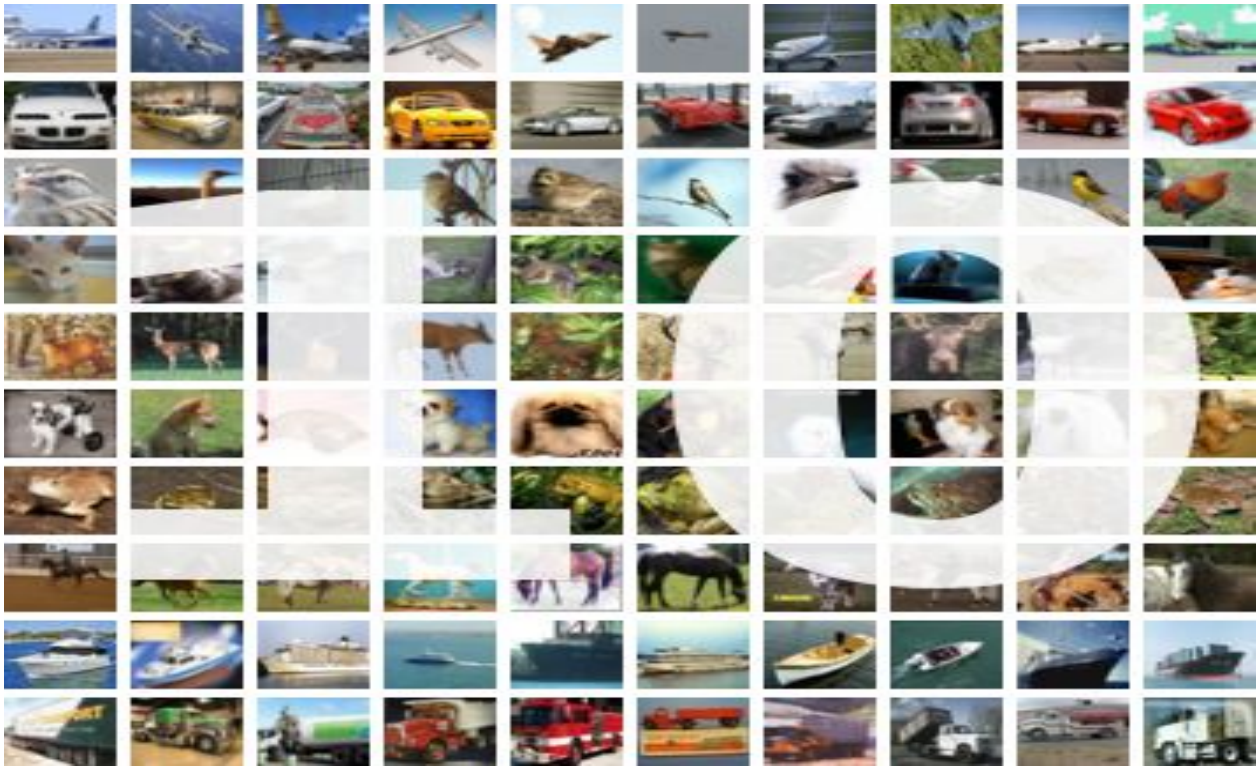


Assignment 5

Visual Recognition

Kartik Gupta - IMT2016128

Swapnil Buchke - IMT2016085

Shubham Gupta - IMT2016118

Objective

The objective is to fine tune the hyperparameter of the model implemented in the code provided which uses Keras deep learning model to classify the CIFAR10 images. The CIFAR10 is consist of 10 different classes which represent aeroplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We are required to work on the different parameters to get the best accuracy.

Preprocessing and Normalisation

We started our tuning process with preprocessing. We used ”henormal” as the weight initialiser and normalised the data by dividing all the columns by 255 so that all the pixel values come in the range 0.0-1.0. Then we used default parameters to train and test the data and got the best validity score = 0. 6893. After this, we moved to fine-tune the model.

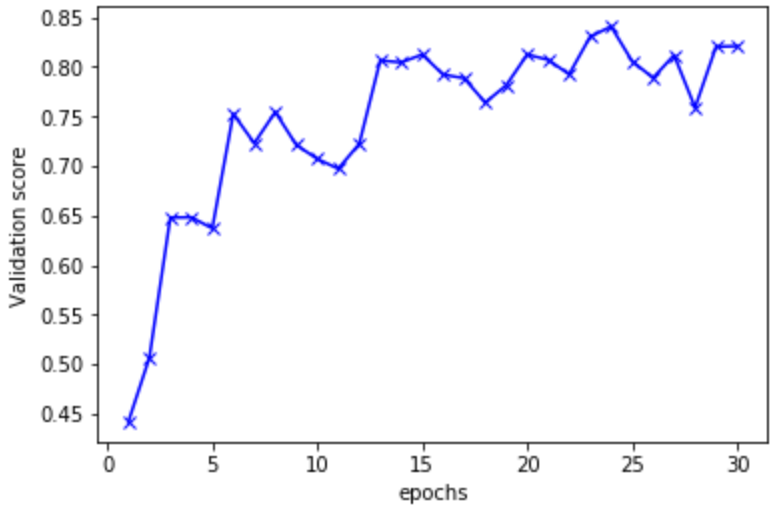
Activation function of layers

For this, we tried different combinations of activation functions on the 3 layers keeping all the other hyper-parameters same. We started with Relu only, then Tanh only and a combination of both at the last. The best we got validity score we got was 0.7982 and that was in the case of [ReLU,ReLU, Tanh] combination.

Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.001	ReLU	ReLU	None	categorical_crossentropy	50	128	0, 0, 0	0.7548
Adamax	0.001	ReLU	ReLU	ReLU	categorical_crossentropy	50	128	0, 0, 0	0.7689
Adamax	0.001	ReLU	None	None	categorical_crossentropy	50	128	0, 0, 0	0.7432
Adamax	0.001	Tanh	Tanh	None	categorical_crossentropy	50	128	0, 0, 0	0.6587
Adamax	0.001	Tanh	Tanh	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.6828
Adamax	0.001	Tanh	None	None	categorical_crossentropy	50	128	0, 0, 0	0.6234
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7982
Adamax	0.001	ReLU	Tanh	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7551

Learning Rate

Although we had the intuition the lesser the learning rate the better it is. But still, we decided to vary it and see the changes in the accuracy. To observe the pattern we plotted the validation score against time (epochs).



We observed that when the learning rate is high their, much distortion in score after each epoch but when it is in range 0.001-0.01 it tends to become constant after a certain number of epoch (most of the times around 30 epochs).

Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7982
Adamax	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7719
Adamax	0.01	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7416
Adamax	0.025	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7148
Adamax	0.05	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7045

Batch size and Dropout

Here we tried varying batch size to 32,64 and 128 and got the following result with the same. Also, it was observed that the lesser batch size the more time it takes to iterate.

Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8109
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	64	0, 0.25, 0	0.7924
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	32	0, 0.25, 0	0.7618

To avoid the model from overfitting, we applied dropout before layers. We tried various dropout values from the set [0.0, 0.25, 0.50] and tried different combinations for different pre-processed data for dropout. If we use 0.25, that means that one in 4 inputs will be randomly excluded from each update cycle. Low-value of dropout like 0, 0.25, 0.5 has been used by us to avoid the under-learning by the network. We also found that applying some dropout at the second layer of the network gave us a good result.

Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.7982
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8109
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.25, 0	0.8025
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.25, 0.25	0.7816

Loss Functions

We have used 3 loss functions namely “categorical cross-entropy”, “mean_squared_logarithmic_error” and “kullback leibler divergence error”. The loss function is basically used by the optimization process to calculate the model error. “ Categorical cross-entropy” gave us the best-suited weight values which are used to get the highest validation score by minimizing the error to the maximum extent.

Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8109
Adamax	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0,25 0.25, 0	0.8064
Adamax	0.001	ReLU	ReLU	Tanh	mean_squared_logarithmic_error	50	128	0, 0.25, 0	0.6027
Adamax	0.005	ReLU	ReLU	Tanh	mean_squared_logarithmic_error	50	128	0,.25 0.25, 0	0.5984
Adamax	0.001	ReLU	ReLU	Tanh	kullback_leibler_divergence_error	50	128	0, 0.25, 0	0.8014
Adamax	0.005	ReLU	ReLU	Tanh	kullback_leibler_divergence_error	50	128	0.25, 0.25, 0	0.7986

Optimiser

We learnt about some of the optimisers which are known to be efficient. Following are the details of the optimisers we used:

1) SGD: Stochastic gradient descent optimizer. Includes support for momentum, learning rate decay, and Nesterov momentum taken as parameters. SGD didn’t work that good and the best validation score was 0.7998.

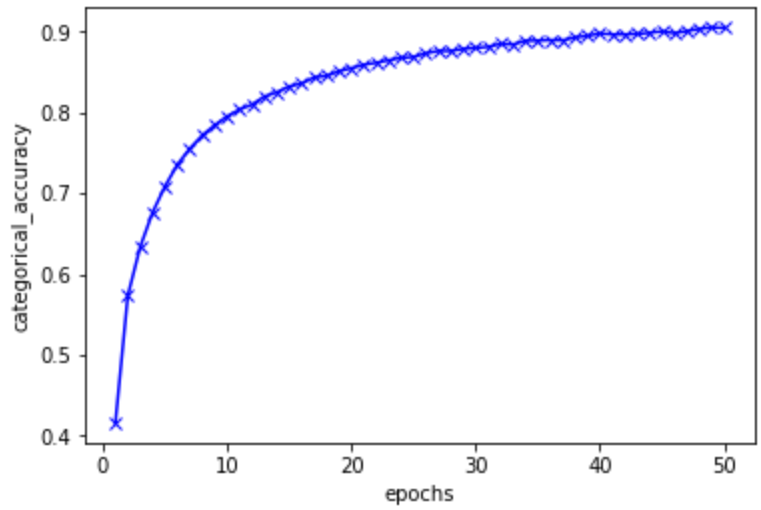
2) Adam: It is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. By using Adam we were getting validation score 0.8094.

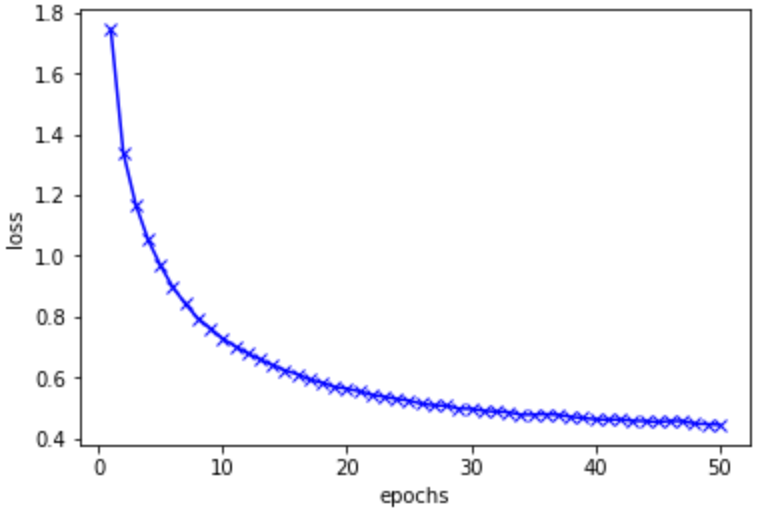
3) Adamax: It is a variant of Adam based on the infinity norm. It worked best for us as the validation score was 0.8289.

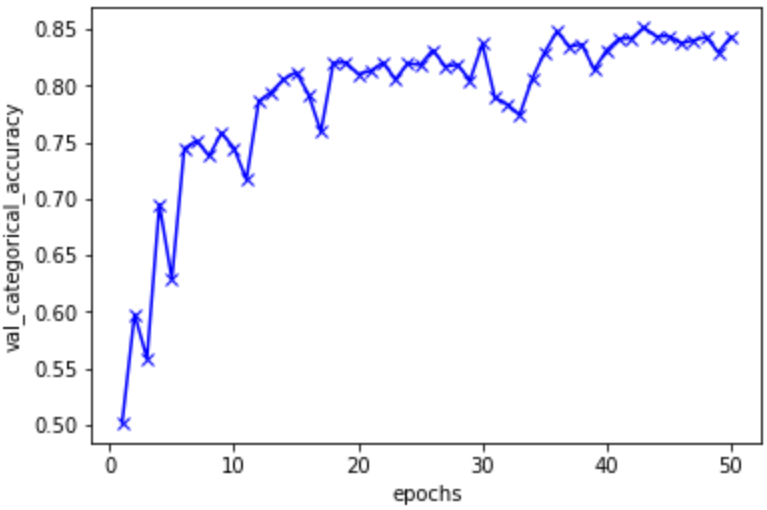
4) Adadelta: Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done. Validation score = 0.8003.

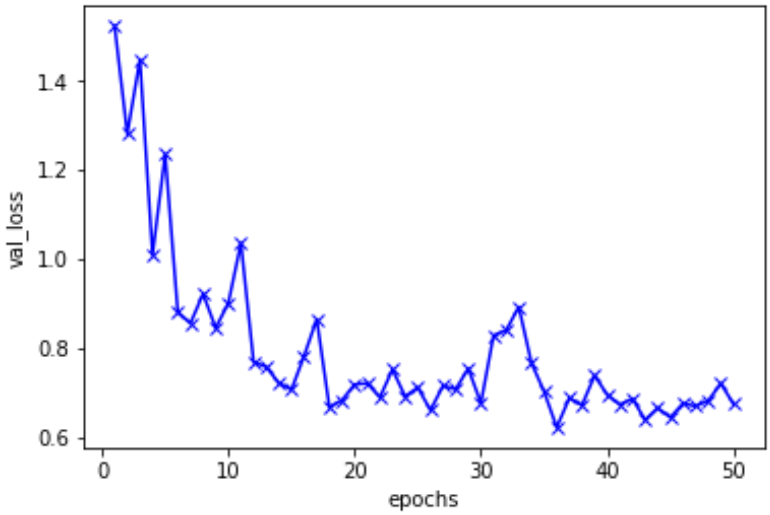
Optimizer	Learning Rate	Layer1	Layer2	Layer3	Loss Function	Epochs	Batch_Size	Dropout	Validation Score
Adamax	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8289
Adamax	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0, 0	0.8109
Adadelta	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8003
Adadelta	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.25, 0	0.7994
Adadelta	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.25, 0	0.7988
SGD	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0,.25 0.25, 0	0.7998
SGD	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.2.5, 0	0.7946
SGD	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.7829
Adam	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.8094
Adam	0.001	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0.25, 0.25, 0	0.7996
Adam	0.005	ReLU	ReLU	Tanh	categorical_crossentropy	50	128	0, 0.25, 0	0.7901

Final Plots









References

- https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/
- https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33
- https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/
- https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
- https://keras.io/optimizers/
- https://github.com/goodboyanush/iiit-bangalore-march-april-2019