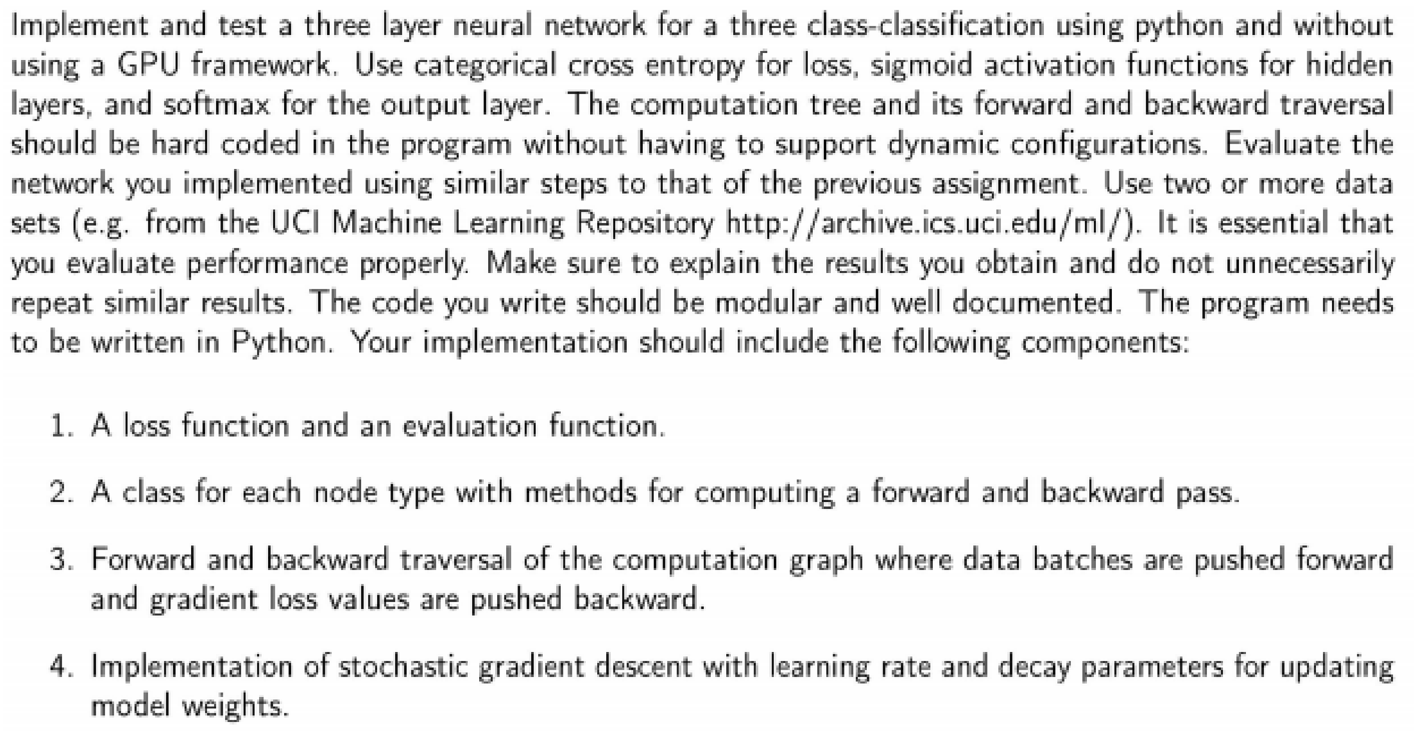
## Problem Statement:


Implement and test a three layer neural network for a three class-classification using python and without using a GPU framework. Use categorical cross entropy for loss, sigmoid activation functions for hidden layers, and softmax for the output layer. The computation tree and its forward and backward traversal should be hard coded in the program without having to support dynamic configurations. Evaluate the network you implemented using similar steps to that of the previous assignment. Use two or more data sets (e.g. from the UCI Machine Learning Repository http://archive.ics.uci.edu/ml/). It is essential that you evaluate performance properly. Make sure to explain the results you obtain and do not unnecessarily repeat similar results. The code you write should be modular and well documented. The program needs to be written in Python. Your implementation should include the following components:

1. A loss function and an evaluation function.
2. A class for each node type with methods for computing a forward and backward pass.
3. Forward and backward traversal of the computation graph where data batches are pushed forward and gradient loss values are pushed backward.
4. Implementation of stochastic gradient descent with learning rate and decay parameters for updating model weights.

We are trying to improve the performance of multiply neuron and multiple data model which use IRIS Data Set and MNIST Data Set.

## Proposed Solution:

Data: IRIS Data Set and MNIST Data Set.

**Mode Details:**

1. Number of Epochs = 25
2. Learning Rate = 0.01 (1%)
3. Hidden Layers = 256
4. 80-20 percentage of partition train and test data in the case of IRIS data set.
5. The target variable is one-hot encoded to categorical variables

## Implementation details:

Files:

1. InnerLayer.py:

- In this file we have a Class Hidden Layer with two function for forward pass and backward pass on the Hidden Layer. Using Sigmoid Activation.

2. OuterLayer.py

- In this file we have a Class Output Layer with two function for forward pass and backward pass on the Output Layer. Using Softmax Activation.
- The Size of Outer Layer change accordingly the number of classes in the target variable.

3. Model.py

- This class is used to define the Model.
- It instantiates the Hidden and Output Layers in accordance to the parameters fed to it.
- Uses Categorical Cross-Entropy as the loss function.

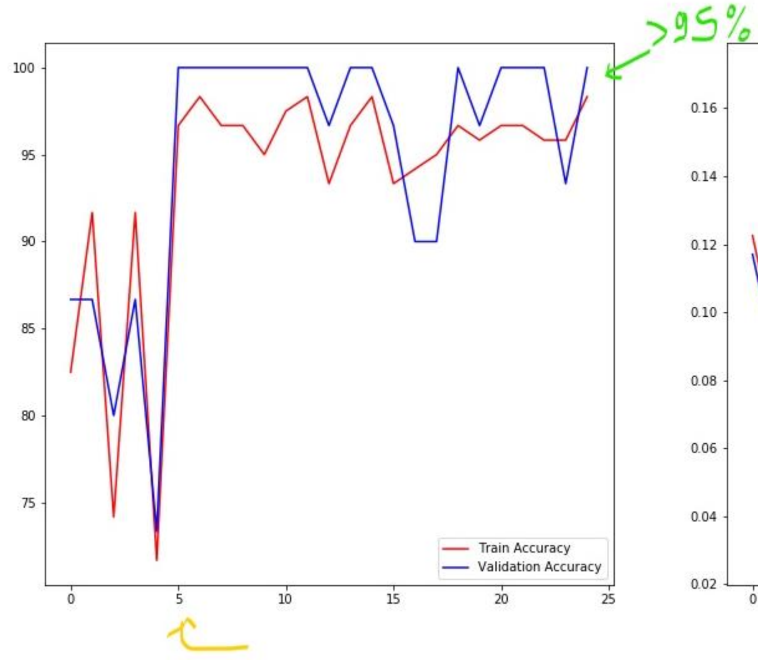
4. Main.py

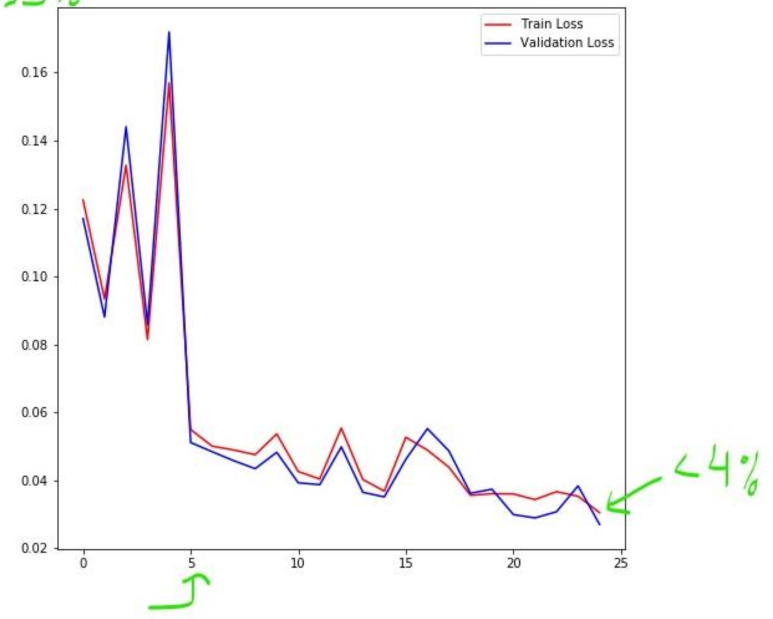
- The Main driver file. Trains models on 2 different datasets and saves the metrics plots.
- Datasets used are – MNIST dataset from keras and IRIS dataset from UCI ML repository.

## Results and Discussion:

• Number of Layers: 3 • Number of hidden layers: [256]• Optimizer Learning Rate: 0.0001 • Activation Function: sigmoid for hidden layers, Softmax for the output layer • Loss: categorical_crossentropy • Epochs: 25

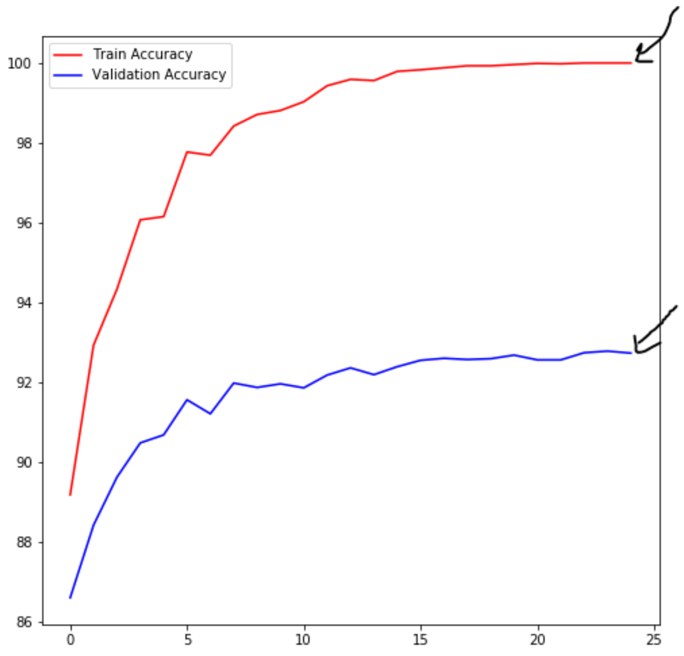
IRIS Data Set Result: default dict(<class 'list'>, {**'acc'**: [82.5, 91.67, 74.17, 91.67, 71.67, 96.67, 98.33, 96.67, 96.67, 95.0, 97.5, 98.33, 93.33, 96.67, 98.33, 93.33, 94.17, 95.0, 96.67, 95.83, 96.67, 96.67, 95.83, 95.83, **98.33**], 'loss': [0.122497424553191, 0.09335304891189276, 0.13274167720717822, 0.08134617195022635, 0.15693440881692997, 0.054882490600174694, 0.04996653962106729, 0.048854101762243606, 0.04745788435766217, 0.05357580621640691, 0.042508089967474676, 0.04029424731583794, 0.05528817609676547, 0.04017140560711258, 0.03672410553969289, 0.052595500725977205, 0.048815960318671775, 0.04377364843019961, 0.035509549797114806, 0.0359564810331503, 0.03589398371696313, 0.0342475632961879, 0.03656628878501944, 0.03522577356763522**, 0.03040083365736947**], 'val_acc': [86.67, 86.67, 80.0, 86.67, 73.33, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 96.67, 100.0, 100.0, 96.67, 90.0, 90.0, 100.0, 96.67, 100.0, 100.0, 100.0, 93.33, **100.0**], 'val_loss': [0.11701309399293577, 0.08802665669781666, 0.14406190014319833, 0.0857431787394865, 0.17196103705613205, 0.05105592436772171, 0.04833826023692448, 0.045696129720542635, 0.04335105126828646, 0.04816118817684175, 0.03917373105056465, 0.0386272388778784, 0.04975525984915055, 0.03638542718924111, 0.03501744081111792, 0.04612313088442523, 0.055147183080115375, 0.04856664234888922, 0.03609127392242406, 0.037290115548870496, 0.029794869237783524, 0.028845518177968807, 0.0306498622080761, 0.038266625271279095, **0.02690852167187692**]})

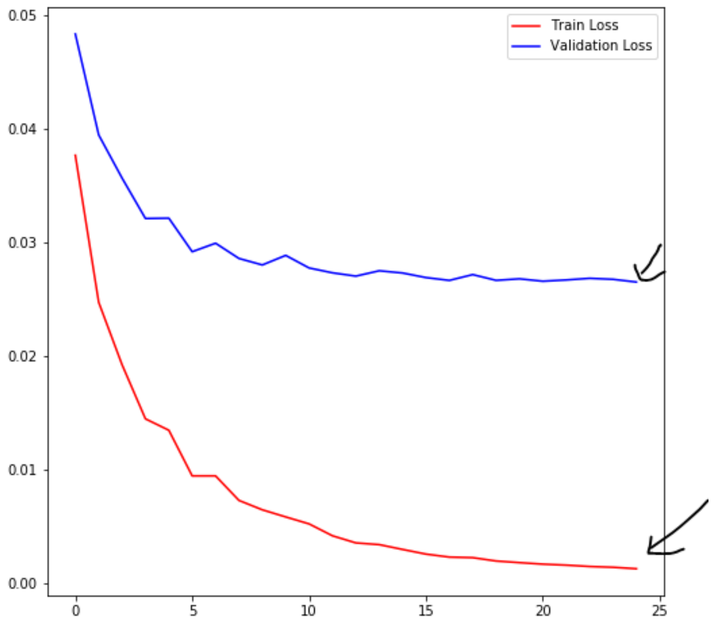




## MNIST Data Set

**Result**: defaultdict(<class 'list'>, {'acc': [89.18, 92.93, 94.33, 96.07, 96.15, 97.77, 97.69, 98.42, 98.71, 98.81, 99.03, 99.43, 99.59, 99.56, 99.79, 99.83, 99.88, 99.93, 99.93, 99.96, 99.99, 99.98, 100.0, 100.0, **100.0**], 'loss': [0.03764625178858418, 0.024709310817861638, 0.01921415497427119, 0.01445356988283287, 0.01345035048918806, 0.009432436691055316, 0.009433343225742574, 0.007275258799361432, 0.006449766001213154, 0.0058229626251875884, 0.005210034892646041, 0.004164012087683291, 0.003537362658155927, 0.003386534483364264, 0.002961416358109077, 0.002551331886506267, 0.0022882168634214524, 0.0022387327136699914, 0.001943928416366116, 0.0018006197178857813, 0.0016672848583260404, 0.0015814324220090232, 0.001460942469277947, 0.0013934123365039763, 0.**001263662320514667**], 'val_acc': [86.6, 88.42, 89.62, 90.48, 90.68, 91.56, 91.21, 91.98, 91.87, 91.96, 91.86, 92.18, 92.36, 92.19, 92.39, 92.55, 92.6, 92.57, 92.59, 92.68, 92.56, 92.56, 92.74, 92.78, **92.73**], 'val_loss': [0.048317482800664203, 0.03943733681003368, 0.035639052585980935, 0.032083511940581774, 0.0321117559554115, 0.029160538280026595, 0.029907309982150646, 0.028566803346557208, 0.027994521933467593, 0.028837400448825913, 0.027730361594949973, 0.027302533949357718, 0.02701215776200918, 0.027482205901289983, 0.027287123808445456, 0.02688411970676804, 0.02663642129264705, 0.027145791741666942, 0.026640641190615304, 0.026773807701572734, 0.026564324058009185, 0.026676648859340627, 0.026819338605852738, 0.026731922143724416, 0.**026491828401404603**]})


Train Accuracy
Validation Accuracy


Train Loss
Validation Loss