Mehran University Research Journal of Engineering and Technology
Vol. 39, No. 3, 583 - 594, July 2020
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: 10.22581/muet1982.2003.13

A Multi-blocked Image Classifier for Deep Learning

Nawaz Muhammad Hamza Shah[1a], Muhammad Junaid[1b], Nawab Muhammad Faseeh[1c], Dong Ryeol Shin[1d]

RECEIVED ON 23.09.2019, ACCEPTED ON 28.11.2019

ABSTRACT

Convolutional Neural Networks (CNN) have been very successful in classification and object recognition. A lot of related work has been done to modify the performance of the networks, but they could either perform better with accuracy at high cost or decrease the time taken by a model on training datasets. We propose a deep neural network model 'Multi-Blocked Model' which tends to decrease this gap and is efficient and accurate with less convolutional layers, easy to deploy online or embedded systems. We choose three datasets that are publicly available and popular for their own uniqueness among the datasets in deep neural networks. These three diverse datasets are: Modified National Institute of Standards and Technology (MNIST), Street View House Number (SVHN) and the Canadian Institute for Advanced Research with 10 Cases (CIFAR-10). Our state-of-the-art Multi-Blocked model is presented well on all three data sets. Dropout is added to overcome the overfitting problem. The multi-blocked model is designed in a way that it uses a minimum number of parameters so that it is able to run on a Graphical Processing Unit (GPU), which requires less power. The experimental results show that our proposed Multi-Blocked model tends to achieve the accuracy of these datasets by 99.40%, 90.8%, 88.07% consuming under 2 GB of graphical memory.

Keywords: **Deep Neural Networks, Image Classification, MNIST, SVHN, CIFAR-10, Convolutional Neural Networks, Multi-Class Image Classification.**

1. INTRODUCTION

In the modern world, technology has made lives easy and there is a massive success in the pursuit of technology. It revolutionized the world and we are trying to make it better. Neural networks have contributed to computer science in a way that has never happened before. They have become backbone in computer science [1]. Almost every field of computer science is using them and enlarging their circle to science fields. Because they are the need for the future. So, we contribute to our future.

Due to the extreme performance of computing systems, especially, GPUs and distributed clusters and thanks to the availability of large public repositories like ImageNet [2], we can use their services and big data for research areas. ImageNet and Kaggle Challenges have served as a platform for large scale image detection, classification, segmentation, *etc.* which [3] led to the improvement of deep learning architectures. Deep Neural Networks (DNNs) consist of neurons that have learnable weights [4]. Experts define deep learning as a feedforward network [5] that has an input layer and output layer in which data flows without back looping. In the beginning, DNNs create virtual neurons and set weights for them. The weights and inputs are calculated and return an output from 0-1. In case of a network does not recognize a particular pattern accurately, then an algorithm would fix and adjust these weights. In image recognition our chosen

[1] Sungkyunkwan University, South Korea. Email:[a]mhamza@skku.edu,(Corresponding Author), [b]mjunaid@skku.edu, [c]faseeh@skku.edu, [d]drshin@skku.edu

datasets: MNIST, SVHN and CIFAR-10 [6] have been widely used for testing and training of the proposed models. State of the Art pre-trained models like Visual Geometry Groups VGG16, VGG19, Exception, *etc*., are the models which are publicly available by Keras, a library used in Python for deep learning [7]. They are very good models usually used for fine-tuning and transfer learning but the problem with the models is that they are very big in size and very complex and are trained on 1000 different classes. Because of the model's size, the training time is long. Considering this problem MobileNet (machine learning model) was introduced. Its size is 2MB in total and less complex but it is not stable compared to the exception model. We realize a need for a DNN which covers all these problems in a sophisticated manner.

Our work focuses on image classification in deep neural network/deep learning. We took 3 datasets namely MNIST [8], SVHN and CIFAR-10. CIFAR has other datasets as well with more classes up-to 100 classes but in our case, we are only dealing with CIFAR-10.

We propose an image classification 'Multi-blocked' model with a minimum number of parameters. This model is suitable for embedded systems because we reduce the complexity and size of Multi-blocked than state of the art model like VGG16/18 or exception model etc. So the Multi-blocked model consumes less GPU memory. We implemented it successfully and we found that it is suitable and stable for all these datasets mentioned above. The model stays the same but to deal with the overfitting or underfitting, we added the Dropout. With the SVHN dataset we added '543,000' more pictures, we call it SVHN (EXTRA). These pictures are taken from the same public archive. We investigated whether given more data would increase the accuracy or loss. We got our results, shown in the 'Results and Discussion' section. The accuracy and loss are mentioned in the experiments section.

All the datasets were run on NVidia GTX 970 GPU which took only 2 GB memory to process all 3 datasets. Training time with all the datasets is dependent on the number of images used for training. Our main contributions in this research are summarized as follows:

- We confronted a classification problem with minimum numbers of training parameters.
- Improved accuracy in digits classification problems like MNIST and SVHN datasets.
- Compared the original SVHN dataset with SVHN (Extra) dataset.
- Improved accuracy in object detection problem i.e, CIFAR-10 dataset.
- Compared results with the difference between the global average pooling (GAP) and max pooling (MP).
- A multi-blocked model is universal for diversity of datasets like digits with grayscale or colored (RBG) images with different classes to deal with Classification or detection problems.
- Quantitative performance is satisfying.
- A multi-blocked model can work stably on low graphical power systems.

This paper is organized into 6 sections. In section 2, we explain the related work. The datasets are discussed in section 3. Section 4 provides information about the proposed model and data preparation information. Experimental setup for training and testing of the Multi-blocked model is presented in section 5. Results and discussion are presented in section 6.

2. RELATED WORK

Image classification is a hot topic of research and it is used in studies for a wide range of applications, to improve the accuracy and functionality of the Artificial neural networks. In the convolutional neural network, multiple approaches [10-12] have been used to optimize the results.

Calic *et al.* in [13], described the evaluation of CIFAR-10 classification using CNN and proposed the image processing under a limited amount of memory using the CNN algorithm. He made a model which is first optimized by considering training and evaluation. It has 4 convolutional layers and 2 fully connected paths. The learning rate is 0.1 and 10% in every 350 epochs.

Garg *et al.* [14] used house number digital classification using CNN model features. They augmented the traditional CNN architecture for a

multistage feature by using Lp pooling. This algorithm has extraction stages of feature extraction and two-layer non-linear classification. First, the convolution layer produced 16 features with 5×5 convolutional filters and 2nd layer's output is 512 features with 7×7 filters. It has 20 hidden units.

MNIST data set is used by Mohapatra *et al.* [15]. In their paper, they performed the analysis for recognition of handwritten character using ANN which involved back propagation neural network. The 400 DCST (Discrete Cosine S-Transform) vector was extracted from and saved in the feature matrix, and the size was 60000×400.

Guo *et al*. [16] performed image classification in a simple neural network on MNIST and CIFAR-10 data sets, together. They used different learning rates, and different optimization algorithms and also optimal parameters that influence the image classification. It also includes Relu which is a linear correlation and purlin's polyline and dropout. The model was trained on random network hidden nodes.

In contrast to the above work, we made it as simple as possible, starting with the convolutional layer then normalizing it. Then applying optimizer and then pooling to it. We used 3 blocks of these layers and in the end, we applied Softmax on output.

3. DATASETS

3.1 MNIST Data Set

The MNIST dataset is a widely used dataset of handwritten digits as it is shown in Fig.1**.** It is commonly used for the training of various image processing models. The MNIST dataset contains 60,000 [17] training images and 10,000 testing images. Dataset 3 (SD-3) and special dataset 1 (SD-1) are MNIST's special databases. It contains greyscale images of size 28×28 and each image vector is 784. The sample images are shown in Fig. 1. MNIST is publicly available.

3.2 SVHN Data Set

The SVHN dataset is a real-world dataset for image processing and machine learning. The advantage of this dataset is that it requires minimum data processing and formatting. Its data nature is similar to MNIST. It contains [18] label data of 60,000 digital images and 10 classes, representing from 0-9 digits. It has two formats in which first has its original images with bounding boxes while second is MNIST 32×32 images centering a single-digit shown in Fig. 2. Testing and training split are 26,032 and 73,257 respectively. However, 531,131 additional images are also available as extra training data.

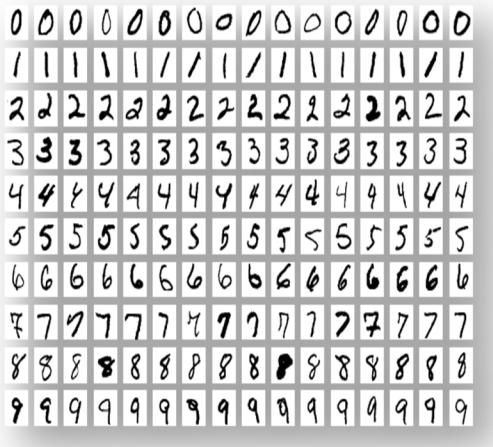

FIG 1: MNIST DATASET

3.3 CIFAR-10

The CIFAR-10 is a group of images that are widely used for Computer vision and Machine learning. It is very well documented [19] dataset. More apparently it is drawn from the large tiny images fine-grained labels (Fig.3). It includes10 classes 5,000 images in each class. The 10 classes represent [20] birds, cars, deer, dog, horses, frogs, trucks, and ships. The dimension of the images is 32×32. The testing and training data are equally split and contain 50,000 training images. The datasets we choose are different but there are few similarities and not entirely different that we have to consider i.e. MNIST and SVHN datasets are on digits from 0-9. But SVHN are colored images and include much more noise. Similarly, SVHN and CIFAR-10 datasets are both colored images, but the level of noise is much higher with CIFAR-10 because it includes entirely different classes and the dimension of both datasets are the same. All three datasets have 10

classes but different content. A summary of these datasets is presented in **Table 3**.

Fig. 2: SVHN DATASET

FIG. 3: CIFAR-10 DATASET

4. PROPOSED MODEL

4.1 Proposed Architecture

We propose a Multi-Blocked Model which is simple and efficient with a minimum number of parameters. The proposed state of the art model performs well and is efficient with all the three datasets we used. It achieves the accuracy higher than [13-16].

TABLE 1: SUMMARY OF THE DATASETS

Data Sets	Training Set Images	Testing Set Images	Classes	Dimension
SVHN	60,000	10,000	10	28×28
SVHN	73,257	26,032	10	32×32
SVHN (Extra Data)	462,083	73,257	10	32×32
CIFAR-10	50,000	10,000	10	32×32

The Multi-blocked Model is consisted of Input layer, Convolutional layers, Batch Normalization layer, Relu, Average Pooling layer, and Softmax. The architecture is shown in Fig. 4. We used 4 block CNN. The first 3 blocks are convolutional blocks, which contain 2 convolutional layers with activation Relu. Before every Relu activation function, the batch normalization layer is used. After that average pooling layer is used for dimension reduction and the last 4th block contains fully connected layers. At the end of every block, we used dropout 0.50, 0.30, 0.30, 0.10 respectively to overcome overfitting problem If needed. we only used dropout in MNIST and SVHN dataset, except the CIFAR-10 dataset. Because we did not face the overfitting problem in CIFAR-10. Below we discuss essential features of the architecture.

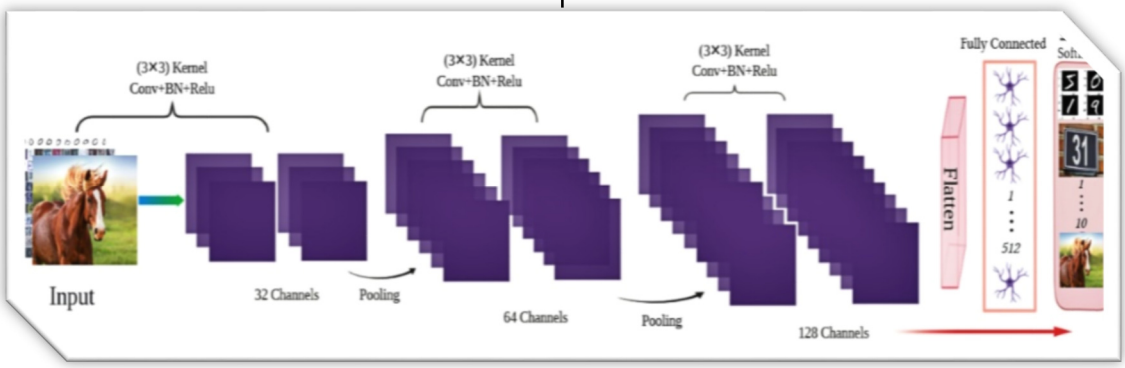

FIG. 4: ARCHITECTURE OF MULTI-BLOCKED MODEL WHERE 3×3 IS KERNEL SIZE, CONV IS CONVOLUTIONAL LAYER, BN IS BATCH NORMALIZATION, RELU IS AN ACTIVATION FUNCTION AND POOLING IS AVERAGE POOLING

4.2 Terminology

1) Convolutional Neural Network:
Convolutional Neural Network's (CNN) hidden layers are called convolutional layers. DNNs are very similar to the CNNs that any network with more than 2 hidden layers is DNN. The dataset used for training the DNN consists of 10 different classes (objects) of images. Pixels of these images form a pattern of rows and columns. Rearranging of this pattern [21-23] and only detecting the local spatial pattern is the job of convolutional layers. Convolutional nets are best to find a nice pattern to classify the images. With each layer, we are required to lay down the number of filters. Actually, these filters are able to detect the patterns [24-25] (shapes, textures, edges, corners, circles, squares, etc). The deeper is the network, the more sophisticated it is [26]. After a few layers, instead of textures and shapes, the filters might detect a particular object that we wanted it to predict. Fully connected (FC) layers connect each neuron in one layer to every neuron in another layer.

2) Activation Function:
In CNN, the activation function states the output of a neuron i.e. sigmoid is an activation function that takes a number as an input if given input is [27] a very negative number then sigmoid will transmute the number close to 0. If the input is a very positive number, then sigmoid will transmute that number close to 1. And if the number is close to 0 or 1 then sigmoid will transform that number between 0-1.

An activation function is biologically inspired by activities happening in our brain where neurons are activated or fired, by different stimuli. For example, if we feel warm then some neurons will fire in our brain, but others won't. This can be represented by '0' for not firing and '1' for firing [28]. If given input is closer to '1' means that the more activated that particular neuron is and the more close to '0', the less activated that particular neuron is. Rectified linear unit (Relu) is an activation function as well that transforms an input to a maximum of either 0 or an input itself, results in more positive the number is the more active the neuron is [29].

It computes the function:

$$f(x) = \max(0, x) \quad (1)$$

In other words, the activation is simply a threshold at 0. There are many activation functions that do different types of transformations.

3) Batch Normalization:
Batch Normalization transforms the data to put all the data points on the same scale. The normalization process includes the scaling of a numerical dataset stepping down to between 0 to 1. We might have numerical data points in our datasets which can be very high or very low. Large data points of non-normalized datasets can result in unpredictability and might be harder to train networks because the relatively large inputs can torrent down via network which can cause unmatched gradients and that could lead to famous exploding gradient problem and can lead to decrease in training speed.

We put all our data on the same scale in order to avoid all problems discussed as well as increase our training speed. But with normalized data, we can face a problem [30, 31] that our weights of a neuron from Convolutional layers can become radically larger than the other weights. If so, then these neurons will be the source of the output from its equivalent neuron to be drastically large. So, this imbalance/gap will again continue to torrent through the DNN causing unsteadiness. Batch norm becomes handy here and applied to the layers we choose to apply to in the network as,

$$0 = y - m / s \quad (2)$$

where, 'm' is the mean, 's' is the standard deviation.

Multiply the normalized output with an arbitrary parameter 'g'.

$$x \times g \quad (3)$$

Add another arbitrary parameter 'b' to follow-on products. In this process, weights in the network do not become imbalance with drastically high or low values, since the normalization is added.

$$(x \times g) + b \quad (4)$$

These parameters are all trainable, it means that 'g' and 'b' will be optimized during the training process. This additional batch norm process can increase the

training speed and reduce the ability of weights that will impact the training process.

4) Pooling:

The pooling layer reduces the dimension of the data by combining the outputs into a single neuron in the next layer. This maneuver is classically added to DNNs following to individual convolutional layers. When it is applied to a model, pooling cuts the dimensionality of images by pruning the number of pixels. Max, Global Max, Average, and Global Avg Pooling are famous pooling techniques [32] these days. We state some n × n region as an equivalent filter for the max-pooling maneuver. Max pooling gets the n × n region and calculates the extreme value of each block and Average pooling takes the middling of n × n values of each block. i.e. If the convolutional layer is 26 × 26 in size then after pooling the dimension will decrease by the vector of 'n'.

5) Loss Function:

The loss function is what SGD (Stochastic Gradient Descent) is attempting to minimize by continuously updating the weights in the model during training. During the training process, after every epoch, the loss will be calculated on the model's predictions. Basically, what is happening is that the model is calculating the error at every input by observing the output it predicted for that particular input and considering the change of that output value for the precise label for that specific input [33]. For example, let the network is classifying images on horses and deers. Let us assume that the label for horse and deer is 0 and 1 respectively. If we pass an image of the horse and the model's output value for the horse is 0.25, then the error between the model's output and true label for the horse's image is:

$$0.25 - 0 = 0.25 \tag{5}$$

So, it does this process for every single input. Then towards the end of each epoch, it will separately evaluate all the errors for each epoch and then carries it on to a loss function. We use categorical Cross-Entropy (CE) loss function for multi-class classification which is defined as:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^c e^{sj}} \tag{6}$$

$$CE = -\sum_i^C t_i \log(\int (s)_i) \tag{7}$$

When we use loss function, we train the network to get the output probability for each image. It is also known as Softmax loss because softmax activation is added to cross-entropy loss. After calculating the loss function that occurs latterly of each epoch during the training process, the value of the loss would be constantly varying, because the weights of the model are constantly changing. The target is to decrease the loss function. Thus, the loss will be minimized as we run more epochs.

6) Over/Under Fitting:

Overfitting happens when a particular model is excellent at predicting a dataset that was used in the training procedure but not as good on test dataset. We can check this difference between accuracy and validation accuracy, at the end of each single epoch. If the validation accuracy and loss are considerably worse than the training matrix then it is an indication of the model's overfitting. The concept of overfitting [24] boils down to the fact that the model is unable to generalize well. It learns all the features well enough on the data on which it has trained on but if given data is slightly deviating from the training data, it is unable to predict the output accurately. It is a very common issue Here are a few steps, we can do to overcome it:

1. Provide more dataset if possible. The more the data, the more the model will learn from the images and also adding more diversity with more data.
2. Data augmentation, with this we can modify our data in our training set. We can do these modifications by rotating, flipping, cropping, and zooming.
3. we can reduce the complexity by changing the number of layers or reducing the number of neurons. This might be able to generalize well.
4. Dropout can also be changed accordingly to reduce overfitting.

The basic idea of dropout is that it randomly drops some subset of nodes given in the convolutional layer during the training process. Thus, called dropout. By

dropout, it prevents the dropout nodes from participating in the prediction on the data. This also helps a lot to generalize well enough.

Underfitting refers to a model that can neither generalize training data nor test data. The model with underfitting is not a good model and has poor performance. One should keep trying for the alternative CNN algorithms. Nevertheless, it provides contrast to a problem of the model's overfitting.

4.3 Data Preparation

MNIST data set have 70,000 images in which there are 60,000 training and 10,000 testing pictures. This gives us approximately 86% training and 14% testing pictures. Then we cropped and re-sized all the images to maximize the performance and efficiency. Because we can reduce the noise by cutting out extra pixels. It also can refer to the data augmentation process, where it creates multiple images i.e. flip, rotate, etc. But with this technique, we need more processing power and memory space. For the SVHN data set, there are 73257 (74.78%) images in training and 26032 (26.22%) images in the test set. We converted it from (width, height, channels, size) into (size, width, height, channels) and labelled the data from 0 to 9 as preprocessing steps. We also added 531,131 more images of the SVHN dataset as a training set which is discussed before. The preprocessing steps are the same. For CIFAR- 10 dataset, the total number of the training set is 50,000 (83.3%) images of total data and 10,000 (16.7%) images are in the test set. We converted class vectors into binary class matrices as a data pre-processing step.

In order to improve the efficiency of the model, we implemented standard preprocessing steps. But the nature of the datasets are different. If the dataset is mature enough then we can skip some specific preprocessing steps as, in case of MNIST data set, it contains grayscale images, cropping and resizing is beneficial in pre-processing but in case of SVHN data set, it holds colorful images of digits which are pre-cropped. In the case of CIFAR-10 dataset, it is also RGB but the nature of the classes is very diverse as discussed in section III-C, so that is why cropping is not productive and can be misleading to errors.

5. EXPERIMENTAL SETUP

We imported all the necessary libraries. Uploaded the dataset into the proposed model of Fig. 4, and compiled using Adam optimizer because it is specifically designed for deep learning [36]. Adam algorithm powers the adaptable learning rates to discover distinct learning rates for every parameter. We set the learning rate of the Adam optimizer as 0.0005. We used categorical cross-entropy loss function and defined accuracy in metrics. Checkpoints have also been added for best weights. Finally, we fit on X and Y train with 100 to 150 epochs and set the batch size to 64. Also, set the validation data to the x and y test. For these experiments, our setup is based on the Intel i5 3.5 quad-core CPU with 24 Gigabyte of memory and Nvidia GeForce GTX 970 GPU.

6. RESULTS AND DISCUSSION

6.1 Overview

All the datasets ran, took less than 2 GB memory of the GPU that is discussed in [13]. Results show that the MNIST dataset is highly accurate with 99.4% which is followed by SVHN with 90.8% and then it is followed by CIFAR-10 with 88.07%. We experienced that this decreased accuracy is because of the level of noise in the data and fewer dimensions of the images.

We compared Multi-blocked model with [13-16] and got the better results. We mention the results of the papers which used these datasets, for the MNIST data set, [14-16] achieved the accuracy of 98.45% 98.8% and 98.9%. For the CIFAR-10 dataset, [13] achieved an accuracy of 85.9%.

Our experimental results shown in Table 2 indicate that the SVHN data set was not stable enough and we faced the overfitting problem. So, we tried to overcome this problem by adding similar extra images. We experienced a significant increase in the accuracy and stability from 90.8% to 97.69%.

The model's accuracy of MNIST and CIFAR-10 is shown in Fig.5 and their loss is shown in Fig.6 respectively.

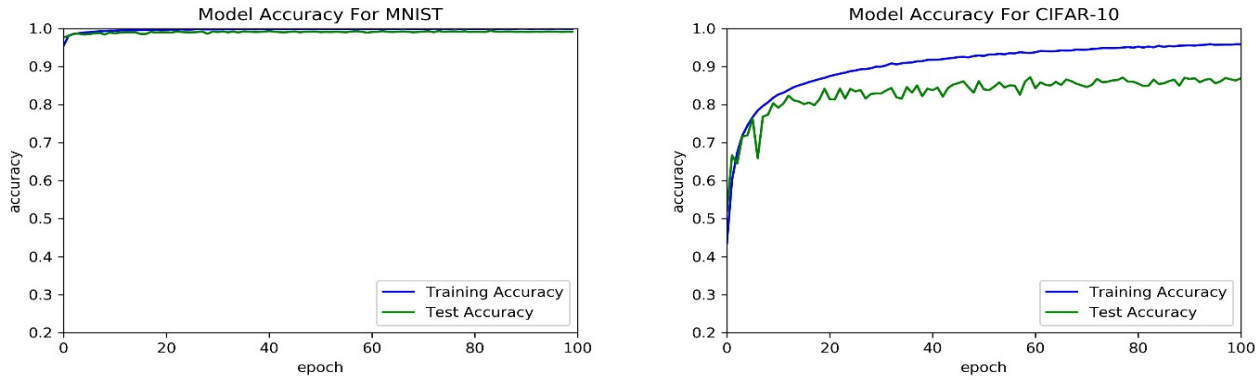

Fig. 5: Accuracy of MNIST AND CIFAR-10 Data Sets

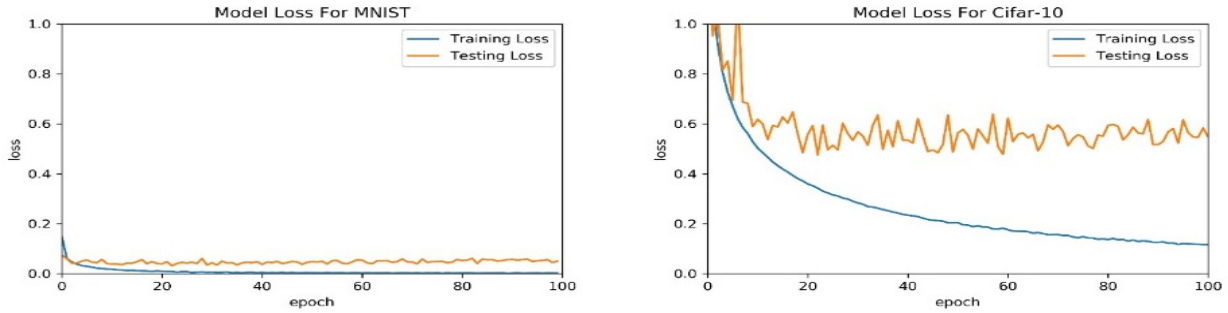

Fig. 6: Loss of MNIST AND CIFAR-10 Data Sets

TABLE 2: SUMMARY OF THE PERFORMNCE OF MULTI-BLOCKED MODEL

Data Sets	MNIT	SVHN	SVHN (Extra)	CIFAR-10
Accuracies	99.40%	90.80%	97.69%	88.07%
Loss	3.17%	26.35%	27.16%	47.56%
Training Time in Sec	46	46	46	46

TABLE 3: COMPARISON

MINIST DATA SET	
Reference	Accuracies
[14]	98.45%
[15]	98.80%
[16]	98.40%
CIFAR-10 DATA SET	
[13]	85.90%
Multi-Blocked Model	88.07%

We have shown this experimental graph in Fig. 7**.** For CIFAR-10, we are not certain what is causing less accuracy if we compare it to the state-of-the-art models with a gap of 8% [1, 37, 38, 39]. But if we try to put some complexity, increase the number of learnable parameters, enhancing with dropout in the convolutional layers, or using data augmentation on dataset, then it is most likely to get better results in accuracy and minimize the loss of CIFAR-10. But we preferred to use the same model for three datasets and not to change it to more complex models, which will also result in acquiring more powerful GPUs and computational systems [34]. We also observed that the gap of classification performance of actual training set in contract with testing datasets are around 5% to 7%.

As discussed in the proposed model (section 4), we used Average pooling in the Multi-Blocked Model, but we also experimented with Average pooling with Max pooling.

Our results shown in Table 4 that Average pooling is slightly better than Max pooling. Max pooling lacks behind for the dataset we choose for the accuracy but as far as loss is concerned Max pooling performed slightly better than Average pooling only for MNIST and CIFAR-10 datasets with 3.01% and 44.91% respectively. Fig. 8 represents the graphs, where [35] we brought all the datasets in one graph. The goal is to minimize the loss as much as possible. Between these two different pooling methods, the average training time for the model remained constant. Our results confirm that the overall performance of Average pooling is better than Max Pooling and is preferable to use for specifically these datasets.

The accuracy improved because of the Multi-Blocked model's architecture. Less number of convolutional layers increase its efficiency. If we include the number of convolutional layers, it may increase the accuracy but will definitely increase the complexity of the model that results in time-consuming and requires more graphical processing power. When we reduce the number of layers in the model, it produces less number of training features, thus consuming less power of the GPU.

TABLE 4: SUMMARY OF MULTI-BLOCKED MODEL'S PERFORMANCE WITH MAX POOLING

Data Sets	MNIST	SVHN	SVHN (Extra)	CIFAR-10
Accuracies	99.31%	90.21%	97.68%	87.88%
Loss	3.01%	139.89%	28.58%	44.91%
Timing Time in seconds	10	19	140	46

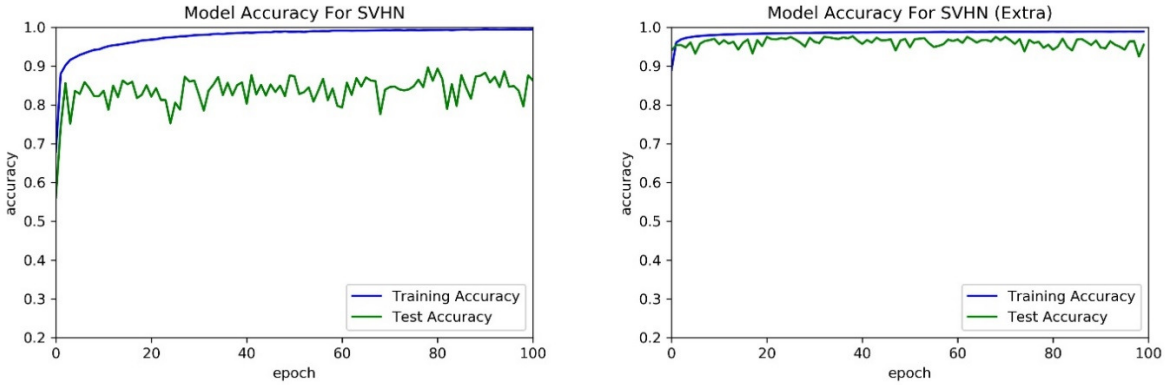

FIG. 7: ACCURACY OF SVHN AND SVHN (EXTRA) DATA SETS

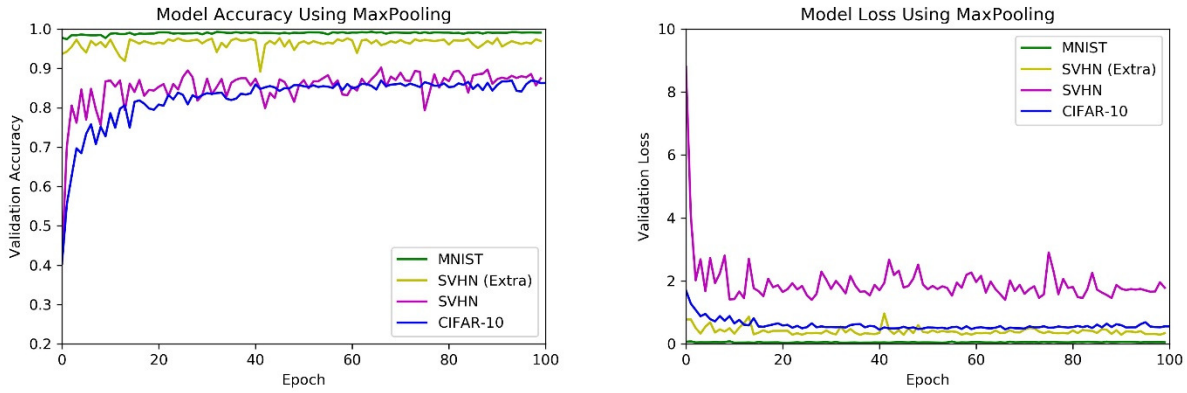

FIG. 8: ACCURACIES AND LOSSES OF ALL THE 4 DATA SETS COLLECTIVELY.

7. CONCLUSION

We have worked on image classification using three datasets, namely MNIST, SVHN, and CIFAR-10. We successfully implemented our purposed model 'Multi-blocked' on these datasets and observed better results. In the light of our results, Adam optimizer performs better than stochastic gradient descent and Average pooling seems to have slightly worked better than Max average pooling for our data sets. In all three datasets, MNIST got the best accuracy and lowest loss followed by SVHN and then CIFAR-10. Due to the fact that the level of noise increased. As we include more data in the SVHN data set, it performed dramatically better. In our opinion, the bounding boxing (that covers only the object in a whole picture) and image segmentation would have performed better than given images as an input to a model. In the future, we aim to work on the bounding boxes and image segmentation for image classification and detection tasks and make stronger and lighter systems that do not have enough computational power.

ACKNOWLEDGMENT

The authors acknowledge the support of Sungkyunkwan University, South Korea to carryout this research.

REFERENCES

[1] Sultana F., Sufian A., and Dutta P., "Advancements in image classification using convolutional neural network", Proceedings of the Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), pages 122–129. IEEE, 2018.

[2] Deng J., Dong W., Socher R., Li L-J, Li K., and Li F.F., "Imagenet: A large-scale hierarchical image database", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 248–255, IEEE, 2009.

[3] M. Shafiq, X. Yu, A.K. Bashir, H. N. Chuahdry, and D., Wang A., "Machine Learning Approach for Feature Selection Traffic Classification Using Security Analysis", Journal of Supercomputing, Springer, Vol. 76, pp. 4867-4892, 2018.

[4] LeCun, Y., "Bengio, Y., and Hinton, G., "Deep Learning", Nature", Vol. 521, No. 7553, pp. 436, 2015.

[5] Xin M, Wang Y., "Research on image classification model based on deep convolution neural network", EURASIP Journal on Image and Video Processing, Vol. 40, 2019, December 2019.

[6] Krizhevsky A.; Convolutional deep belief network son cifar-10. Available at https://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf

[7] Keras applications. https://keras.io/applications

[8] Sermanet P., Chintala S., and LeCun Y., "Convolutional neural networks applied to house numbers digit classification". arXiv preprint arXiv:1204.3968, 2012.

[9] Doon R., Rawat T.K., and Gautam S., "Cifar-10 classification using deep convolutional neural network", In IEEE Punecon, pp. 1–5, 2019.

[10] Kowsari K., Heidarysafa M., Brown D.E., Meimandi K.J., and Barn,s L.E.. "Rmdl: Random multimodel deep learning for classification", Proceedings of the 2nd International Conference on Information System and Data Mining, pp. 19–28, ACM, 2018.

[11] Liu J., Gong M., Miao Q., Wang X., and Li H., "Structure learning for deep neural networks based on multi objective optimization", IEEE Transactions on Neural Networks and Learning Systems, Vol. 29, No. 6, pp. 2450–2463, 2017.

[12] Suganthi S.R.L, Hanumanthappa M, and Kavitha S.. "Event image classification using deep learning". Proceedings of the International Conference on Soft-computing and Network Security (ICSNS), pp. 1–8. IEEE, 2018.

[13] Calik R.C. and Demirci M.F.. "Cifar-10 image classification with convolutional neural networks for embedded systems", Proceedings of the 15th IEEE International Conference on Computer Systems and Applications (AICCSA), p. 1–2, 2018.

[14] Garg A., Gupta D., Saxena S., and Sa-Hadev P.P. "Validation of random dataset using an efficient CNN model traineds on mnist handwritten dataset", Proceedings of the 6th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 602–606, IEEE, 2019.

[15] Mohapatra R.K., Majhi B., and Jena S.K., "Classification performance analysis of mnist dataset utilizing a multi-resolution technique", Proceedings of the International Conference on Computing, Communication and Security (ICCCS), pp. 1–5, IEEE, 2015.

[16] Guo T., Dong J., Li H., and Gao Y., "Simple convolutional neural network on image classification", Proceedings of the 2nd IEEE International Conference on Big Data Analysis (ICBDA), pp. 721–724, 2017.

[17] The MNIST database. http://yann.lecun.com/exdb/mnist

[18] The SVHN database. http://ufldl.stanford.edu/housenumber

[19] Recht B., Roelofs R., Schmidt L., and Shankar V., "Do cifar-10 classifiers generalize to cifar-10?" arXiv preprintarXiv:1806.00451, 2018.

[20] The CIFAR-10 database. https://www.cs.toronto.edu/ kriz/cifar.html

[21] Bengio Y., "Learning deep architectures for AI, Foundations and Trends in Machine Learning", Vol. 2, No. 1, pp. 1–127, 2009.

[22] Vailaya A., Jain A., and Zhang H.J. "On image classifica-tion: City images vs. land scapes. Pattern Recognition", Vol. 31, No. 12, pp. 1921–1935,1998.

[23] Lin M., Chen Q., and Yan S., "Network in network". arXiv preprint arXiv:1312.4400, 2013.

[24] Berg A.C, Berg T.L., and Malik J., "Shape matching and object recognition using low distortion correspondences", Computer Vision and Pattern Recognition, Vol. 1, pp. 26–33, 2005.

[25] Martin D.R., Fowlkes C.C., and Malik J., "Learning to detect natural image boundaries using local brightness, color, and texture cues". IEEE Transactions on Pattern Analysis & Machine Intelligence, Vol. 5, pp. 530–549, 2004.

[26] LeCun Y., Bengio, Y., "Convolutional networks for images, speech, and time series". The Handbook of Brain Theory and Neural Networks, 1995.

[27] Efros A.A., Berg A.C., Mori G., and Malik J., "Recognizing action at a distance". In In IEEE International Conference on Computer Vision, Nice, France, pp. 726, IEEE, 2003.

[28] Jin K.H., McCann M.T., Froustey, and Unser E.M., "Deep convolutional neural network for inverse problems in imaging". IEEE Transactions on Image Processing, Vol. 26, No. 9, pp. 4509–4522, 2017.

[29] Tan M., Le Q.V.,. "Efficient net: Rethinking model scaling for convolutional neural networks", arXiv preprint arXiv:1905.11946, 2019.

[30] Deep lizard. Machine Learning and Deep Learning Fundamentals, https://deeplizard.com/ 2017.

[31] Fergus R., Fei-Fei L., Perona P. , and Zisserman A, Learning Object Categories from Google's Image Search, In ICCV, pp. 1816–1823, 2005.

[32] Kamavisdar P., Saluja S., and Agrawal S., "A survey on image classification approaches and techniques", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, No. 1, pp. 1005–1009, 2013.

[33] Sharif M., Kausar A., Park J.H., Shin D-R., "Tiny image classification using Four-Block convolutional neural network", Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC) Jeju Island, Korea, October 2019.

[34] Ku J., Harakeh A., Waslander S.L., "In defense of classical image processing: Fast depth completion on the cpu", Proceedings of the 15th IEEE Conference on Computer and Robot Vision (CRV), pp. 16–22. IEEE, 2018.

[35] Gonzalez T.F., Approximation algorithms for multilevel graph partitioning, Handbook of

Approximation Algorithms and Metaheuristics", pp. 943–958, Chapman and Hall/CRC, 2007.

[36] Kingma D.P., Ba. Adam J.L., "A method for stochastic optimization", arXiv:1412.6980v9, 2014.

[37] Hassan H., Bashir A.K., Abbasi R., Ahmad W., Luo B., "Single image defocus estimation by modified gaussian function", Transactions on Emerging Telecommunications Technologies, Wiley, 2019.

[38] Ahmad M., Bashir A.K., and Khan A.M., "Metric Similarity Regularizer to Enhance Pixel Similarity Performance for Hyperspectral Unmixing. Optik", Journal for Light and Electron Optics, Elsevier, Vol. 140, pp. 86-95, 2017.

[39] Shafiq M., Yu X., Bashir A.K., Chuahdry H.N., and Wang D., A Machine Learning Approach for Feature Selection Traffic Classification Using Security Analysis, Journal of Supercomputing, Springer, Vol. 76, pp.4867-4892, 2018.