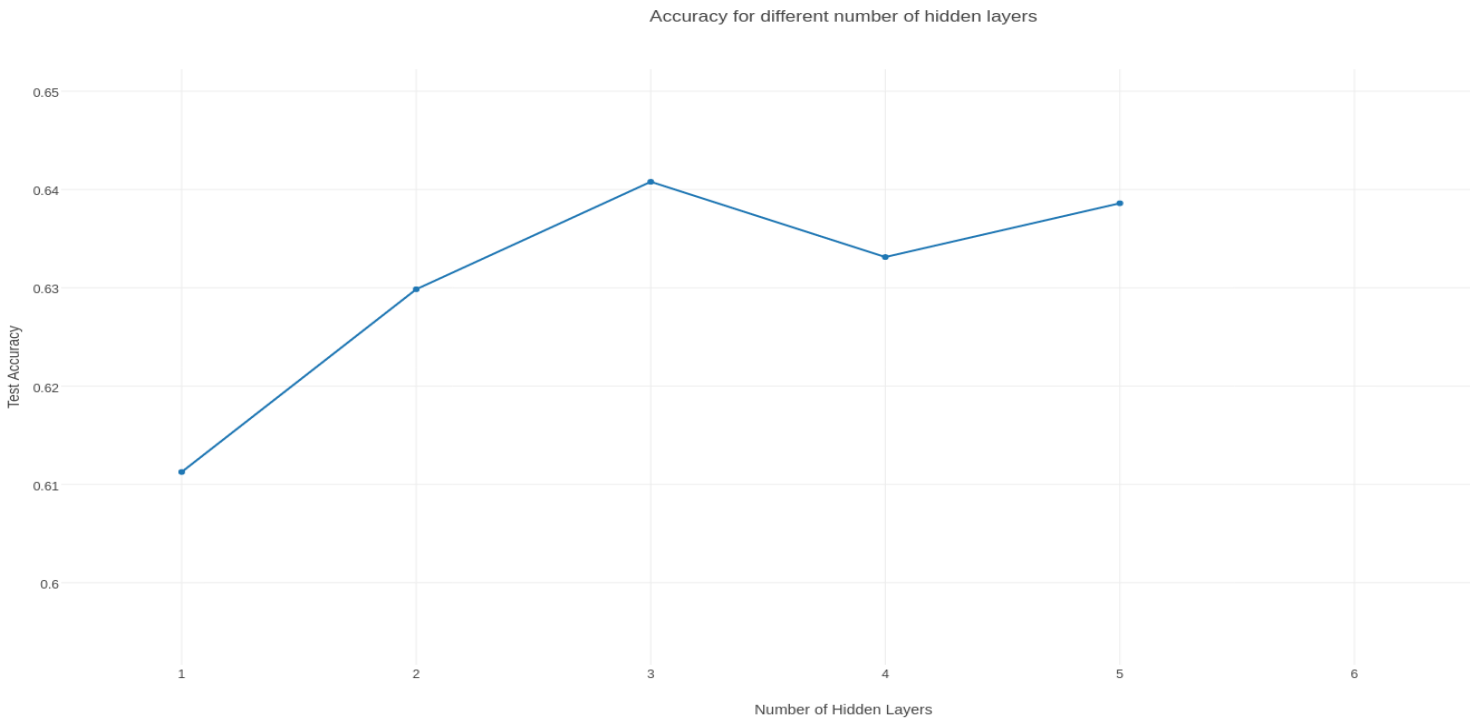
CS726 Assignment 1

Accuracy vs Number of Hidden Layers

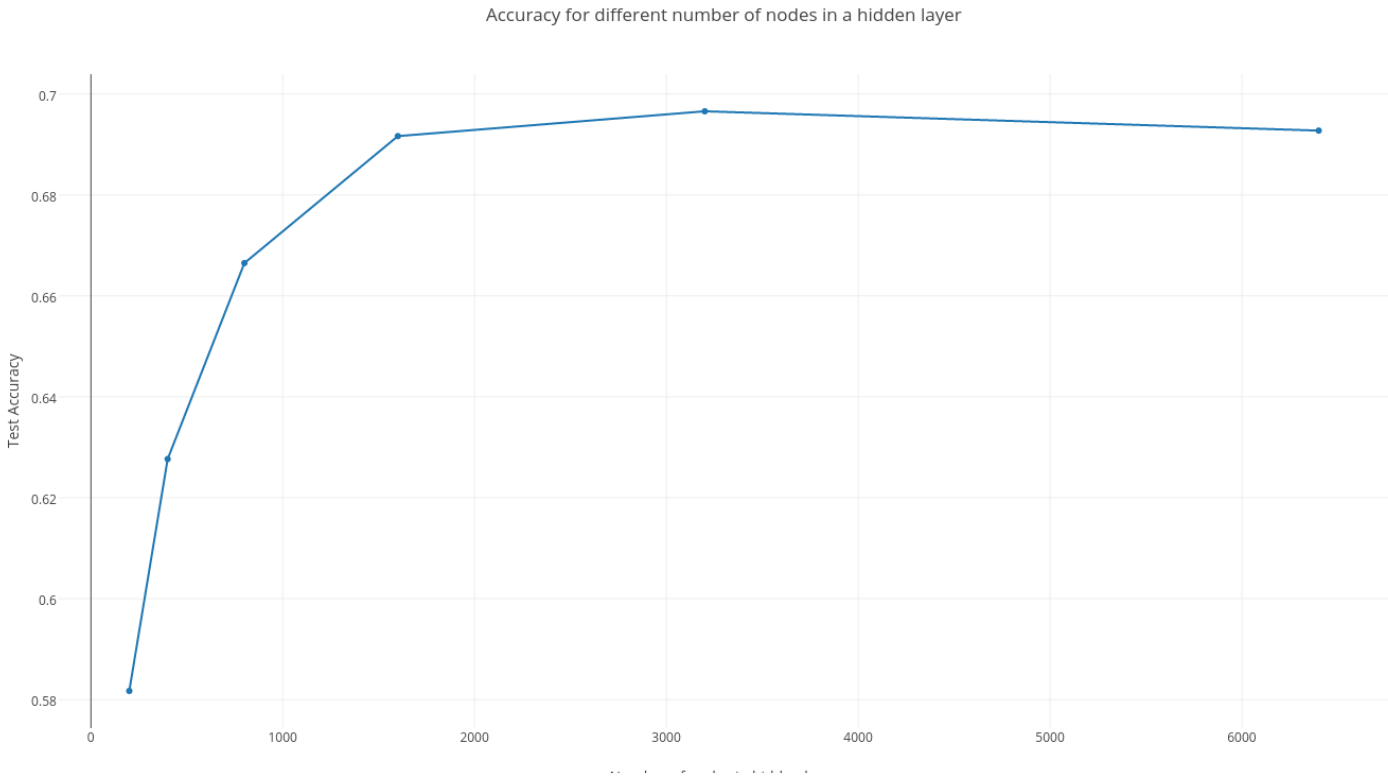


Number of hidden layers	Test Accuracy
1	0.611263
2	0.629852
3	0.640787
4	0.63313
5	0.6386

The input image has been resized to 80x80 pixels. Hence, the input layer has 1600 nodes. Each of the hidden layers has 400 nodes, and the output layer has 104 nodes. The training was done for 15 epochs (1 epoch being a pass over all the 17205 training images, with a batch size of 208).

As can be seen, increasing number of layers increases accuracy till the number of hidden layers is 3, after which it flattens out.

Accuracy vs Number of Nodes in a Hidden Layer

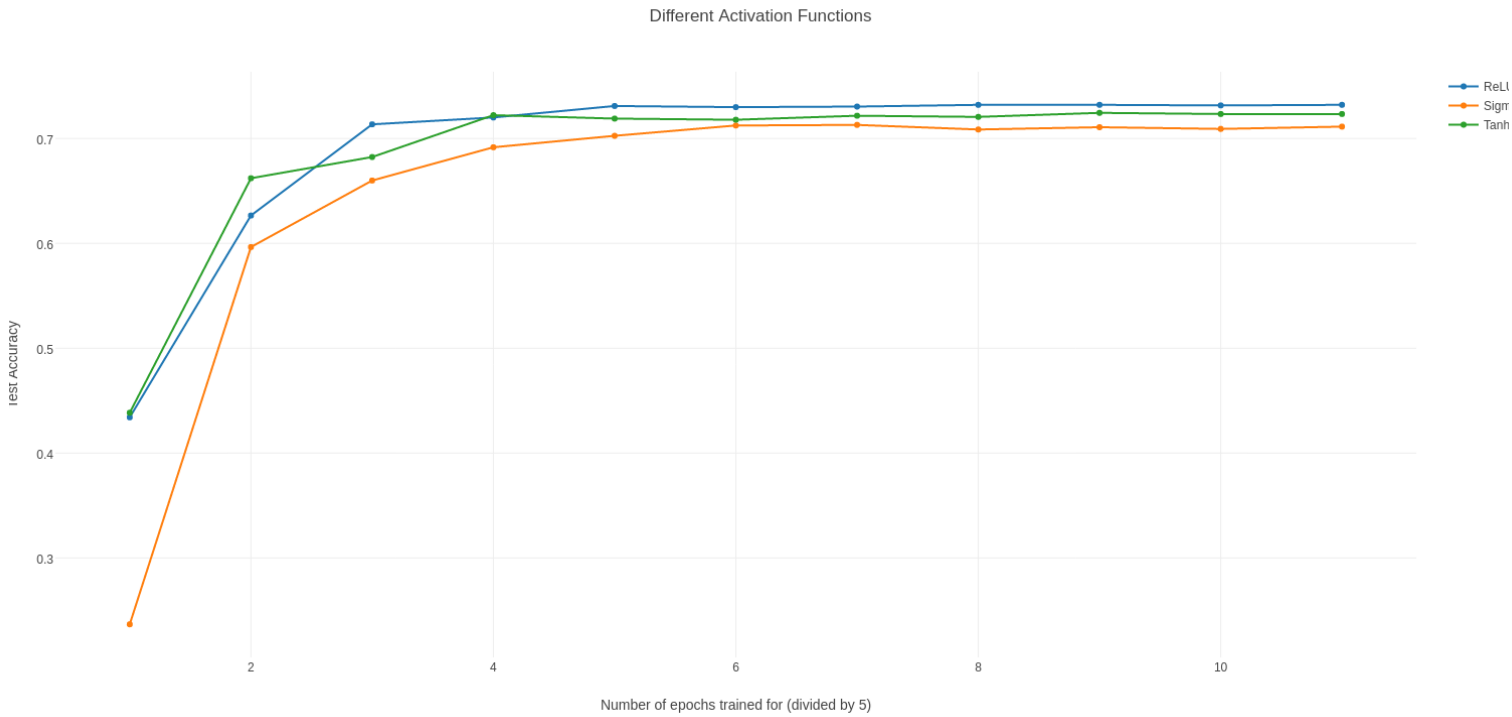


Number of nodes in hidden layer	Test Accuracy
200	0.5817394
400	0.6276658
800	0.66648416
1600	0.69163532
3200	0.69655564
6400	0.6927284

The model was trained for 15 epochs, having 1 hidden layer.

As can be seen, increasing the number of hidden layer nodes increases the accuracy uptil a given number, after which, it flattens out or saturates.

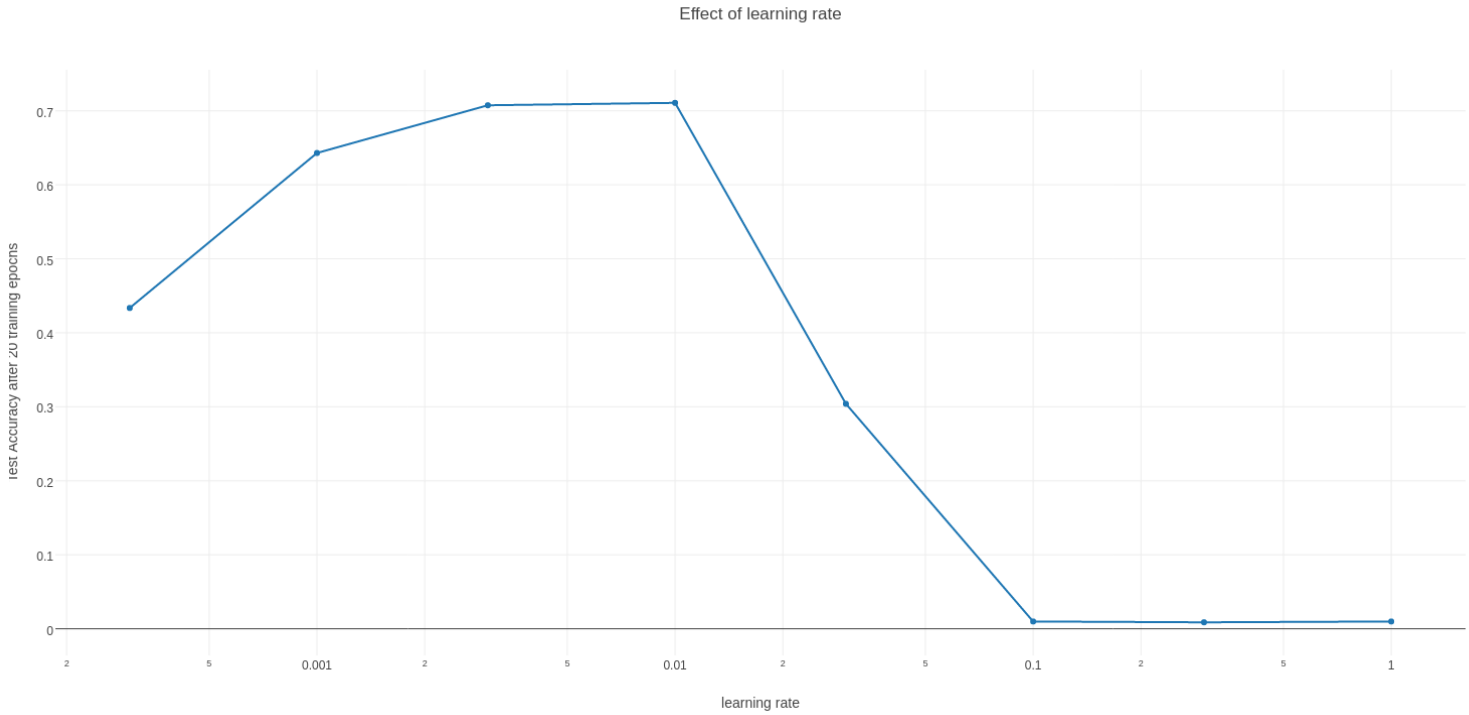
Accuracy for different activation functions



Final accuracy for ReLU neurons : 0.73209405
Final accuracy for sigmoid neurons : 0.71131766
Final accuracy for tanh neurons : 0.72334611

The model has two hidden layers, the first one having 1600 nodes and the second one, 400.
Sigmoid neurons are a little slower to train as compared to ReLU and tanh and converges at a test accuracy lower than those. Tanh is faster to converge than ReLU, but converges at a lower test accuracy.

Accuracy with different learning rates



Learning Rate	Test Accuracy
0.0003	0.43357
0.001	0.642974
0.003	0.70749
0.01	0.710771
0.03	0.303991
0.1	0.00984144
0.3	0.00874795
1	0.00984144

Using a gradient descent optimiser, for 20 training epochs.
A learning rate which is smaller than optimum doesn't allow the model to learn fast enough to reach as high a test accuracy(in 20 epochs) as the optimum rate does. A learning rate which is higher, may lead to large gradient descent

steps which do not decrease the cost function and hence, there is no further learning.

Effect of Regularizers

L2 regularizer : L2 regularizers didn't seem to achieve a higher validation accuracy in 50 training epochs as compared to the unregularised model being trained for the same number of training epochs.

Lambda	Test Accuracy
0.1	0.709131
0.25	0.683434
0.5	0.674139
0.75	0.696009
1.5	0.630946
2	0.588846

For larger values of the parameter lambda (coefficient of the L2 norm in the regularization term), the model under fits and results in a lower test accuracy. The model, when trained for 50 training epochs gave an accuracy of 72.6%

Dropout regularizer : Dropout regularizers helped increase the test accuracy of the model by around 2%, when tuned at the right value of the drop probability. The model was trained using the same keep probability at all three layers.

Keep probability(=1-drop probability)	Test Accuracy
0.4	0.746309
0.5	0.744122
0.6	0.743576
0.7	0.732094
0.8	0.699289

Final model:

The final model was trained using ReLU perceptrons and a dropout regularizer with keep probability = 0.5. The network has 2 hidden layers with sizes 1600 and 400 nodes respectively. The final test accuracy achieved was 74.9%

Link to code:

https://github.com/samarth4149/CS726-Assignment.git

Instructions to run:

The final model is saved in the file model.ckpt. The file load_n_test.py is the script that takes in the image as the argument(only 1 argument needed by the script, which is the path to the image), pre-processes it and outputs probability values for the 104 output classes. The file devnflow.py was the one used to train the model and pre-process.py was used to pre-process all training and test images. These scripts need specific folders containing the images to exist in a folder ‘devnagri’, which resides in the same directory as these scripts.

Hence, for the desired output, run:
python load_n_test.py <path_to_test_image>

The code depends on libraries scikit-image, tensorflow and numpy.