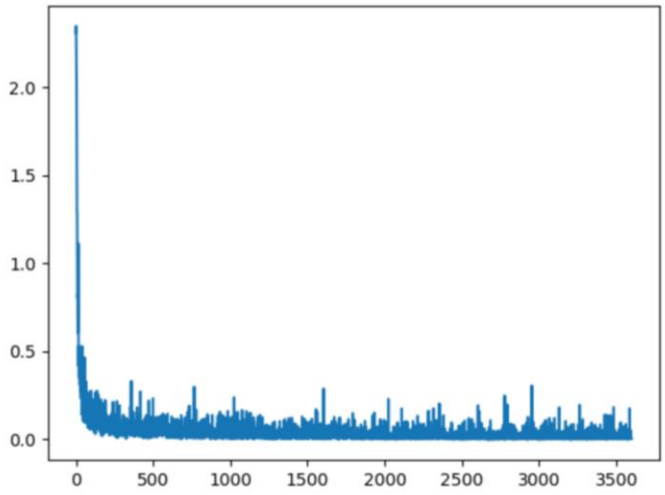
Project 2 Week 1

Project 2 Problem 1:

Figure:



Here in number of iterations is represented in X-axis and Y-axis represented error percentage.

Citation:

Wan Zhu**, "Classification of MNIST Handwritten Digit Database using Neural Network".**

Figure: 7, Page: 5

Url: http://users.cecs.anu.edu.au/~Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_117.pdf

Problem Setting of Problem 1:

Here we will use MNIST handwritten digit recognition having the pixel value of all handwritten digits. We will be going to use CNN, LeNet-5 model, to classify the digits. The final output will be a figure where the scenario of error vs iteration will be represented. So here we can learn LeNet-5 algorithm, the way to implement activation function, pooling, dropout.

Data Source of Problem 1:

Here problem is to classify the handwritten digits and measure the error base on iteration size which helps to find out efficient way to classify handwritten digits. Here we are going to use MNIST data set.

Data Set:

MNIST Handwritten digit dataset

Developed by: Yann LeCun, Corinna Cortes, and Christopher J.C. Burges

Url: http://yann.lecun.com/exdb/mnist/

Dataset Dimension and attributes:

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. In the dataset, Images of digits were taken from a

variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required. Each image is a 28 by 28 pixel square (784 pixels total). Each sample image is 28x28 and linearized as a vector of **size** 1x784. So, the training and test **datasets** are 2-d vectors of **size** 60000x784 and 10000x784 respectively It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict.

The MNIST digits are grayscale images, with each pixel represented as a single intensity value in the range 0 (black) to 1 (white). You can think of the whole image as consisting of 784 numbers arranged in a plane of 28 rows and 28 columns. For color (RGB) images, however, each pixel consists of three numbers (one for Red intensity, one for Green, and one for Blue). Therefore color images are represented as arrays of shape *rows* × *columns* × 3, where the 3 indicates the *depth* of the image. For consistency, the grayscale MNIST images are treated as images of depth 1, with shape *rows* × *columns* × 1

Input X features:

Input is image in a array from. The input image is a 28 × 28 × 1 array of floating-point numbers representing grayscale intensities ranging from 0 (black) to 1 (white)

Input Summary will be

shape : (28, 28, 1) ,range : (0.0, 1.0)

Output Y label:

In output/ target it shows the classification from 0 to 9

So, the output / target summary will be

shape : (10,) range : (0.0, 1.0)

Algorithm of Project 1

Pseudo Code:

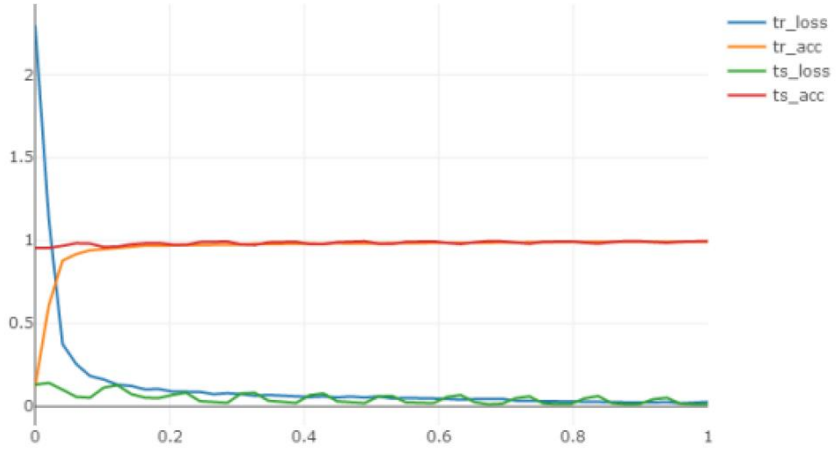
1. **Process Data to train and test**
2. **Implement 2-D CNN**
   Conv: 28 filters, kernel (3, 3), stride (1, 1), Padding (1, 1)
3. **Implementing ReLu activation function**
   1/(1+e-z)
4. **Implementing Pooling**
   kernel = 2, stride = 2, padding=0
5. **Repeat Step 2-4 again**
6. **Implement droupout**
7. **Implement linear function again**
8. **Measure accuracy**
9. **Measure loss**
10. **Plot the figure**

I am going to use keras/ pytorch to implement the CNN. When I will implement keras, tensorflow will be run in the background. So, it will help to learn this libraries. Also I will learn how to implement deep

learning to classify the images which will help me to implement deep learning algorithm to my research work.

Project 2 Problem 2:

Figure:



The loss function and accuracy in training and test process of Deep CNN

Citation:

Yan Wen et al. **, "A Novel Deep Convolutional Neural Network Structure for Off-line Handwritten Digit Recognition"**

Figure: 5, Page: 3.

Url: https://dl.acm.org/citation.cfm?id=3358585

Problem Setting of Problem 2:

Here we will use MNIST handwritten digit recognition having the pixel value of all handwritten digits. We will be going to use Deep CNN to classify the digits for different size of data set. The final output will be a figure where the scenario of error and accuracy vs iteration will be represented. For two test we will take two different size of dataset. In the following figure the different size of data set is represented.

Set	Count	Size(byte)
Train-images	60000	9912422
Train-labels	60000	28881
Test-images	10000	1648877
Test-labels	10000	4542

So here we can learn Deep CNN algorithm, the way to implement activation function, pooling, dropout and how to use GPU.

Data Source of Problem 2:

Here problem is to classify the handwritten digits and measure the error base on iteration size which helps to find out efficient way to classify handwritten digits. Here we are going to use MNIST data set.

Project 2 Week 1

Data Set:

MNIST Handwritten digit dataset

Developed by: Yann LeCun, Corinna Cortes, and Christopher J.C. Burges

Url: http://yann.lecun.com/exdb/mnist/

Dataset Dimension and attributes:

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. In the dataset, Images of digits were taken from a variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required. Each image is a 28 by 28 pixel square (784 pixels total). Each sample image is 28x28 and linearized as a vector of **size** 1x784. So, the training and test **datasets** are 2-d vectors of **size** 60000x784 and 10000x784 respectively It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict.

The MNIST digits are grayscale images, with each pixel represented as a single intensity value in the range 0 (black) to 1 (white). You can think of the whole image as consisting of 784 numbers arranged in a plane of 28 rows and 28 columns. For color (RGB) images, however, each pixel consists of three numbers (one for Red intensity, one for Green, and one for Blue). Therefore color images are represented as arrays of shape *rows* × *columns* × 3, where the 3 indicates the *depth* of the image. For consistency, the grayscale MNIST images are treated as images of depth 1, with shape *rows* × *columns* × 1

Input X features:

Input is image in a array from. The input image is a 28 × 28 × 1 array of floating-point numbers representing grayscale intensities ranging from 0 (black) to 1 (white)

Input Summary will be

shape : (28, 28, 1) ,range : (0.0, 1.0)

Output Y label:

In output/ target it shows the classification from 0 to 9

So, the output / target summary will be

shape : (10,) range : (0.0, 1.0

Algorithm of Project 2:

Pseudo Code:

1. **Process Data to train and test**
2. **Implement 2-D CNN**
   Conv: 28 filters, kernel (3, 3), stride (1, 1), Padding (1, 1)
3. **Implementing ReLu activation function**
   1/(1+e-z)
4. **Implementing Pooling**

kernel = 2, stride = 2, padding=0

5. **Repeat Step 2-4 again**
6. **Repeat Step 2-3, 2 times**
7. **Repeat Step 2-4 again**
8. **Implement dropout**
   P = 0.5
9. **Implement linear function**
   $z=w \cdot x+b$
10. **Implementing ReLu activation function**
    1/(1+e-z)
11. **Implement Step 8-10 again**
12. **Implement linear function again**
13. **Measure accuracy**
14. **Measure loss**
15. **Plot the figure**

This project is similar to first one but here we need to implement a deep CNN. To implement deep CNN we need to implement multiple Convolution, relu and pooling function. That's why this project requires high computational power. We may need to use two GPUs to implement the project.

I am going to use keras/ pytorch to implement the CNN. When I will implement keras, tensorflow will be run in the background. So, it will help to learn this libraries. Also I will learn how to implement deep learning to classify the images which will help me to implement deep learning algorithm to my research work.