Machine Learning Engineer Nanodegree

Capstone Proposal

Rachit Kumar Agrawal 1st Sept 2017

Proposal

(approx. 2-3 pages)

Domain Background

(approx. 1-2 paragraphs)

I have picked up the domain of Computer Vision for my capstone project. In last decade, the amount of data has exploded exponentially. These data are in different forms - text, image and video. With each of these forms of data exploding, it becomes extremely important to categorize them or classify them and classification of these images is not humanly possible. So, it become important to let machine do it for us. I am personally motivated for this problem because, I feel this is first step to my ultimate goal to create a prototype for the Autonomous Driving.

Problem Statement

(approx. 1 paragraph)

The problem I am trying to solve is - "Digit Recognition on MNIST dataset". The goal is to recognize hand-written digits in images and identify the correct digit it represents.

2. Datasets and Inputs

(approx. 2-3 paragraphs)

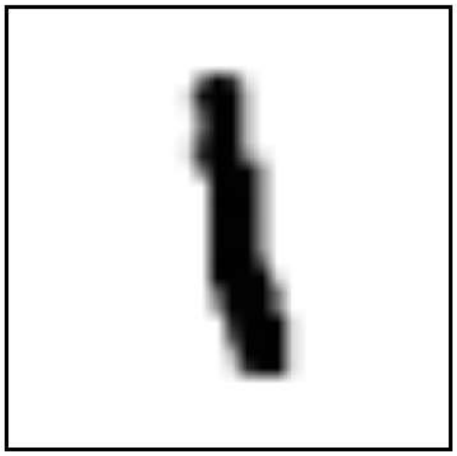
2.1 Input Dataset

For this problem, I am using MNIST image dataset. The MNIST dataset consists of images of handwritten images like these:

It includes labels for each of these input image, telling us which digit it is. For example, the labels for the above images are 5, 0, 4 and 1.

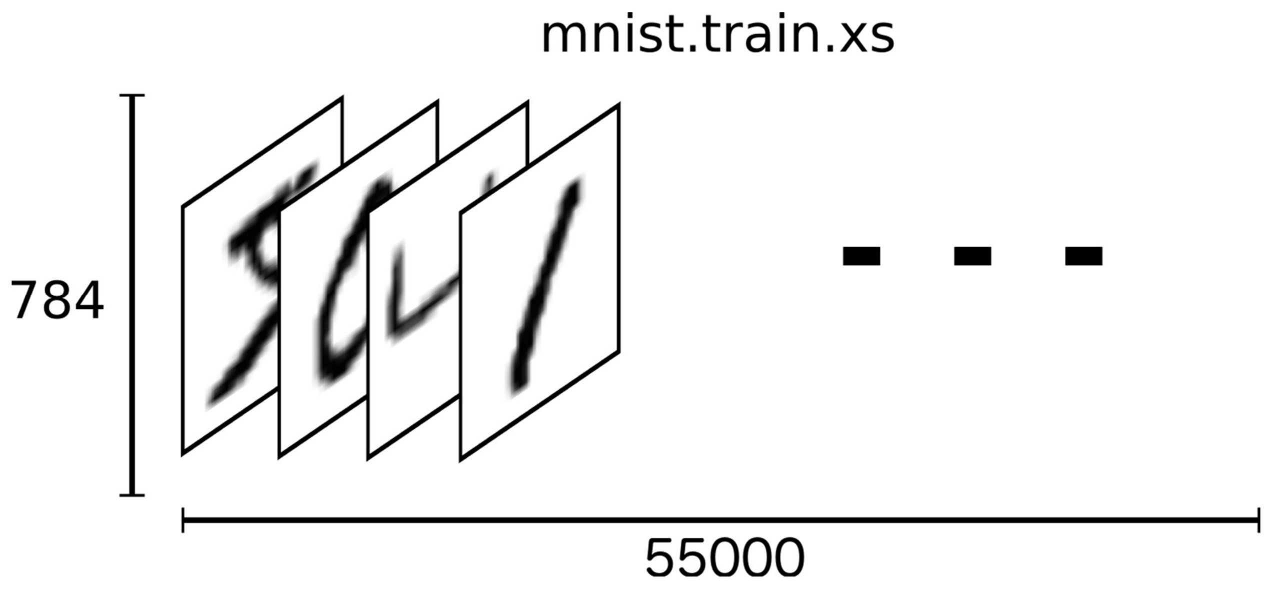
The MNIST data is split into three parts: 55,000 data points of training data, 10,000 points of test data and 5,000 points of validation data. This split is essential to make sure that what we learn from the model actually generalizes.

Every MNIST data point has two parts: an image of a handwritten digit and a corresponding label. I will call the images “x” and the labels “y”. Both the training set and test set contain images and their corresponding labels. Each image is 28 pixels by 28 pixels. We can interpret this as a big array of numbers:

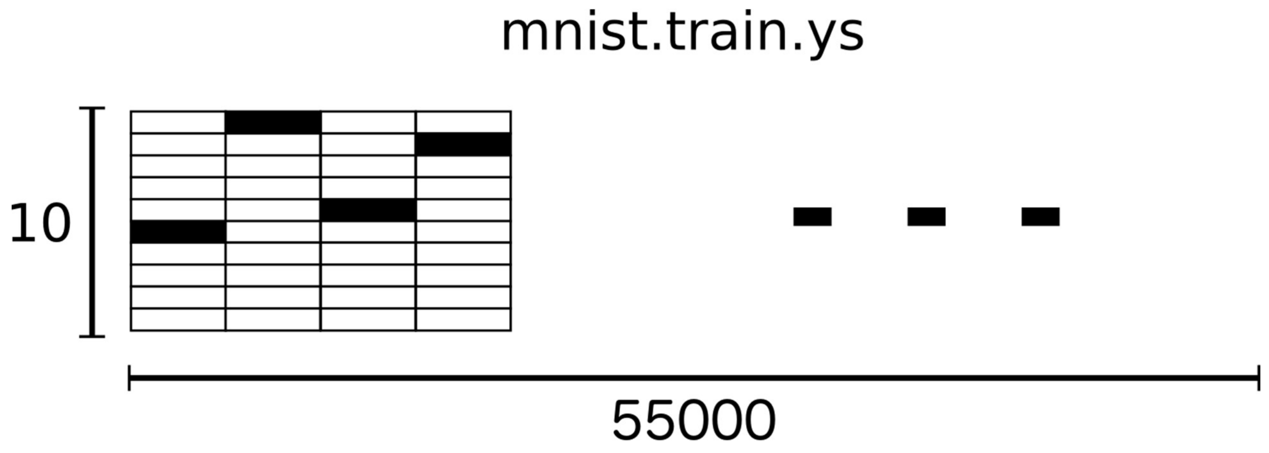


$$\simeq \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

A 28 x 28 image has total 28 x 28 = 784 numbers. So, the training data set can be visualized as the figure shown below:



Similarly, the labels are the integer number representing the correct label. We are going to convert these labels into “one-hot-vectors”. A one-hot vector is a vector which is 0 in most dimensions, and 1 in a single dimension. In this case, the nth digit will be represented as a vector which is 1 in the nth dimension. For example, 3 would be [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]. So, these labels can be visualized as:



The MNIST database is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. This database was created and released by Yann LeCun, Corinna Cortes and Christopher J.C. Burges.

The input data is provided in four separate files:

FileName	**Details**
Train-images-id3-ubyte.gz	Training Set Images (9.9 MB)
Train-labels-idx1-ubyte.gz	Training Set Labels (28 KB)

T10k-images-idx3-ubyte.gz	Test Set Images (1.6 MB)
T10k-labels-idx1-ubyte.gz	Test Set Labels (4.5 KB)

The data is stored in a very simple file format designed for storing vectors and multidimensional matrices.

2.2 Evaluation Metric:

3. Solution Statement

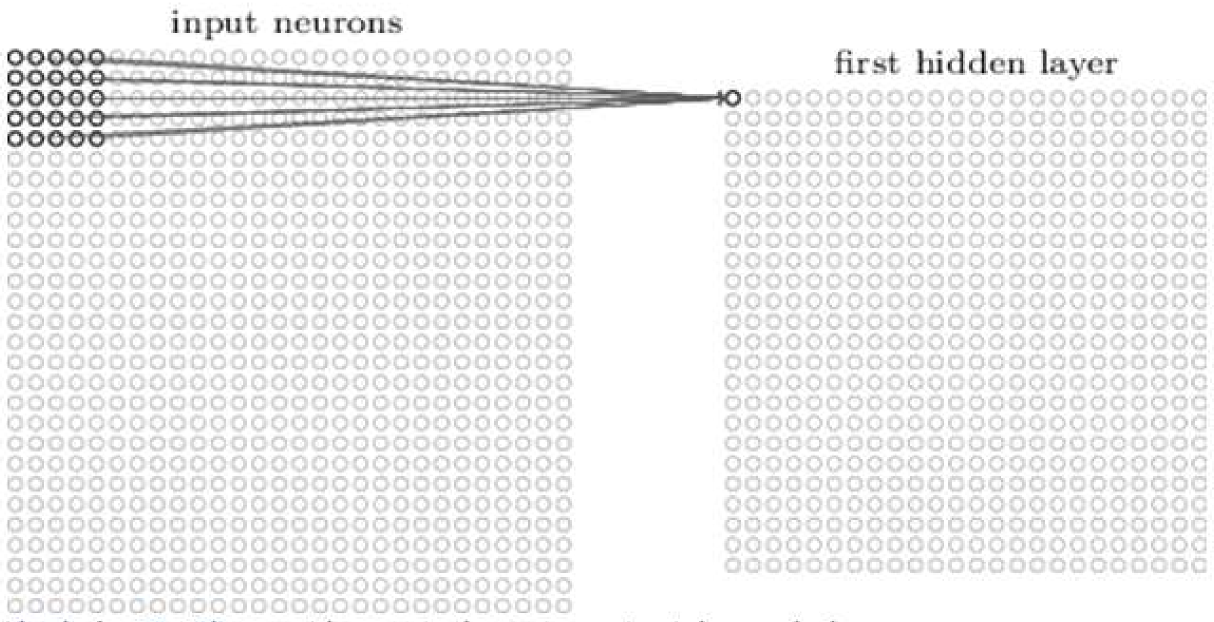
3.1 Neural Network

I plan to use Convolution Neural Network (CNN) for this project. In machine learning, a convolution neural network 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 perceptron designed to require minimal preprocessing [1]. Convolution networks were inspired by biological processes [2].

A CNN consists of an input layer, a output layer and multiple hidden layers. The hidden layers can be either convolution(CNV), pooling(PL) or fully connected(FC). We will look into each of these layers in detail in the following section.

3.1.1 Convolution Layer (CNV):

The Convolution Layer is the core building block of a Convolution Network that does most of the computation heavy lifting. The CNV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a CNV might have size 5(W)x5(H)x3(D). During the forward pass, we slide(convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume. An example of convolution operation is shown in figure below:



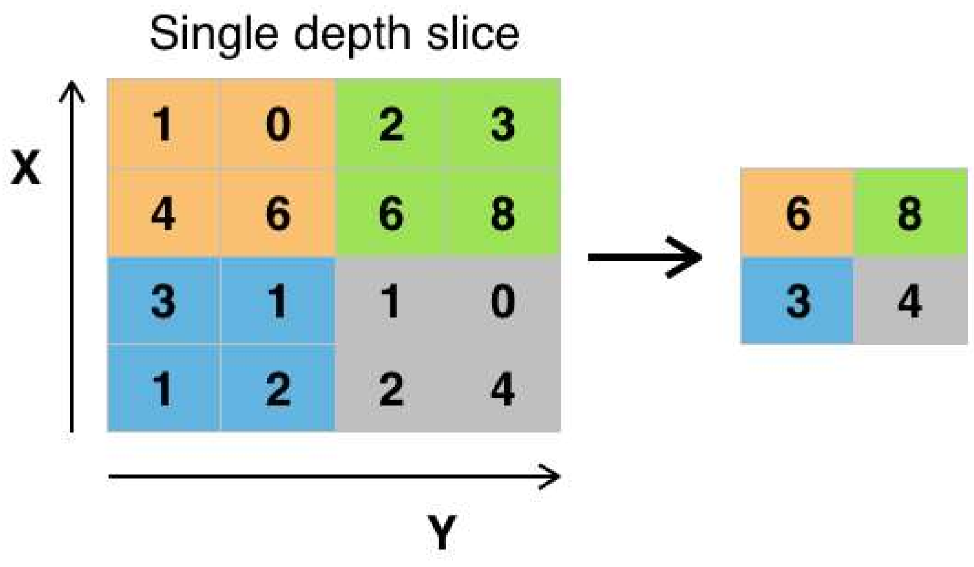
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

3.1.2 Activation Layer

After each conv layer, it is convention to apply non-linear layer (or activation layer) immediately afterward. The purpose of this layer is to introduce nonlinearity to a system that basically has just been computing linear operations during the conv layers (just element wise multiplications and summations). There are many type of activation functions like sigmoid, tanh and ReLU but researchers found out that ReLU layers work far better because the network is able to train a lot faster without making a significant difference to the accuracy. The tanh and sigmoid function suffers from vanishing gradient problem because of the slope being too low for high numbers. ReLU doesn't suffer from this problem and hence it trains faster. The ReLU[3] layer applies the function $f(x) = \max(0, x)$ to all of the values in the input volume. In basic terms, this layer just changes all the negative activations to 0. This layer increases the nonlinear properties of the model and the overall network without affecting the receptive fields of the CNV layer.

3.1.3 Pooling Layer (PL):

After some ReLU layers, programmers may choose to apply a pooling layer. It is also referred to as a downsampling layer. In this category, there are also several layer options, with maxpooling being the most popular. This basically takes a filter (normally of size 2x2) and a stride of the same length. It then applies it to the input volume and outputs the maximum number in every subregion that the filter convolves around.



Example of Maxpool with a 2x2 filter and a stride of 2

Other options for pooling layers are average pooling and L2-norm pooling. The intuitive reasoning behind this layer is that once we know that a specific feature is in the original input volume (there will be a high activation value), its exact location is not as important as its relative location to the other features.

3.1.4 Fully-connected layer (FC):

After several convolution and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

3.1.5 Classification Layer (CL):

This is typically final layer of a CNN that transforms all the net activations in the final output layer to a series of values that can be interpreted as probabilities. One of the most common function is softmax function. The property of the softmax function is that it makes sure that the total probability of all the possible classes is 1.

4. Evaluation Metrics

(approx. 1-2 paragraphs)

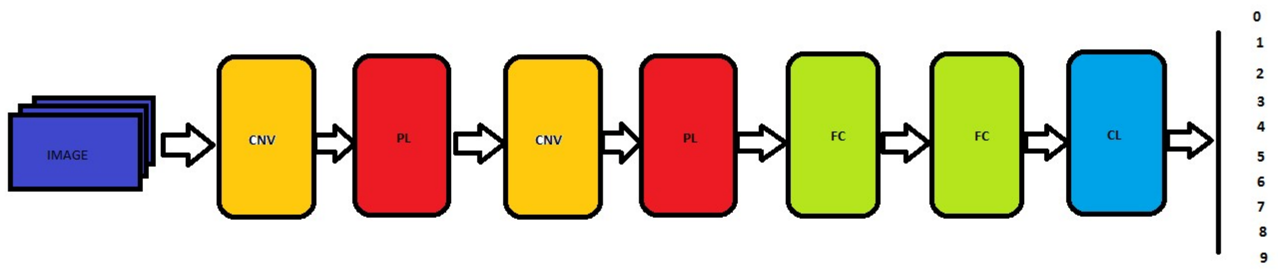
The evaluation metric used in this work is *accuracy*. The accuracy is defined as follows:

$$Accuracy = \frac{\text{Number of images correctly predicted}}{\text{Total images in the dataset}}$$

5. Project Design

5.1 Network Architecture

In this work, we use a CNN that uses 2 CNV layers followed by 2 PL layers which is followed by 2 FC layers as illustrated in figure below:



The above model receives 28x28 image inputs from the MNIST dataset. I have used 32 filters of kernel size 5x5 in the first convolution layer and 64 filters of kernel size 5x5 in the second convolution layer. The parameters for the pooling layer is pooling layer of 2x2 with a stride of 2. There are 1024 neurons used in the first fully connected layer and 10 neurons in the second fully connected layer. In the classification layer, a softmax function is used to convert the output of second fully connected layer into probabilities values. Based on these probabilities, one with the highest probability is taken as the prediction.

5.2 Neural Network Framework

Managing a neural network involves multiple steps:

1. Forward propagation.
2. Loss Calculation
3. Gradient Descent to figure out the new weights.
4. Backward propagation to propagate the updated values to all the layers in the system.

All these steps are essential and important for any architecture design. To simplify the design of any architecture, there are many open source frameworks that takes care of all these nuances and let the research focus on the more important part of identifying the right parameters and hyperparameters.

With so many choice of these framework, it is difficult to pick one from them. We have evaluated multiple popular frameworks and selected TensorFlow as the choice of our framework. This is keeping in mind the long term personal goal of building a prototype for Autonomous vehicle. The frameworks evaluated are as follows:

1. Caffe
2. Neon
3. TensorFlow
4. Theano
5. Torch

The detailed study of Bahrampour et. al. in [4]

5.3 Training

For the training, we have divided the 55,000 images into batche size of 100 and then use the tensorflow framework to do the training. The model gives a good result with 20000 steps of backward propagation.

5.3.1 Handle Overfitting

Typically, all such complex models suffer from the problem of over-fitting and it is one of the common problems in creating a complex neural network. To avoid this problem, I have used drop-out mechanism to do regularization. In this mechanism, we randomly drop some of the data going from one layer to another randomly with certain probability. We have a used a dropout probability of 40% in our architecture. Other methods of regularization is increasing the number of training data. Since we are using a standard database, we do not have additional dataset to use this technique. Hence, we resort to using regularization using drop-out mechanism.

6. Conclusion

We picked a simple CNN algorithm and trained it with MNIST dataset to do classification of hand-written digits. There had been many linear regression, k-means clustering and other techniques to do this classification in the past but the CNN algorithm is one of the best way to do image classification. This is proved through our experiment that shows the accuracy of 98.41%.

Apart from this, this work achieves the goal of modeling this architecture in the tensorflow framework. Usage of a framework like tensorflow relaxes the researcher from the small details of managing a complex neural network and he can focus on identifying the right architecture for any complex problem.

All these lays a strong foundation for me to take me closer to my ultimate goal of coming up with a CNN that gives very good accuracy for the Object Detection for a Autonomous Vehicle and then this technique/tool is so generic that it can be applied in multiple fields of interest.

7. References

1. LeCun, Yann. *LeNet-5, convolution neural networks.*
2. Matusugu, Masakazu; Katshhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). *Subject independent facial expression recognition with robust face detection using a convolution neural network.* Neural Networks. 16 (5): 555-559.
3. Vinod, Nair; Geoffrey E. Hinton. *Rectified linear units improve restricted Boltzmann machines.* ICML' 10 Proceedings of the 27th International Conference on International Conference on Machine Learning, June 21- 24, 2010, USA.
4. Soheil, Bahrampour; Naveen Ramakrishnan; Lukas, Schott; Mohak, Shah. *Comparative Study of Deep Learning Software Frameworks.* arXiv preprint arXiv; 1511.06435, 2015.