# Deep Learning Assignment 3

The objective of this assignment is to train a neural network using **Tensorflow(use latest version 1.5)** and to get an intuition about the output of each of those hiddens layers.

**Dataset** : Same as Assignment 2.

**Task**: You will create a neural network with 3 hidden layers and final output layer over all possible classifications. Train it end to end. Use relu activations in all layers.

After training the above NN, get the hidden layer representation of layer 1 for the data-points and use it as input for training a logistic regression based classifier for the data. Use the same train-test split for training the logistic regression classifier. Report the results on the test set for it. Similarly do the same for hidden layers 2 and 3.

Submit a brief report on what inferences you could draw from experiment. (in pdf only)

**Constraints and Considerations:**

1. You must **NOT** use **tf.nn or tf.layers or tf.contrib.layers** modules for using various functions, instead implement them on your own. (For example you must implement the relu activation on your own using tf.Variables, tf.placeholder, etc. But you cannot use tf.nn.relu(x)). You are allowed to use the tf.train module and other basic modules necessary.
2. For implementing logistic regression you can directly use Scikit Learn's Logistic Regression classifier
   http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
3. Your logistic regression classifier should take the layer activations as features and the image class labels as ground truth labels. Train the classifier on your dataset's training set and test on the dataset's test set.
4. Place the data in '../data' folder and use the data_loader file (if needed) as done exactly in last file. You should not submit data. Always create a train-validation split on the training set and train and test on appropriate data only. You must train properly with proper stopping criterion. All decisions you take for model parameters must be logical.
5. Keep the number of hidden units in each of the hidden layer same.
6. Store all weights of your model in "weights/" folder. The weights folder must be inside the directory in which you are coding and must be submitted to Moodle.
7. If the weights are exceeding the upload constraints imposed by Moodle (may not happen), then upload the weights in some public site like github and download them first, by writing appropriate code for it.
8. Do NOT hardcode any paths we may replace the data in '../data' folder to evaluate your work.
9. Include all training and testing codes. A file named *'assignment3.py'* must be present in your code. It takes in some command line arguments. '*python assignment3.py* --train' should train the model, *'python assignment3.py --test '* should load all model weights, test the model and report the accuracy on the test data. If '*python assignment3.py --layer=1'* is run, output the results for logistic regression trained using layer 1's representation. Do similarly for value of argument equal to 2 and 3. For argument parsing you can use python's argparse library or tf.flags(more preferred).

10. Name the **ZIP file as well the folder that you compressed to get the zip** as "Assignment3_<Roll No.>.zip" and Assignment3_<Roll No.>" respectively. Upload the zip file with codes and report to moodle. Note that the zip file should contain your python files, codes and a pdf file explaining your model parameters and results/inferences. Please note that all explanations should be short (1-3 sentences).
11. Small differences in your method's accuracy would not be penalized (larger may be; remember to set a fixed random seed for reproducing your result). Your experimentation technique should be sound. (like, do not test on training data)