Digit Recognition Using Neural Network Classifiers

Md. Monoarul Islam Bhuiyan
Dept. of Computer Science & Engineering
Khulna University of Engineering & Technology
Khulna, Bangladesh
monoarul.islam.amit@gmail.com

I. ABSTRACT

Fully connected neural networks and convolutional neural networks are at the very core of large-scale deep learning architectures in the modern era for image classification tasks. Solving problems, for example, classification on MNIST dataset with near human-level accuracy is much more convenient with the help of a fully connected neural network and convolutional neural network than traditional machine learning algorithms. Although classification on the MNIST dataset is an image classification task, a simple fully connected neural network architecture could possibly give us quite fascinating results with some minor improvements before feeding the data into the model. However, especially when it comes to images, there seems to be little correlation or relation between two individual pixels unless they are close to each other thus introducing the concepts of convolution and pooling layers. Additionally, to find finer data representations, both models use a non-linear activation function to capture complex patterns between inputs and outputs. Experimentally, when the dataset is fed into a convolutional neural network classifier, it outperforms a fully connected neural network classifier with less number of trainable parameters.

II. INTRODUCTION

Solving supervised classification problems with the help of deep learning methods has shown great success in recent years. Before that, we used to solve classification problems using traditional machine learning models, for example, logistic regression, naive-bayes, k-nearest neighbors, decision trees, support vector machines, etc. However, these methods have limited generalizability compared to neural networks. Neural networks work on large-scale datasets as well as improve generalization on unseen data. They are able to capture complex patterns with more ease than traditional machine learning algorithms. With hyperparameter tuning and applying regularization methods, the performance could be improved both in the training and in the testing phase while maintaining low bias and variances for the models.

In this work, the approaches for building two neural networks are very simple and still quite efficient. Unlike traditional machine learning algorithms to solve classification problems, neural networks reduce the human effort of feature engineering while trying to capture complex patterns automatically through learnable parameters. The process of training the model parameters involves gradient descent variants such as adaptive momentum estimation (Adam) [5], and root mean squared propagation (RMSProp) [6] and utilizes cross entropy as a loss function. For improved generalization, models are introduced with regularization techniques like dropout [3] where randomly selected neurons are ignored during training and thus initiating randomness in the models. Therefore tone down high-variance models into low-variance models and make them much more robust to unseen data.

The primary purpose of this project is to build a fully connected neural network classifier and convolutional neural network [1] classifier to correctly classify handwritten digits. The evaluation shows that both models give significant results on the testing dataset and specifically convolutional network performs much better than the fully connected network while using fewer epochs and parameters.

III. DATASET DESCRIPTION

The MNIST [2] database (Modified National Institute of Standards and Technology) database is a large database of digits from 0-9 that are handwritten by high school students and employees of the United States Census Bureau. The dataset was created using the samples from NIST's original datasets by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges in 1998. It contains 70,000 handwritten images of digits split into 60,000 training and 10,000 testing images. Every sample image has an associated target label featuring a digit from 0-9. The handwritten images were fitted into 28x28 pixel bounding and also anti-aliased hence minimizing distortion artifacts when representing a high-resolution image to a low-resolution image.

All the fields of each sample image consist of pixel values ranging from 0 to 255. They are then normalized between 0 and 1 for faster computational performance and better generalization. The images of the dataset are transformed into 784 input features that are primarily the 784-pixel fields of 28x28 images for the experiment of a fully connected neural network classifier.

The normalized images are sent directly to the convolutional neural network classifier as 28x28 pixel values. After convolution and pooling operations, the resultant output is flattened out and fed into the fully connected layer.

IV. EXPERIMENTAL SETUP

For the classification task, two network architectures namely a fully connected neural network classifier and a convolutional

neural network classifier have been configured. Hand-written sample images of 28x28 pixels are split into training and testing datasets. There are 60,000 training images to train the models and 10,000 testing images to find out how accurately the models capture the relationship between inputs and outputs.

The fully connected neural network classifier has four layers including two hidden layers of sizes 128 and 64. For each input sample, the model takes 784 normalized input features as they are scaled down from 0-255 to 0-1 for faster computation and better generalization and gives back 10 output probabilities with the class being determined by the highest probability among those outputs. During training and testing, batches of 64 images are passed onto the network for the convenience of data management. The model tries to recognize patterns between inputs and outputs throughout the training phase. With a learning rate of 0.001 and dropout value of 0.1 as suited best for the models, the training loss gets smaller with a higher number of epochs, resulting in capturing complex patterns between inputs and outputs. To train the network, cross-entropy is used as the cost function, and rectified linear unit (ReLU) is used as an activation function to capture the non-linearity of the dataset. The fully connected layer classifier has 109386 parameters including 109184 weights and 202 biases.

$$Cross\ Entropy = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

The convolutional neural network classifier has two convolutional layers with 8 and 16 filters respectively for generating feature maps from the earlier layer. While the samples in the dataset are grayscale images, the number of input channels is thus 1. The kernel size of the filters for both convolutional layers is 3x3. Padding and stride of 1x1 are used to obtain the same convolution i.e. the size of the input doesn't change after each convolution operation. Applying 2x2 max pooling after each convolution operation reduces the size of the input images to half. Thus, extracting more important features as the model performs its operations. After convolution and pooling operations, the resultant output looks like a 3-d tensor of size 16x7x7. By flattening out the tensor and feeding the resultant output into the fully connected layer the model then tries to capture the patterns between the inputs and outputs during training. The cost function, activation function, batch size, and learning rate have been used for the fully connected layer classifier and are similar to the other model. There are 51056 trainable parameters distributed between the convolutional and fully connected parts of the convolutional neural network classifier. There are 240 trainable parameters for the convolutional layer portion with 216 weights and 24 biases and the fully connected layer part consists of 51130 parameters with 50816 weights and 74 biases.

All Models are evaluated by accuracy and F1 score. F1 score uses the equation of the harmonic mean of precision and recall:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2}$$

While the MNIST dataset is balanced where each target class is represented by the same number of input samples, the accuracy and f1 score are the same.

V. Results and Discussion

Table I and table II shows evaluation metric f1 score at different epochs and dropouts for fully connected network classifier and convolutional network classifier. Convolutional neural network is a class of deep neural networks for classification tasks and because of its internal architecture, it takes less training time and less trainable parameters than the fully connected neural network classifier and yet outperforms the fully connected neural network classifier. During convolution and pooling operation the important features are extracted fittingly due to the inner mechanism of the convolutional neural network. After convolution and pooling operation the resultant features are flattened out and fed into the network which then produces near human-level performance on each digit class. RMSProp and Adam are used for optimizing weights and biases. RMSProp uses the second moment with a decay rate to speed up while Adam uses both the first and second moments. Adam uses the combination of two momenta which generates more momentum than RMSProp. As a result during convergence, adam takes a longer time because it oscillates more near the global minima than RMSProp and thus RMSProp generalizes better on the unseen data for both models.

Optimizer	**Dropout**	**F1 Score**
		Epoch 35
Adam	0.0	97.93
	0.1	97.92
	0.2	98.05
	0.3	98.14
	0.4	97.79
RMSProp	0.0	97.89
	0.1	**98.15**
	0.2	97.91
	0.3	98.03
	0.4	97.76

[1]Learning rate = 0.001
[2]Batch Size = 64

TABLE I
Fully Connected Neural Network Classifier on MNIST

Optimizer	**Dropout**	textbfF1 Score	
		Epoch 5	**Epoch 15**
Adam	0.0	**98.63**	98.97
	0.1	98.62	98.98
	0.2	98.57	98.98
RMSProp	0.0	98.51	98.90
	0.1	98.56	**99.04**
	0.2	98.38	98.93

[1]Learning rate = 0.001
[2]Batch Size = 64

TABLE II
Convolutional Neural Network Classifier on MNIST

Fully connected network classifier uses twice as many parameters as convolutional network classifier. Thus it takes more epochs to generalize well compare to the other classifier. Feeding the data into only fully connected network on image-based problems, such as MNIST digit recognition will not provide the best output. Hence, a second network has been built on the concept of convolution to see how it overcomes the limitation of the first model. Adding randomness using dropout for both the models tends to generalize better on unseen data than models without any dropout values, because during training the model sees randomness through dropping out some of the network's neurons. Although this increase the value of training loss than the models without the dropouts, the models perform better on the unseen data as randomness produces lower variance for the models. Thus reducing overfitting. The most important factor for every machine learning model is to have low values for both bias and variance [4], and the models created in this project take great care of this phenomenon. As the models are almost flawless during training, bias is already low.

Table III shows the training and testing accuracy for each class of the MNIST dataset for both models. While the digits are handwritten, certain digits like 4 and 9 or 7 and 1 may look the same for our models during prediction and thus it predicts some incorrect digits due to the condition. Additionally, some unseen digits may seem noisy for our models to predict and thus generate different predictions. For these reasons, the accuracy couldn't be closer to the human level.

Class	FCN		CNN	
	Train Acc.	**Test Acc.**	**Train Acc.**	**Test Acc.**
0	99.881	98.877	99.949	**99.693**
1	99.985	99.295	99.881	**99.911**
2	99.966	99.127	99.932	**99.224**
3	99.934	98.514	99.885	**99.207**
4	99.726	96.741	99.914	**99.287**
5	99.907	97.421	99.631	**99.439**
6	99.949	98.434	99.814	**98.851**
7	99.888	97.276	99.760	**98.832**
8	99.777	97.433	99.470	**98.459**
9	99.966	**98.116**	98.856	97.423

TABLE III
CLASS PERCENTAGE FOR BEST MODELS

A. Prediction Visualization

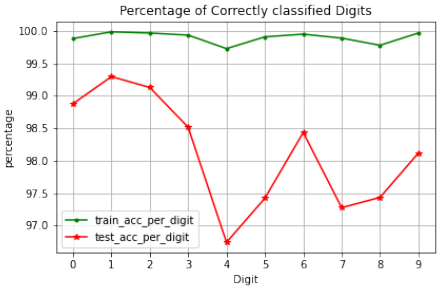

Fig. 1. Class Accuracy Percentage FCN

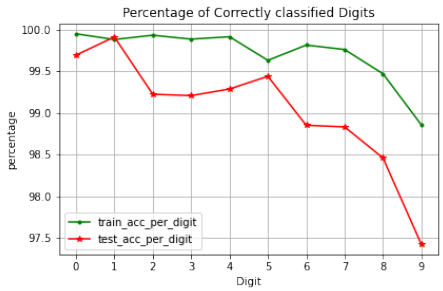

Fig. 2. Class Accuracy Percentage CNN

Fig. 1. and Fig. 2. shows the accuracy percentage of each class for fully connected neural network classifier and convolutional neural network classifier respectively.

Both models perform exceptionally well on the training dataset but fail to generate the same level of accuracy on the testing dataset. For the fully connected network classifier, the margin is slightly higher between training and testing accuracy than the convolutional network classifier. This demonstrates that the fully connected network classifier is an overfitted model and thus convolutional network classifier would be the ideal model to perform classification on the MNIST dataset with a 10 percent dropout value for this project.

VI. CONCLUSION

In this project, two different classifier models are trained and evaluated using the famous MNIST dataset to classify handwritten digits. Although the models perform significantly well and in some cases nearly as close as a human can predict, the convolutional neural network model outperforms the fully connected neural network. Convolutional neural network classifier merely uses about half as trainable parameters as of fully connected neural network classifier, yet outperforms the fully connected network classifier.

Future work would consider state-of-the-art activation functions, optimizers and even batch normalization to be used in the models to improve generalization on unseen data. Further, the addition of a recurrent neural network could give slightly better accuracy than the project's models.

REFERENCES

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). Deep Learning. MIT Press. p. 326.

[2] THE MNIST DATABASE of handwritten digits. Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond.

[3] Dropout: A Simple Way to Prevent Neural Networks from Overfitting Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; 2014.

[4] S. Geman, E. Bienenstock, and R. Doursat (1992). Neural networks and the bias/variance dilemma. Neural Computation 4, 1–58.

[5] Adam: A Method for Stochastic Optimization Diederik P. Kingma, Jimmy Ba 2014

[6] Geoffrey Hinton, "Coursera Neural Networks for Machine Learning lecture 6", 2018