

**Program:**

Digit\_reg.py:

import numpy as np import mnist\_loader as ml import Network as net import testi as tt import bar as br np.random.seed(1)

weights = 2\*np.random.random((784,50)) - 1 weights1 = 2\*np.random.random((50,10)) - 1

tr\_data, val\_data, test\_data = ml.load\_data()

tr\_inputs = [np.reshape(x, (784, 1)) for x in tr\_data[0]]

tr\_outputs = [ml.vectorized\_result(x) for x in tr\_data[1]]

for i in range(50000):

weights , weights1 = net.train(tr\_inputs[i],tr\_outputs[i],weights,weights1) if(i % 500) == 0 :

br.progress(i, 50000)

br.progress(50000, 50000, cond = True)

print ("\n") print ("Network Trained and ready to be operated")

te\_inputs = [np.reshape(x, (784,1)) for x in test\_data[0]] te\_outputs = test\_data[1] tt.check(te\_inputs,te\_outputs,weights,weights1)

mnist\_loader.py:

import pickle import gzip import numpy as np

def load\_data(): f = gzip.open('mnist.pkl.gz', 'rb') training\_data, validation\_data, test\_data = pickle.load(f, encoding = 'latin1') f.close() return (training\_data, validation\_data, test\_data)

def vectorized\_result(j): e = np.zeros((10, 1)) e[j] = 1.0 return e

Network.py:

import numpy as np

def sigmoid(x): return 1/(1+np.exp(-x))

def deriv\_sigmoid(x): return x\*(1-x) def train(inputs,output,weights,weights1):

x = inputs.T y = output.T

l1 = sigmoid(np.dot(x,weights)) l2 = sigmoid(np.dot(l1,weights1))

error = y - l2

l2\_del = error \* deriv\_sigmoid(l2)

error0 = l2\_del.dot(weights1.T)

l1\_del = error0 \* deriv\_sigmoid(l1)

weights1 += np.dot(l1.T,l2\_del) weights += np.dot(x.T,l1\_del) return weights,weights1

bar.py:

import sys

def progress(count, total, cond=False):

bar\_len = 60 filled\_len = int(round(bar\_len \* count / float(total)))

percents = round(100.0 \* count / float(total), 1) bar = '|' \* filled\_len + '-' \* (bar\_len - filled\_len)

if cond == False: sys.stdout.write('[%s] %s%s\r' % (bar, percents, '%')) sys.stdout.flush()

else: sys.stdout.write('[%s] %s%s' % (bar, percents, '%'))

testi.py:

import Network as net import numpy as np

def feedforward(x,weights,weights1):

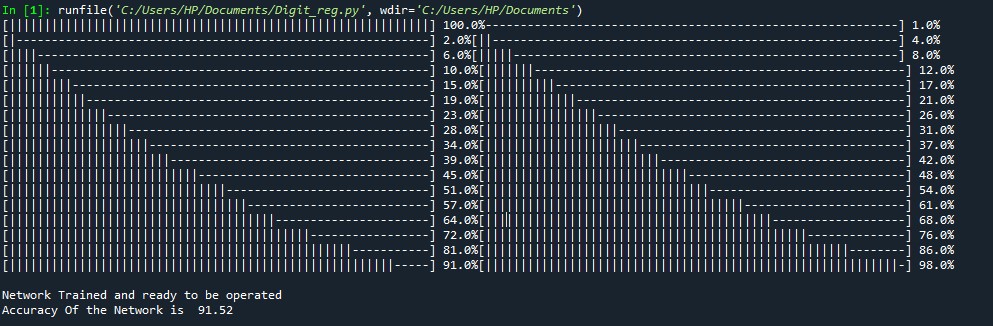
l = x.T l1 = net.sigmoid(np.dot(l,weights)) l2 = net.sigmoid(np.dot(l1,weights1)) return l2; def check(te\_inputs,te\_outputs,weights,weights1):

correct = 0

for i in range(len(te\_inputs)):

out = feedforward(te\_inputs[i],weights,weights1) f\_out = np.argmax(out) if(f\_out == te\_outputs[i]): correct += 1 print ("Accuracy Of the Network is " , ((correct/10000)\*100))

**Output:**



**Result:**

Thus using MNIST dataset handwritten digit recognition is implemented using ANN with backpropagation algorithm.