Neural Networks and Deep Learning

CSCI 5922 – Assignment 3 (Fall 2017) by Akshit Arora (108631342)

Part 1:

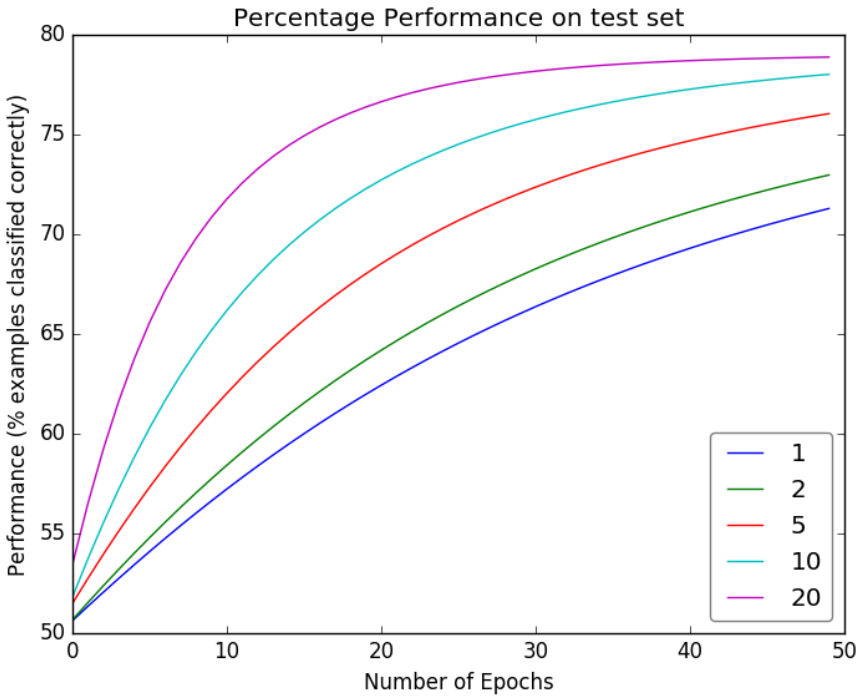
Question 1a (learning rate: 0.001, number of epochs: 50, 5 input units, [1,2,5,10,20] hidden units, 1 output unit)

Code available: ***assign3_p1.py, assign2_p2f.py***

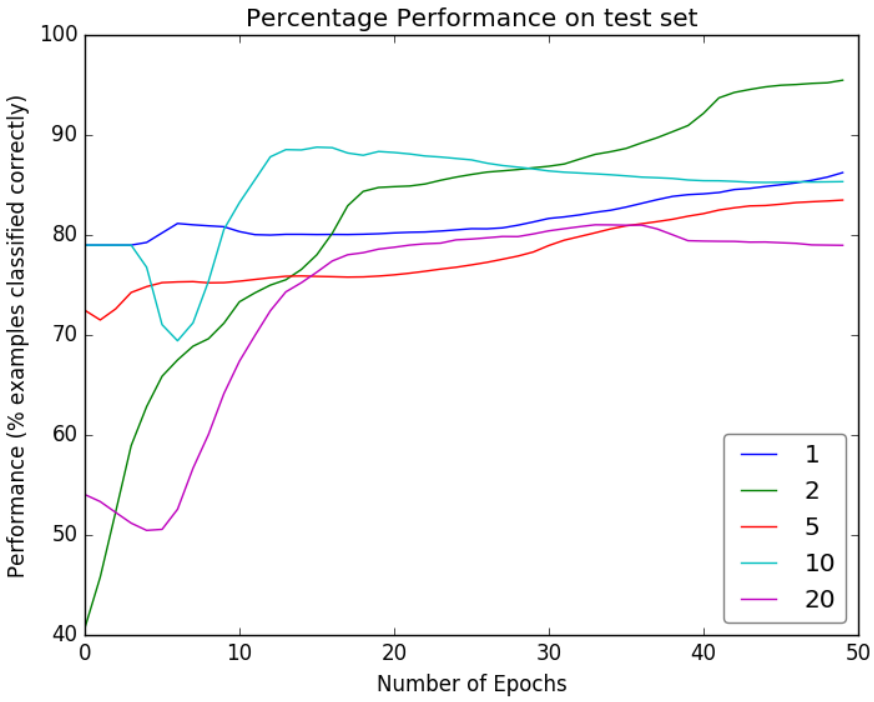
Implemented assignment 2 (part 2f) using tensor flow (used AdamOptimizer, relu activation unit, with mean-squared error function) => **assign3_p1.py**

And made some modifications to my original submission for assignment 2 (part 2f) so that both outputs are comparable (used mini-batch gradient descent, using simple absolute difference between target and output as error function) => **assign2_p2f.py**

Here is the plot I obtained from **assign2_p2f.py:** (naïve implementation; without using Tensorflow)



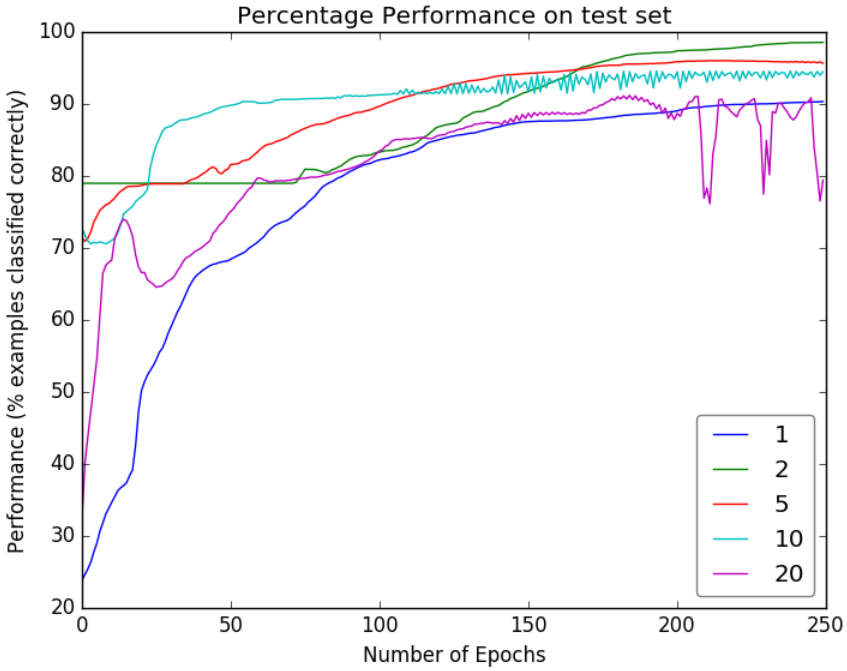
And here is the plot I obtained from **assign3_p1.py**: (using Tensorflow)



Question 1b

Observations:

- Tensorflow's AdamOptimizer does a better job (learning rate decays overtime) than naïve mini-batch gradient descent in which learning rate remains constant throughout the training.
- Also, tensorflow's relu unit performs better than sigmoid units.
- Naïve code for implementing neural network was way slower than tensorflow implementation.
- I used only 50 epochs (because naïve implementation was too slow), maybe increasing the number of epochs give a clearer view for how using different hidden units affect the performance. Have a look here for 250 epochs for tensorflow representation:



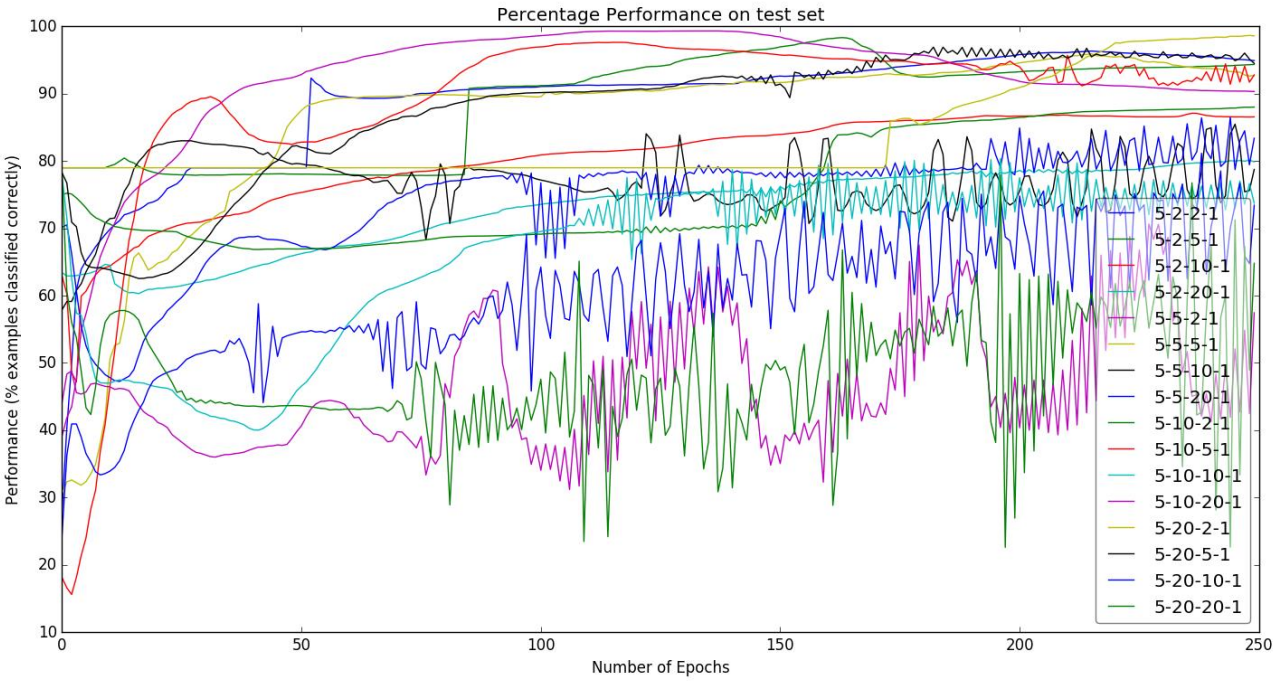
Number of Hidden Units	Percentage Performance on test set at 250th epoch
20	79.42
10	94.473
5	95.662
2	**98.554 (best)**
1	90.34

Table 1: Test set performance of architectures with single hidden layer.

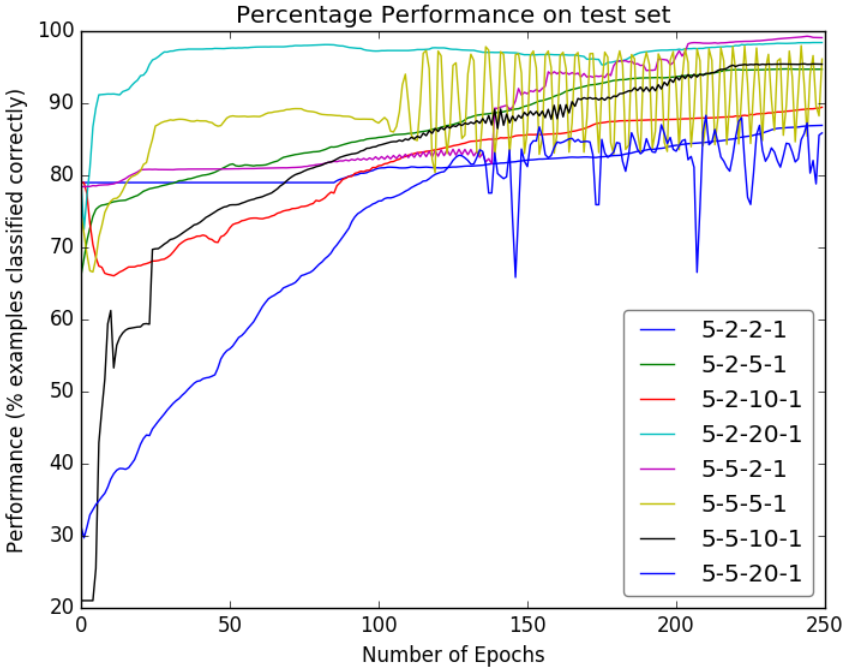
Question 1c (AdamOptimizer, ReLU units, 250 epochs) *Code available: **assign3_p1_v.py***

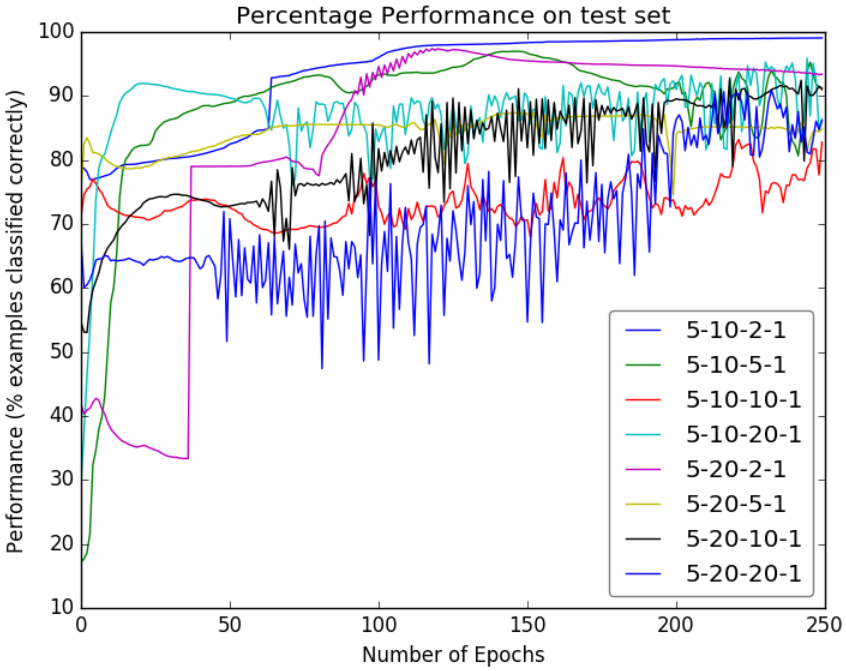
I tried the following architectures: '5-2-2-1','5-2-5-1','5-2-10-1','5-2-20-1','5-5-2-1','5-5-5-1','5-5-10-1','5-5-20-1','5-10-2-1','5-10-5-1','5-10-10-1','5-10-20-1','5-20-2-1','5-20-5-1','5-20-10-1','5-20-20-1'.

This was the result I got for 250 epochs:



To de-clutter it, I divided the graph into two parts:





Architecture	Maximum Percentage Performance on test set
5-2-2-1	86.926 at epoch 249
5-2-5-1	94.749 at epoch 237
5-2-10-1	89.4380639869 at epoch: 249
5-2-20-1	98.4105824446 at epoch: 246
5-5-2-1	**99.2924528302 at epoch: 244 (best)**
5-5-5-1	97.9901558655 at epoch: 242
5-5-10-1	95.4573420837 at epoch: 243
5-5-20-1	88.3100902379 at epoch: 210
5-10-2-1	**99.0668580804 at epoch: 249 (better than table1)**
5-10-5-1	96.9852337982 at epoch: 147
5-10-10-1	83.2034454471 at epoch: 221
5-10-20-1	95.8982772765 at epoch: 244
5-20-2-1	97.4056603774 at epoch: 118
5-20-5-1	87.3974569319 at epoch: 146
5-20-10-1	92.7399507793 at epoch: 218
5-20-20-1	90.9454470878 at epoch: 232

Table 2: Test set performances of architectures with 2 hidden layers.

Observations:

- **5-10-2-1 and 5-5-2-1 architectures outperform single layer architecture (5-2-1)** in the previous part. (compare table 1 and table 2)
- Took the readings at 250 epochs.

Part 2:

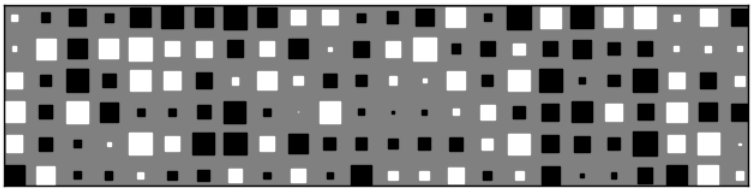
Question 2a (3000 epochs, AdamOptimizer, sigmoid activation function, mean square loss function): *Code available: fix.py*

After implementing Hinton's architecture in tensorflow, I got the following values:

Mean for test set accuracy: **17.33%**, Standard deviation for test set accuracy: **13.04**

Mean for training set accuracy: **51.6%**, Standard deviation for training set accuracy: **15.64**

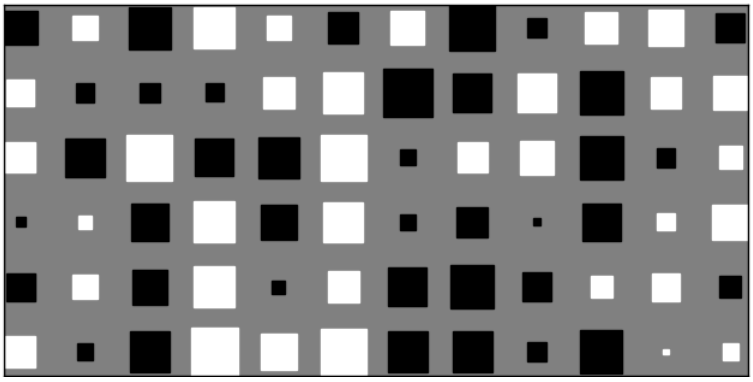
Question 2b & 2c: *Code available: fix_2b.py (run for 8000 epochs)*



Weights from input (36 units) to hidden layer (6 units)

[left to right where F = French and E = English]: ['Alfonso-F' 'Andrew-E' 'Angela-E' 'Arthur-E' 'Charles-E' 'Charlotte-E' 'Christine-E' 'Christopher-E' 'Colin-E' 'Emilio-F' 'Francesca-F' 'Gina-F' 'James-E' 'Jennifer-E' 'Lucia-F' 'Marco-F' 'Margaret-E' 'Maria-F' 'Penelope-E' 'Pierro-F' 'Roberto-F' 'Sophia-F' 'Tomaso-F' 'Victoria-E']

Observation: **The first row "roughly" classifies English vs French! (F EEEEEEEE FFF EE FF E F E FFFF E) French = white and English = black.**



Weights from hidden layer (6 units) to output (24 units).

[Left to right]: ['aunt' 'brother' 'daughter' 'father' 'husband' 'mother' 'nephew' 'niece' 'sister' 'son' 'uncle' 'wife']

Observation: **The first layer represents Gender! (F M F M M F M F F M M F) => F = Female (black) and M = Male (white)**