

Machine Learning Practical Coursework 3

UUN:s1579563

1 Research Questions

As part of this coursework the performance of feed forward neural network on CIFAR-10 and CIFAR-100 data sets was analysed. Many variation were incorporated so as to come up with a good baseline model. In all, five kinds of variations were performed as detailed below with the motivation behind them.

Experiment 1: Vary the Number of hidden layers

Increasing the number of hidden layers can help in training a network better if the data is not linearly separable. Linearly separable data implies that there exists a boundary that divides the sample space into different groups each representing a class. Learning such decision boundary helps to classify effectively.

Experiment 2: Vary the number of neurons(hidden units)

Neurons are small mathematical units which take as input a value generated by neurons from previous layer, apply a mathematical function on it, and pass the value to the next neuron in next layer. The output value of these neurons that present the best accuracy are vital and hence stored for later classifying unseen data. These values are also called weights. Thus increasing the number of neurons increases computation but may help in modeling the complex model better.

Experiment 3: Different optimizers

Some of the optimizers that were tried include Adagrad, Adam and Gradient Descent(with learning rates 0.03 and 0.3). Different optimizers minimize the error function differently. The variation in techniques may help find a minima better, or quicker. Gradient descent moves against the slope of the gradient of error function by a fixed amount. The scale factor is a hyperparameter called learning rate and denoted by greek alphabet α . Adagrad is also a gradient descent technique which performs larger updates for infrequent parameters, and smaller for frequent ones.

Experiment 4: Different non-linear functions

The motivation behind this experiment is to analyse what is the effect on prediction of different activation function. Non-linear activation functions are used to model the non-linearities in the data which cannot be captured by linear transformations such as matrix multiplication. For the purpose of the report, two non-linear functions have been explored: Hyperbolic tangent(Tanh) and Rectified Linear Units(ReLU). Both have different characteristics which make them interesting to explore. Tanh function varies smoothly between $[-1,1]$ whereas ReLU remains dormant till $[0,0]$ and then varies linearly. The point of inflexion for each function can be adjusted, but has been kept default for all the experiments conducted in his report.

Experiment 5: Regularization

While conducting the above experiments, a deep sense of overfitting was observed. Regularization helps reduce overfitting by penalizing the parameters by a small quantity. This experiment tests whether we can use regularization to reduce overfitting our model and increase generality. Two regularizers were explored: L1 Regularizer and L2 regularizer each with penalty(β) set to 0.1 and 0.001.

2 Method

For conducting above experiments Tensorflow 0.12 and Python 2.7 were used in an Anaconda 2 environment. Tensorflow is a machine learning library developed at Google. It works on the concept of computation graphs where in each mathematical operation is realised as a node in the graph and value from it is then used as input to another node. The value is exported from one node to another in the form of a data type called Tensor

For all the experiments, unless stated, a feed-forward neural network with activation function as ReLU was used and weight parameters were initialised according to Glorot initialisation. The number of input units was equal to the size of the image (784 pixels) and the output neurons were same as the number of classes. Two hundred hidden units in each layer were used for all the different layer experiments. This architecture is stored under a graph object using Tensorflow's name scope. Thus, we can assign a graph with name *graph_one_layer* that describes a neural network with one hidden layer. The graph is then set as default and session variables picks up this graph and initialises variables which are associated with the graph. It is important that the session is terminated after the computation has been done to avoid confusing with different graph variables. Hence, preferable all the training and initialisation is done with a *with..* session active block. The default Adam optimizer defined in `tf.train` class was used to optimize the results and softmax cross entropy defined in `tf.nn` class was used to define the error function. The validation score was calculated at every five epoch. The experiments were conducted for hundred epochs each.

Each layer is defined by three key elements: parameters such as weights and bias, and non-linear function. The parameters are instances of Variable class and are mutable objects. Hence, any update in parameter would overwrite the previous value stored. This is useful since we continuously update the parameters based on the error function. Non-linear functions such as ReLU and Tanh are defined in the `tf.nn` class and the same are used to compute the output for each layer. Input and output layer is instantiated using Placeholder class. Input and output layer access the values associated with the observed data. These values do not take in mathematical operation directly, all operations are applied in hidden layers. Hence, an immutable class was used to instantiate these two layers.

Experiment 1: Different number of hidden layers

The first experiment explores the effect of increasing the number of hidden layers on the classification performance. This is done by defining another fully connected layer, which takes output from previous layer as input, and sends its output to the next layer. We can stack as many layers we want, but an increase in training time was observed with increasing the number of hidden layers. Hence, the number of hidden layers was kept between one to five for CIFAR-10 dataset, and one to three for CIFAR-100 dataset.

Experiment 2: Different number of neurons(hidden units)

This experiment explores the effect of changing the number of hidden units in each layer. Hidden units, at the lowest granularity, are responsible for mathematical operation and correspond to each value in the weight matrix. The same is then stored and used as model to make future prediction. Hence, the number of hidden units can play a major role in capturing the complexity of the model. For this experiment, the number of hidden units were varied between 100,200,300,400,500 for CIFAR-10 dataset and for CIFAR-100 it was 100,300,500. The number of hidden layers was kept fixed as three.

Experiment 3: Different optimizers

Optimization algorithms are used to find the optimum value error function by readjusting the parameters of the model. This is done until the least value of error function for a set of parameters is found. Gradient descent is one popular technique which updates the parameters in the direction opposite the slope of the error function by a fixed amount, called learning rate. Setting this learning rate large

may result in algorithm missing a minima and it could keep oscillating around some different value. Learning rate(α) is a hyperparameter which we must set by ourself. This experiment explores gradient descent for α values 0.3 and 0.03. Adam, α 0.001 and Adagrad with α 0.01 were explored for CIFAR 10 data set. For CIFAR-100 dataset, Adam and gradient descent with learning rate 0.03 were explored. All the optimization algorithms were defined by default in `tf.train` class.

Experiment 4: Different non-linear functions.

In this experiment instead of using only ReLU, a Tanh function was used and the results for both were recorded. The objective was to find if using a different non-linear function would model our input distribution better. ReLU and Tanh functions defined in `tf.nn` class were used for all the hidden layers. Both the functions were used on CIFAR-10 and CIFAR-100 datasets.

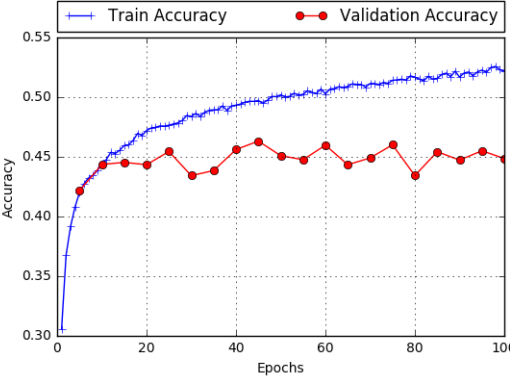
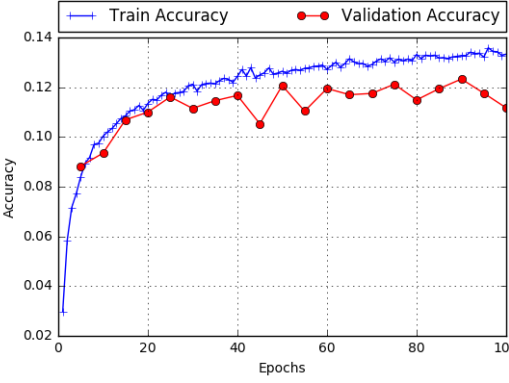
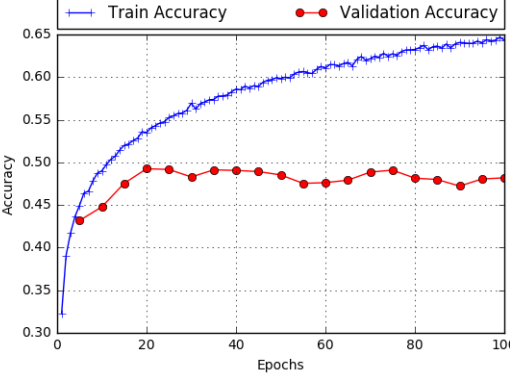
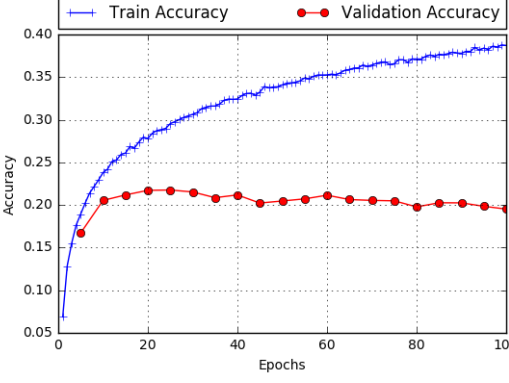
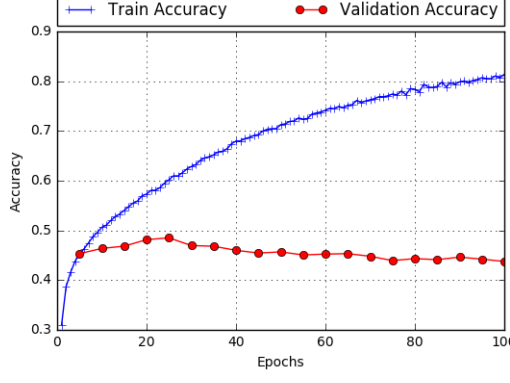
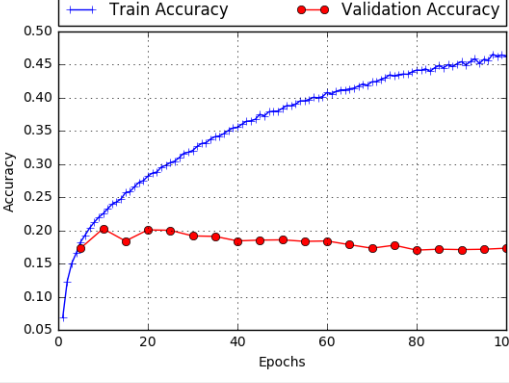
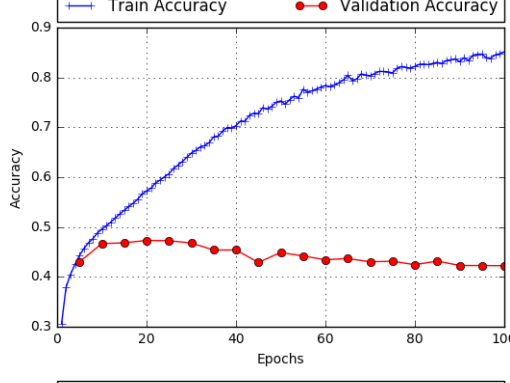
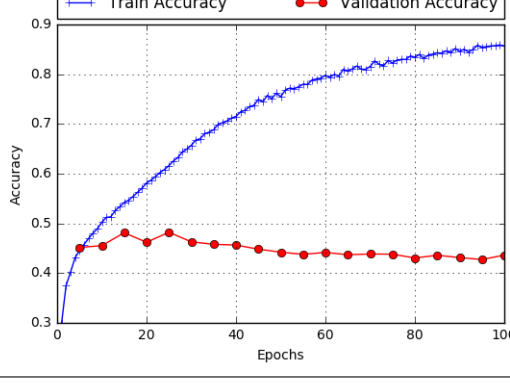
Experiment 5: Regularization

Regularization is used to reduce overfitting in the model. A penalty is added to the error function, that reduces the value of parameters by a scalar quantity, thus, reducing the effect of parameter. Two regularization techniques were explored, L1 and L2 regularization with a scaling factor of 0.01 and 0.001 for CIFAR-10 and 0.01 for CIFAR-100 dataset. L2 regularization is implemented in `tf.nn` class and the same was used. L1 regularization was implemented by adding scaled sum of absolute values of the weights computed for each hidden layer to the error function.

3 Results and Discussion

Experiment 1 Number of Hidden Layer

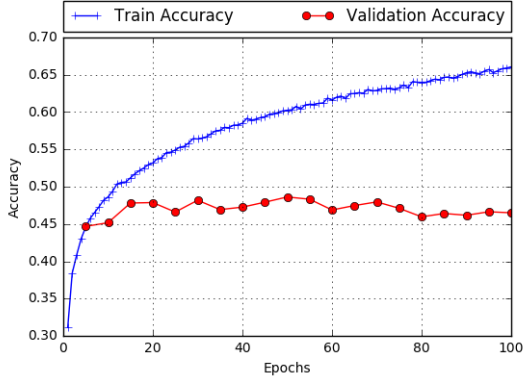
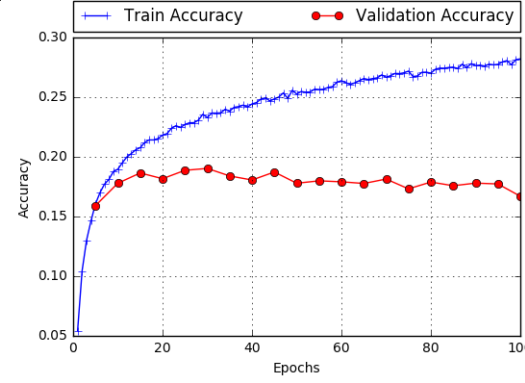
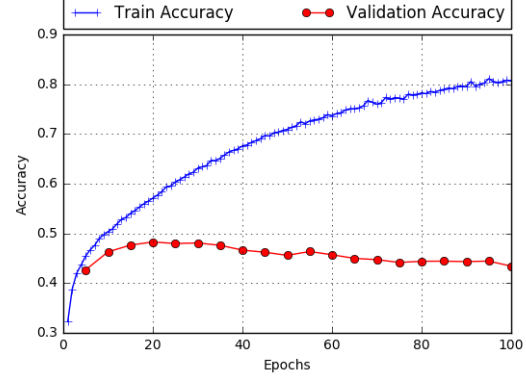
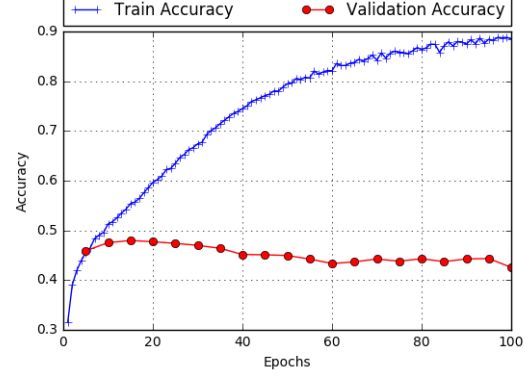
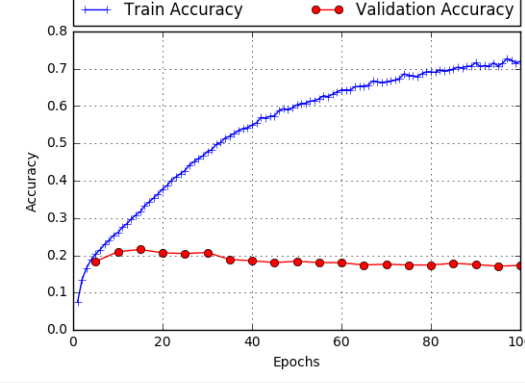
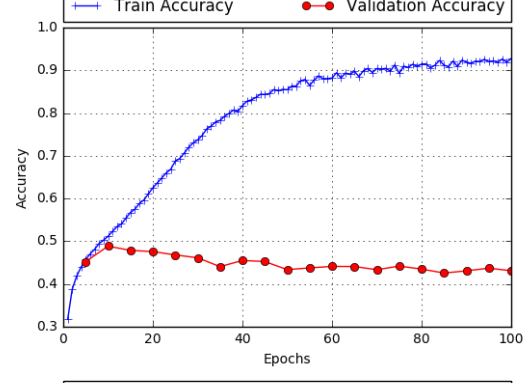
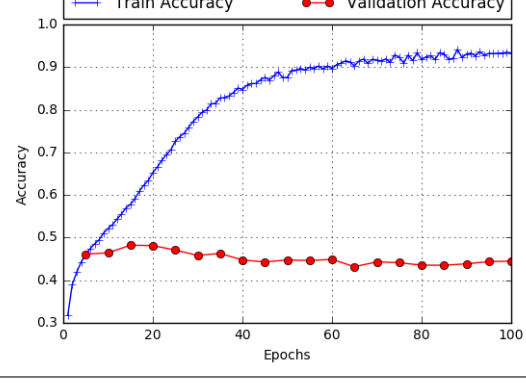
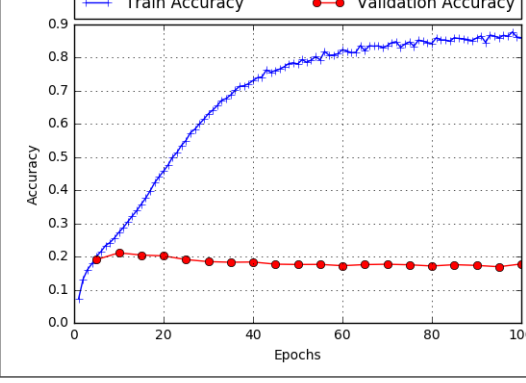
Table 1: Training and Validation Accuracy plots for Experiment 1

Layers	Accuracy Plot for CIFAR 10	Accuracy Plot for CIFAR 100
One		
Two		
Three		
Four		-NA-
Five		-NA-

Discussion: It is observed from graphs in Table 1 that with increasing the number of hidden layer, the training accuracy for CIFAR-10 dataset increases, however the validation set accuracy is fairly between the same interval. The large difference between the accuracies of two sets imply that with increasing the number of layers we are able to capture the variation in training data but fail to generalise since validation accuracies are lower. A similar trend is observed in CIFAR-100 dataset with an exception for CIFAR-100 dataset for one hidden layer neural network which does not show much variation in training and validation set accuracy. This may lead to the conclusion that one layer feed forward neural network does not overfit on CIFAR-100 dataset but the low level of accuracies achieved across all epochs suggests that this might not be a good conclusion.

Experiment 2: Different number of hidden units

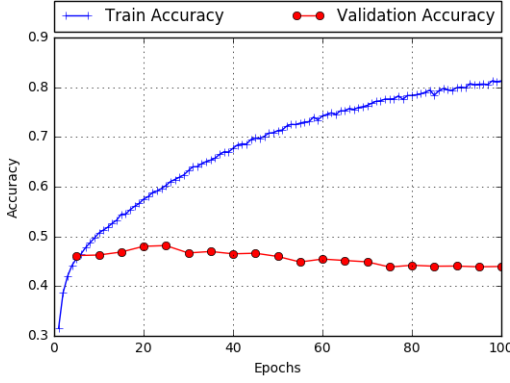
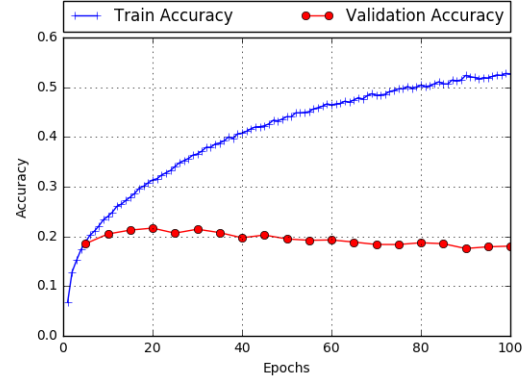
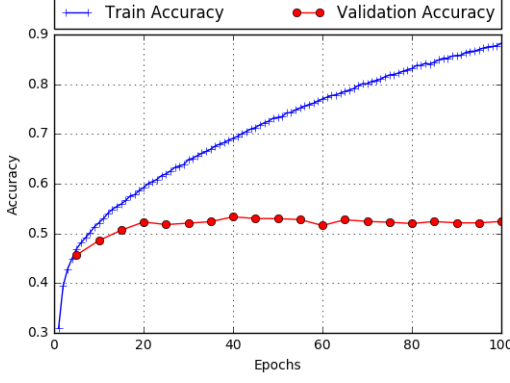
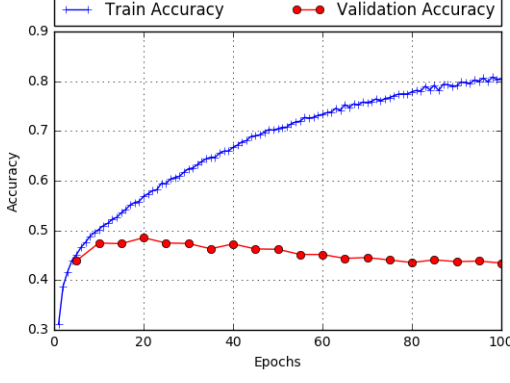
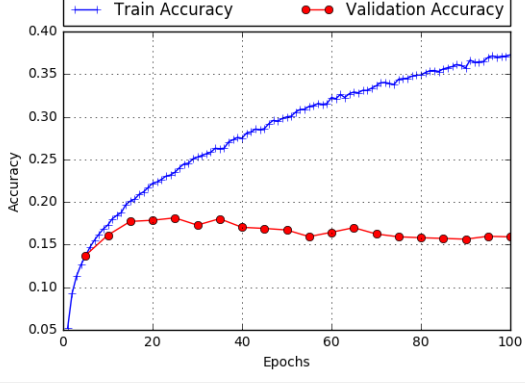
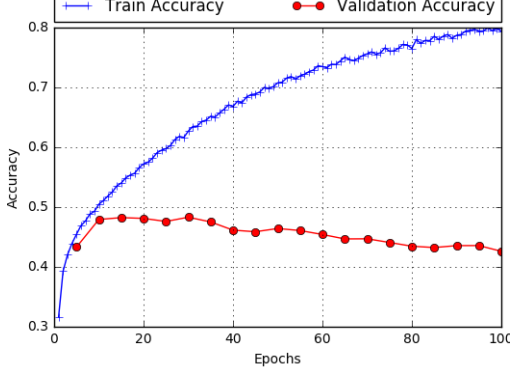
Table 2: Training and Validation Accuracy plots for Experiment 2

Neurons	Accuracy Plot for CIFAR 10	Accuracy Plot for CIFAR 100
100		
200		<p>—NA—</p>
300		
400		<p>—NA—</p>
500		

Discussion: Increasing the number of neurons had little effect on improving the validation set accuracies for both CIFAR-10 and CIFAR-100 data sets as shown by plots in Table 2. Especially for CIFAR-100 dataset, the accuracy remains almost the same on increasing the number from 300 to 500 neurons. Since increasing the number of neurons directly correspond to more computation and time taken, it is good to evaluate if increasing number of neurons would be able to help improve accuracy.

Experiment 3: Different optimizers

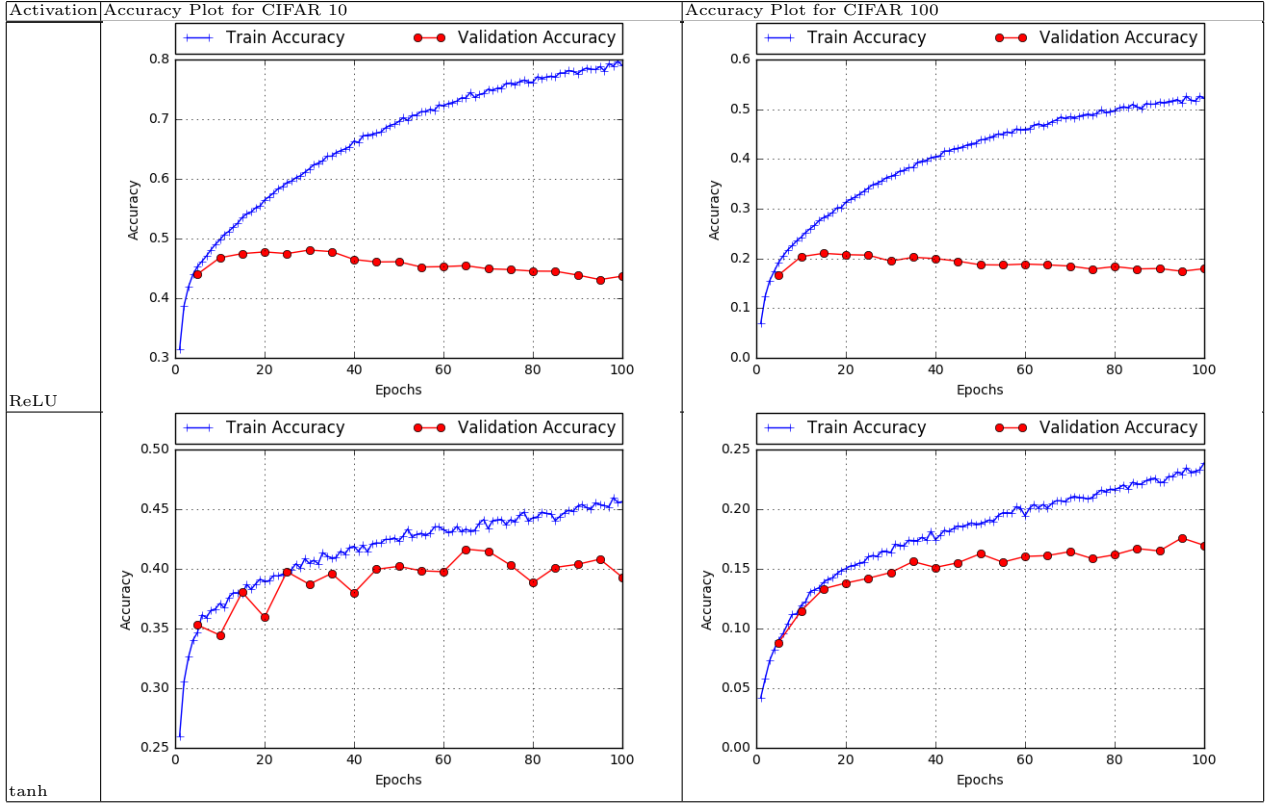
Table 3: Training and Validation Accuracy plots for Experiment 3

Optimizers	Accuracy Plot for CIFAR 10	Accuracy Plot for CIFAR adam
Adam		
Adagrad		-NA-
Gradient Descent($\alpha=0.3$)		
Gradient Descent($\alpha=0.03$)		-NA-

Discussion: As shown in graphs in Table 3, Adagrad optimization technique yielded better performance for CIFAR-10 dataset. Also, the validation set accuracy remained fairly constant for this optimization technique signalling that there was no overfitting. The validation accuracy fell for other three optimization techniques after 20 epochs. For CIFAR-100 data set it is observed that Adam optimizer performs better than the gradient descent technique, but the difference is too small to conclude something concrete. Further reducing the learning rate for both the techniques and then comparing the validation accuracy curves could be one area to explore.

Experiment 4: Different non-linear functions

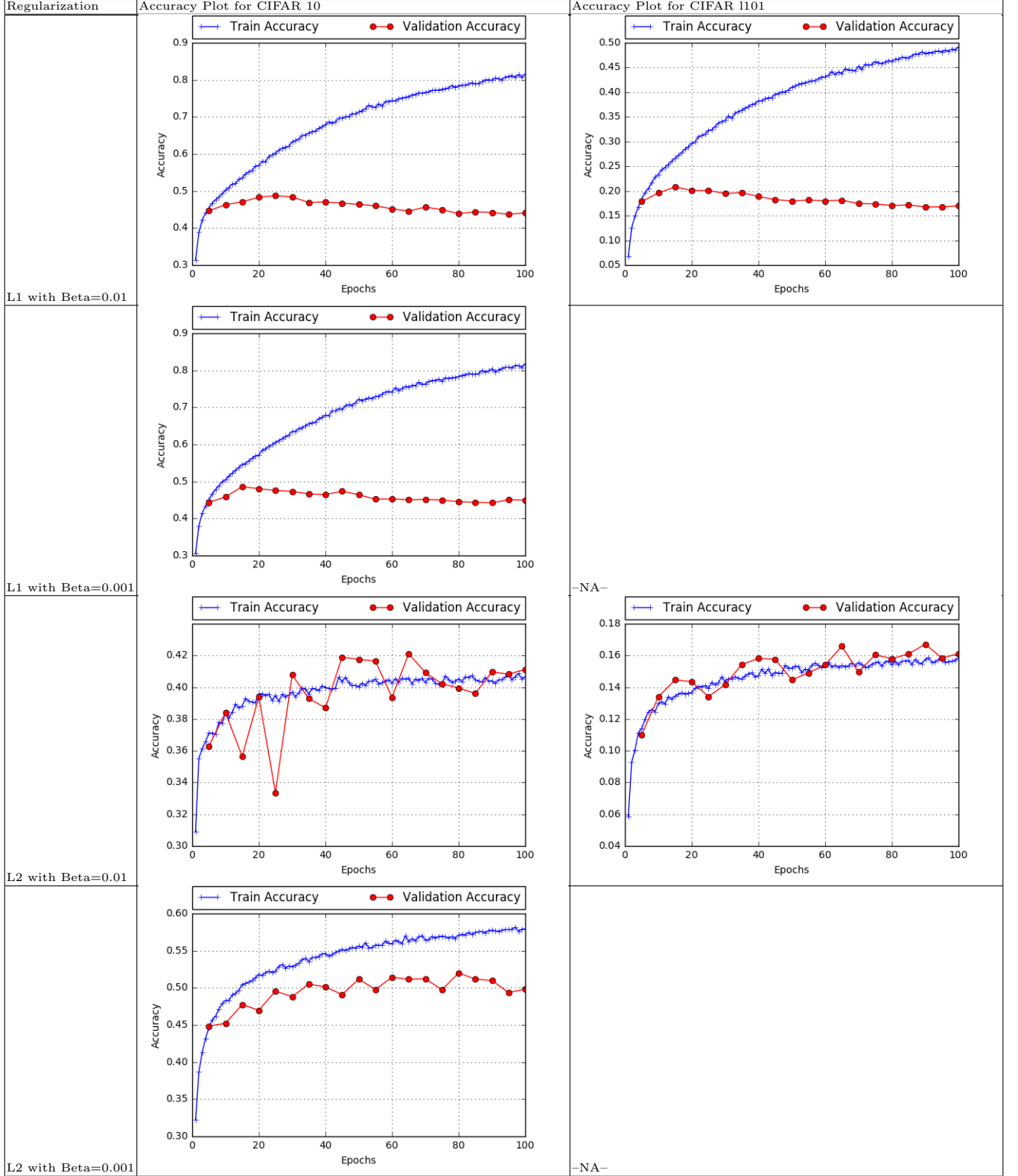
Table 4: Training and Validation Accuracy plots for Experiment 4



Discussion: ReLU activation function outperforms Tanh activation function for both CIFAR-10 and CIFAR-100 dataset as is evident from plot of experiments in Table 4. The ReLU activation function seems to overfit after epoch 20, whereas the hyperbolic tangent function experiment has increasing validation accuracy. The experiment could be run for longer epochs, and with a different optimizer to find a better parameter configuration.

Experiment 5: Different regularization techniques

Table 5: Training and Validation Accuracy plots for Experiment 5



Discussion: From Table 5 it is observed that L1 regularization seems to have little effect compared to L2 regularization. L2 regularization achieved higher accuracy level (and also the best of all experiments). The validation accuracy curve for L2 with beta as 0.01, oscillates a lot, implying that a good minima was not found for the error function. This is cured by reducing the value of beta, which in turn scales down the effect of parameters also. Finally we get better accuracy with reduced β for L2 and decreased overfitting.

4 Future Work

The performance of multi layer feed forward neural networks tested in this part of the report has not been overwhelmingly good. This implores a necessity to compute more intricate neural network architecture. Convolution neural networks(CNN) are one such class of neural network that try to model variation in small windows of images. The database used for the experiments, present a lot of variation in terms of colour attributes as well as orientation of targets. Hence, such specialised network might be able to capture the differences in orientation and colours.

In this part of report, regularization yielded good results. This should be tested with CNN as well along with other techniques such as drop out. More variation in data at training time would be incorporated using data augmentation techniques, such as rotating the image, or just feeding half of the image original image and blanking the rest.