Optimizing Convolutional Neural Network Parameters for Better Image Classification

Manik Dhingra
mdhingra@lakeheadu.ca
Department of Computer Science
Lakehead University

Sarthak Rawat
srawat@lakeheadu.ca
Department of Computer Science
Lakehead University

Jinan Fiaidhi
jinan.fiaidhi@lakeheadu.ca
Department of Computer Science
Lakehead University

***Abstract*—The concept of convolution in neural networks has been one of the most challenging and rewarding fields in the field of deep learning. It has improved the way neural network look at and process images, videos and text (Natural Language Processing), anything which can be converted to numbers for processing by a machine. The purpose of this report is to tackle the problem of image classification using convolutional neural networks in terms of optimizing the parameters to be used for such problems. The experiments performed for comparison of certain techniques to achieve specific goals incline towards the ultimate objective of classifying images with reasonable accuracy and within time constraints. For the final part of the project, the classification model developed is deployed as a web-service on cloud for better visualization of its working.**

***Index Terms*—Convolutional Neural Networks (CNN), Natural Language Processing (NLP), hyper-parameters, Extreme Learning Machine (ELM), web-services, MNIST, classification accuracy**

I. Introduction

The relevance of neural networks can be seen in machine learning applications worldwide. It is with no doubt that a neural network is one of the most widely known methods of solving machine learning (ML) problems. However, with passage of time, neural networks have displayed limitations when it comes to processing of large-scale data-sets and other types of data, such as images or video, text, audio signals, and many more. The development of Convolutional Neural Networks (CNN) was a breakthrough in the field of ML for solving these problems.

A neural network is essentially a mathematical model which tries to map an input to either a continuous or a discrete target function, usually unknown to the machine. The aim of developing such machines if to minimize the error between the actual target values and the predicted values. The problem statement for this project is the task of image classification – the machine takes as input an image of certain dimensions and outputs the class to which it belongs, such as objects, places, or even handwritten digits.

This report covers the entire development process and tasks involved in deploying of the classification model implemented to improve performances for image classification. Previous works in the domain of ML for the task of image classification are vividly reviewed in section II. The data-set used for the training and testing of the classifier is discussed elaborately in section III and the network structures of various techniques to improve performances are explained in sections IV, V and VI. Finally, section VII reports the analysis of experiments for the problem statement and the results of the research study and demonstrates the ultimately developed web-service for the task.

II. Literature Review

The concept of a neural network was first coined back in 1957, when Dr. Frank Rosenblatt invented the first neural network model, *Perceptron* [1]. Dr. Bernard Widrow is believed as the 'Father of Neural Networks', and was the co-inventor of the Widrow-Hoff least square filter [2]. The first recurrent neural network (RNN) was invented in 1982 by Dr. John J. Hopfield [3]. The method of 'backpropagation' was first proposed in Dr. Paul Werbos' 1974 PhD thesis. Developments in late 1980s and early 1990s were significant in the field of Machine Learning, with the advent of Support Vector Machines (SVM), Radial Basis Function (RBF) network, maxpooling, and one of the most renowned research topics today, the CNN. With such a magnificent contribution towards the exploration of machine intelligence, researchers like Dr. Yann Lecun, Dr. Andrew Ng and Dr. Feifei Li, to name a few, implemented deep neural networks for sophisticated problems pertaining to images, like the LeNet model [4], the ImageNet project [5], the Places database [6], etc.

The first convolutional neural network model invented was the LeNet-5 by Dr. Yann LeCun in 1998 [4], which aimed to solve the document recognition problem using CNNs. Many state-of-the-art neural network models which have achieved human level accuracy for image classification task have been developed. Using a CNN, a high-dimensional image (or text) can be 'encoded' in lower-dimensional data as a

means to reduce the complexity of models. Some of these image classification models include AlexNet [7], VGGNet [8], GoogleNet [9] and ResNet [10].

The performances of neural networks for the task of classification on tabular data was further improved by introducing deep neural networks with a large number of hidden neurons in many hidden layers, thus contributing to the architecture complexity. Although the development of better and faster processors and graphic cards aided this, a much better and faster solution to the problem was introduced by Huang et al., (2006) [11] – the Extreme Learning Machine (ELM). The concept of ELM is based on random networks, randomly generating input-layer weights for and employing a one-shot learning mechanism (single run) using those random weights. This significantly improved the training times of models.

In addition to this, working on image data can be laborious and time consuming in terms of training the classification model for better accuracy. As a step towards this, the ImageNet project was introduced by Deng et al., (2009) [5] which is an enormous collection of 14M+ images categorized into more than 21K classes. However, this was a database of object-centric images. The Places database was yet another set of 10M+ scene-centric images of 400+ unique scene categories [6]. As improvements to the existing CNN-based image classifiers, they were now being trained on these huge data-sets and the trained models are available open-source for anyone to use. These pre-trained CNN models converged much faster as compared to those trained from scratch.

III. Dataset Specification

There are a plethora of image data-sets available today which have a variety of image types – clothes, digits, vehicles, scenes – an infinite number of options to choose from. Even one class of images can be obtained through more than one data-sets. For the task of image classification in our project, we decided to use the MNIST Handwritten Digits data-set [12], which consists of images of the digits from 0 to 9, ten classes in total. The data-set in total contains 60,000 images in the training-set and 10,000 images in the test-set, each image is of 28×28 pixels and is gray-scaled. The data-set can be downloaded online, or from the *keras* library in python environment, and some examples from the corpus can be viewed in Figure (1).

IV. Data Augmentation

One of the most crucial steps in improving performances for any task is to have a large number of training samples for the model. This is essential in cases where limited volume of data is present since it gets extremely difficult for the machine to learn patterns form that data. Creating new data instances can sometimes prove to be beneficial and sometimes not. In our case, creating (new) fake images from existing ones will increase the size of training samples as well as enable the classifier to be more robust if some images have noise or are distorted. There are a number of ways of augmenting the data-set for better training – (1) affine transformation, (2) rotation, (3) blur, (4) flip (not useful for digits since a flipped image can be a different digit, for instance '5' and '2'), (5) distortion, etc. Such image augmentation tasks were implemented in our project which yielded new images to be fed to the model for training (can be seen in Figures 2–6).

Fig. 1: MNIST Data-set for Handwritten Digit Recognition [12]

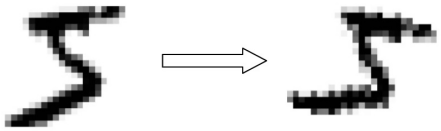

Fig. 2: Image Augmentation by Rotation (clockwise 30°)

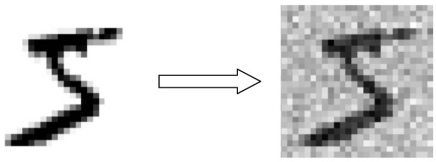

Fig. 3: Image Augmentation by Adding Noise (blur)

As part of an experiment, it was observed that with a general classification model, better performances can be achieved by augmenting the data-set with new generated images than with the original data-set. The types of data augmentation methods used for the analysis on MNIST dataset include the affine transformations and elastic distortions, and the results obtained are displayed in Table I.

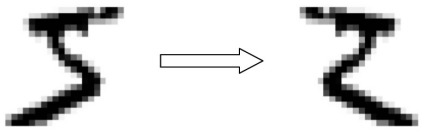

Fig. 4: Image Augmentation by Flipping (not useful here since the digit 5 when flipped looks like the digit 2)

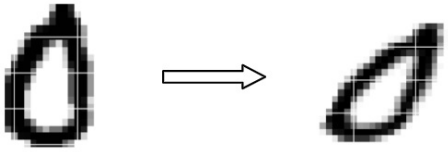

Fig. 5: Image Augmentation by Affine Transformations

V. Extreme Learning Machine (ELM)

Based on the concept of one-shot learning, the Extreme Learning Machine (ELM) architecture is relatively simple to understand and implement. For a single-layer feed-forward ELM network, the structure consists of one input-layer, one hidden-layer, and one output-layer, as in Figure (8), as compared to the dense fully-connected feed-forward network which uses backpropagation for training over a number of epochs, as in Figure (9).

One important step for using the ELM-based architecture is that the data needs to be in a tabular form, i.e. organized in rows and columns. Relating to our problem statement, we plan to use ELM to classify images (not rows and columns, but also channels). Due to this reason, and for better comparisons between ELM and other iterative learning methods, we use a different tabular data-set, called the letter-recognition data-set. This can be downloaded online on the UCI (University of California, Invine) data-set repository [13]. A snapshot of the data-set can be seen in Figure (7), in which the total number of input features is 16, and the last column of the data-set is the class-label – one of the 26 alphabets. The letter-recognition data-set contains a total of 20,000 instances, which are divided into 14,000 training and 6,000 testing samples.

Unlike the traditional backpropagation-based feed-forward networks, the ELM does not use a gradient-based learning algorithm. With this method, all the parameters are tuned exactly once and not iteratively. The input weights for such a network are randomly generated (without training or fine-tuning) and the hidden-layer weights are calculated based on the target mappings. The most important step in implementing an ELM is the selection of number of hidden

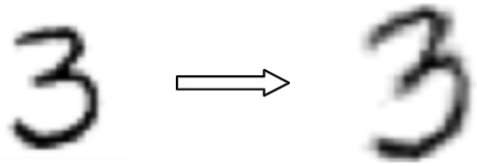

Fig. 6: Image Augmentation by Elastic Distortion

Transformation	Training	Testing
None	97.21%	90.26%
Affine Transformation	97.61%	93.16%
Elastic Distortion	98.02%	95.37%
Affine + Elastic	98.34%	96.16%

TABLE I: Data Augmentation Performance Comparison

layer nodes and the penalty factor. The comparison of ELM with a traditional dense feed-forward network is shown in Table II. This comparison is for a simple multi-class classification problem of alphabet letter-recognition based on 16 input features. In addition to this, the comparison between the two models in terms of training times is also shown in Figure (10).

Model Type	Performance		Training Time
	Training	Testing	
Traditional	90.73%	90.56%	30m 56s
ELM	98.77%	94.76%	02m 04s

TABLE II: ELM vs Traditional Feed-forward Network

VI. Transfer Learning

Transfer Learning is best understood as training a classification model on a huge data-set and then using those trained neuron-weights to test on a relatively smaller data-set. It is one of the most frequently used techniques to boost classification performances, specially in the case of images. Pre-trained CNN weights can be easily found online and used for any task. This technique works much better and converges much earlier then those models trained from scratch. Moreover, since a large data-set is used to train these weights, the performance on the smaller data-set is almost always high.

As mentioned before, a number of CNN architectures are used today for image-related problems, such as the VGGNet, AlexNet, and more. There also exist pre-trained weights for the same classification models. These belong to two categories – (1) pre-trained on Imagenet (for object-centric images), or (2) pre-trained on Places365 (better suited for scene-centric images). Models tested using pre-trained weights and those trained from scratch were evaluated on a

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8	T
1	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10	I
2	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9	D
3	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8	N
4	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10	G
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	2	2	3	3	2	7	7	7	6	6	6	4	2	8	3	7	D
19996	7	10	8	8	4	4	8	6	9	12	9	13	2	9	3	7	C
19997	6	9	6	7	5	6	11	3	7	11	9	5	2	12	2	4	T
19998	2	3	4	2	1	8	7	2	6	10	6	8	1	9	5	8	S
19999	4	9	6	6	2	9	5	3	1	8	1	8	2	7	2	8	A

Fig. 7: Data-set For ELM and Dense Network Models Comparison

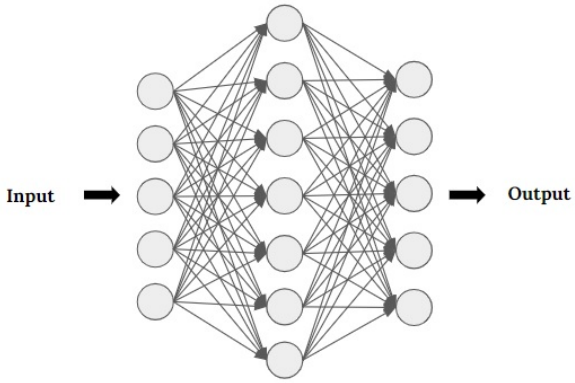

Fig. 8: Extreme Learning Machine Architecture

small data-set, the Scene-15, which consists of 4,485 images divided unequally over 15 scene classes. After a 50-50 split between training and testing images, the two said models obtained the following performances, Figure (12).

VII. Web Service

Now with the sub-optimal model architecture has been designed for image classification, the focus was on developing a web-service to better visualize the predictions made by the proposed model. For the purpose of deploying a computationally intensive CNN model on Google Cloud, there had to be certain constraints to work with:

- small model size
- high performance
- fast speed

For this, the aforementioned model architectures were compared and the analysis is done to get the best fit model for the task, i.e. fast and accurate predictions along with a small size for easy deployment. The results are tabulated in Table III.

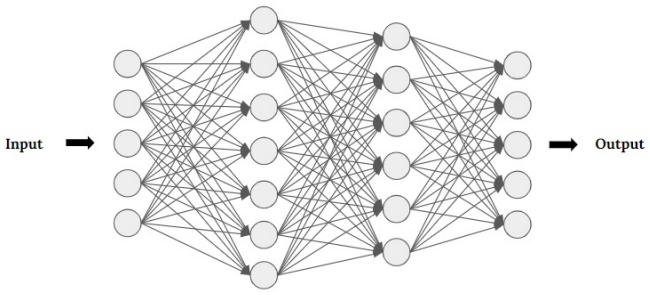

Fig. 9: Dense Feedforward Network Architecture

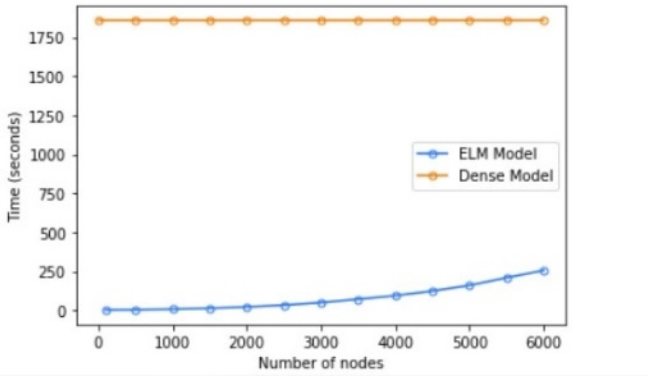

Fig. 10: ELM vs Dense Network Time Comparison

	VGG-16 Transfer Learning	ELM	LeNet-5
Model Size	300MB	60KB	105KB
Accuracy	98%	92%	96%
Training Time	1 hour	20 secs	15 mins

TABLE III: Deployment Model Comparison Metrics

Due to the size and performance trade-off, the LeNet-5 model architecture was used to be deployed on cloud for predictions.

The ultimate goal of our project was the create the web-service which could host our designed image classification model for handwritten digits' prediction. There were a number of sub-goals to be met with the web-service, some of the most important ones are:

- black board for drawing the digit
- variable-size marker for drawing
- label to display the prediction

The web-service is developed using the Flask framework in the python environment. Flask is a lightweight Web

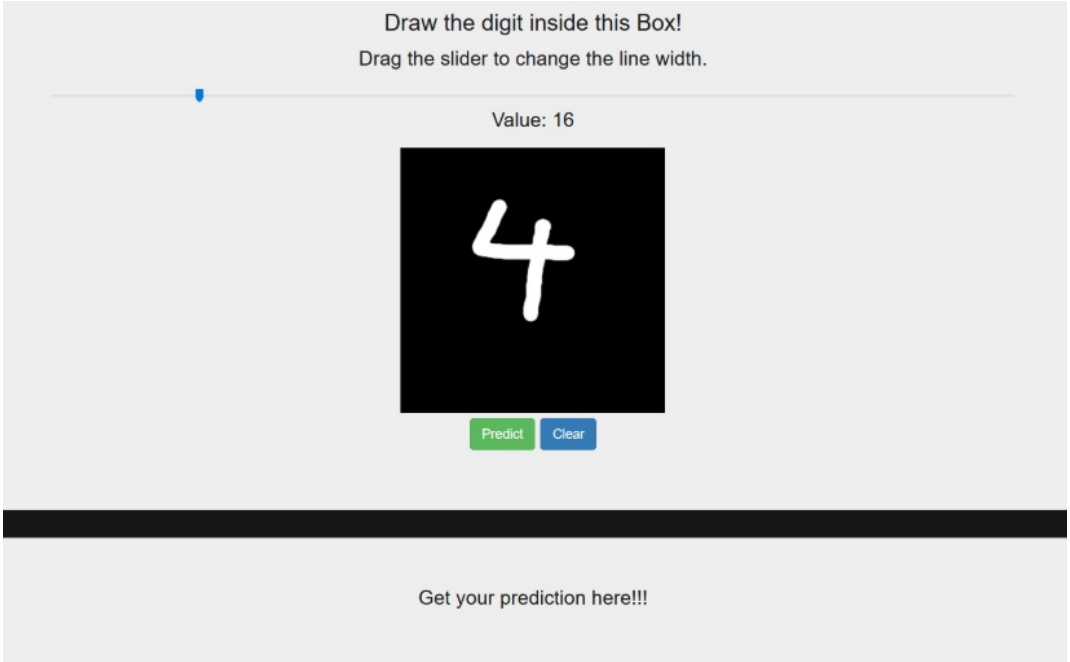

Fig. 11: Web-service: first page

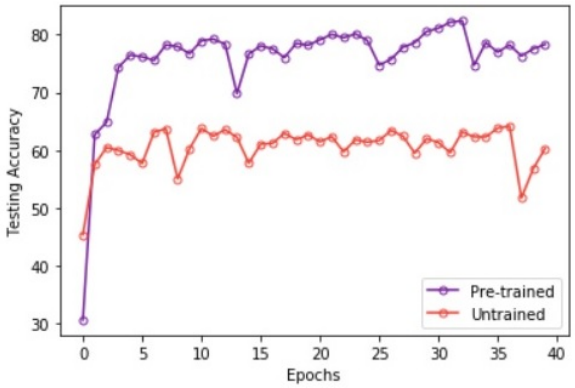

Fig. 12: Transfer Learning Performance Comparison

Server Gateway Interface (WSGI) web application framework. WSGI is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request. Flask is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around *Werkzeug* and *Jinja* and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

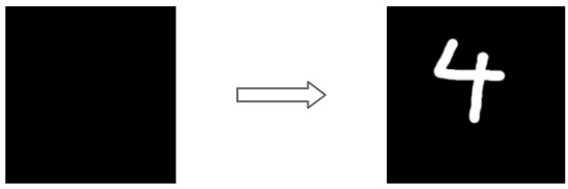

Fig. 13: Web-service: startup page layout (empty black-box to draw the digit)

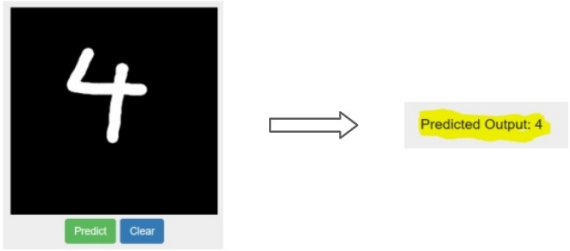

Fig. 14: Web-service: get predictions for the drawn digit

When the web-service is started, the web-page shows a black-box in the middle of the screen along with a bunch of placeholder labels, Figure (11). The user is supposed to draw a (single) digit – from 0 to 9, inside the black-box using the pointer (mouse), as can be seen in Figure (13).

After having drawn the desired digit, the user can click one of 'Predict' and 'Clear' buttons just below the black-box – 'Predict' will give a prediction of the image captured inside the black-box, and 'Clear' will clear with contents of the box, i.e. erase the drawing completely. On clicking 'Predict', the processing is pushed to the back-end server, where our designed classification model is run and the predictions are obtained for the extracted image, and can be seen as in Figure (14).

VIII. Conclusion

The task of image classification is one of many in the field of deep learning. There have been many proposals for enhancement in CNN performances, some of which have been tested in our project. Moving above from images, one can use similar techniques to achieve higher performances for text-based problems such as Spam Classification, Text Generation, Sentiment Analysis, and even video analysis, such as Video Segmentation, Object Detection/Tracking, etc. This is one of the future scopes of this project, that these approaches are useful even for such cases. In addition, the model trained by us just works for digit-recognition, which can be improved to detecting more than single-digit numbers and even letters (alphabets). The scalability of deep learning solutions are endless, and as for us, the sky is the limit.

References

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] B. Widrow and M. E. Hoff, "Adaptive switching circuits," tech. rep., Stanford Univ Ca Stanford Electronics Labs, 1960.

[3] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[6] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

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

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[11] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[12] D. Keysers, "Comparison and combination of state-of-the-art techniques for handwritten character recognition: topping the mnist benchmark," *arXiv preprint arXiv:0710.2231*, 2007.

[13] "Letter recognition data-set." http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.