Neural Network for Digit Classification

ylhao

**Abstract**—This project implements a Neural Network and a Convolutional Neural Network for the purpose of digit classification. The two neural networks are implemented in python by using TensorFlow. The network parameters are learned though the backpropagation algorithm. MNIST dataset is used to train and test the network

**Index Terms**—Neural Network, Convolutional Neural Network, Convulotional, Dropout, Pooling, Adam, Back Propagation, Digit Classification

# 1 Introduction

Digit classification is the task of classifying a given image into one of the digits from 0 to 9. This is a typical classification problem. A classification problem consists of a set of classes and the machine has to classify a given object into one of the classes. Other examples of a classification problem include facial recognition, image classification (identifying objects as car, bus, dog) etc. This classification problem can be addressed using many techniques such as Linear Regression, K-nearest-neighbours, SVM, Neural Network, Convolutional Neural Network and many other such machine learning techniques. In this project we have done the task of digit classification using a Neural Network and a Convolutional Neural Network. The motivation for this project comes up from the fact that digit classification has been an important research area for exploring several learning techniques ranging from automatically learning feature representations, learning classifiers invariant to distortions, matching and alignment based distances, and learning multilayered representations of data[1].

# 2 Review

## 2.1 Neural Network

In this report The key idea of ‘neural network’ is the novel structure of the artificial networks[2]. ANN consists of an input layer, any number of hidden layers (including 0), and an output layer. Each layer has a collection of neurons and all the layers are connected in an acyclic manner. In other words, the outputs of some neurons in a layer will be the inputs to other neurons in the above layer. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including but not limited to sales forecasting, business marketing, customer research, data validation, risk management, medicine, robot learning etc.

## 2.2 Convolutional Neural Network

A convolutional neural network (CNN) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks.

Like other kinds of artificial neural networks, a convolutional neural network has an input layer, an output layer and various hidden layers. Some of these layers are convolutional, using a mathematical model to pass on results to successive layers. This simulates some of the actions in the human visual cortex.

CNNs are a fundamental example of deep learning, where a more sophisticated model pushes the evolution of artificial intelligence by offering systems that simulate different types of biological human brain activity.

## 2.3 Max Pooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned[3].

We have a 4x4 matrix representing our initial input and we hanve a 2x2 filter that we'll run over our input. We'll have a stride of 2 and we won’t overlap regions.

For each of the regions represented by the filter, we will take the max of that region and create a new, output matrix where each element is the max of a region in the original input.

## 2.4 Dropout

Neural network with large number of parameters are very powerful but they are prone to the issue of overfitting.

Dropout is a [regularization](https://en.wikipedia.org/wiki/Regularization_(mathematics)) technique for reducing [overfitting](https://en.wikipedia.org/wiki/Overfitting) in [neural networks](https://en.wikipedia.org/wiki/Neural_networks) by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a [neural network](https://en.wikipedia.org/wiki/Neural_network)[4].

# 3 Method

## 3.1 Architecture

In this project, we have implemented a neural network and a convolution neural network for the purpose of digit classification. Our implementation is in python by using TensorFlow.

### 3.1.1 Nueral Network Architecture

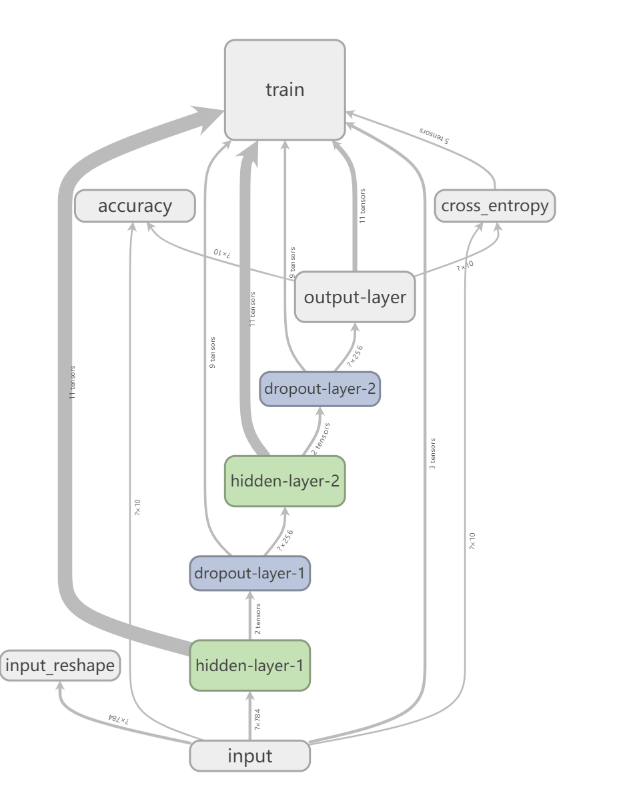
There are two hidden layers with 256 neurons in each hidden layer and an output layer with 10 neurons corresponding to the ten digits from 0 to 9 in the neural network. The dataset that we have used in this project is the MNIST dataset in which each image is represented by 28 x 28 pixels. Each pixel’s value ranging from 0 to 255. We use these 784-pixel values to represent the input and to train the network. So, we have 784 neurons in the input layer. The structure of our neural network as shown in Fig. 3.1.1

Fig. 3.1.1. Structure of our Neural Network

### 3.1.2 Convolutional Nueral Network Architecture

There are one convolutional layer with 32 kernals, one convolutional layer with 64 kernals, two max pooling layers, one hidden layer with 1024 neurons and an output layer with 10 neurons corresponding to the ten digits from 0 to 9 in the convolutional neural network. The sturcture of our convolutional neural network as shown in Fig.3.1.2.

Fig. 3.1.2. Structure of our Convolutional Neural Network

## 3.2 Preparing the training data

MNIST database [3] of handwritten digits is used to train the network and to test the accuracy. The data is available in a byte format and we have used the python-mnist library [4] to read the dataset which will return a list of all the images and the labels corresponding to each image. Each image returned by this library will be in the form of a list of 784 features corresponding to the 784 pixels with a value from 0 to 255. The label corresponding to that image is a number from 0 to 9. As the loss that we are about to calculate is categorical cross entropy, we have encoded the labels as one-hot vectors [9]. So, that means a label of ‘1’ has been encoded as a list of 10 in which the first index is 1 and the rest are zeros i.e. [1, 0, 0, 0, 0, 0, 0, 0, 0, 0].

There are 65000 images in the data set and we have sampled randomly 5000 out of them for validation and 10000 for testing. Since we have randomly sampled all the images, our assumption is that there are equal number of samples from each class in all the three splits. The image data has pixel values ranging from 0 to 255, so we have normalized each pixel value to be in the range [0, 1] by dividing each value by 255. This can help the algorithm to converge faster.

## 3.3 Network Parameters

### 3.3.1 Nueral Network Parameters

We have defined six network parameters as follows:

W1: From input layer to the first hidden layer. Since input layer has 784 neurons and the first hidden layer has 256 neurons., this will be a 784 × 256 vector.

B1: This is the bias applied to each of the neuron in the first hidden layer, a 256 × 1 vector.

W2: From the first hidden layer to the second hidden layer. Since each of the hidden layers has 256 neurons, this will be a 256 × 256 vector.

B2:This is the bias applied to each of the neuron in the second hidden layer, a 256 × 1 vector.

W3: From the second hidden layer to the output layer. Since the second hidden layer has 256 neurons and the output layer has 10 neurons, this will be a 256 x 10 vector.

B3: This is the bias applied to each of the neuron in the

Output layer, a 10 × 1 vector.

W1, W2, W3 parameters are initialised with the values follow a truncated normal distribution with mean of 0 and standard deviations of 0.1. B1, B2, B3 parameters are initialized with 0.1.

### 3.3.2 Convolutional Nueral Network Parameters

We have defined six network parameters as follows:

W1: From input layer to the first convolutional layer. Since there are 32 kernels in the first convolutional layer, the kernel size is 5 and the number of input channels is 5, This will be a 5 × 5 × 32 vector.

B1: Since there are 32 kernels in the first convolutional, This will be a 32 × 1 vector.

W2: From the first max pooling layer to the send convolutional layer. Since there are 64 kernels in the second convolutional layer, the kernel size is 5 and the number of input channels is 32, This will be a 5 × 5 × 32 × 64 vector.

B2: Since there are 64 kernels in the second convolutional, This will be a 64 × 1 vector.

W3: From the flatten layer to the first hidden layer. Since the output shape of faltten layer is (3136, 1) and the hidden layer has 1024 neurons, this will be a 3136 x 1024 vector.

B3: This is the bias applied to each of the neuron in the

first hidden layer, a 1024 × 1 vector.

W4: From the first hidden layer to the output layer. Since the first hidden layer has 1024 neurons and the output layer has 10 neurons, this will be a 1024 x 10 vector.

B4: This is the bias applied to each of the neuron in the

Output layer, a 10 × 1 vector.

W1, W2, W3, W4 parameters are initialised with the values follow a truncated normal distribution with mean of 0 and standard deviations of 0.1. B1, B2, B3, B4 parameters are initialized with 0.1.

## 3.4 Batch Shuffle

In each iteration, the training dataset is shuffled around so that each sample has no dependency on its previous or the next sample and is trained as if it is chosen independently. This leads to a faster convergence and is more efficient [5].

## 3.5 Testing

Once the training is completed, we have run the neural network on the test dataset to calculate the accuracy and thus measuring the performance of our digit classifier.

# 4 Experiments and Conclusion

We have trained the network on 50000 samples and for 5 epochs.

## 4.1 Select Dropout Rate

### 4.1.1 Nueral Network Dropout Rate

We have changed the dropout rate from 0.1 to 0.9 and print the accuracy on the testset after training for one epoch. We observed that a dropout rate of 0.1(keep\_prob=0.9) yields the highest accuracy. So, we have chosen 0.1 dropout rate for the rest of the experiments(set keep\_prob as 0.9).

|  |  |
| --- | --- |
| Dropout Rate(keep\_prob) | Accuracy |
| 0.1 | 97.21% |
| 0.2 | 97.89% |
| 0.3 | 98% |
| 0.4 | 98.17% |
| 0.5 | 98.43% |
| 0.6 | 98.34% |
| 0.7 | 98.5% |
| 0.8 | 98.41% |
| 0.9 | 98.61% |

### 4.1.2 Convolutional Nueral Network Dropout Rate

We have changed the dropout rate from 0.1 to 0.9 and print the accuracy on the testset after training for one epoch. We observed that a dropout rate of 0.1(keep\_prob=0.9) yields the highest accuracy. So, we have chosen 0.1 dropout rate for the rest of the experiments(set keep\_prob as 0.9).

|  |  |
| --- | --- |
| Dropout Rate(keep\_prob) | Accuracy |
| 0.1 | 47.56% |
| 0.2 | 90.97% |
| 0.3 | 90.98% |
| 0.4 | 94.51% |
| 0.5 | 95.13% |
| 0.6 | 95.78% |
| 0.7 | 95.88% |
| 0.8 | 96.03% |
| 0.9 | 96.68% |

## 4.2 Learning Rate

We have analysed various learning rates. For a learning rate of 0.001, a high accuracy is reached after 5 epoch. The accuracy is more better if we use learning rate decay strategy in our Nueral Network.

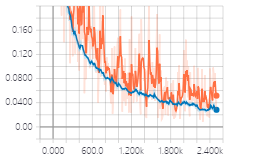
In our leraning rate decay strategy, we set the decay rate to 0.96 and set the decay step to 100.

|  |  |
| --- | --- |
| Model And Stategy | Accuracy |
| nn, fix lr = 0.001 | 97.71% |
| nn, lr decay | 98.09% |
| cnn, lr = 0.001 | 98.83% |
| cnn, lr decay | 98.99% |

## 4.3 Loss Behavior

### 4.3.1 Nueral Network Loss Behavior

Every ten step, the network is evaluated on the validation dataset and the training dataset. The loss is plotted against the step in Fig. 4.3.1 and Fig 4.3.2

As expected, the loss reduces as the iterations increase. In the first 500 steps, the loss reduces significantly.

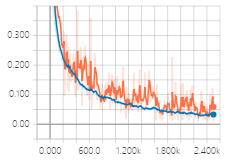
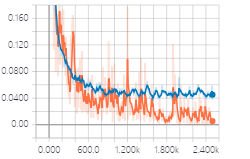
Fig. 4.3.1. Neural Network Loss(lr=0.001)

Fig. 4.3.2. Neural Network Loss(lr decay)

### 4.3.2 Convolutional Nueral Network Loss Behavior

Every ten step, the network is evaluated on the validation dataset and the training dataset. The loss is plotted against the step in Fig. 4.3.3 and Fig. 4.3.4.

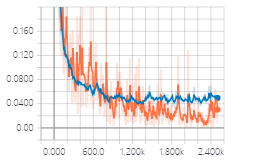
Fig. 4.3.3. Convolutional Neural Network Loss(lr=0.001)

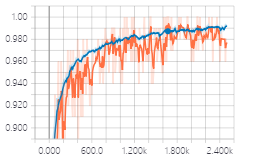
Fig. 4.3.4. Convolutional Neural Network Loss(lr decay)

As expected, the loss reduces as the iterations increase. In the first 500 steps, the loss reduces significantly.

## 4.4 Accuracy

### 4.4.1 Nueral Network Accuracy

As the training progressed, we have plotted accuracy of the network on the validation dataset in Fig. 4.4.1 and Fig 4.4.2. Like the loss behavior, we have a very high accuracy after the first 500 steps(one epoch) and it continues to improve.

Once the training is complete, the network is evaluated on the testset and the accuracy is obtained as 98.09% (with using learning decay strategy).

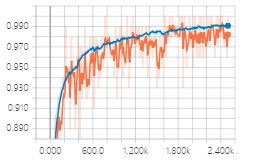
Fig. 4.4.1. Neural Network Accuracy(lr=0.001)

Fig. 4.4.2. Neural Network Accuracy(lr decay)

### 4.4.2 Convolutional Nueral Network Accuracy

As the training progressed, we have plotted accuracy of the network on the validation dataset in Fig. 4.4.3 and Fig 4.4.4. Like the loss behavior, we have a very high accuracy after the first 500 steps(one epoch) and it continues to improve.

Once the training is complete, the convolutional neural network is evaluated on the testset and the accuracy is obtained as 98.99% ((with using learning decay strategy).

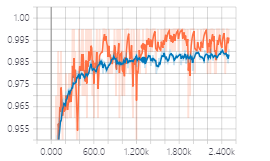
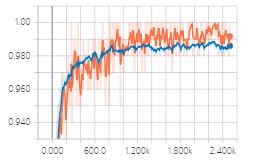
Fig. 4.4.3. Neural Network Accuracy(lr=0.001)

Fig. 4.4.4. Neural Network Accuracy(lr decay)

# 6 Conclusion and Future Scope

We have seen that Neural network is a powerful tool. We have also seen that choosing the activation functions and the loss functions are key to the purpose of the network. We need enough training samples to learn the network parameters through the back-propagation algorithm which is a combination of the chain rule and dynamic programming. However powerful the neural network is, there is always the problem of overfitting when the network is overtrained. We have seen one regularization scheme called Dropout that prevents the network from overfitting and thus providing a better performance. Dropout effectively prevents the nodes of the hidden layers from coadapting too much and thus making sure that each node has enough information to decide.

We have also seen the Convolutional Neural Network is more suitable for this problem and the Convolutional Neural Network has less parameters.

Generally Speaking, the accuracy is more better if we use learning rate decay strategy in our Nueral Network.

**References**

1. [Subhransu Maji and Jitendra Malik. “Fast and Accurate Digit Classification. “](https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-159.pdf)
2. [Christos Stergiou and Dimitrios Siganos. “NEURAL NETWORKS.”](https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html)
3. <https://computersciencewiki.org/index.php/File:MaxpoolSample2.png>
4. <https://en.wikipedia.org/wiki/Dropout_(neural_networks)>
5. <https://arxiv.org/pdf/1707.09725.pdf#page=95>