## AML ASSIGNMENT 2 – IT1761

**Assignment Name:** Impact of number of hidden neurons to model performance

**Name:** Ankit Kumar

**Reg. Number:** 159102017

**Phone number:** +91-8809149699

**E-mail:** [ankitk0711@gmail.com](mailto:ankitk0711@gmail.com)

## ABSTRACT

Different number of hidden layers define different learning ability for a neural network. The idea is to make a neural network simple to understand. Number of neurons plays a vital role in the hidden layer. When a neural network has too few hidden neurons (< 16), it does not have the capacity to learn enough of the underlying patterns to distinguish between 0 - 9 effectively. When the neural network has >= 16 neurons, the neural network start to do better. At increasing number of hidden neurons (>= 128), the number of hidden neurons does not help too much for this problem.

## INTRODUCTION

**Machine Learning**

Machine Learning is that branch of computer studies that gives the potentiality to the computer to grasp without being characteristically programmed. Machine learning is employed in a wide range of computing functions where building and designing specific algorithms with better performances is difficult or impractical. Machine Learning is also firmly attached to computational statistics which makes prediction through computers easier and feasible. In commercial terms Predictive Analysis is machine learning used to design multiple algorithms and models that helps the process of prediction. Here the machine learns itself and divide the data provided into the levels of prediction and in a very short period of time gives the accurate results.

**Deep Learning**

Deep learning is a part of the broader family of machine learning wherein the learning can be supervised, unsupervised or semi supervised. Deep learning unlike machine learning uses a large dataset for the learning process and the number of classifiers used gets reduced substantially. The training time for the deep learning algorithm increases because of the usage of the very large dataset. Deep learning algorithm chooses its own features unlike the machine learning making the prediction process easier for the end user as it does not use much of pre-processing.

## ALGORITHM

The algorithm I used in my assignment is **Artificial Neural Network**. I increased the numbers of neurons in the hidden layers. A Neural Network is a computational model that works in a similar way to the neurons in the human brain. Each neuron takes an input, performs some operations then passes the output to the following neuron.

## DATASET

I collected the dataset from the Kaggle competition (DIGIT RECOGNIZER). The link to the dataset is: <https://www.kaggle.com/c/digit-recognizer/data>

The data files train.csv and test.csv contain gray-scale images of hand-drawn digits, from zero through nine.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

The training data set, (train.csv), has 785 columns. The first column, called "label", is the digit that was drawn by the user. The rest of the columns contain the pixel-values of the associated image.

Each pixel column in the training set has a name like pixelx, where x is an integer between 0 and 783, inclusive. To locate this pixel on the image, suppose that we have decomposed x as x = i \* 28 + j, where i and j are integers between 0 and 27, inclusive. Then pixelx is located on row i and column j of a 28 x 28 matrix, (indexing by zero).

The test data set, (test.csv), is the same as the training set, except that it does not contain the "label" column.

## METHODOLOGY

## Impact of number of hidden neurons to model performance

This notebook investigates how the number of hidden neurons affect the model performance. We will see that increasing the number of hidden neurons increases the performance of a model using the MNIST dataset. The MNIST dataset is a common standard dataset used to evaluate machine learning models performance, which is just a task of recognizing digits from 0 to 9.

This notebook has dependencies on Keras, Scikit-Learn and MatPlotLib.

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
%matplotlib inline  
  
from keras.models import Sequential  
from keras.layers.core import Dense, Activation, Dropout  
from keras.optimizers import SGD, Adam, RMSprop  
from sklearn.preprocessing import \*  
from sklearn.cross\_validation import \*  
from sklearn.metrics import \*

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/sklearn/cross\_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model\_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.  
 "This module will be removed in 0.20.", DeprecationWarning)

TRAIN\_FILE = 'data/train.csv'  
TEST\_FILE = 'data/test.csv'

train\_data = np.loadtxt(TRAIN\_FILE, skiprows = 1, delimiter = ',', dtype = 'float')  
X = train\_data[:, 1:]  
  
# Preprocess the data to make features fall between 0 and 1. Neural networks perform a lot better in this way.  
  
X = X/255  
raw\_Y = train\_data[:, 0].reshape(-1, 1)

X\_test = np.loadtxt(TEST\_FILE, skiprows = 1, delimiter = ',', dtype = 'float')  
  
# Preprocess the data to make features fall between 0 and 1. Neural networks perform a lot better in this way.  
  
X\_test = X\_test/255

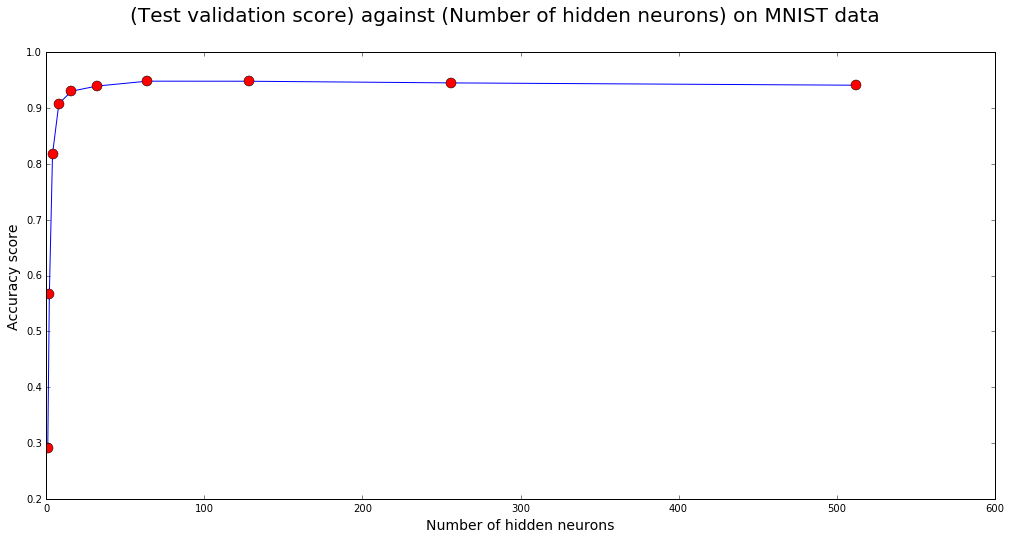
X\_train, X\_cv, raw\_Y\_train, raw\_Y\_cv = train\_test\_split(X, raw\_Y, test\_size = 0.20)  
  
# Converter to transform input into one hot encoding, i.e. [3] => [0, 0, 1, 0, 0, 0, 0, 0, 0, 0].  
# Can use the np\_utils from Keras instead.  
  
Y\_expander = OneHotEncoder().fit(raw\_Y)  
Y\_train = Y\_expander.transform(raw\_Y\_train).astype(int).toarray()  
Y\_cv = Y\_expander.transform(raw\_Y\_cv).astype(int).toarray()

n\_hiddens = [512, 256, 128, 64, 32, 16, 8, 4, 2, 1]  
scores = []  
for n\_hidden in n\_hiddens:  
 # Build a simple neural network.  
 model = Sequential()  
 model.add(Dense(input\_dim = X.shape[1], output\_dim = n\_hidden))  
 model.add(Activation('tanh'))  
 model.add(Dense(output\_dim = 10))  
 model.add(Activation('softmax'))  
 sgd = SGD(lr=0.2, decay=1e-7, momentum=0.1, nesterov=True)  
 model.compile(loss='categorical\_crossentropy', optimizer='sgd')  
  
 model.fit(X\_train, Y\_train, nb\_epoch = 10, batch\_size = 10, show\_accuracy = True, verbose = 1, validation\_split = 0.05)  
 Y\_cv\_pred = model.predict\_classes(X\_cv, batch\_size = 10, verbose = 1)  
  
 score = accuracy\_score(raw\_Y\_cv, Y\_cv\_pred)  
 scores.append(score)  
 print('Using [%d] number of hidden neurons yields. Accuracy score: %.4f' % (n\_hidden, score))  
 print('')

Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.4771 - acc: 0.8713 - val\_loss: 0.3271 - val\_acc: 0.9054  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3105 - acc: 0.9121 - val\_loss: 0.2945 - val\_acc: 0.9119  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.2806 - acc: 0.9207 - val\_loss: 0.2707 - val\_acc: 0.9185  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2605 - acc: 0.9250 - val\_loss: 0.2533 - val\_acc: 0.9232  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2416 - acc: 0.9320 - val\_loss: 0.2368 - val\_acc: 0.9262  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.2240 - acc: 0.9367 - val\_loss: 0.2233 - val\_acc: 0.9321  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.2068 - acc: 0.9407 - val\_loss: 0.2104 - val\_acc: 0.9357  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.1913 - acc: 0.9461 - val\_loss: 0.1997 - val\_acc: 0.9387  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.1771 - acc: 0.9506 - val\_loss: 0.1872 - val\_acc: 0.9411  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1640 - acc: 0.9540 - val\_loss: 0.1785 - val\_acc: 0.9429  
8400/8400 [==============================] - 0s   
Using [512] number of hidden neurons yields. Accuracy score: 0.9412  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.4966 - acc: 0.8655 - val\_loss: 0.3360 - val\_acc: 0.9048  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3118 - acc: 0.9112 - val\_loss: 0.2957 - val\_acc: 0.9137  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.2772 - acc: 0.9205 - val\_loss: 0.2687 - val\_acc: 0.9155  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2527 - acc: 0.9278 - val\_loss: 0.2465 - val\_acc: 0.9292  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2305 - acc: 0.9350 - val\_loss: 0.2358 - val\_acc: 0.9286  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.2110 - acc: 0.9404 - val\_loss: 0.2155 - val\_acc: 0.9357  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.1932 - acc: 0.9459 - val\_loss: 0.2009 - val\_acc: 0.9393  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.1774 - acc: 0.9512 - val\_loss: 0.1895 - val\_acc: 0.9440  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.1632 - acc: 0.9552 - val\_loss: 0.1836 - val\_acc: 0.9446  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1510 - acc: 0.9592 - val\_loss: 0.1714 - val\_acc: 0.9476  
8400/8400 [==============================] - 0s   
Using [256] number of hidden neurons yields. Accuracy score: 0.9454  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.5168 - acc: 0.8630 - val\_loss: 0.3357 - val\_acc: 0.9030  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3111 - acc: 0.9107 - val\_loss: 0.2855 - val\_acc: 0.9125  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.2703 - acc: 0.9223 - val\_loss: 0.2552 - val\_acc: 0.9220  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2401 - acc: 0.9312 - val\_loss: 0.2340 - val\_acc: 0.9351  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2161 - acc: 0.9386 - val\_loss: 0.2131 - val\_acc: 0.9435  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.1956 - acc: 0.9452 - val\_loss: 0.2018 - val\_acc: 0.9429  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.1791 - acc: 0.9503 - val\_loss: 0.1865 - val\_acc: 0.9476  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.1644 - acc: 0.9544 - val\_loss: 0.1789 - val\_acc: 0.9476  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.1518 - acc: 0.9578 - val\_loss: 0.1695 - val\_acc: 0.9476  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1412 - acc: 0.9619 - val\_loss: 0.1625 - val\_acc: 0.9512  
8400/8400 [==============================] - 0s   
Using [128] number of hidden neurons yields. Accuracy score: 0.9483  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.5449 - acc: 0.8554 - val\_loss: 0.3485 - val\_acc: 0.9012  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3154 - acc: 0.9113 - val\_loss: 0.2837 - val\_acc: 0.9202  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.2685 - acc: 0.9234 - val\_loss: 0.2516 - val\_acc: 0.9214  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2377 - acc: 0.9321 - val\_loss: 0.2304 - val\_acc: 0.9315  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2135 - acc: 0.9388 - val\_loss: 0.2114 - val\_acc: 0.9357  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.1942 - acc: 0.9449 - val\_loss: 0.1982 - val\_acc: 0.9423  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.1785 - acc: 0.9501 - val\_loss: 0.1849 - val\_acc: 0.9423  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.1651 - acc: 0.9538 - val\_loss: 0.1764 - val\_acc: 0.9470  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.1532 - acc: 0.9571 - val\_loss: 0.1712 - val\_acc: 0.9458  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1433 - acc: 0.9596 - val\_loss: 0.1581 - val\_acc: 0.9512  
8400/8400 [==============================] - 0s   
Using [64] number of hidden neurons yields. Accuracy score: 0.9485  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.5879 - acc: 0.8494 - val\_loss: 0.3675 - val\_acc: 0.8982  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3271 - acc: 0.9102 - val\_loss: 0.2959 - val\_acc: 0.9149  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.2780 - acc: 0.9222 - val\_loss: 0.2647 - val\_acc: 0.9226  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2481 - acc: 0.9306 - val\_loss: 0.2408 - val\_acc: 0.9333  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2257 - acc: 0.9376 - val\_loss: 0.2268 - val\_acc: 0.9381  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.2085 - acc: 0.9421 - val\_loss: 0.2116 - val\_acc: 0.9440  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.1940 - acc: 0.9470 - val\_loss: 0.2033 - val\_acc: 0.9452  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.1816 - acc: 0.9498 - val\_loss: 0.1938 - val\_acc: 0.9482  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.1714 - acc: 0.9526 - val\_loss: 0.1903 - val\_acc: 0.9482  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1628 - acc: 0.9547 - val\_loss: 0.1879 - val\_acc: 0.9458  
8400/8400 [==============================] - 0s   
Using [32] number of hidden neurons yields. Accuracy score: 0.9396  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.6751 - acc: 0.8353 - val\_loss: 0.4077 - val\_acc: 0.8929  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.3600 - acc: 0.9049 - val\_loss: 0.3196 - val\_acc: 0.9101  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.3040 - acc: 0.9168 - val\_loss: 0.2844 - val\_acc: 0.9202  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.2735 - acc: 0.9251 - val\_loss: 0.2576 - val\_acc: 0.9208  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.2531 - acc: 0.9294 - val\_loss: 0.2460 - val\_acc: 0.9310  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.2372 - acc: 0.9344 - val\_loss: 0.2364 - val\_acc: 0.9315  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.2249 - acc: 0.9367 - val\_loss: 0.2261 - val\_acc: 0.9321  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.2143 - acc: 0.9406 - val\_loss: 0.2254 - val\_acc: 0.9363  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.2054 - acc: 0.9430 - val\_loss: 0.2173 - val\_acc: 0.9369  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.1974 - acc: 0.9453 - val\_loss: 0.2181 - val\_acc: 0.9339  
8400/8400 [==============================] - 0s   
Using [16] number of hidden neurons yields. Accuracy score: 0.9305  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 0.9143 - acc: 0.7975 - val\_loss: 0.5725 - val\_acc: 0.8679  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.4821 - acc: 0.8799 - val\_loss: 0.4302 - val\_acc: 0.8875  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.3984 - acc: 0.8942 - val\_loss: 0.3778 - val\_acc: 0.9000  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.3587 - acc: 0.9017 - val\_loss: 0.3469 - val\_acc: 0.9077  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.3345 - acc: 0.9081 - val\_loss: 0.3352 - val\_acc: 0.9065  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.3178 - acc: 0.9123 - val\_loss: 0.3209 - val\_acc: 0.9131  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.3048 - acc: 0.9143 - val\_loss: 0.3136 - val\_acc: 0.9101  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.2941 - acc: 0.9167 - val\_loss: 0.3106 - val\_acc: 0.9125  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.2862 - acc: 0.9176 - val\_loss: 0.3100 - val\_acc: 0.9131  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.2788 - acc: 0.9202 - val\_loss: 0.3049 - val\_acc: 0.9131  
8400/8400 [==============================] - 0s   
Using [8] number of hidden neurons yields. Accuracy score: 0.9081  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 1.2546 - acc: 0.6927 - val\_loss: 0.8366 - val\_acc: 0.7988  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 0.7538 - acc: 0.7991 - val\_loss: 0.6797 - val\_acc: 0.8185  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 0.6600 - acc: 0.8143 - val\_loss: 0.6277 - val\_acc: 0.8304  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 0.6239 - acc: 0.8230 - val\_loss: 0.6077 - val\_acc: 0.8268  
Epoch 5/10  
31920/31920 [==============================] - 3s - loss: 0.6027 - acc: 0.8256 - val\_loss: 0.5839 - val\_acc: 0.8351  
Epoch 6/10  
31920/31920 [==============================] - 3s - loss: 0.5892 - acc: 0.8306 - val\_loss: 0.5785 - val\_acc: 0.8315  
Epoch 7/10  
31920/31920 [==============================] - 3s - loss: 0.5790 - acc: 0.8316 - val\_loss: 0.5758 - val\_acc: 0.8321  
Epoch 8/10  
31920/31920 [==============================] - 3s - loss: 0.5709 - acc: 0.8342 - val\_loss: 0.5739 - val\_acc: 0.8339  
Epoch 9/10  
31920/31920 [==============================] - 3s - loss: 0.5654 - acc: 0.8342 - val\_loss: 0.5581 - val\_acc: 0.8345  
Epoch 10/10  
31920/31920 [==============================] - 3s - loss: 0.5599 - acc: 0.8374 - val\_loss: 0.5736 - val\_acc: 0.8256  
8400/8400 [==============================] - 0s   
Using [4] number of hidden neurons yields. Accuracy score: 0.8177  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 1.6598 - acc: 0.3836 - val\_loss: 1.4273 - val\_acc: 0.4155  
Epoch 2/10  
31920/31920 [==============================] - 3s - loss: 1.3708 - acc: 0.4233 - val\_loss: 1.3183 - val\_acc: 0.4298  
Epoch 3/10  
31920/31920 [==============================] - 3s - loss: 1.2975 - acc: 0.4403 - val\_loss: 1.2685 - val\_acc: 0.4536  
Epoch 4/10  
31920/31920 [==============================] - 3s - loss: 1.2581 - acc: 0.4591 - val\_loss: 1.2310 - val\_acc: 0.4542  
Epoch 5/10  
31920/31920 [==============================] - 2s - loss: 1.2294 - acc: 0.4855 - val\_loss: 1.2125 - val\_acc: 0.5048  
Epoch 6/10  
31920/31920 [==============================] - 2s - loss: 1.2047 - acc: 0.5040 - val\_loss: 1.1922 - val\_acc: 0.5310  
Epoch 7/10  
31920/31920 [==============================] - 2s - loss: 1.1818 - acc: 0.5237 - val\_loss: 1.2201 - val\_acc: 0.5262  
Epoch 8/10  
31920/31920 [==============================] - 2s - loss: 1.1645 - acc: 0.5443 - val\_loss: 1.1387 - val\_acc: 0.5780  
Epoch 9/10  
31920/31920 [==============================] - 2s - loss: 1.1491 - acc: 0.5534 - val\_loss: 1.1416 - val\_acc: 0.5542  
Epoch 10/10  
31920/31920 [==============================] - 2s - loss: 1.1351 - acc: 0.5653 - val\_loss: 1.1171 - val\_acc: 0.6000  
8400/8400 [==============================] - 0s   
Using [2] number of hidden neurons yields. Accuracy score: 0.5676  
  
Train on 31920 samples, validate on 1680 samples  
Epoch 1/10  
31920/31920 [==============================] - 3s - loss: 1.9758 - acc: 0.2119 - val\_loss: 1.8660 - val\_acc: 0.2583  
Epoch 2/10  
31920/31920 [==============================] - 2s - loss: 1.8501 - acc: 0.2623 - val\_loss: 1.8218 - val\_acc: 0.3000  
Epoch 3/10  
31920/31920 [==============================] - 2s - loss: 1.8202 - acc: 0.2602 - val\_loss: 1.7980 - val\_acc: 0.3018  
Epoch 4/10  
31920/31920 [==============================] - 2s - loss: 1.8035 - acc: 0.2698 - val\_loss: 1.7855 - val\_acc: 0.2565  
Epoch 5/10  
31920/31920 [==============================] - 2s - loss: 1.7889 - acc: 0.2808 - val\_loss: 1.7755 - val\_acc: 0.3083  
Epoch 6/10  
31920/31920 [==============================] - 2s - loss: 1.7775 - acc: 0.2811 - val\_loss: 1.7672 - val\_acc: 0.3357  
Epoch 7/10  
31920/31920 [==============================] - 2s - loss: 1.7674 - acc: 0.3117 - val\_loss: 1.7552 - val\_acc: 0.3375  
Epoch 8/10  
31920/31920 [==============================] - 2s - loss: 1.7580 - acc: 0.3014 - val\_loss: 1.7477 - val\_acc: 0.2982  
Epoch 9/10  
31920/31920 [==============================] - 2s - loss: 1.7489 - acc: 0.3145 - val\_loss: 1.7322 - val\_acc: 0.3476  
Epoch 10/10  
31920/31920 [==============================] - 2s - loss: 1.7419 - acc: 0.3189 - val\_loss: 1.8143 - val\_acc: 0.2798  
8400/8400 [==============================] - 0s   
Using [1] number of hidden neurons yields. Accuracy score: 0.2919

# Plot the results for comparison  
  
fig = plt.figure()  
fig.suptitle('(Test validation score) against (Number of hidden neurons) on MNIST data', fontsize = 20)  
fig.set\_figwidth(17)  
fig.set\_figheight(8)  
ax = fig.add\_subplot(111)  
ax.plot(n\_hiddens, scores, '-o', markersize = 10, markerfacecolor = 'r')  
ax.set\_xlabel('Number of hidden neurons', fontsize = 14)  
ax.set\_ylabel('Accuracy score', fontsize = 14)

<matplotlib.text.Text at 0x7f98e4af1e10>



### What can we learn?

* From here we can see the the number of hidden neurons does affect the model performance.
* When a neural network has too few hidden neurons (< 16), it does not have the capacity to learn enough of the underlying patterns to distinguish between 0 - 9 effectively.
* When the neural network has >= 16 neurons, the neural network start to do better. At increasing number of hidden neurons (>= 128), the number of hidden neurons does not help too much for this problem.

## REFERENCES

I went through various research papers for this project.

Digital Open Science Index, Computer and Information Engineering Vol:8, No:11, 2014 waset.org/Publication/10000264

## ANNEXURE

Enclosed in the project folder.