ASSIGNMENT 2

ELL-793 COMPUTER VISION

Deepak (2021JOP2393)
&
RAHUL SAHA (2021JTM2223)

A Report Presented for the assignment on
Convolutional Neural Networks

Bharti School Of
Telecommunication Technology and Management
IIT Delhi India
Mar 06, 2022

CONTENTS :

QUESTION 1

1. **MNIST is a handwritten Digit Classification dataset that contains 60,000 training and 10,000 test images. Using this dataset, build a three-layer feedforward neural network. Use 10K images from the training set as validation data. The two hidden layers have a dimension of 500 each. Using a suitable loss function, train the network for 250 epochs, and report the classification accuracy on test data. Do not use regularizations. Plot the error and classification accuracy on both training and validation data over the epochs. Justify the loss used to train the network. Also, normalize the data appropriately.**

2. **Repeat the above experiment, but train the network with the following regularizations:**
   • l2 regularization
   • Dropout
   • Early stopping
   Compare with the results in the previous experiment and detail your observations after adding the regularizations. Note: No need to implement them on your own; the software framework (Tensorflow/Pytorch) typically provides implementations for l2 regularization and dropout. Early stopping is employed during training, so you need to tune your training code accordingly.

ANSWER :

In this question, we took an MNIST dataset and passed it through a neural network that has 2 hidden layers of size 500. The output layer size is 10. We have applied SoftMax in the output layer so that we can get the output as a probability. Relu function was used as activation function which adds non-linearity in the neural network.
We carried out our experiment first using no regularization technique and then we added different regularization technique and tried to see the difference in performance.
The different regularization techniques used in this network are

1. L2 regularization
2. Dropout
3. Early Stopping

1.1 L2 regularization

While training a model , it may become overfit which can result into lower training loss and higher validation loss. This is not the right thing. We should always try to avoid overfitting of the model. We add a small penalty to stop over fitting from happening.

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

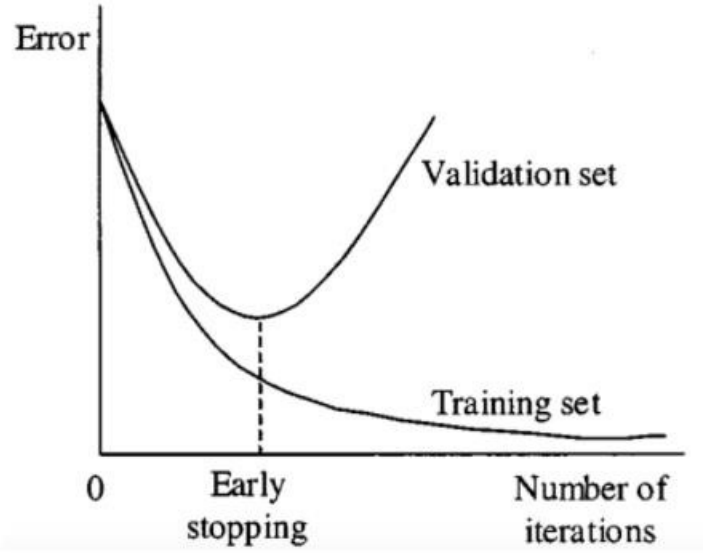
We add regularization term with the loss function which acts as a penalty . Lambda is a hyperparameter which can be adjusted according to our need. While calculating the loss function in the gradient calculation step , the loss function tries to minimise the loss by subtracting it from the average of the data distribution.

1.2 Dropout

In this method we randomly drop units from neural network which can stop from overfitting of the neural network. This prevents from co-adapting too much. We generally drop units from the layers which has more number of connection in order to made it thin. Dropout can be applied to fully connected layers, convolution layers, recurrent layers and also the hidden layers. Dropout is not used in output layer.

1.3 Early Stopping

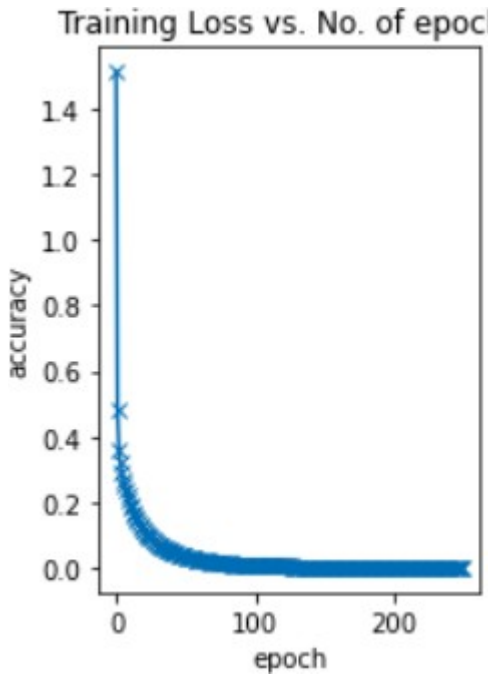
Early stopping is a form of regularization method where we stop gradient descent algorithm if validation error increases beyond a certain limit or it reaches a saturated level of accuracy . This helps in stopping the model from overfitting.

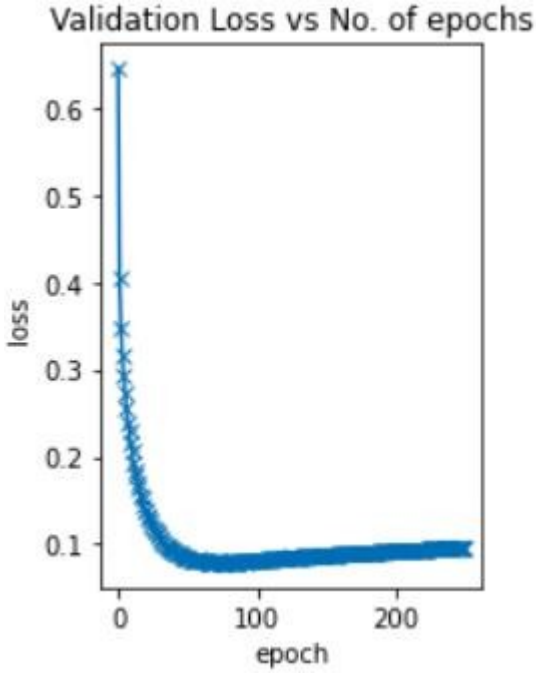


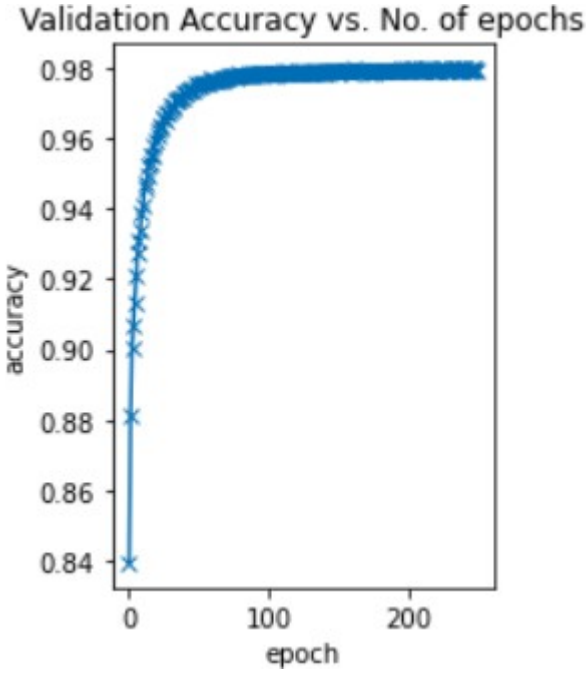
1.4 RESULTS

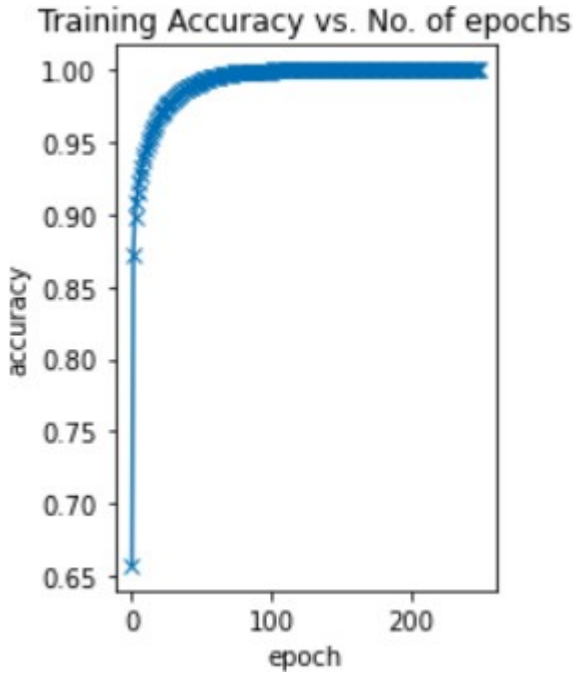
- **no regularization**

test_loss : 0.0766433402
test_acc : 0.9814453125





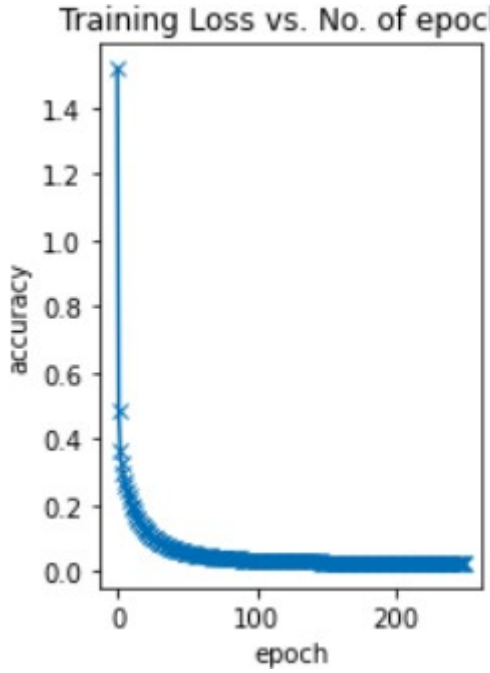


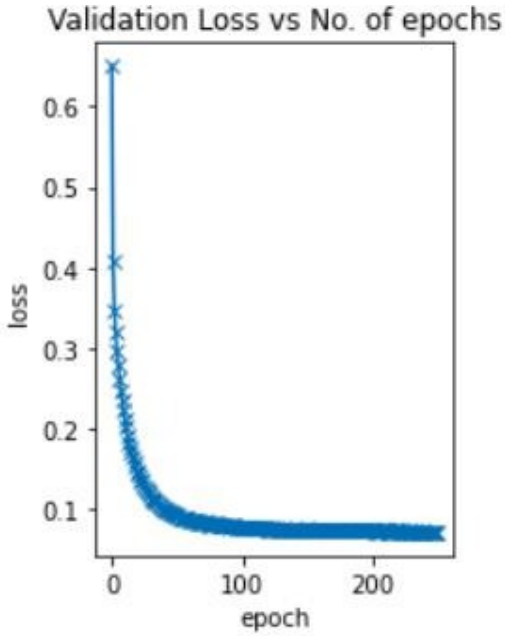


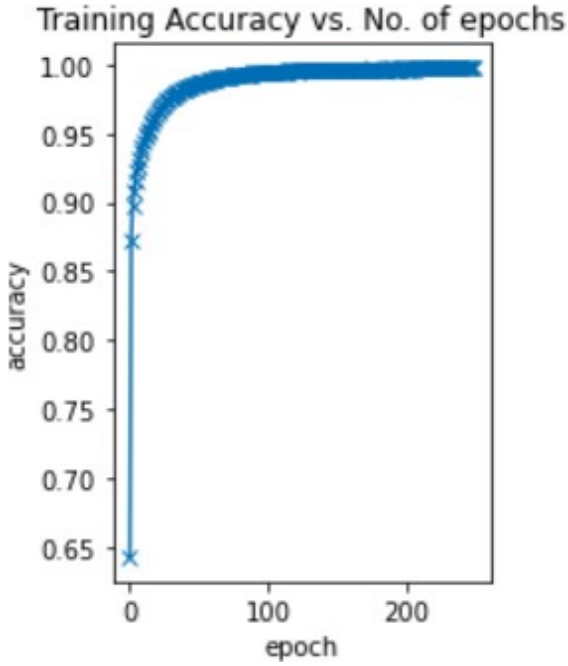
- **L2 Regularization**

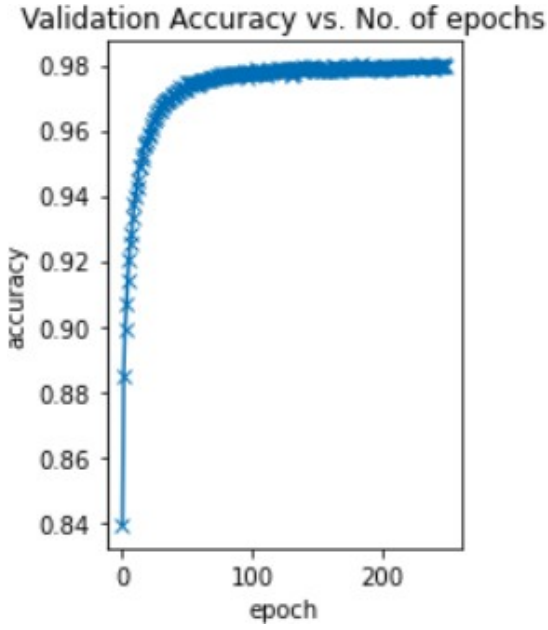
test_loss: 0.0601287782

test_acc: 0.982421875





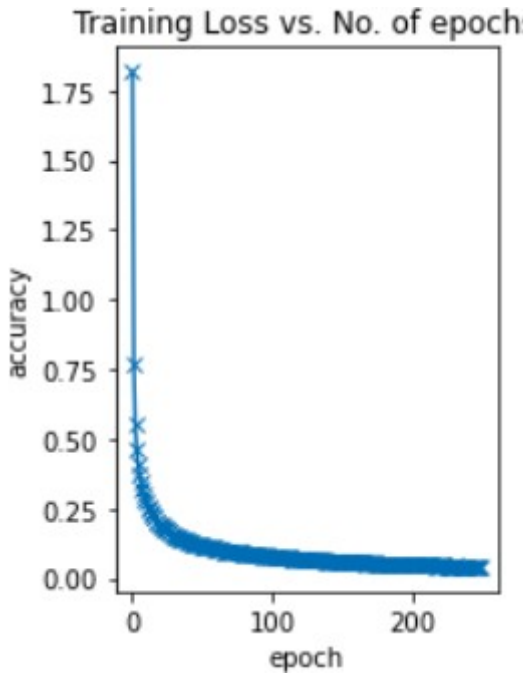


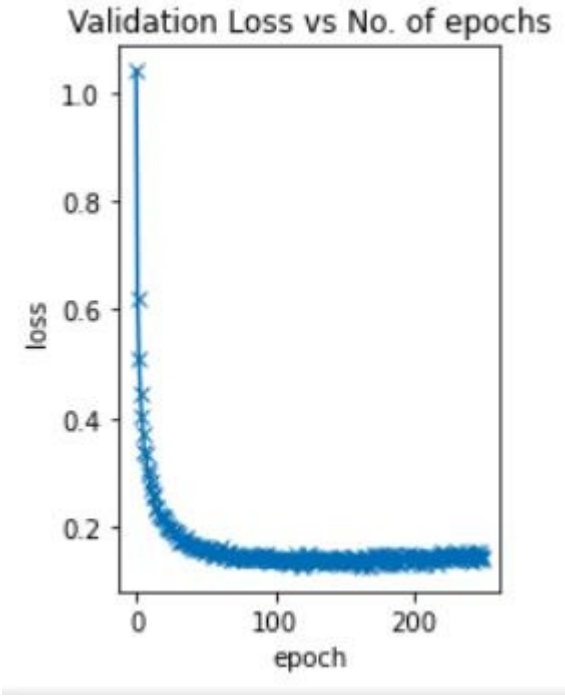


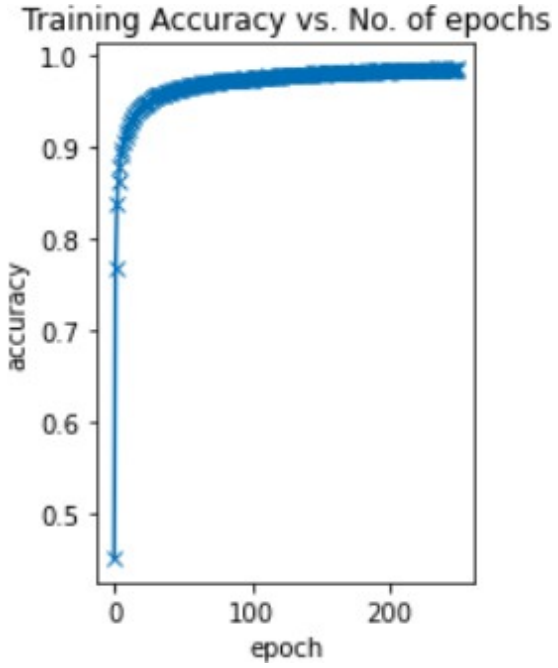
- **Dropout**

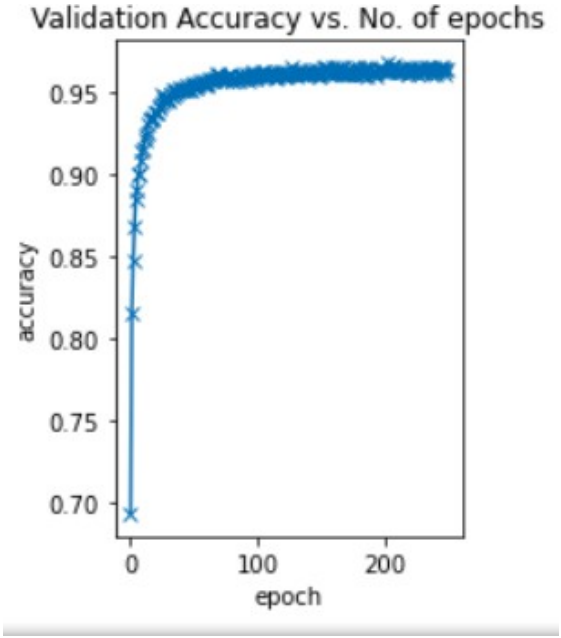
  test_loss: 0.13864083588123

  test_acc: 0.966015636920929





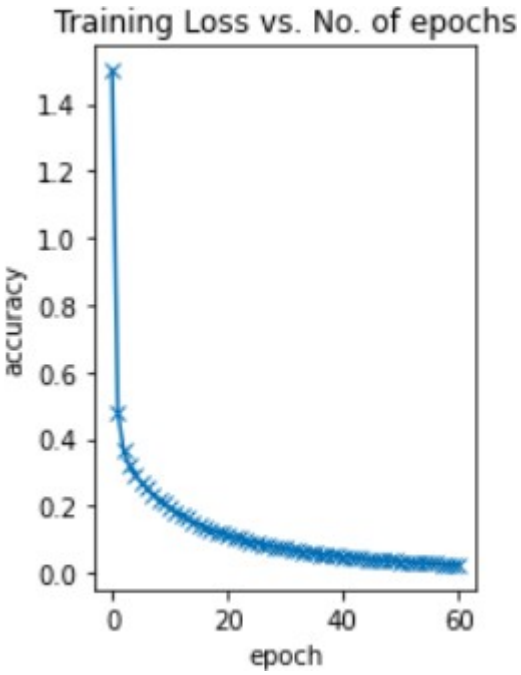


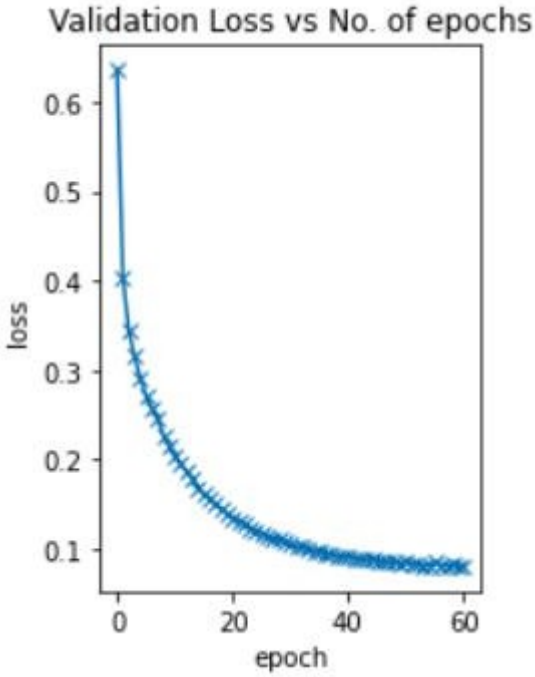


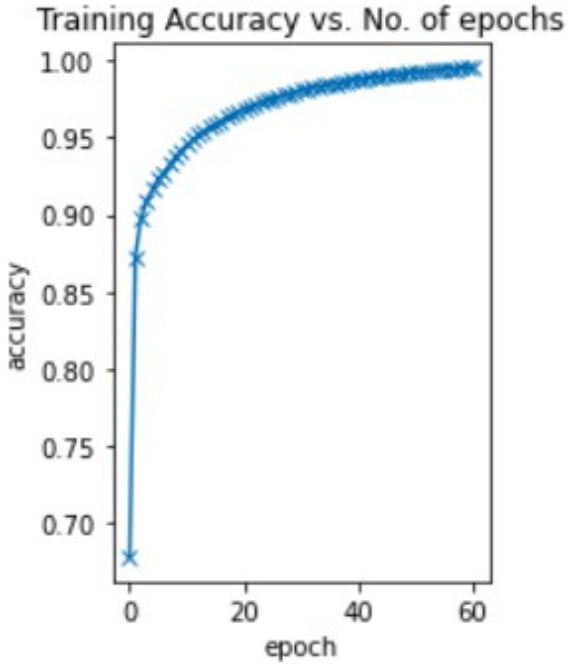
- **Early stopping**

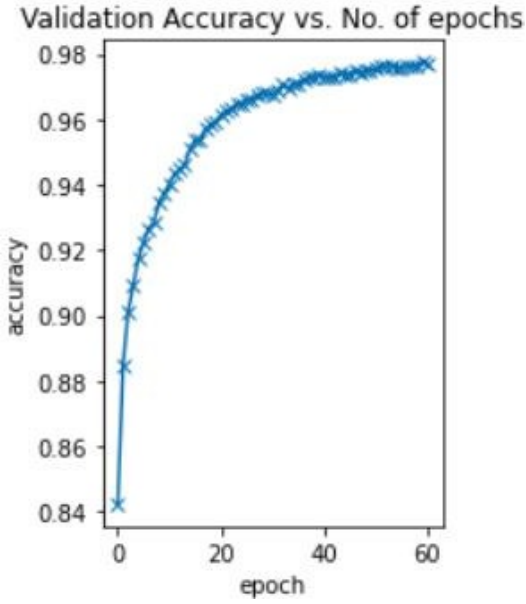
  test_loss: 0.065050162374973

  test_acc: 0.9803711175918579









1.5 CONCLUSION

- In no regularization, we can clearly see that overfitting occurs. After 100 epochs , validation loss starts increasing , while training loss keeps decreasing or remains same. This happens only when overfitting occurs. To stop this different regularization techniques are employed

- The validation loss will always be more than the training loss and validation accuracy will always be less than training accuracy.

- We can see a slight improvement in test loss and test accuracy after we apply regularisation compared to no regularisation. However the improvement is less because we ran the model for 250 epochs and saturation occurred in validation and training losses.

- The increase in validation loss after 100 epochs which we were able to see in no regularization model , has been eliminated in the models where regularization has been applied.

- In Early stopping regularization technique, we can see that only 60 epochs ran and after that the training was halted. After 60 epochs the validation loss started increasing and thus early stopping stopped training. The patience given for Early stopping was 20 , which means if the value of loss remains same for 20 epochs then it means that saturation has occurred and thus training is stopped.

- So we can conclude that all three regularization techniques were able to achieve better performance by decreasing the validation loss and stopping it from overfitting.

- In our simulation L2 regularisation was able to reach lowest test loss.

QUESTION 2.1

Train a CNN network with ResNet-18 as a backbone from scratch with CIFAR-10 and note down the performance.

2.1.1 THEORY

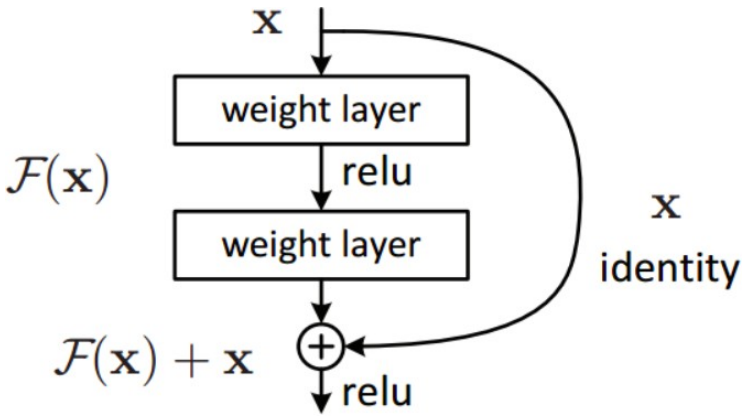
ResNet −18 is a convolution neural network that has 18 layers. We made a ResNet-18 architecture from scratch and applied regularization techniques such as learning rate scheduling, weight decay and gradient clipping to stop overfitting.

The regularisation techniques used to prevent overfitting are

- **Learning rate scheduling:** We will utilise a learning rate scheduler, rather than a set learning rate, to vary the learning rate after each batch of training. The "One Cycle Learning Rate Policy" entails starting with a low learning rate, gradually raising it batch by batch to a high learning rate for roughly 30% of the epochs, and then gradually dropping it to a very low value for the remaining epochs.
- **Weight decay:** Weight decay is another regularisation strategy that prevents the weights from growing too big by adding an extra term to the loss function.
- **Gradient clipping**: In addition to layer weights and outputs, it's also a good idea to keep gradient values within a narrow range to avoid undesired changes in parameters caused by excessive gradient values. Gradient clipping is a straightforward yet powerful method.

Skip connection:

The inclusion of the residual block, which adds the original input back to the output feature map created by passing the input through one or more convolutional layers, is one of the most significant additions to our CNN model this time.



The following is the ResNet 18 architecture.

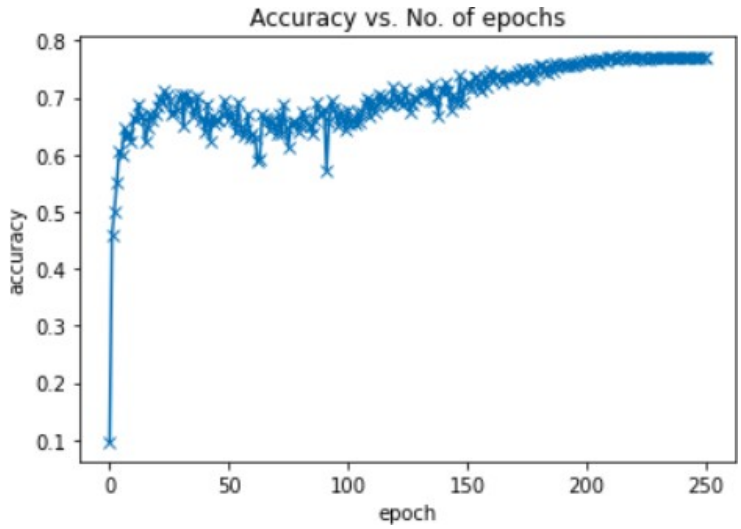
Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

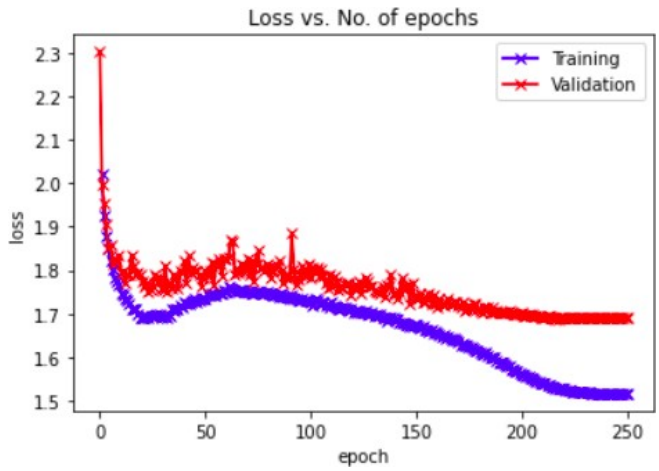
- We use the ResNet18 architecture as a backbone to find different classes in CIFAR10 dataset.

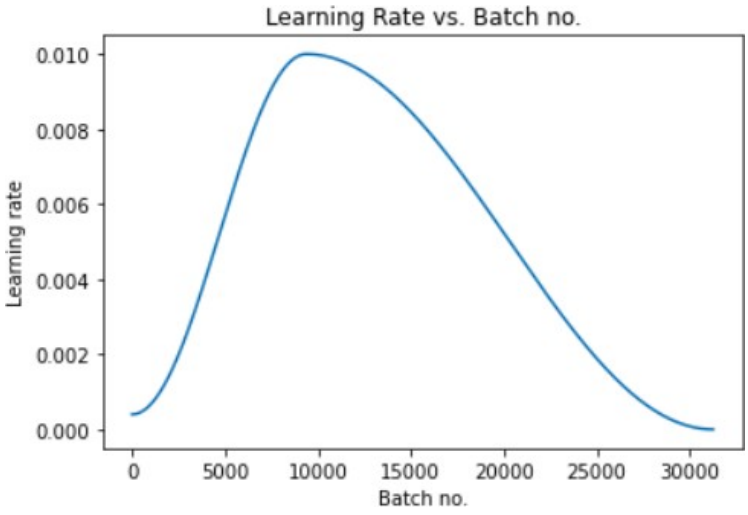
- The following are the performance graphs of our ResNet18 architecture-

2.1.2 RESULTS

Epoch [249], last_lr: 0.00000, train_loss: 1.5164, val_loss: 1.6909, val_acc: 0.7692







2.1.3 CONCLUSION

- We are able to reach accuracy of 76% and we are able to stop overfitting using different regularization techniques.

- The Training loss and validation loss keeps on decreasing as number of epochs keeps on increasing. Thus, we can clearly see that overfitting does not occur.

- We have selected learning rate scheduling regularization which helps to stop overfitting.

QUESTION 2.2

Initialize the ResNet-18 network with pre-trained weights from ImageNet and then try to use these weights to improve the training for the CIFAR-10 dataset. Try to come up with different ways of using these weights to improve the performance and play with the hyper-parameters to get the best performance. Document the results of your experiments.

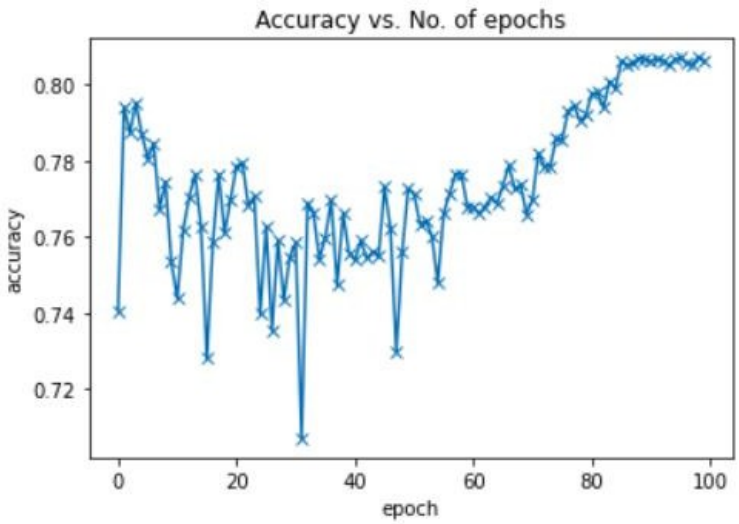
2.2.1 THEORY

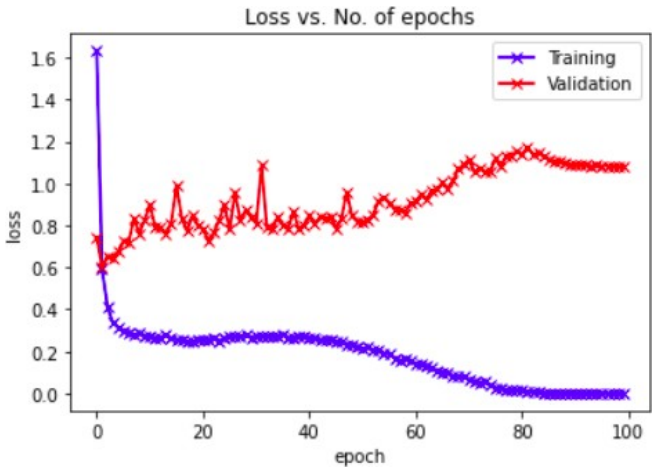
We have used pretrained model of ResNet18 and have again trained the model on our dataset and have recorded various parameters. Hyperparameters can be adjusted by the user to get the best performance out of the model. We have taken different combinations of hyperparameters and tried to come up with a model which gives best performance. We ran each model for 100 epochs and tried to predict the best hyperparameter.

2.2.2 RESULTS

The different hyperparameters and the observations are as follows

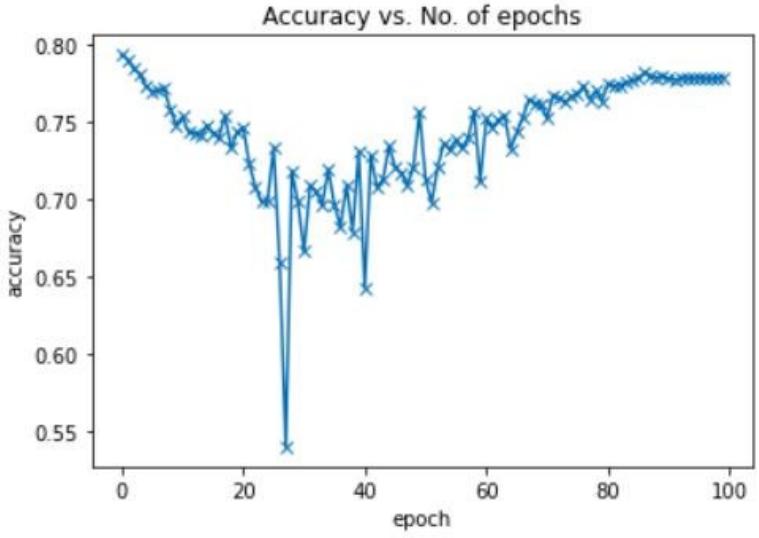
max_lr: 0.01 -- grad_clip: 0.1 weight_decay: 0.0001

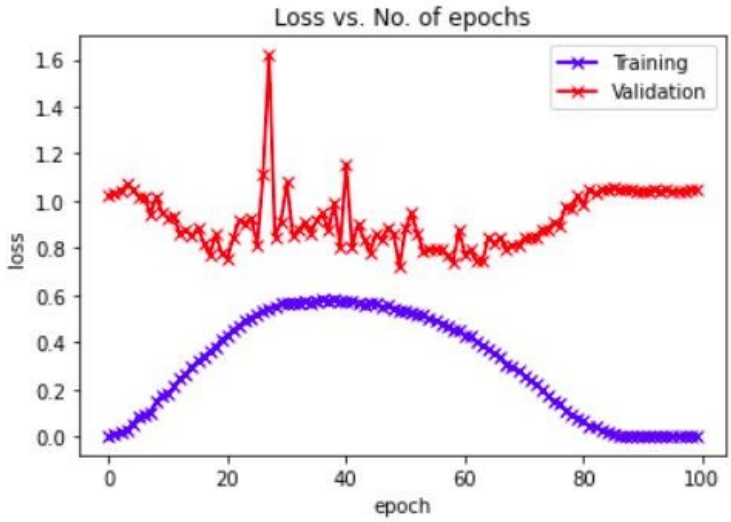




With this set of hyperparameter we can clearly see that the validation errors keeps on increasing and training errors keeps on decreasing , thus this model overfits. This are not the best hyperparameter and thus it needs to be improved further.

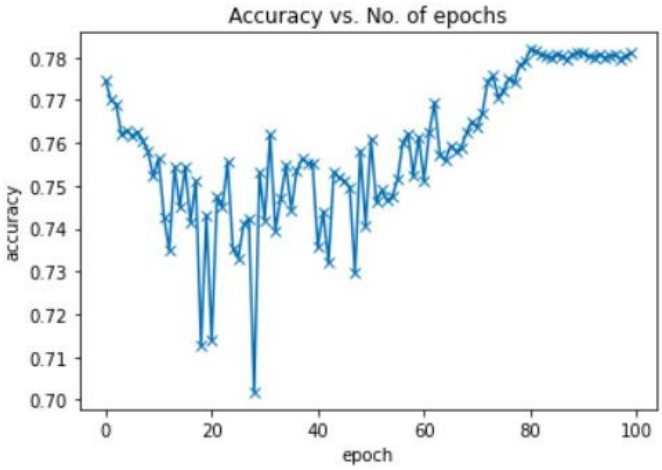
max_lr: 0.01 -- grad_clip: 0.1 weight_decay: 0.000600000000000000

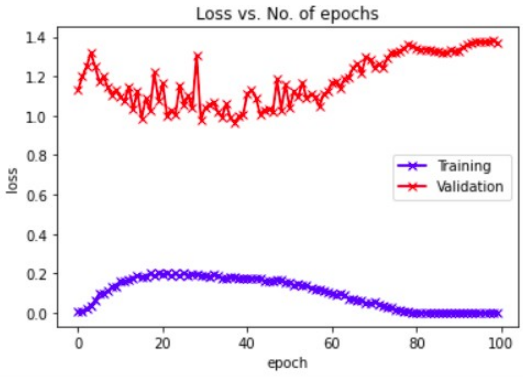




There is huge difference between training loss and validation loss which refers to overfitting because validation loss increase after 60 epochs while training loss decreases.

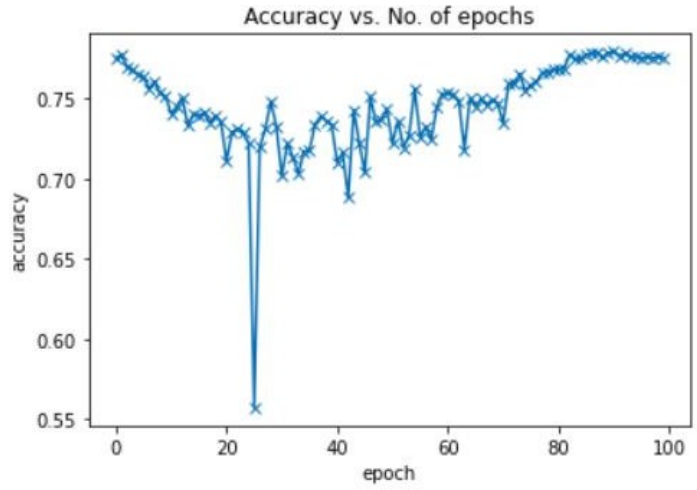
max_lr: 0.01 -- grad_clip: 0.6 weight_decay: 0.0001

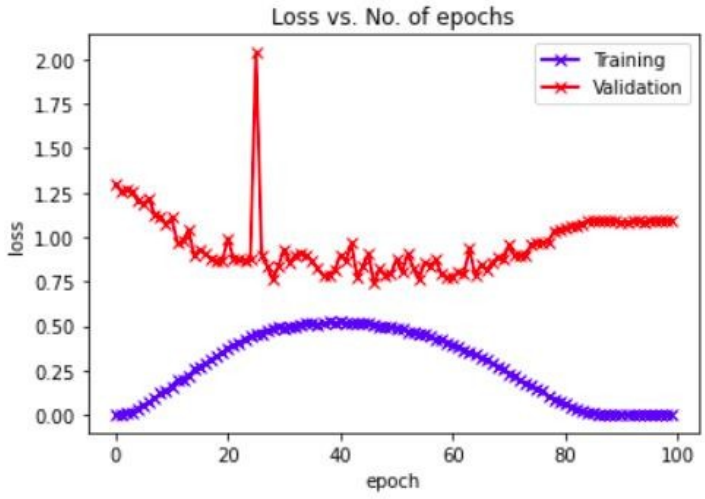




The following hyperparameter also leads to overfitting . The accuracy of the model increases but it is due to overfitting. After epoch 60 , the validation loss starts increasing while training loss starts decreasing

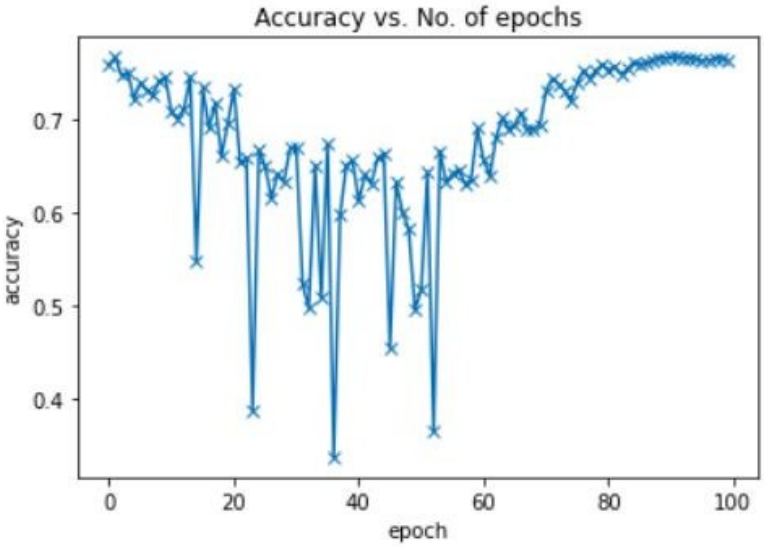
max_lr: 0.01 -- grad_clip: 0.6 weight_decay: 0.0006000000000000001

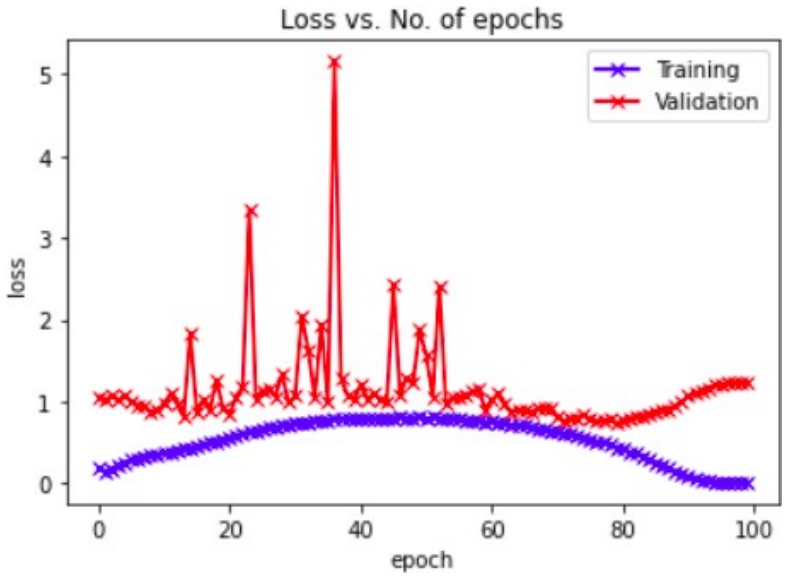




This hyperparameter performs better than the last one because the difference between validation loss and training loss is less. However, this model starts overfitting after 80 epochs. If this model was run for 60 epochs, then this could have been a better model. With the increase of weight decay this performance improvement was achieved.

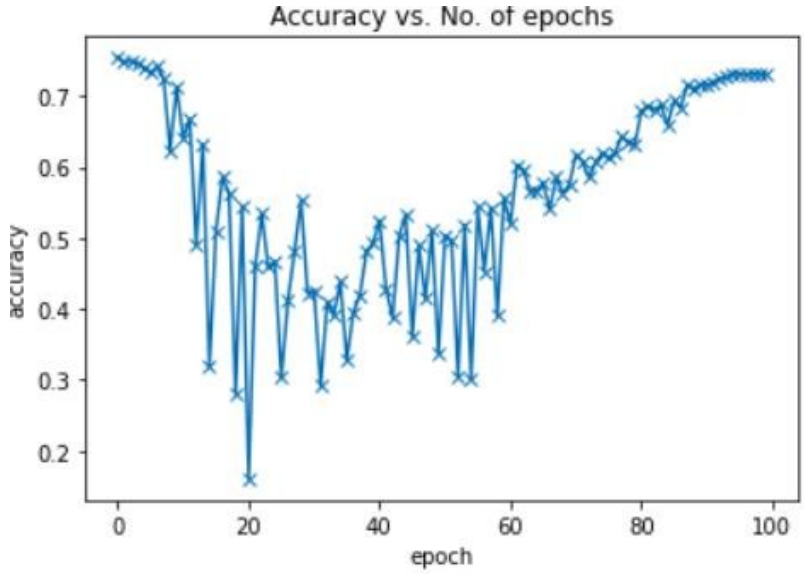
max_lr: 0.060000000000000005 -- grad_clip: 0.1 weight_decay: 0.0001

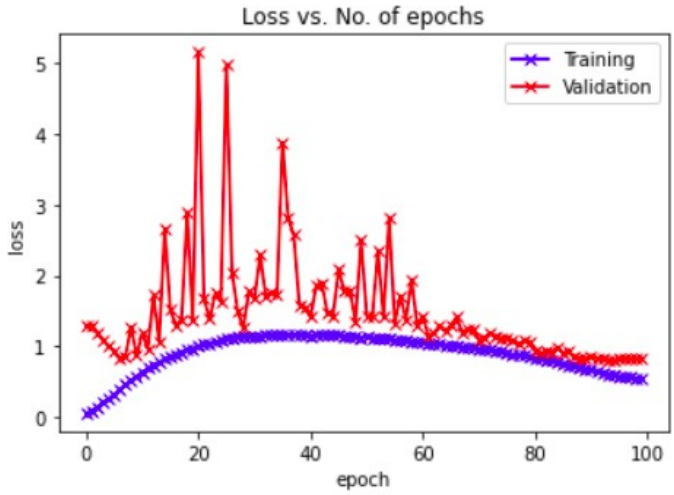




With the following hyperparameters we were able to achieve a better model. The difference between training loss and validation loss is least. However, just like the last model , it starts overfitting from 80 epochs.

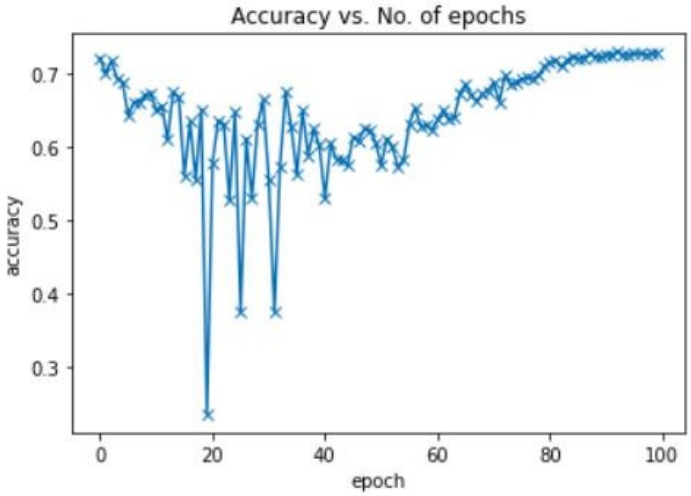
max_lr: 0.060000000000000005 -- grad_clip: 0.1 weight_decay: 0.0006000000000000001

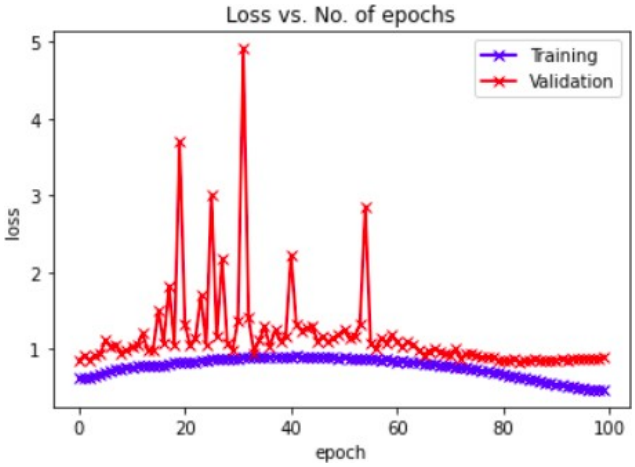




The difference between the validation loss and training loss is least and there is no overfitting in the model. However the accuracy decreases to some extend which is negligible. This model is better because we were able to control overfitting till 100 epochs.

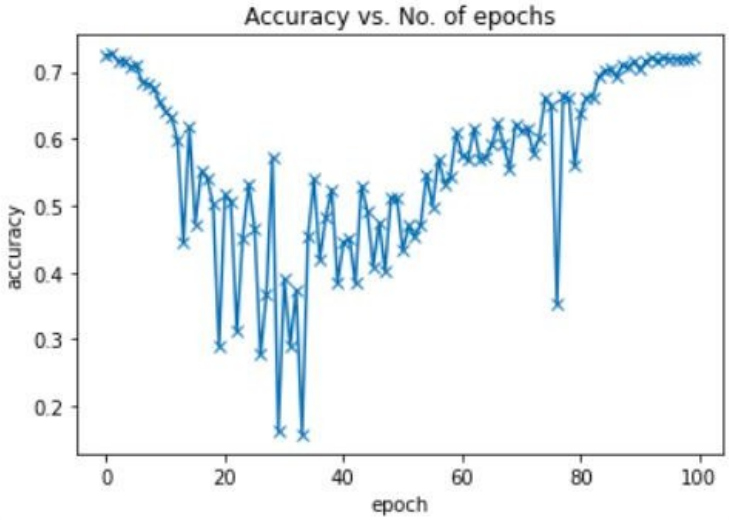
max_lr: 0.060000000000000005 -- grad_clip: 0.6 weight_decay: 0.0001

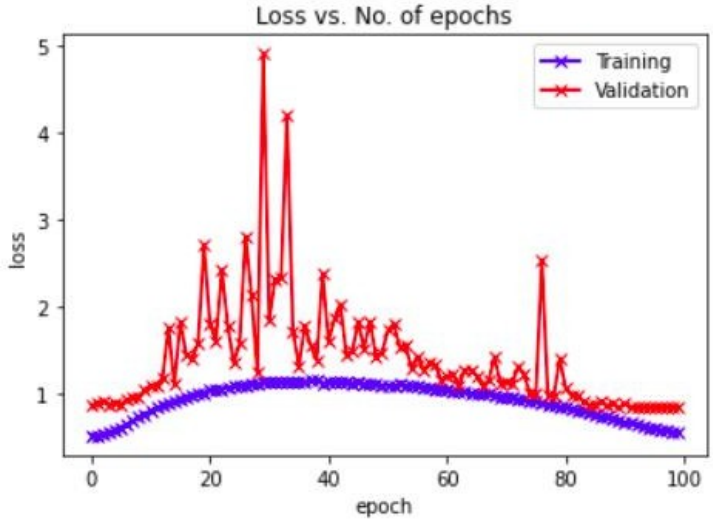




This hyperparameters are better than last model hyperparameters because we were able to improve accuracy and still stop the model from overfitting till 90 epochs. The difference between validation loss and training loss is least. However there is a slight increase of validation error after 90 epochs. The hyperparameters should be changed to control this.

max_lr: 0.060000000000000005 -- grad_clip: 0.6 weight_decay: 0.0006000000000000001





In this model there is no overfitting and the validation loss and training loss are almost similar. The difference between them is least and accuracy almost remains same after 100 epochs.

2.2.3 CONCLUSION

We see that the model with

- max learning rate of 0.060000000000000005
- Gradient clipping of 0.6
- Weight decay of 0.0006000000000000001

performs best as the validation loss and training loss keeps on decreasing and the difference between them is less. There is no overfitting and accuracy is also good and thus this is the best model.

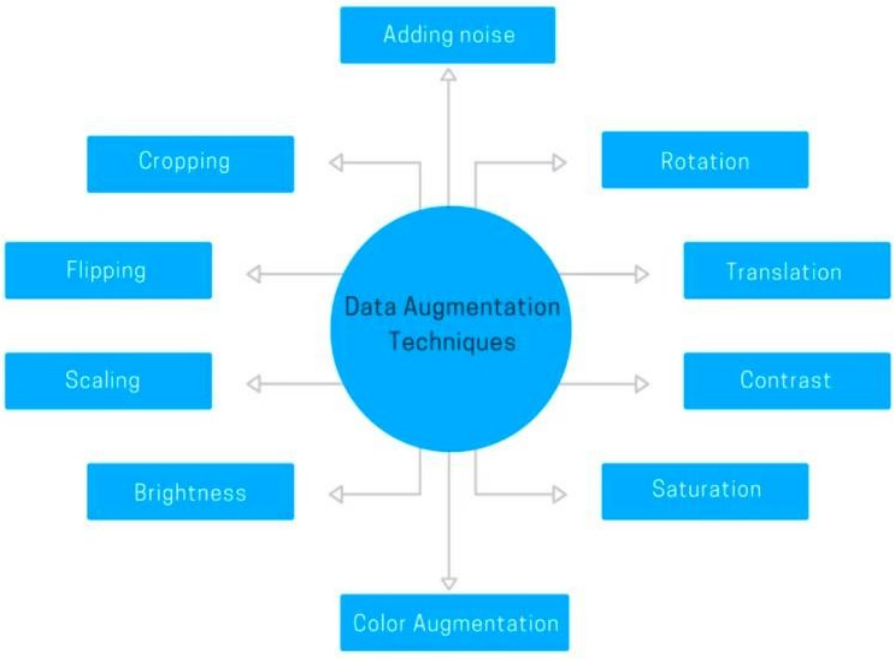
We observe that having higher maximum learning rate, higher gradient clipping and higher weight decay prevents the model from overfitting and performs better. Thus higher regularization does make a better model.

QUESTION 2.3

Train the network from scratch with the Tiny-CIFAR-10 dataset. Try using as many data augmentation techniques as you can think of to try to improve the performance. Try dropout after different layers and with different dropout rates. Document the results of your experiments.

2.3.1 THEORY

We took Tiny –CIFAR-10 dataset which has only 500 photos in each classes. Thus the amount of dataset is less and thus we need to find other ways to increase the dataset so that we can train the model . We use data augmentation techniques to change the properties of the photos and then train the model upon them . This also helps in preventing the model from overfitting.



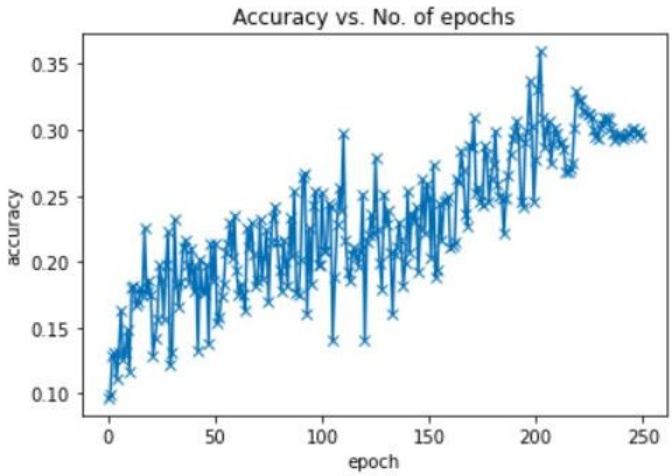
The above figure shows different data augmentation techniques that can be used to prevent model from overfitting. While loading images from the training dataset, we'll apply transformations at random. We'll pad each image by 4 pixels, then take a random 32 by 32 pixel crop and flip the image horizontally with a 50% chance. The model sees somewhat different images in each epoch of

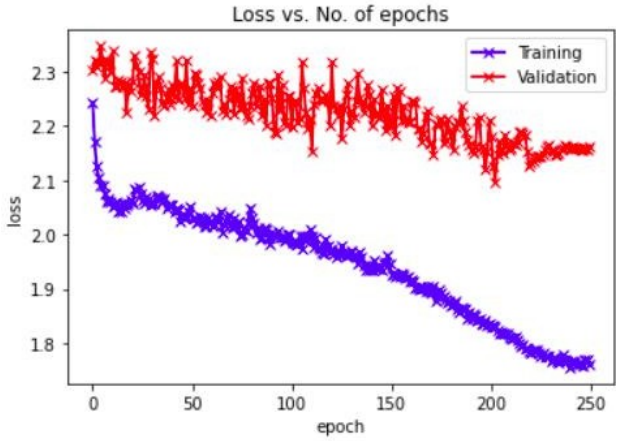
training since the transformation is done randomly and dynamically each time a given image is loaded, allowing it to generalise better.

We have used dropout method in this model as a regularisation technique. There are many layers in ResNet18 artitecture, so we thought that the best layer to apply dropout should be the layers where most number of connection is present. In ResNet18 , most number of connection is present in first convolution layer and in the fully connected layer , so we applied dropout in both first convulution layer and fully connected layer and recorded the performance

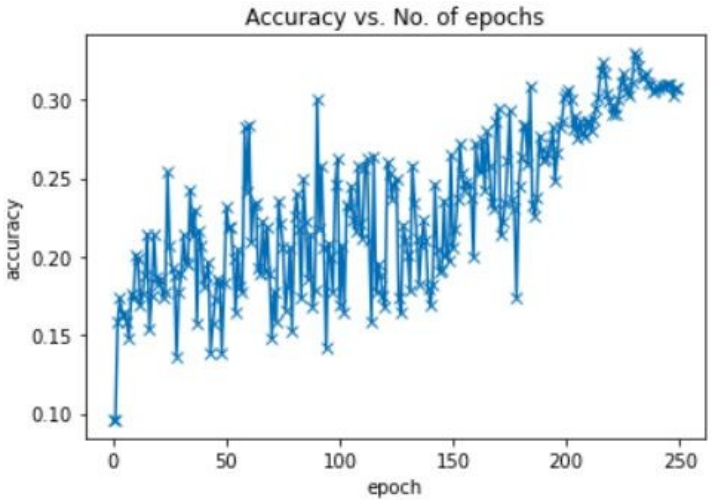
2.3.2 RESULTS

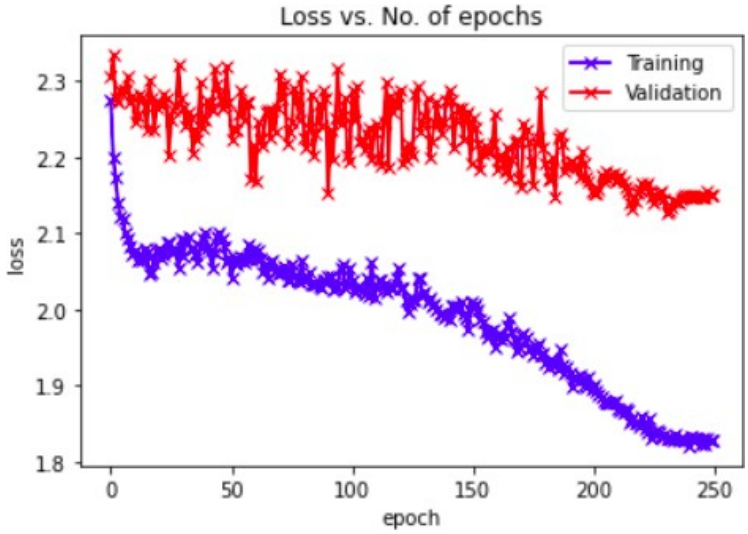
- Drop in first convolution layer =0.8
- Drop in fully connected layer=0.6



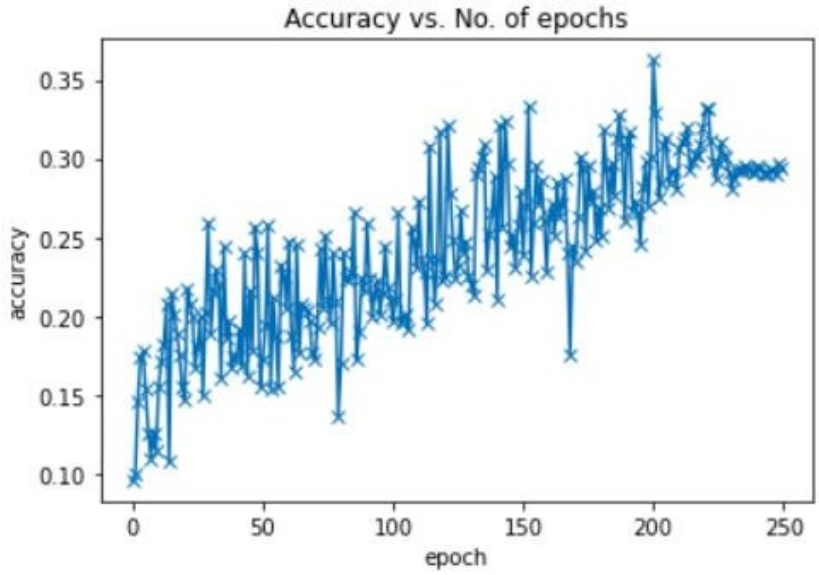


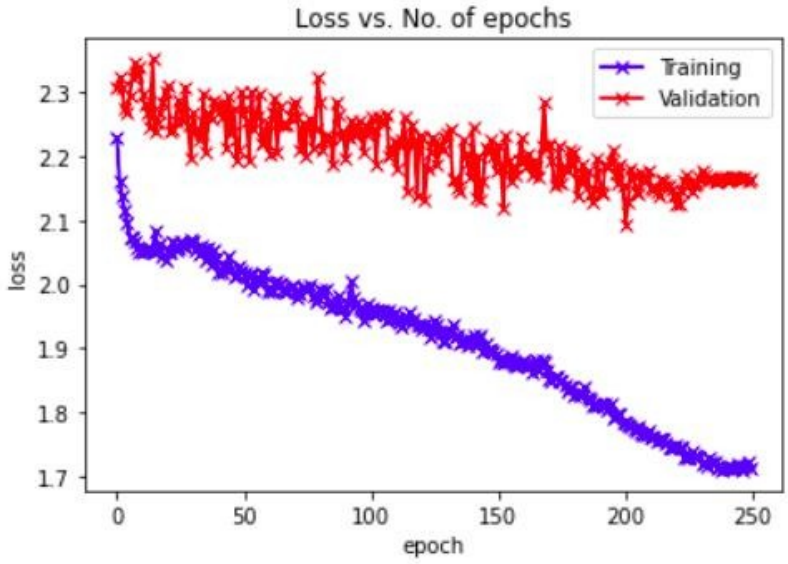
- Drop in first convolution layer =0.8
- Drop in fully connected layer=0.8



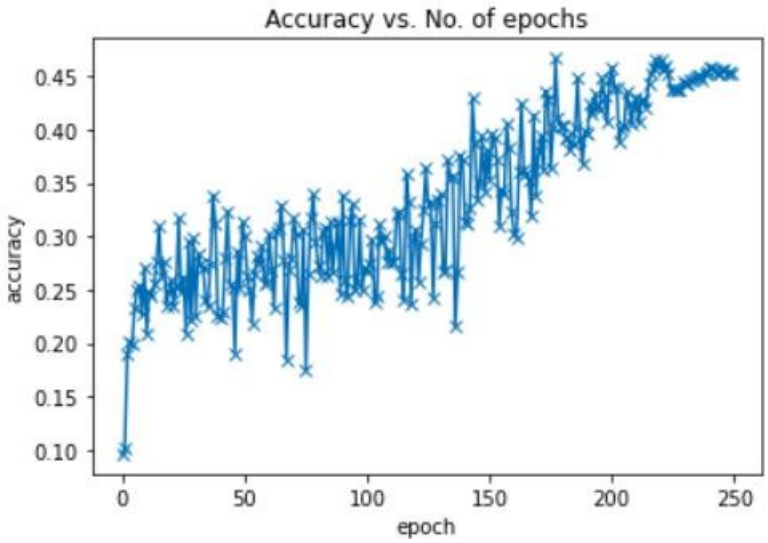


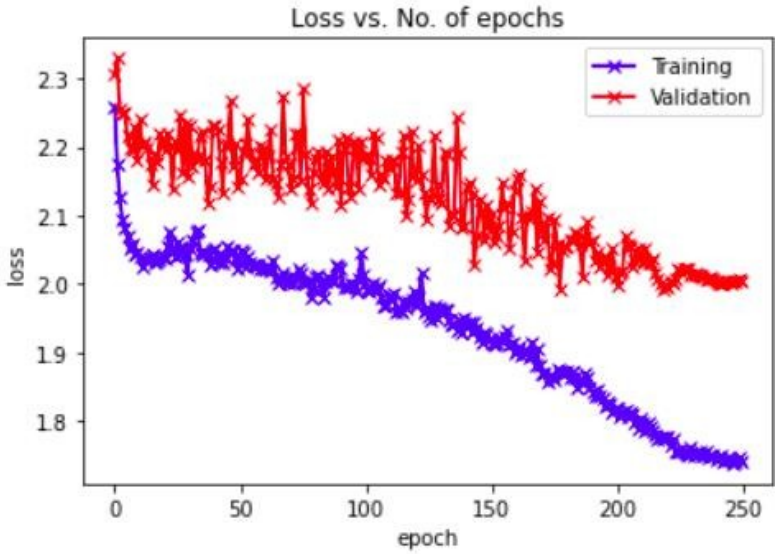
- Drop in first convolution layer =0.8
- Drop in fully connected layer=0.2



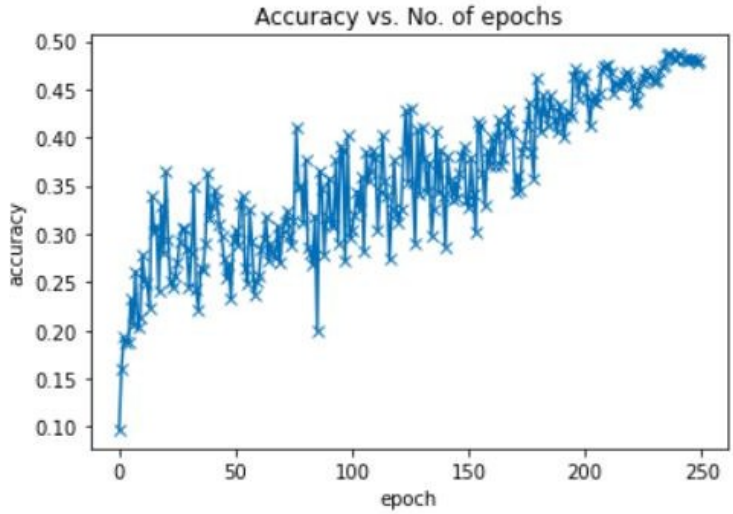


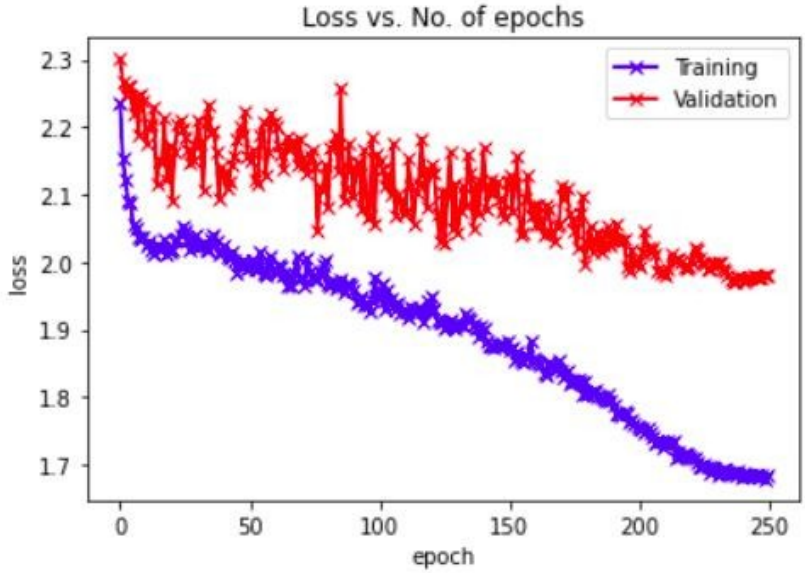
- Drop in first convolution layer =0.6
- Drop in fully connected layer=0.8



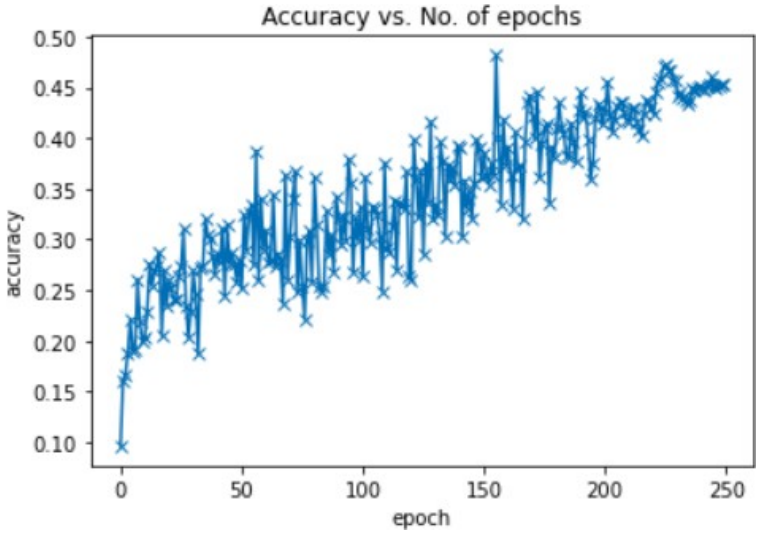


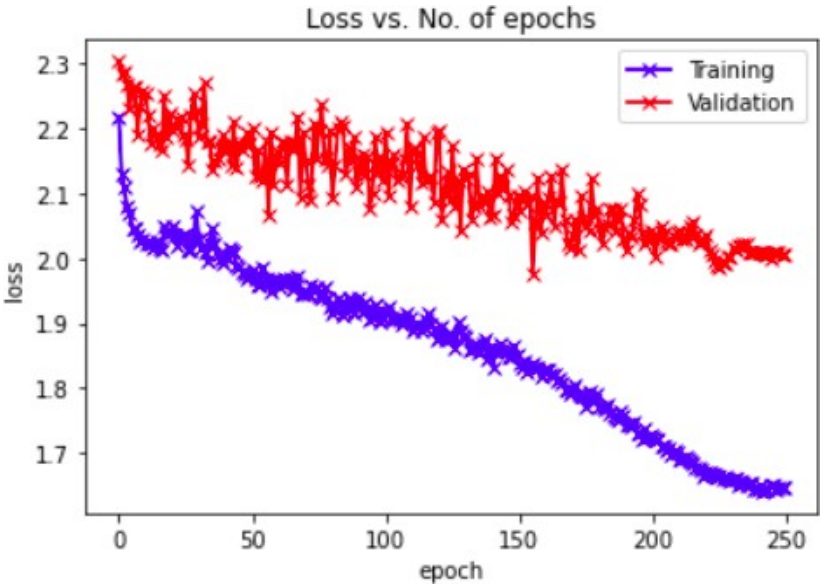
- Drop in first convolution layer =0.6
- Drop in fully connected layer=0.6



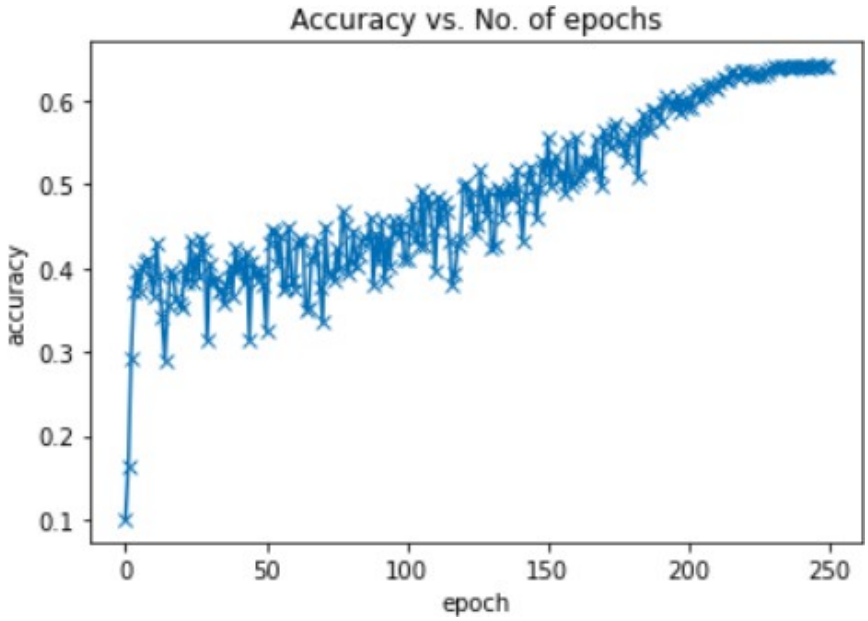


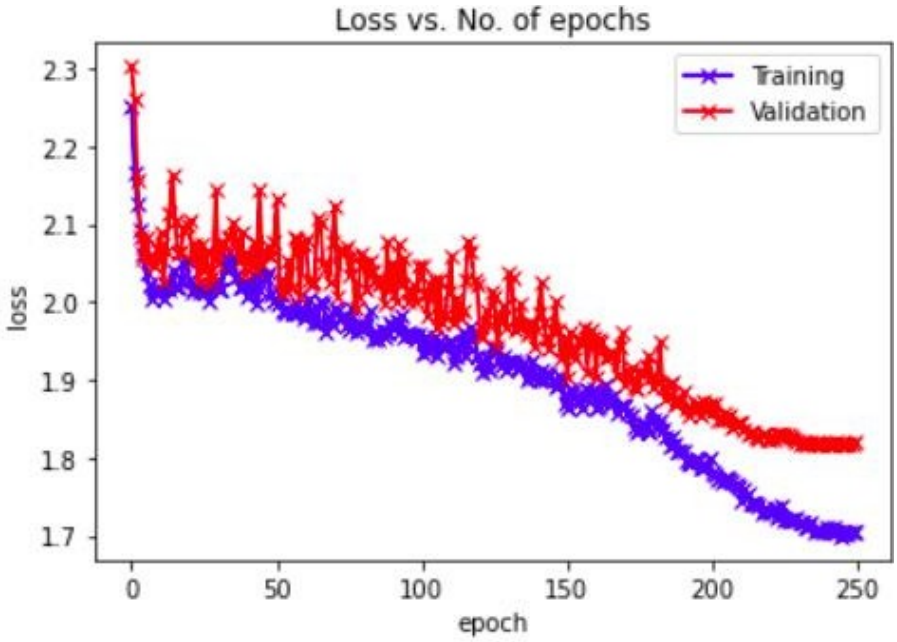
- Drop in first convolution layer =0.6
- Drop in fully connected layer=0.2



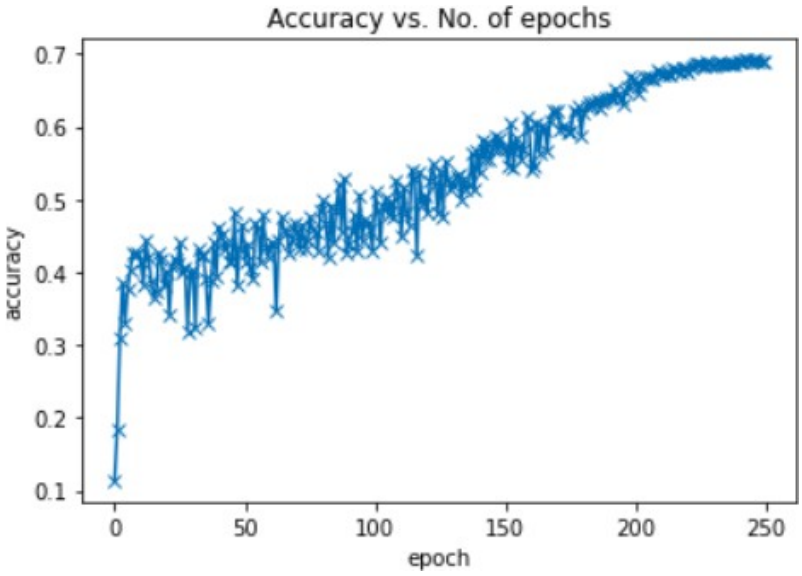


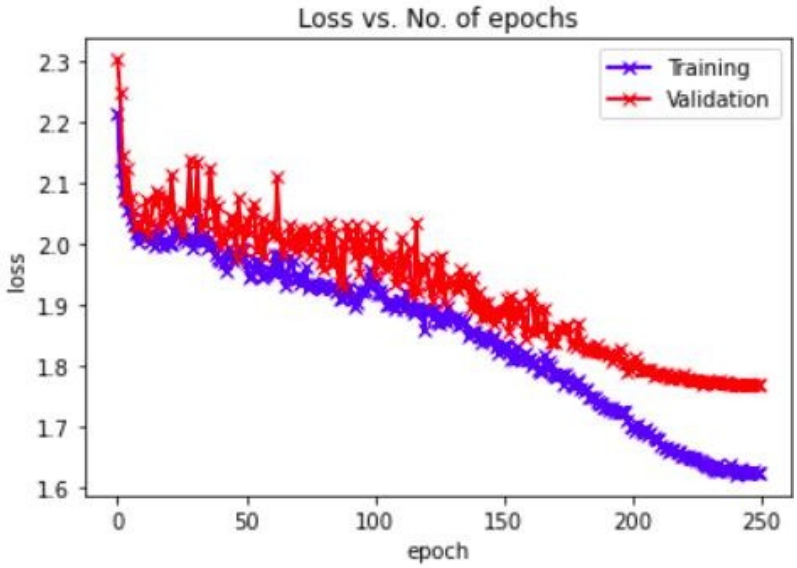
- Drop in first convolution layer =0.2
- Drop in fully connected layer=0.8



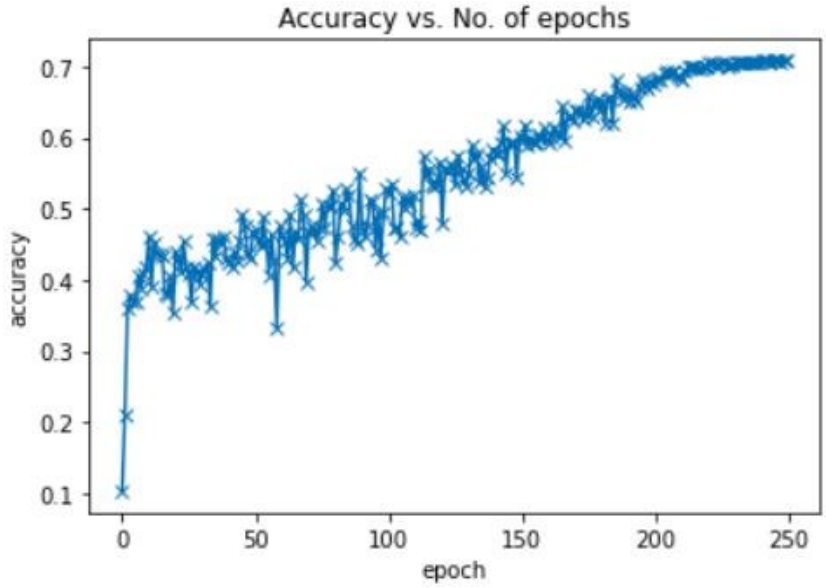


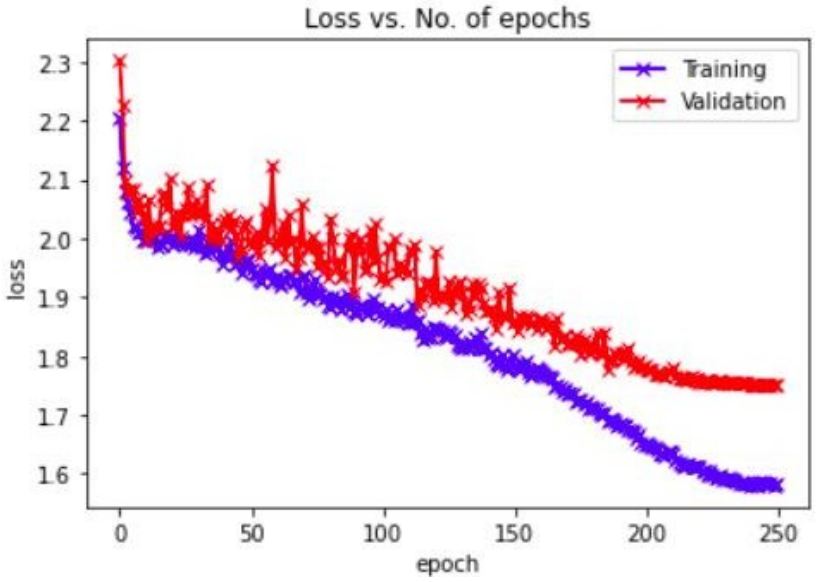
- Drop in first convolution layer =0.2
- Drop in fully connected layer=0.6



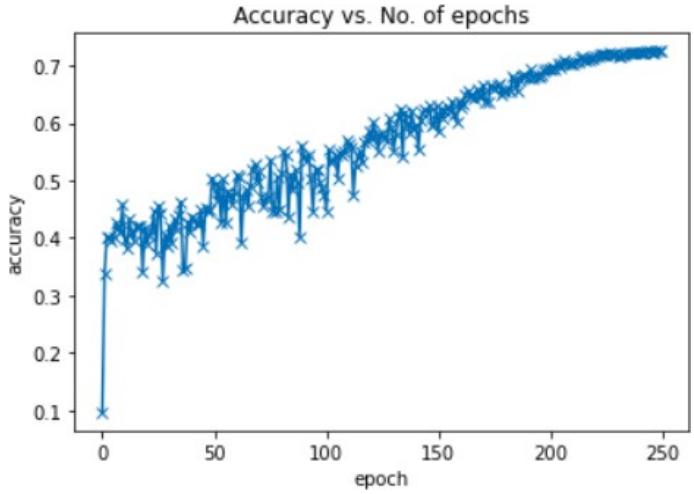


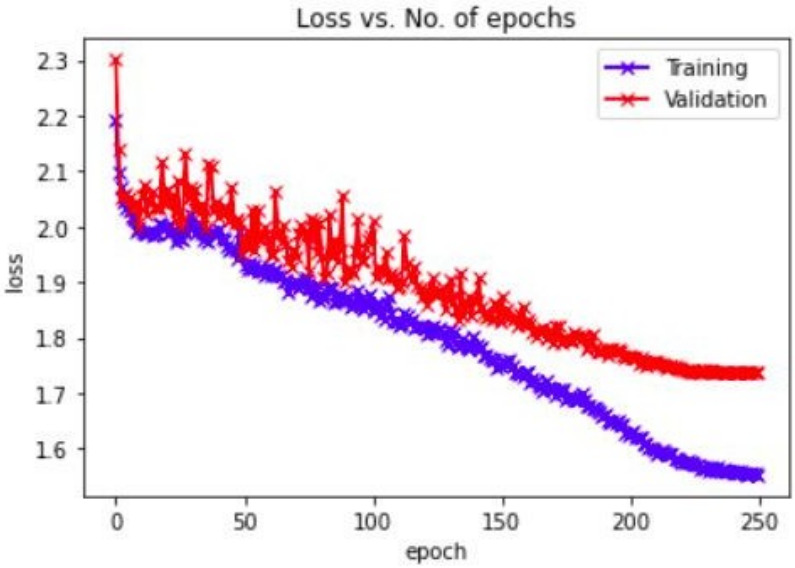
- Drop in first convolution layer =0.2
- Drop in fully connected layer=0.2





- Drop in first convolution layer =0 (NO dropout)
- Drop in fully connected layer=0 (NO dropout)





2.3.3 CONCLUSION

Dropout parameter of 0.2 in first convolution layer and 0.6 in fully connected layer is best in our observation. We conclude that having less number of dropout in first convolution layer and having relative higher amount of dropout in fully connected layer is best for the model . The difference between the validation loss and training loss is least among others and the accuracy is also comparable with others and thus this is the best parameter for dropout.

QUESTION 2.4

Visualize the activations of the CNN for a few test examples in each of the above cases. How are the activation in the first few layers different from the later layers?
Note: For the Tiny-CIFAR-10 dataset, take 500 images per class from CIFAR-10 for training. Use the same 10,000 images for testing as per the CIFAR-10 dataset. It is recommended to use Pytorch for this problem.

2.4.1 THEORY

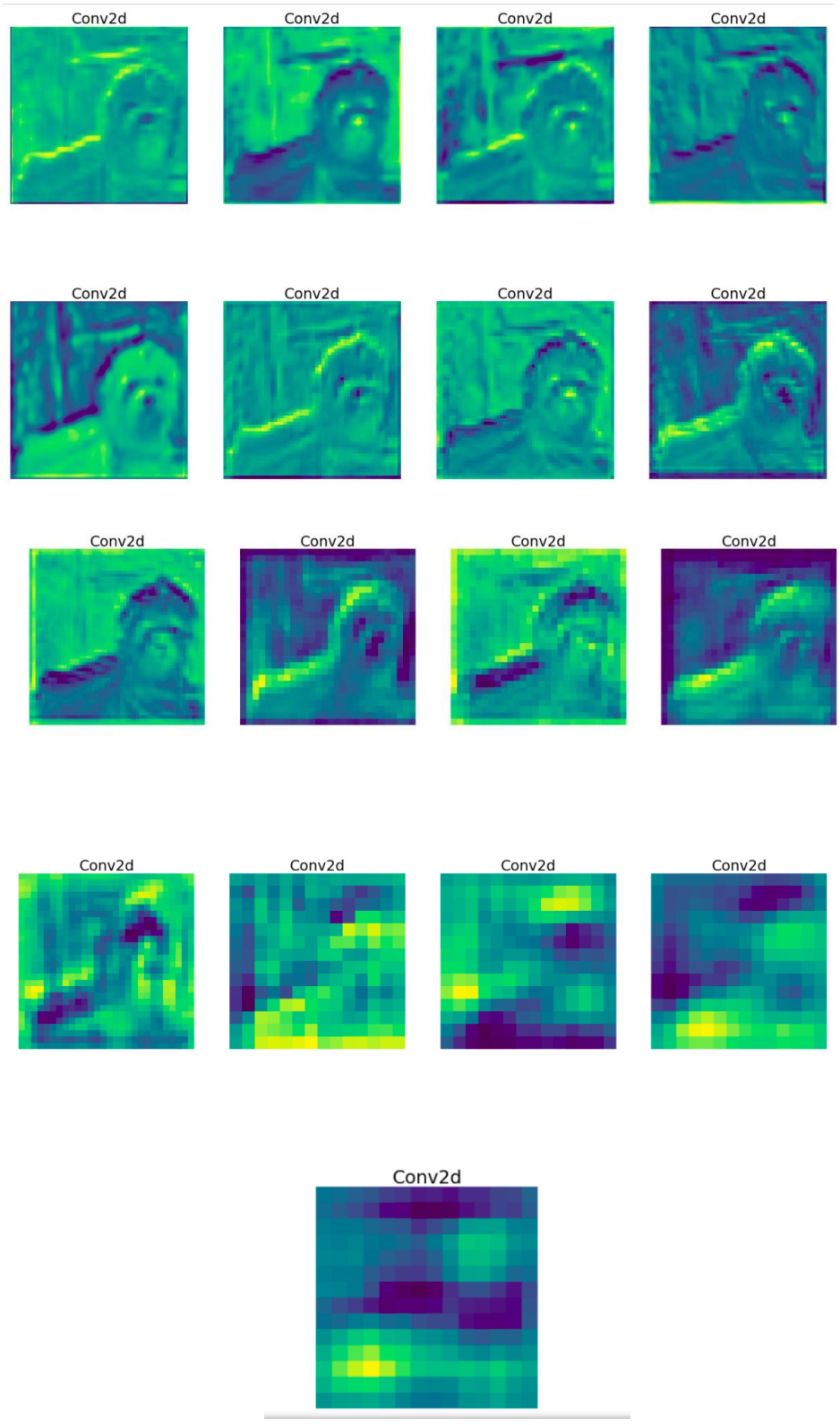
Activation maps are just a visual depiction of these activation values as a particular picture moves through the network because of different linear algebraic computations.

The activations in ReLU activation-based networks are often blobby and dense at first, but they become sparse and localised as the training advances. Some activation maps may be entirely zero for many distinct inputs, which can suggest dead filters and be an indication of excessive learning rates, which is an easy design issue to catch using this view.

Here we try to visualise activation map of each layer and try to understand what each layer is learning from an image. We use a test image and give it to a already trained model and see the visualization of different activation layers.

2.4.2 RESULTS

The activation maps of different layers are given below.



2.4.3 CONCLUSION

We saw that the first layer convolution layer detects the edges of the photo and the last convolution layer detects high level features of the photo. The internal narrative of how CNN learns to recognise distinct aspects present in images may be visualised to provide a better understanding of how the model works. It will also aid in determining why the model is failing to correctly categorise some of the photos and, as a result, fine-tuning the model for increased accuracy and precision.