

HANDWRITTEN DIGIT RECOGNITION

CS 354N Minor Project



UNDER THE GUIDANCE OF

Dr. Aruna Tiwari



INTRODUCTION:

Handwriting Recognition is the ability of a computer to interpret intelligible handwritten input from various sources such as paper documents, photographs, touch-screens and various other devices. Handwriting Recognition principally entails Optical Character Recognition. One such category is Handwritten Digit Recognition. Handwritten character recognition is a field of image processing as well as pattern recognition.

Now-a-days we are facing problems converting hard copy matter into digital information for further usage, for example, in bank cheques the digits are written by hand on cheques. To convert this handwritten digits into digital form for further use, we need human effort and lots of time. Another such example is recognition of zip codes on envelope for postal services. Since the handwriting of different writers is different, building a general recognition system that would recognize all characters with good reliability is not possible in every application.

Recognizing handwritten digits isn't easy. The difficulty of visual pattern recognition becomes apparent if you attempt to write a computer program to recognize digits. Simple intuitions about how we recognize shapes - "a 9 has a loop at the top, and a vertical stroke in the bottom right" - turn out to be not so simple to express algorithmically. Neural networks approach the problem in a different way. The idea is to take a large number of handwritten digits, known as training examples, and then develop a system which can learn from those training examples. In other words, the neural network uses the examples to automatically infer rules for recognizing handwritten digits. Furthermore, by increasing the number of training examples, the network can learn more about handwriting, and so improve its accuracy.

PROBLEM STATEMENT:

In this minor-project, we are developing a program using Deep Neural Network (DNN) and Convolutional Neural Network (CNN). We are implementing a neural network with 3 hidden layers. Output Layer has 10 output nodes for classifying 0-9 digits.

Handwritten digit recognition system can be divided into four stages

1. Data acquisition
2. Pre-processing
3. Feature extraction
4. Classification

DATASET:

Dataset is taken from MNIST (Modified National Institute of Standards and Technology) database. Images are extracted from this dataset. The set of images in the MNIST database is a combination of two of NIST's databases: Special Database 1 and Special Database 3. Special Database 1 and Special Database 3 consist of digits written by high school students and employees of the United States Census Bureau, respectively.

Each image is a 28 by 28 pixel square (784 pixels total). A standard spit of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it.

Example Dataset Images:

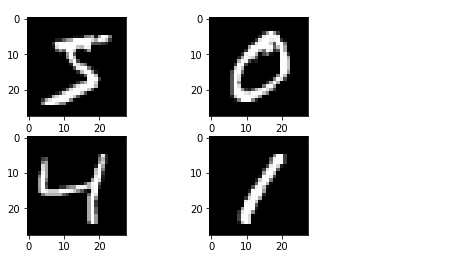
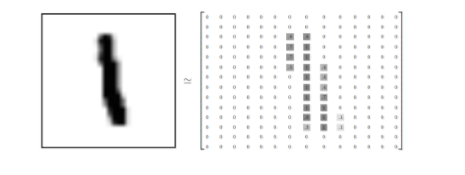


Image is converted in to 28X28 Matrix.Example below:



FEATURESET:

Complete image is our Feature set in this program.

PRE-PROCESSING:

The images in the dataset are in the grayscale format i.e. in the form of pixel density value between 0 to 255. In binarization, the grey scale image is thresholded and converted into binarized image in the form of 0 and 1. Binarization is done by dividing every pixel value with threshold value (in our case maximum value of grayscale - 255) and we are encoding the obtained binarized image using One Hot encoder for using it as input in the Deep Neural Network or Convolutional Neural Network.

CLASSIFICATION:

Output Variable is an integer from 0-9. So, this becomes a multi-class classification problem. We use one hot encoding of class value, transforming class integers into binary. The

ALGORITHM OVERVIEW:

Algorithm-1 (Deep Neural Network):

**Description**:

This neural network has 3 hidden layers, 1 input and 1 output layer. Output layer has 10 output node for classification of 0-9 values learning using by backpropagation approach.

**Overview:**

1. Modified input data is input is feed into input layer
2. Input layer passes the values to hidden layer 1 with each having an edge weight.
3. All the values approached in neurons in the hidden layer 1 are summed with bias and pass through activation function.
4. Output of hidden layer 1 is fed into hidden layer 2 with multiplying the respectively weights.
5. In hidden layer 2, all the values approached in neurons are summed with bias and pass through activation function.
6. Output of hidden layer 2 is fed into hidden layer 3 with multiplying the respectively weights.
7. In hidden layer 3, all the values approached in neurons are summed with bias and pass through activation function.
8. Output of hidden layer 3 is fed into output layer with multiplying the respectively weights.
9. In output layer, all the values approached in neurons are summed with bias and output is shown.

As learning is by backpropagation approach, we compare output with desired output by cross entropy function and optimizer used here is Adam optimizer or SGD.

We run 10 to 20 epochs for more accurate. One epoch is one forward pass + backward pass of all data set.

Algorithm-2 (Convolutional Neural Network):

**Description**:

We make convolutions from image and make maxplots from the image this is done for two layers and next third layer is fully connected layer and last layer is output layer.

Libraries

For Algorithm-1:

1. tensorflow

Reason for using tensorflow is Many company prefer and do ANN in Tensorflow.so,we choosed.

2. matplotlib

For Algorithm-2:

1. tensorflow or tflearn or keras

2. matplotlib

Functions:

For activation function: relu (rectified linear unit)

For cost (output - input): cross\_entropy\_with\_logits

For optimizer: Adam optimizer or SGD.

DEEP NEURAL NETWORK:

A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers. DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitive data types The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow(opp. of deep) network.

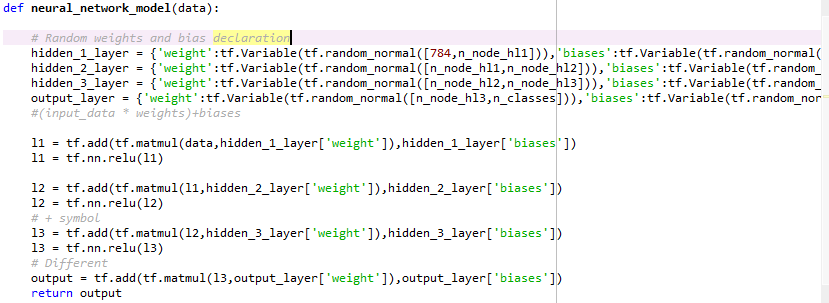
DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back.

**Abstract view of the solution**:

Input -> Weights -> hidden layer 1 (with bias) -> activation function -> Weights -> hidden layer 2 (with bias) -> activation function->Weights -> hidden layer 3 (with bias) -> activation function->weights->output layer(with bias).

We did the code in Tensorflow in Python 3.6. In this architecture, learning method is Backpropagation.

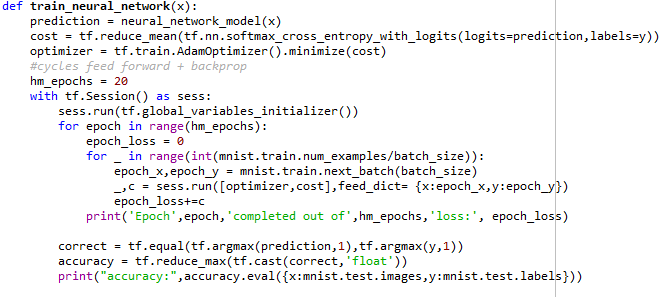
Neural Network function is :



In this NN, weights and bias are initialized randomly.

Activation function used is ReLu (Rectilinear Activation Function).

This function returns the output in OneHot form.



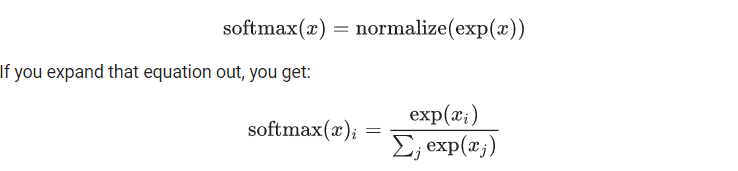
Above function is a training neural network function

Here we used

Cost function: softmax cross entropy with logits

Optimizer is AdamOptimizer for backpropagation learning.

Softmax cross entropy function is:



We run 10 epochs and the dataset is made into batches and trained.

Here accuracy is the float value of the correct function of all batches in dataset.

Data is taken from MNIST dataset and dataset output is in OneHot form.

Network has 500 neural per layer.

Input is taken in form of 1x784 matrix form.

Since the output is going to be from 0-9, total no. of output neurons are only 10.

CONVOLUTIONAL NEURAL NETWORK:

A Convolutional Neural Network (CNN) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is the advantage it has compared to others.

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

**Convolutional:**

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images.

A very high number of neurons would be necessary, even in a shallow architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

**Pooling:**

Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

**Fully Connected:**

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

**Abstract view of the solution**:

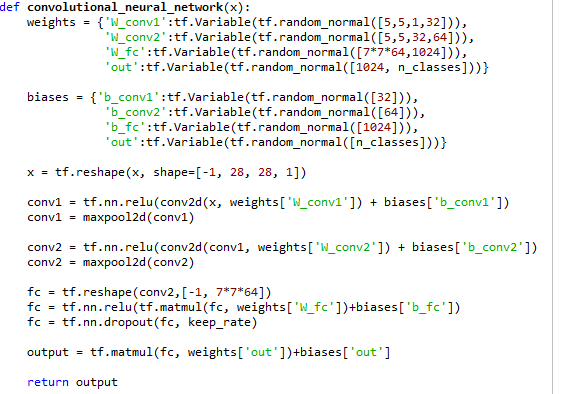
Input ->(Convolutional + Maxpool)Hidden Layer 1 ->(Convolutional + Maxpool)Hidden Layer 2 ->Fully connected layer -> Output Layer.

Example:

How number is identified is similar to face identification.

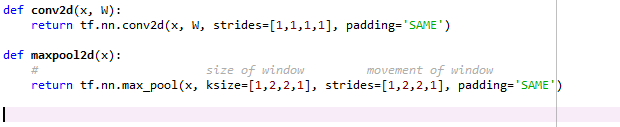


Code:



At first weight and bias variables are declared randomly.

Input is reshaped and convolution & max pooling are done.

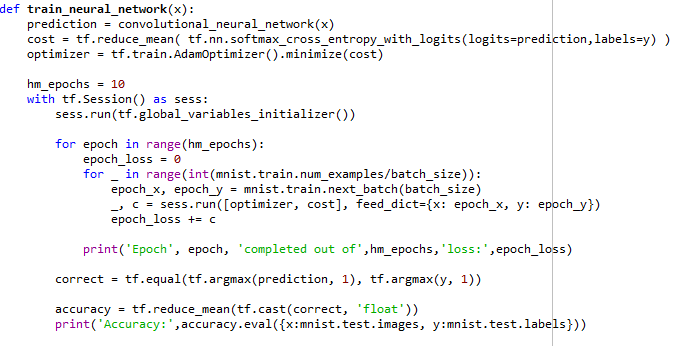


Convolution and MaxPooling functions

In Convolution we made “x” as “W” because we took matrix of size 1x1.

In max pooling, Matrix of 2x2 is taken and matrix is moved by 2 pixels.

Training Set Code:



This code is same as before method.

In second method, we applied dropout function of neurons with some rate to have minimal neurons.

**Reasons for using Rectilinear Activation Function, Softmax cross entropy function and AdamOptimizer**:

Rectilinear activation function derivative is bigger than Sigmoidal, Tanh activation function. So, learning will be faster.

We are comparing the output with actual output which is in One Hot format. So, using softmax is better and faster than Squared error.

AdamOptimizer learning is larger and AdamOptimizer mainly used for image identification.

OBSERVATION:

Time taken to run first program is 10 min (Approx).

Time taken to run second program is 15 min (Approx).

More time is taken to load the library into the terminal.

Output of the programs are error of each iteration and accuracy.

**Note:**

We are doing the solution in second method (CNN) because sometimes accuracy is coming to be 1.0.

Reasons are

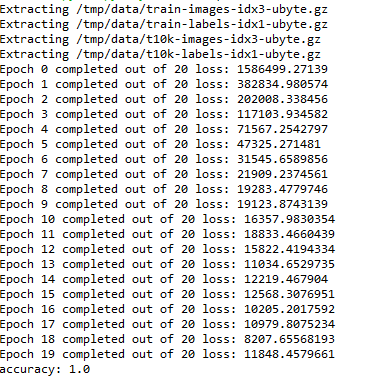
1. Neural Network may be memorizing
2. Accuracy calculation float conversion may have mistake

PERFORMANCE MEASUREMENT:

This program’s performance is measured by accuracy. In general, accuracy is measured as

RESULT 1:

Here is a partial transcript of the output of one training run of the neural network. The transcript shows the number of epochs completed and error value recognized by the neural network after each epoch of training. As you can see, after just a single epoch this has an error of 382834.98, and the number continues to decrease in general, suggesting the increase in accuracy.

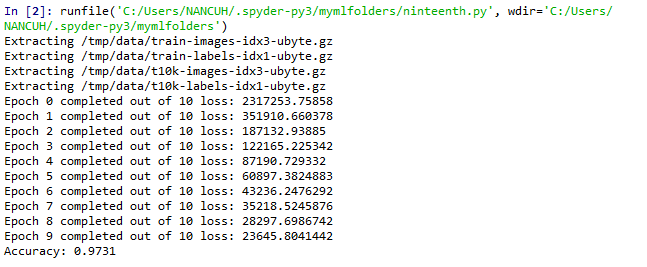


That is, the trained network gives us a accuracy of about 1.0.

However, the results are not necessarily going to be same since we'll be initializing our network using (different) random weights and biases.

RESULT 2:

Here is a partial transcript of the output of one training run of the neural network. The transcript shows the number of epochs completed and error value recognized by the neural network after each epoch of training. As you can see, after just a single epoch this has an error of 351910.66, and the number continues to decrease, suggesting the increase in accuracy.



Thus, this trained network gives accuracy of 0.9731 or 97.31%.

CONCLUSION:

In this project, we created a Neural Network that can train and recognize the handwritten digits upto 97% accuracy using Convolutional Neural Networks and accuracy of “1” using Deep Neural Network. Hence, Convolutional neural Network or CNN is more accurate and gives precise results than Deep Neural Network.

REFERENCES:

1. Internet
2. MNIST
3. tensorflow.org (<https://www.tensorflow.org/versions/r1.2/get_started/mnist/pros> and <https://www.tensorflow.org/versions/r1.2/get_started/mnist/beginners>)
4. tflearn.org
5. Other sources.