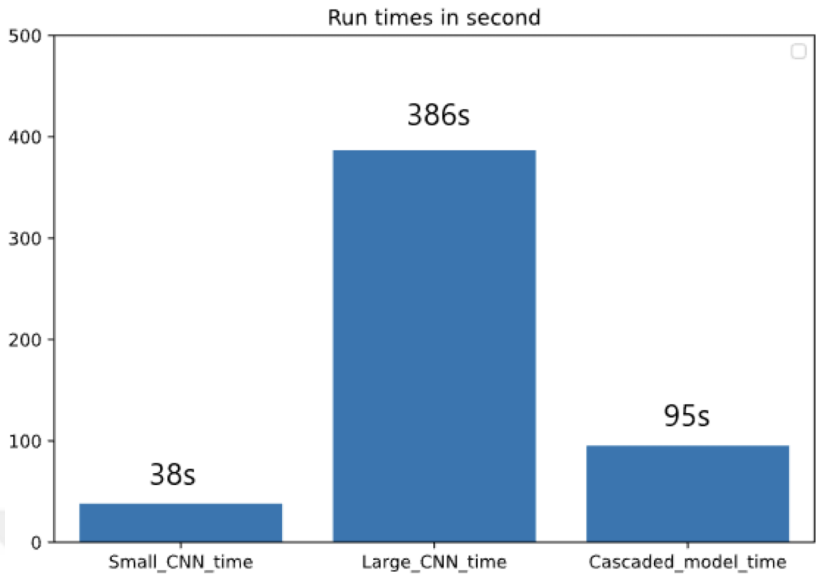

Figure 3.19 Time consumption of Small3, LeNet-5 and cascaded model in EMNIST dataset

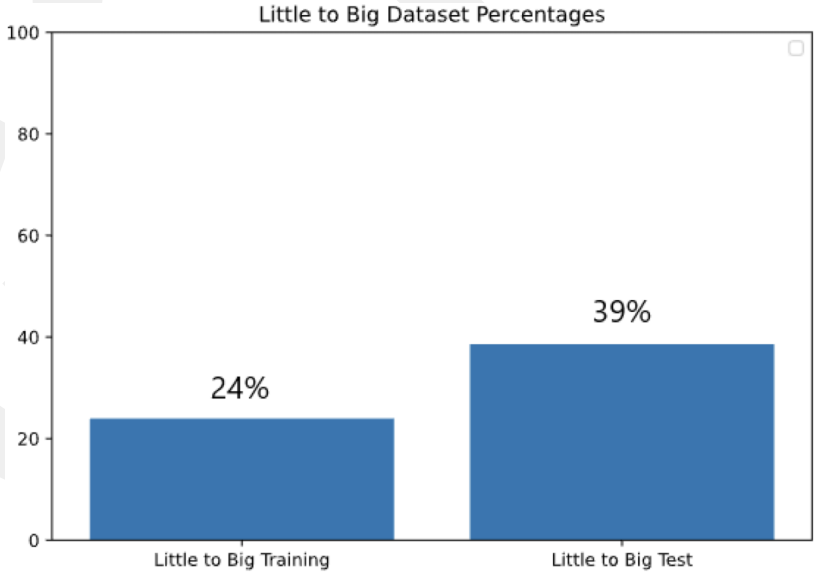

Figure 3.20 Ratio of samples passed from Small3 CNN to big CNN in the cascaded model for EMNIST dataset

3.2.4 Small4 + LeNet5

This section reports on the experimental evaluation of the cascaded network, which utilizes the small4 and big networks described in Tables 2.7 and 2.5 in Chapter 2. The test accuracies of the small CNN, large CNN, and cascaded model obtained using the introduced network selection algorithm are presented in Figure 3.21. In addition to evaluating the models in terms of accuracy, their runtimes were also measured and presented in Figure 3.22, with runtimes of 159 seconds, 386 seconds, and 218 seconds for the small CNN, large CNN, and cascaded model, respectively. The small4 network has the same depth as the small2 network. However, the convolutional layer is configured to use a stride of 3. Even though there is an increment of 0.6% in small module performance, advancement in cascaded model performance is 2.5% compared to the small2 network.

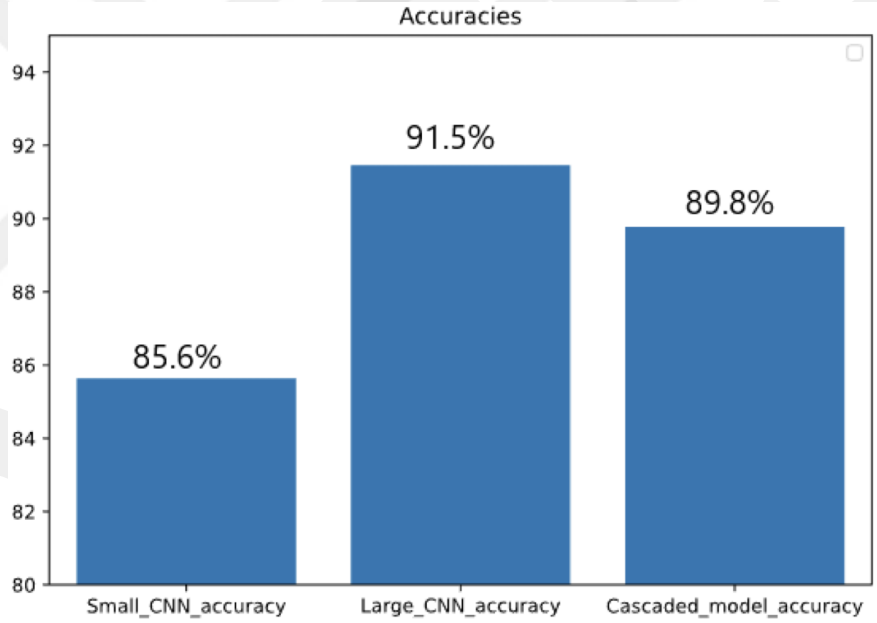

Figure 3.21 Performance of Small4, LeNet-5 and cascaded model in EMNIST dataset

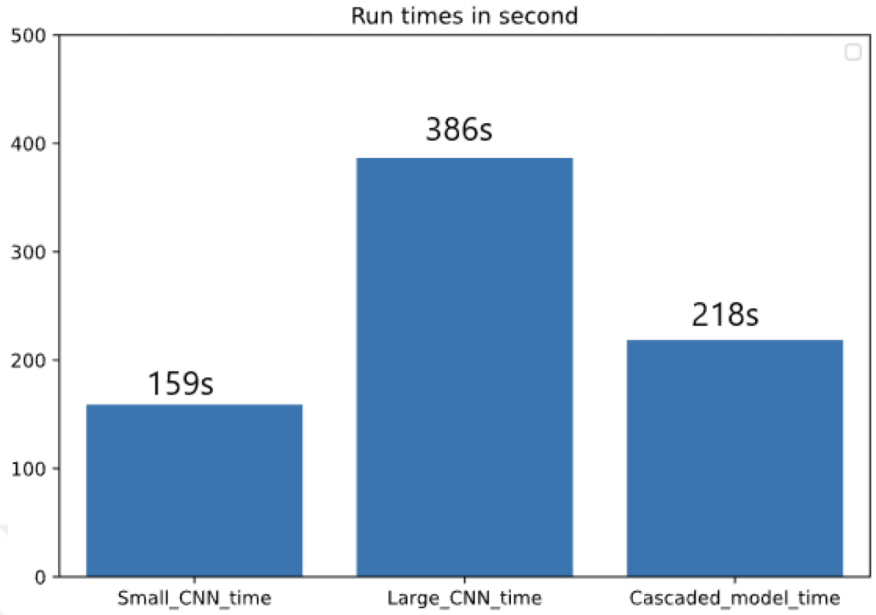

Figure 3.22 Time consumption of Small4, LeNet-5 and cascaded model in EMNIST dataset

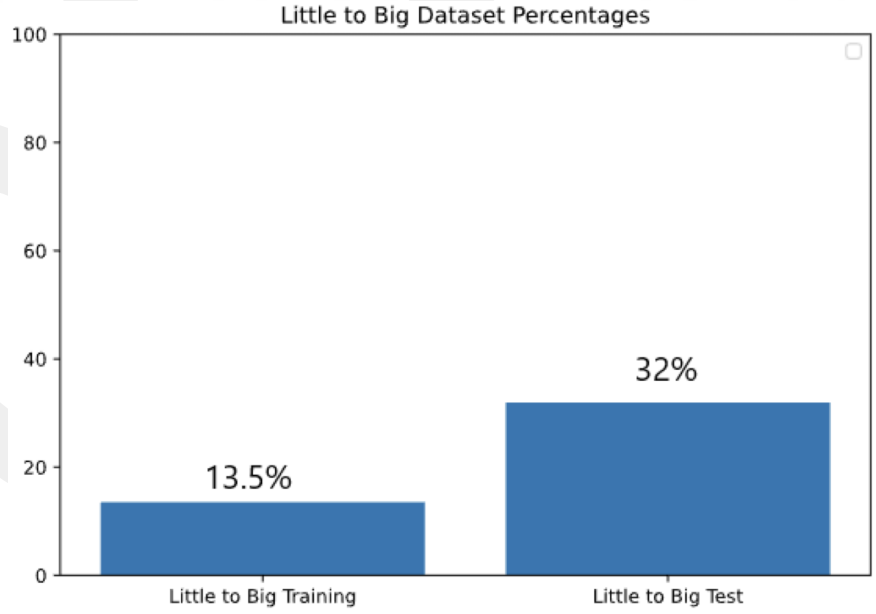

Figure 3.23 Ratio of samples passed from Small4 CNN to big CNN in the cascaded model for EMNIST dataset

3.3 Fashion MNIST Dataset

LeNet-5 model is utilized with hyperparameters optimized in paper [6] and results in 88.91% test accuracy and 92.57% training accuracy with 179 seconds of total run-time (training and test phase run-times are combined). Small network structures utilized in this part are demonstrated in Table 2.10. Test accuracy and run-time of LeNet-5 are employed as benchmark for the assessment of cascaded deep neural network models in the next sections. Performance of the benchmark model is presented in Figure 3.24.

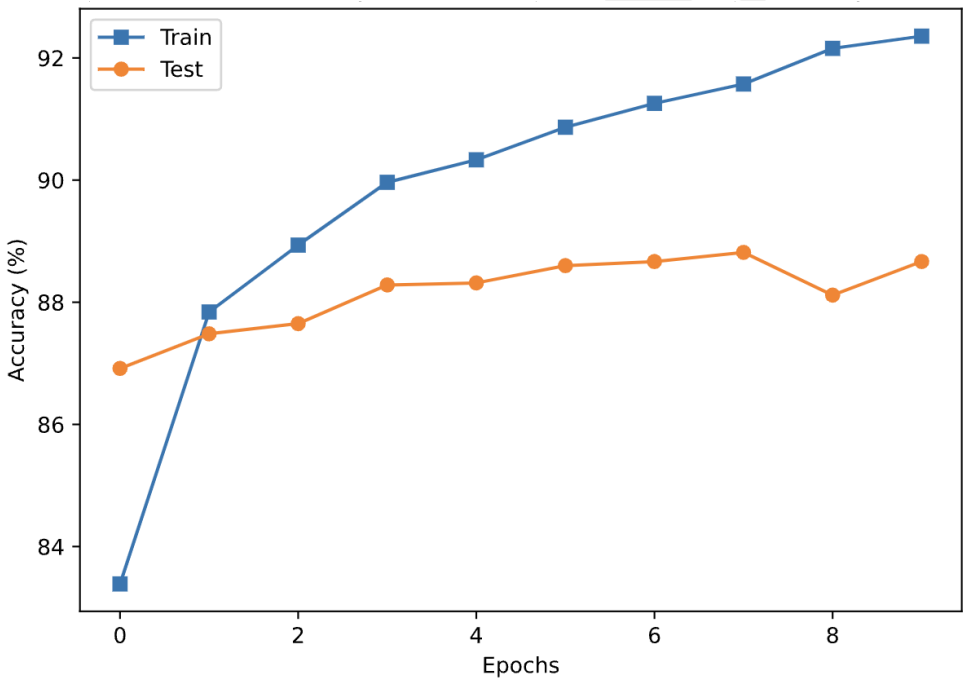

Figure 3.24 Performance of benchmark model in Fashion-MNIST dataset

3.3.1 Small1 + LeNet5

This section reports on an experimental evaluation of the cascaded network, which utilizes both small and big networks described in Tables 2.10 and 2.9 from Chapter 2. Figure 3.25 presents test accuracies for the small CNN, large CNN, and cascaded model obtained using the introduced network selection algorithm, and Figure 3.26 shows their corresponding runtimes. The results reveal that, although the model with the highest accuracy achieved a score of 88.91%, it had a much longer runtime of 179 seconds compared to the other models. Interestingly, the cascaded model exhibits

only a small reduction in accuracy when compared to the benchmark network, yet achieved a significant decrease in runtime. Figures 3.25, 3.26, and 3.27 reveal that around 20% of training data is routed from little to big network is sufficient for the large module of the cascaded system to have a good performance, hence performance of the cascaded system is very similar to the benchmark model. The proposed cascaded model achieved 60% more efficiency while having only 1% less accuracy than the benchmark model. These results highlight the potential of the proposed network selection algorithm to enhance the efficiency of cascaded models while maintaining high accuracy.

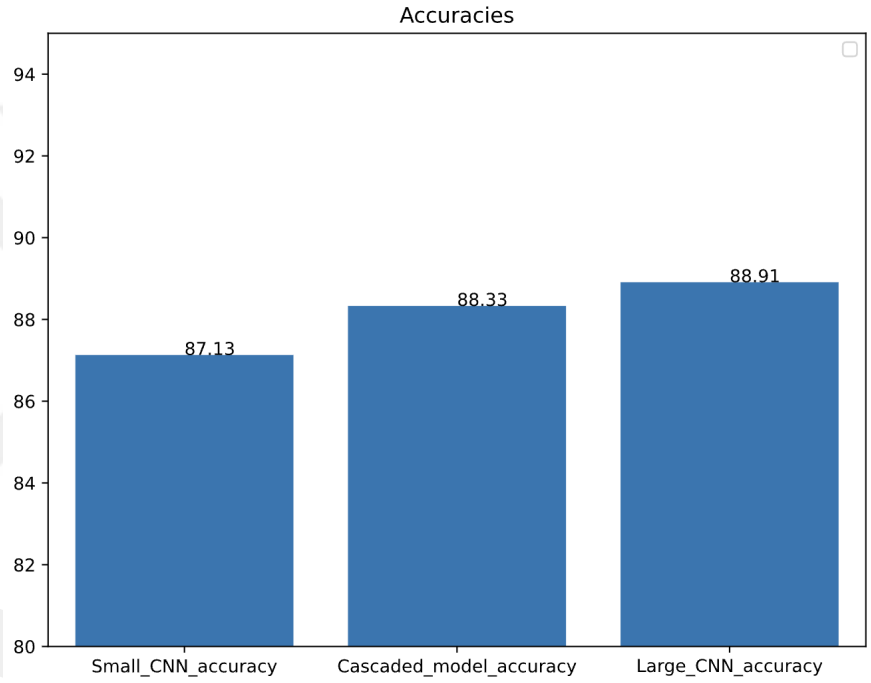

Figure 3.25 Performance of Small1, LeNet-5 and cascaded model in Fashion-MNIST dataset

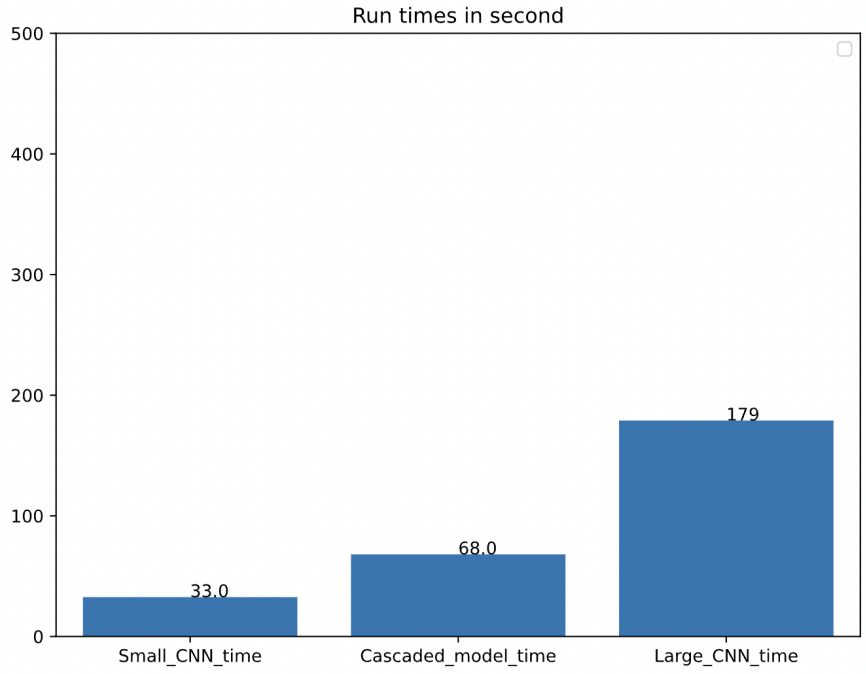

Figure 3.26 Time consumption of Small1, LeNet-5 and cascaded model in Fashion-MNIST dataset

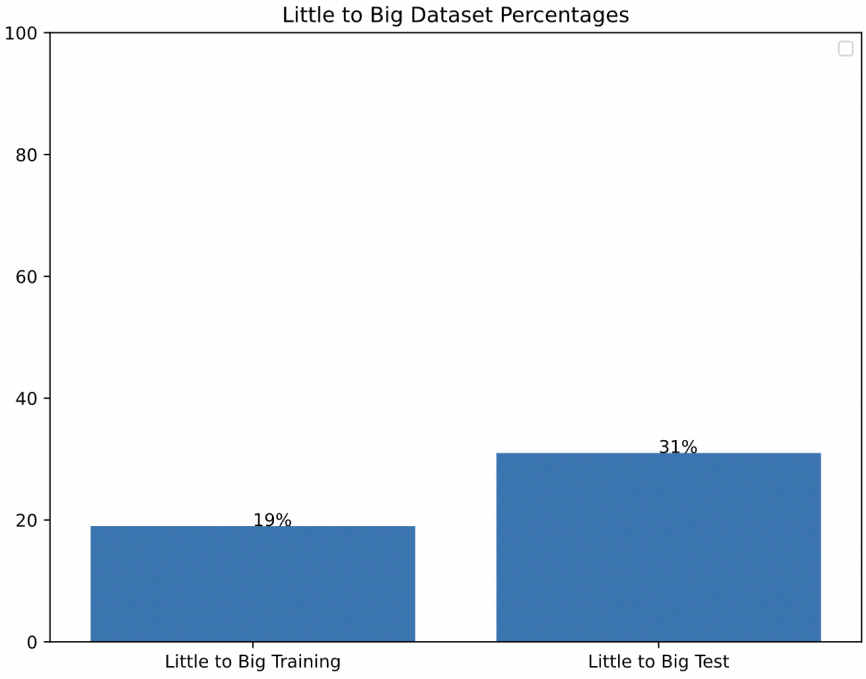

Figure 3.27 Ratio of samples passed from Small1 CNN to big CNN in the cascaded model for Fashion-MNIST

3.3.2 Small2 + LeNet5

This section reports on an experimental evaluation of the cascaded network, which utilizes both small and big networks described in Tables 2.10 and 2.9 from Chapter 2. A smaller CNN model than the Small1 network in the previous subsection is implemented in this section by decreasing the number of filters in the convolutional filter. Reduction of layer width results in a faster little CNN and therefore faster cascaded model. However, the accuracy of the little CNN module is 0.5% higher, hence performance gap between little and big modules is too small and results in a performance decrease for the proposed cascaded model. This proves that if the performance difference between the little CNN module and big CNN module is less than 1.5%, the accuracy of the cascaded model is more close to the small neural network, than the benchmark model. Figure 3.28 presents test accuracies for the small CNN, large CNN, and cascaded model obtained using the introduced network selection algorithm, and Figure 3.29 shows their corresponding runtimes. The findings indicate that even though the benchmark model has 1.4% better performance than the proposed cascaded model, it takes 68% longer to run. It's worth noting that the cascaded model has only a slight decrease in accuracy compared to the benchmark network, but it has a significant decrease in runtime.

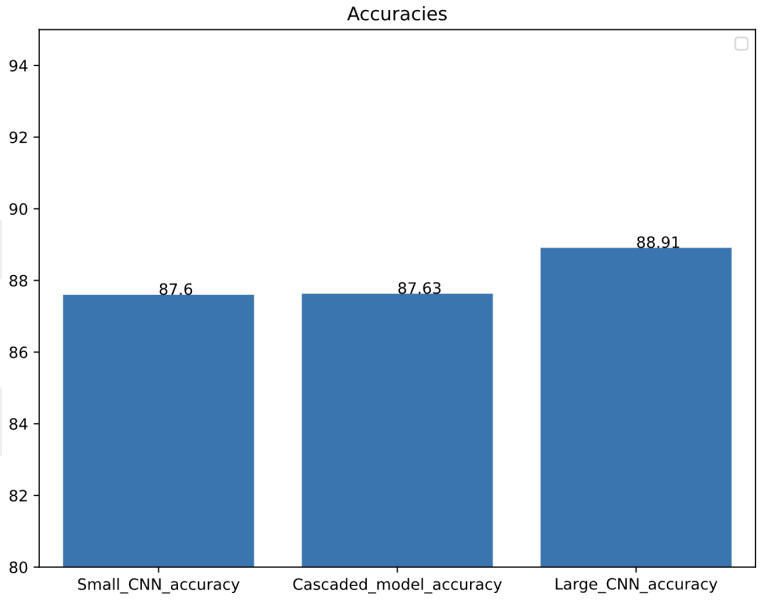

Figure 3.28 Performance of Small2, LeNet-5 and cascaded model in Fashion-MNIST dataset

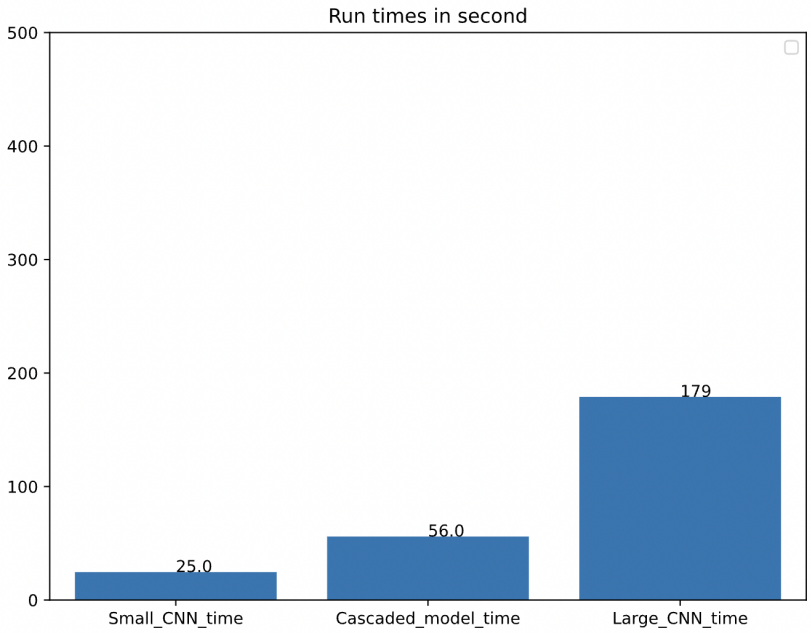

Figure 3.29 Time consumption of Small2, LeNet-5 and cascaded model in Fashion-MNIST dataset

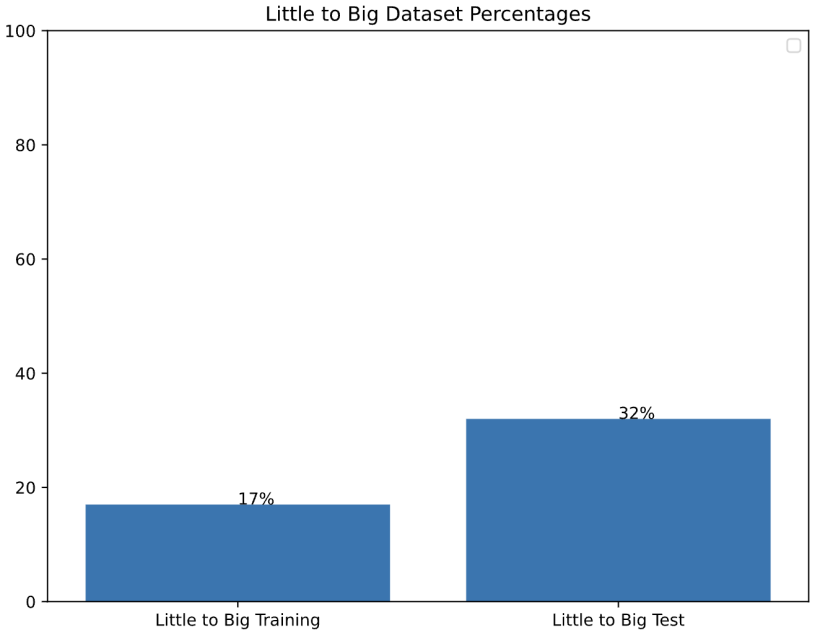

Figure 3.30 Ratio of samples passed from Small2 CNN to big CNN in the cascaded model for Fashion-MNIST

3.3.3 Small3 + LeNet5

This section describes an experiment evaluating a cascaded network that combines a third small network explained in Table 2.10 and a big network explained in Table 2.9. Small3 is the smallest neural network implemented for experimenting in the Fashion MNIST dataset, with a narrower convolutional layer and larger stride value, hence fewer parameters to be trained. This results in lower accuracy of 78% and fast computation time of 13 seconds as demonstrated in Figures 3.31 and 3.32 respectively. Low performance of the small CNN module leads to a higher ratio of samples passed from little CNN to big CNN during both training and test phases which is demonstrated in Figure 3.33. Therefore the cascaded model is more similar to the benchmark model than to the small CNN in the matter of performance; while keeping a higher runtime gap between the benchmark model. The findings indicate that when a very simple and fast neural network with low performance is utilized, the proposed network selection algorithm is still able to enhance the efficiency of cascaded models while maintaining high accuracy.

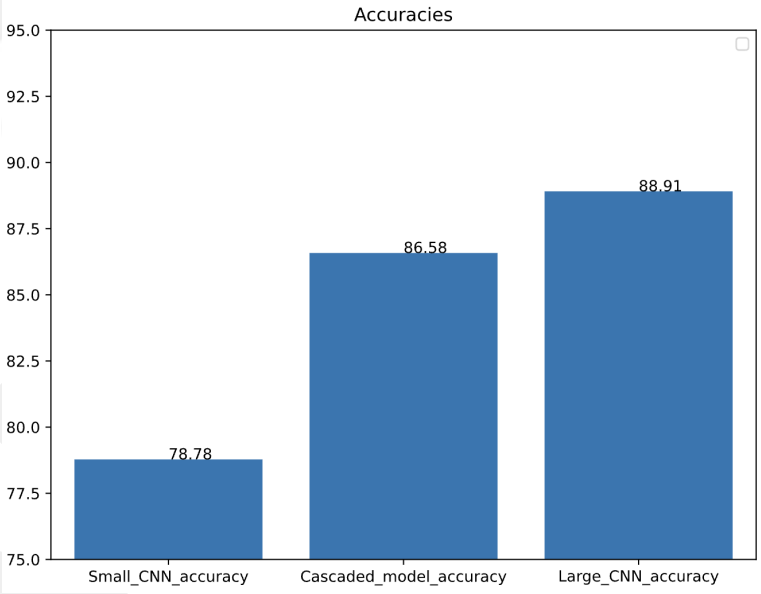

Figure 3.31 Performance of Small3, LeNet-5 and cascaded model in Fashion-MNIST dataset

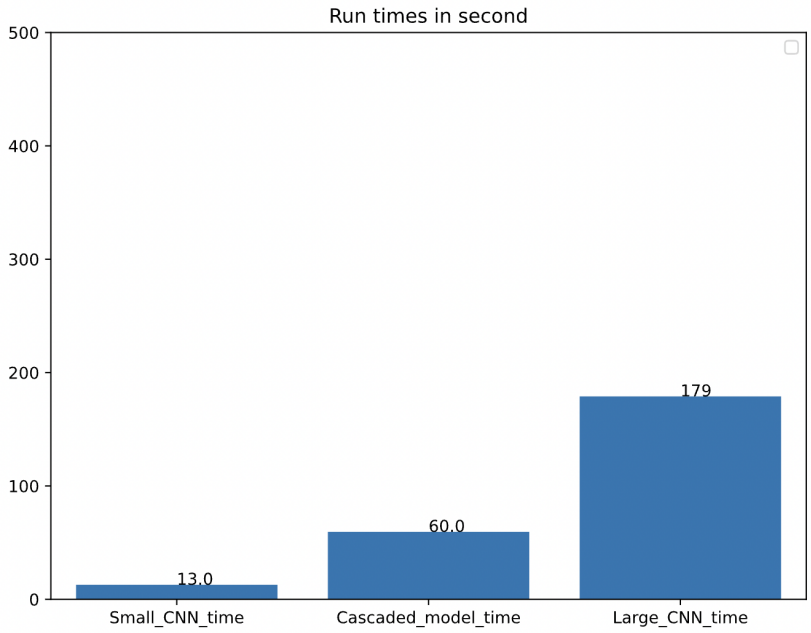

Figure 3.32 Time consumption of Small3, LeNet-5 and cascaded model in Fashion-MNIST dataset

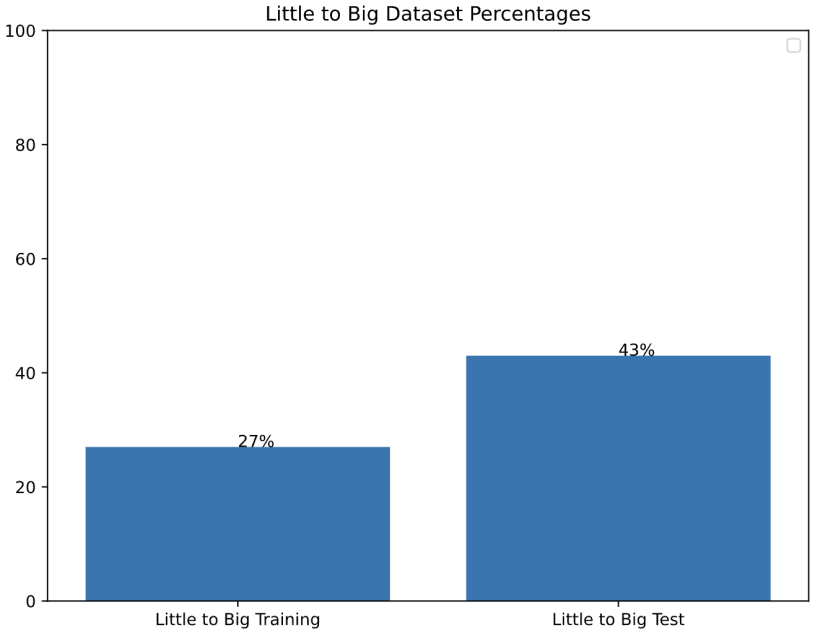

Figure 3.33 Ratio of samples passed from Small3 CNN to big CNN in the cascaded model for Fashion-MNIST

3.4 Discussion of Experimental Results

Numerical results demonstrate that cascaded models with the proposed network selection algorithm outperform benchmark models in matters of efficiency under various scenarios. The comparison between cascaded models employing different small modules indicates that; although their performance and efficiency depend on the small modules, efficiency gain is remarkably larger than performance decrease even in the worst-case scenario. Worst results are observed when the small module of the cascaded model is relatively large, which causes a decrease in both performance and efficiency. Experimentation on different datasets shows that more complex classification tasks can limit the efficiency gain to around 60%. However, performance in the 5% domain of the benchmark model is achievable. The summary of experimental results is demonstrated in Table 3.1.

Table 3.1 Summary of experimental results

MNIST	Acc	Benchmark Acc	Acc Decrease(%)	Runtime	Benchmark Runtime	Efficiency Gain(%)
Small1+ LeNet	97.75	98.5	0.76%	67	421	84%
Small2+ LeNet	96.7	98.5	1.82%	31	421	92%
Small3+ LeNet	94.0	98.5	4.56%	177	421	58%
EMNIST	Acc	Benchmark Acc	Acc Decrease(%)	Runtime	Benchmark Runtime	Efficiency Gain(%)
Small1+ LeNet5	89.51	91.5	2.1%	156	386	59%
Small2+ LeNet5	87.32	91.5	4.5%	279	386	27%
Small3+ LeNet5	89.9	91.5	1.7%	95	386	75%
Small4+ LeNet5	89.8	91.5	1.8%	218	386	43%
FASHION MNIST	Acc	Benchmark Acc	Acc Decrease(%)	Runtime	Benchmark Runtime	Efficiency Gain(%)
Small1+ LeNet5	88.33	88.91	0.6%	68	179	62%
Small2+ LeNet5	87.63	88.91	1.4%	56	179	68%
Small3+ LeNet5	86.58	88.91	2.6%	60	179	66%

CHAPTER 4

CONCLUSION

In this thesis, two essential questions are addressed: is it possible to implement a parameter-free network selection algorithm based on softmax margin for cascaded neural networks, and how this technique can be employed in both training and test phases? In Chapter 2, cascaded neural networks are explained and the definition of softmax margin is given. A detailed description of the proposed technique is provided and how it is adapted to cascaded convolutional neural networks is explained in detail, including the introduction of cross-entropy as a loss function and the derivation of softmax margin for dynamic threshold updates. The chapter concludes with the experimental setup, architectures of little and big networks utilized on every dataset, and a comprehensive description of datasets.

All experimental results are summarized in Table 3.1. It can be observed that the cascaded models are 2 to 4 times faster and only loses 2.6% accuracy at most in every experiment except two worst-case situations. These two worst-case scenarios occur if the small network is redundantly overtrained, and overfitting of the small network causes wrong high confidence, so the router sends fewer samples to the large network, even if it should send more. Even in the worst-case scenarios, the proposed technique achieves only a 4.5% accuracy decrease and a 25 to 50% efficiency gain. However, this situation can be easily fixed with domain knowledge. It is also possible to notice from the experiments that even when the small network has low accuracy, the cascaded system is still able to perform well enough. This situation can be explained by the adaptive nature of the proposed network selection technique. The router is able to notice when the small module is performing poorly and increases the ratio of samples

fed to the large module. Therefore, similar to the benchmark model performance is achieved even if the small network module is chosen poorly.

In conclusion, this thesis proposes a novel network selection technique to be utilized in cascaded deep neural networks and demonstrates its performance and efficiency through numerical simulations on MNIST, EMNIST, and Fashion-MNIST datasets. The results suggest that the proposed model can perform more efficiently than benchmark models without sacrificing accuracy, making it a promising approach for solving similar problems in the future.

The proposed network selection technique is capable of increasing efficiency with only a small decrease in effectiveness and is also able to adjust to different datasets and small network modules. However, the introduced approach is only suitable for batch-processing datasets. It is not possible to directly use this technique in real-time data since the threshold is calculated for a batch of data. As a future study, the proposed network selection algorithm can be improved so that it is also suitable for real-time data.

While the introduced technique is only applied to double cascade systems in the presented thesis, As a future study, the proposed network selection algorithm can be adapted to triple cascade systems. Furthermore, the network selection technique can be generalized to a layer selection technique so that early exiting in a large network can be achieved. This means, instead of utilizing a small and a large network; a large network can be used with varying active layers for creating a single network with dynamic depth. Another research direction can be transforming action-selection algorithms, used for solving reinforcement learning tasks, to utilize for network selection.

REFERENCES

[1] Choi, H., Kim, J., Joe, S. and Gwon, Y., "Evaluation of bert and albert sentence embedding performance on downstream nlp tasks", *International conference on pattern recognition*, pp. 5482-5487, January 2021.

[2] Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. "Bert: Pre-training of deep bidirectional transformers for language understanding",*Proceedings of naacL-HLT*, pp. 4171-4186 ,2019.

[3] Brock, A., De, S., Smith, S.L. and Simonyan, K., "High-performance large-scale image recognition without normalization". *International Conference on Machine Learning* ,pp. 1059-1071, July 2021.

[4] Pfister, T., Simonyan, K., Charles, J. and Zisserman, A. , "Deep convolutional neural networks for efficient pose estimation in gesture videos." *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision* , pp. 538-552, 2014.

[5] Voulodimos, Athanasios, et al. "Deep learning for computer vision: A brief review" *Computational intelligence and neuroscience* , pp. 1-13 ,2018.

[6] Y. Lecun, L. Bottou, "Gradient-based learning applied to document recognition" *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[7] Krizhevsky, A., Sutskever, I. and Hinton, G.E., "Imagenet classification with deep convolutional neural networks". *Communications of the ACM*, pp.84-90, 2017.

[8] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition.*, pp. 770-778, 2016.

[9] C. Szegedy et al., "Going deeper with convolutions." *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9, 2015.

[10] Huang, Gao, et al. "Multi-scale dense networks for resource efficient image classification." *arXiv:1703.09844*, 2017.

[11] Shazeer, Noam, et al. ”Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.”*arXiv:1701.06538*, 2017.

[12] Cubuk, Ekin D., et al. ”Autoaugment: Learning augmentation strategies from data.” *IEEE/CVF conference on computer vision and pattern recognition.*, pp. 113-123, 2019.

[13] Kingma, Diederik P., and Jimmy Ba. ”Adam: A method for stochastic optimization.” *arXiv:1412.6980*, 2014.

[14] Huang, Gao, et al. ”Condensenet: An efficient densenet using learned group convolutions.”*Proceedings of the IEEE conference on computer vision and pattern recognition.*, pp. 2752-2761, 2018.

[15] Hubara, Itay, et al. ”Binarized neural networks.” *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4114-4122. 2016.

[16] Yang, Brandon, et al. ”CondConv: conditionally parameterized convolutions for efficient inference.” *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 2019.

[17] Murata, Akira, et al. ”Selectivity for the shape, size, and orientation of objects for grasping in neurons of monkey parietal area AIP.” *Journal of neurophysiology* pp. 2580-2601. 2000.

[18] Y. Han, G. Huang, S. Song, L. Yang, H. Wang and Y. Wang, ”Dynamic Neural Networks: A Survey” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7436-7456, 1 Nov. 2022.

[19] T. Bolukbasi, ”Adaptive neural networks for efficient inference” *International Conference on Machine Learning*, pp. 527-536, July 2017.

[20] E. Park et al., ”Big/little deep neural network for ultra low power inference” *International Conference on Hardware/Software Codesign and System Synthesis*, pp. 124-132, 2015.

[21] W. Xin et al., ”IDK Cascades: Fast Deep Learning by Learning not to Overthink” *arXiv:1706.00885*, 2017.

[22] Bishop, Christopher M. ”Pattern Recognition and Machine Learning” *Springer*, Vol. 4, No. 4, pp. 1-738, 2006.

[23] Simard, P. Y., Steinkraus, D., Platt, J. C. ”Best practices for convolutional neural networks applied to visual document analysis” *Proceedings of the Seventh International Conference on Document Analysis and Recognition*,pp. 958-962, 2003.

[24] Cireşan, D. C., Meier, U., Gambardella, L. M., Schmidhuber, J.”Deep, big, simple neural nets for handwritten digit recognition” *Neural computation*, pp. 2375-2410 ,2012.

[25] Salakhutdinov, R., Hinton, G. E.” Deep boltzmann machines” *International conference on artificial intelligence and statistics* pp. 448-455, 2009.

[26] Jaderberg, M., Simonyan, K., Zisserman, A. ”Spatial transformer networks” *Advances in neural information processing systems*, pp. 2017-2025, 2015

[27] P. Sermanet and Y. LeCun. ”Traffic Sign Recognition with Multi-Scale Convolutional Networks” *International Joint Conference on Neural Networks* pp. 2809-2813, 2011.

[28] Krizhevsky A, Sutskever I, Hinton GE. ”Imagenet classification with deep convolutional neural networks” *Communications of the ACM* pp.84-90, May 2017

[29] Xiao H, Rasul K, Vollgraf R. ”Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms” *arXiv:1708.07747.*, August 2017

[30] Vijayaraj, A., Raj, V., Jebakumar, R., Gururama Senthilvel, P., Kumar, N., Suresh Kumar, R. and Dhanagopal, R., ”Deep Learning Image Classification for Fashion Design” *Wireless Communications and Mobile Computing*, pp. 71-84, 2022.

[31] Chen, Z., Lv, M., Tian, S. and Yin, S., ”Fashion-MNIST Classification Based on CNN Image Recognition”, *Highlights in Science, Engineering and Technology*,pp.196-202, 2023.

[32] Kadam, S.S., Adamuthe, A.C. and Patil, A.B., ”CNN model for image classification on MNIST and fashion-MNIST dataset”, *Journal of scientific research*, pp.374-384, 2020

[33] Meshkini, K., Platos, J. and Ghassemain, H., ”An analysis of convolutional neural network for fashion images classification (Fashion-MNIST)”, *Proceedings of the Fourth International Scientific Conference*, pp. 85-95, 2020

[34] LEITHARDT, V., ”Classifying garments from fashion-MNIST dataset through CNNs”, *Advances in Science, Technology and Engineering Systems Journal*, pp. 989-994, 2021