COL341 : MACHINE LEARNING (Spring 2023)

Name: Stitiprajna Sahoo
Entry Number: 2020CS10394

Date: April 25 2023
Assignment: 4

Assignment 4: Neural Network

1 Implementation with Numpy

- All the layers: **MaxPool, Convolution, Relu, Softmax** were implemented using vectorisation methods of numpy.
- Inputs were passed as 4d matrices to the convolutional layers, and 2d convolution was performed along the last 2 axis using `np.einsum` methods
- Maxpooling was done using reshape and taking max along specific axes.
- For computing back gradients of convolution, similar `np.einsum` methods were used.

Parameters:
Optimiser: ADAM Optimizer
Learning Rate: 0.001
Loss function: Cross Entropy Loss

Loss curves:

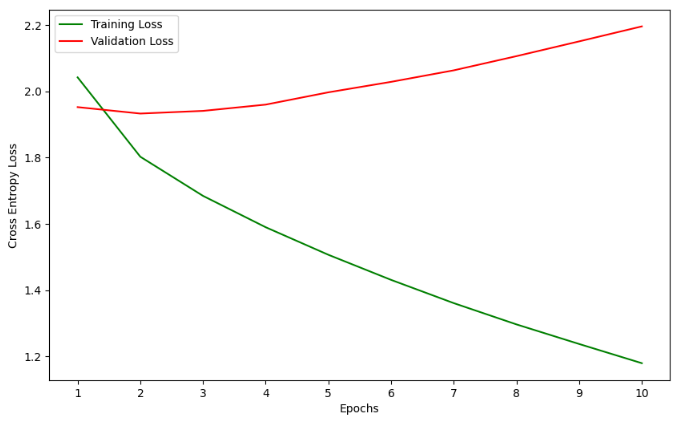


Figure 1: Loss curves for Train and Validation data with LR = 0.001

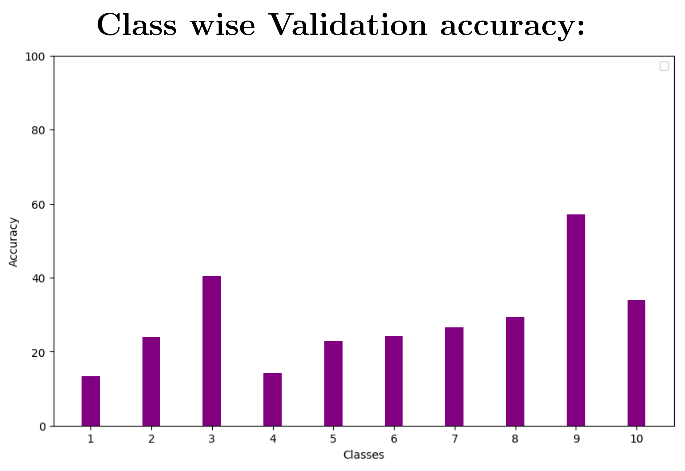


Figure 2: Classwise accuracy Validation data with LR = 0.001

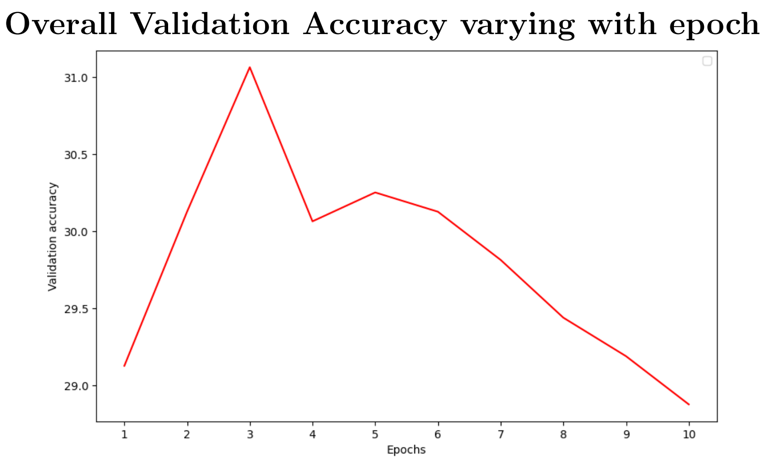


Figure 3: Accuracy on Validation data varying with epochs with LR = 0.001

Overall Validation Accuracy
31.0625

Analysis:

- The training loss is decreasing in a slower rate on applying learning rate = 0.001, whereas validation loss saturates.
- The training loss for learning rate = 0.002 is decreasing exponentially and reducing to good extent. I have run the experiment for 10 hours which took me around 16 hours.

2 Implementation with pytorch

2.1 Hyperparameter Tuning:

2.1.1 Variation in LR

LR = 0.001
Loss curves and Class-wise Accuracy Graphs:

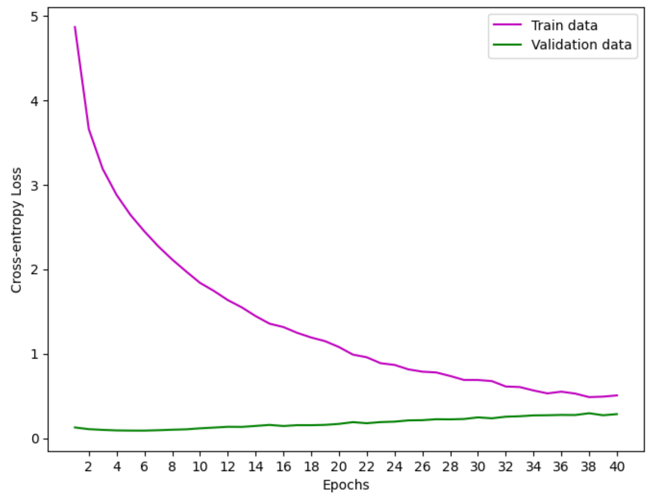


Figure 4: Cross Entropy Loss curves for Learning Rate = 0.001

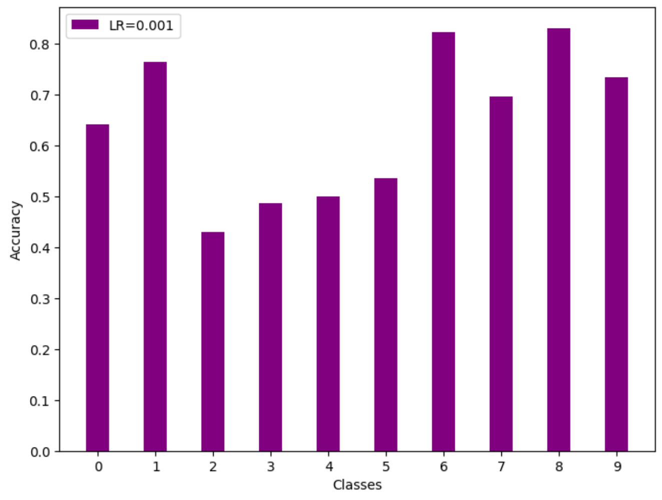


Figure 5: Bar Graph showing Class wise Accuracy for Learning Rate = 0.001

Overall Validation Accuracy
0.72

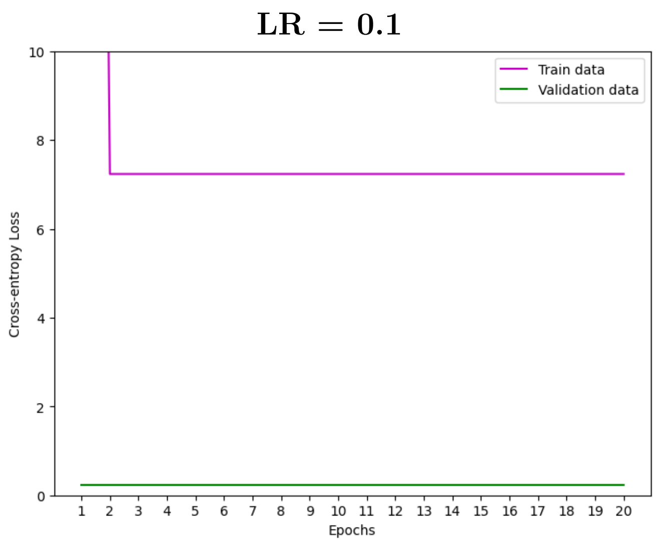


Figure 6: Cross Entropy Loss curves for Learning Rate = 0.1

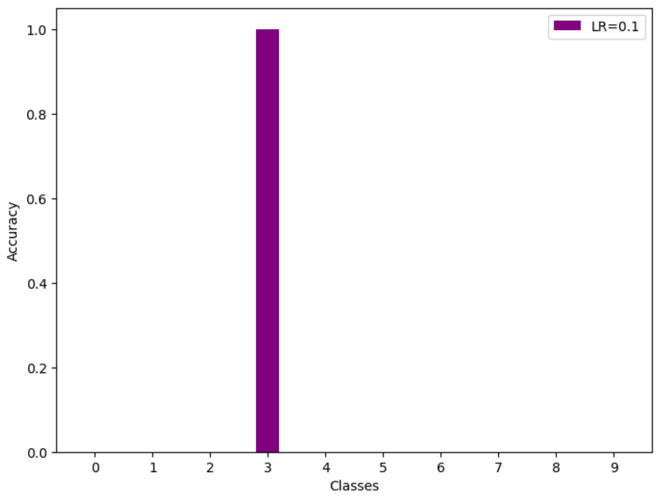


Figure 7: Bar Graph showing Class wise Accuracy for Learning Rate = 0.1

Overall Validation Accuracy
0.10

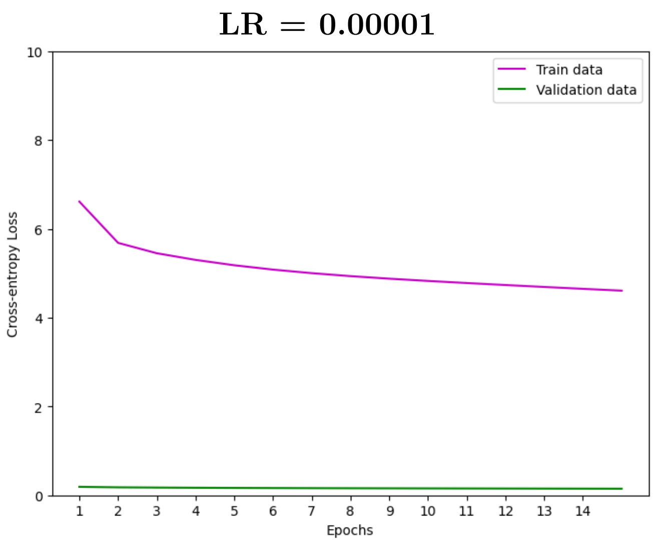


Figure 8: Cross Entropy Loss curves for Learning Rate = 0.00001

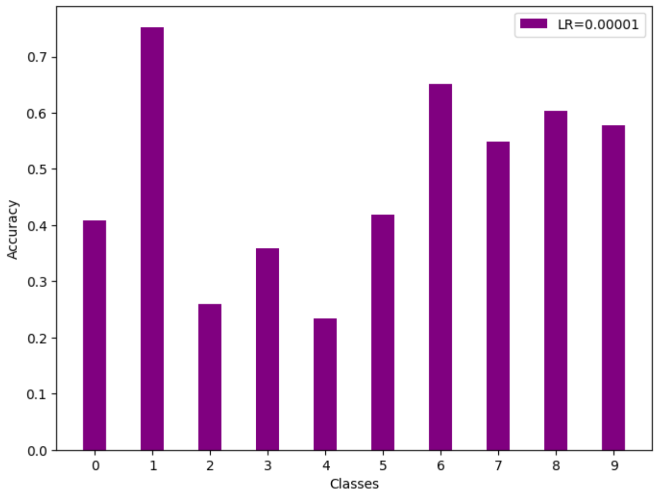


Figure 9: Bar Graph showing Class wise Accuracy for Learning Rate = 0.00001

Overall Validation Accuracy
0.4850

Comparing with different LRs (0.1, 0.001, 0.00001):

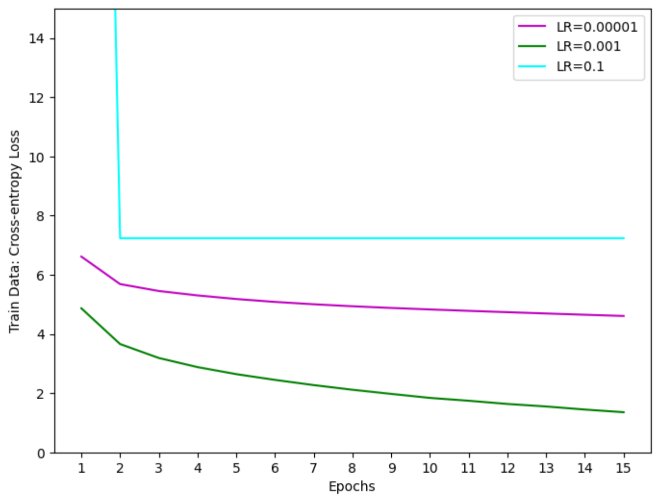


Figure 10: Cross Entropy Loss curves on Train data for 3 learning rates (0.00001, 0.001, 0.1)

Analysis:

- The training loss decreases for lr = 0.00001 and 0.001, but remains stagnant after reducing to a certain value for lr = 0.1. This may be because going by a higher step along the gradient (as lr = 0.1 seems to be higher here) results in some local optima, and it gets stuck there.
- The training loss on lr 0.00001 decreases at a very slower rate than the loss for lr = 0.01, as visible from fig. 10
- The higher learning rates result in local optimum, which corresponds to the prediction which predicts all inputs to be the label that is maximum present in the set. This can be seen from classwise accuracy. But for smaller rates, it is distributed among the classes.

2.2 LR Scheduling

Exponential Leaning Rate Scheduler - Loss curves:

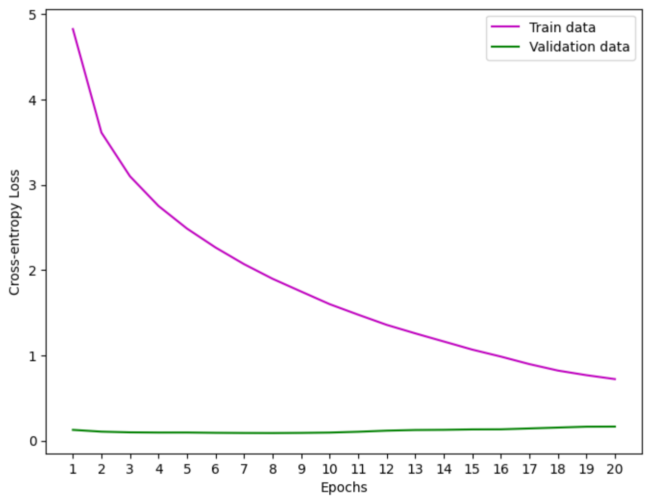


Figure 11: Cross Entropy Loss curves for Learning Rate Scheduler with Exponential LR Scheduler, initial LR = 0.001, gamma = 0.99

Overall Validation Accuracy
0.6589999794960022

Class-wise Accuracy:

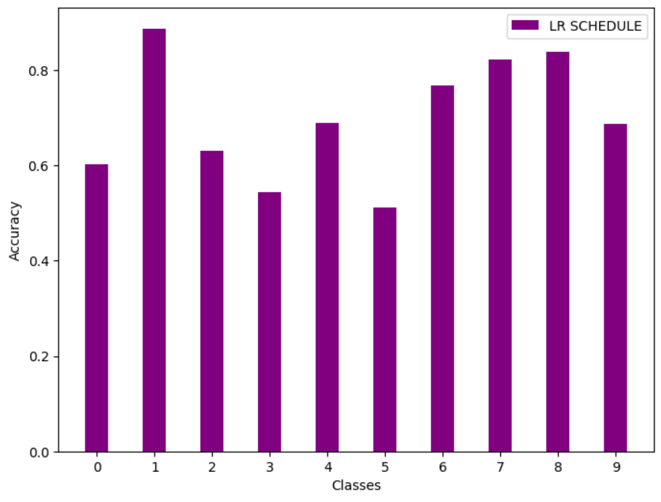


Figure 12: Classwise Accuracy Graph with Exponential LR Scheduler

Comparision with Fixed LR:

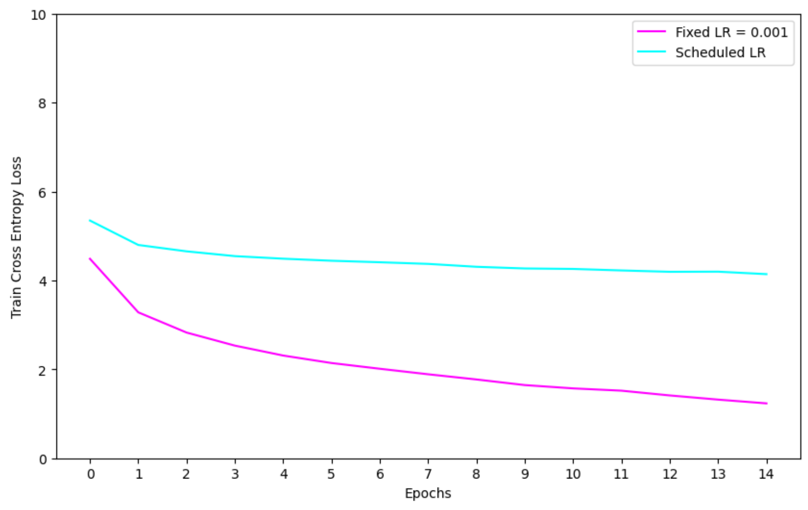


Figure 13: Cross Entropy Loss curves on Train data

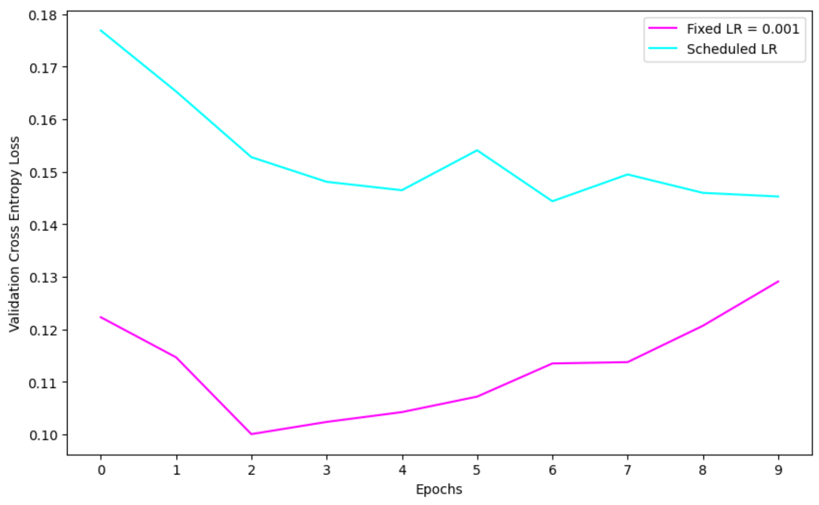


Figure 14: Cross Entropy Loss curves on Validation data

Analysis:

- As LR scheduler uses a decay function to reduce the learning rate gradually, (in this case I have chosen the exponential decay function with gamma = 0.99), we can see from the graph of Training loss, that fixed lr results in fast decrease on the loss, whereas scheduled lr decreases gradually, and decreases loss function by less amount as it near the minima.
- In case of validation loss, the loss decreases very gradually and thus results in preventing overfitting. As near the minima, lr decreases so overfitting cannot

occus, i.e. it doesnt jump to some local optimum. So it learns the data well, but fits poor.

2.2.1 Number of Training Epochs

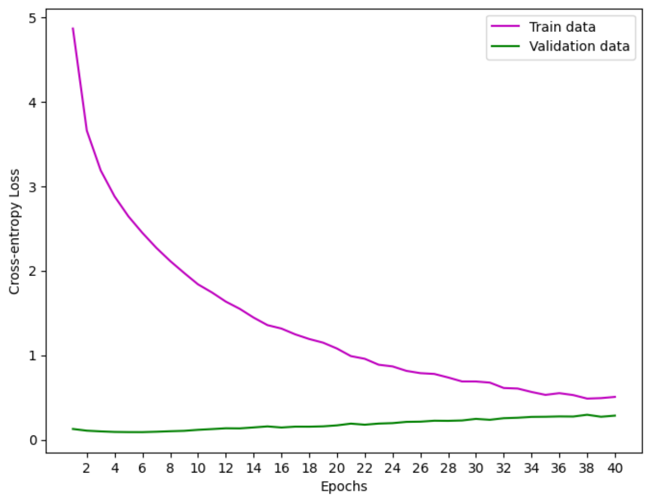


Figure 15: Cross Entropy Loss curves for Learning Rate = 0.001, done for 40 epochs

Overall Validation Accuracy
0.7139999866485596

Analysis:

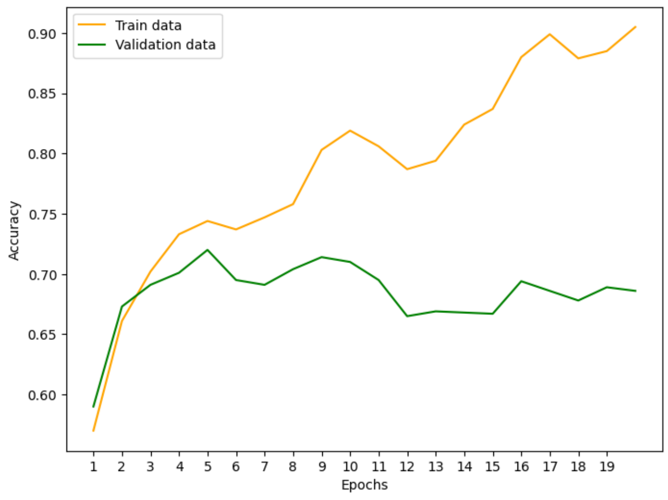


Figure 16: Accuracy curves for Learning Rate = 0.001, done for 10 epochs

Analysis: Does increasing training epochs always help?

- Increasing the epochs does not help always, because the model reaches a saturation (optimal point), after which the model starts overfitting.
- As observed from fig. 15 and 16, training loss decreases further as we go on increasing epochs, but this doesnt do any good to the model, as it memorises the training data, thus increasing validation loss. Validation loss decreases upto a certain minima, and then starts increasing due to overfitting.
- Similar trend can be seen in the accuracy, as training accuracy keeps on increasing with increasing epochs, but validation accuracy decreases after a certain number of epochs (6 in this case).
- However if we set an optimal learning rate, such as scheduled in a way that decreases its value near optimal values, then increased number of epochs may increase accuracy, as the model will learn the complexities of model intricately.

2.2.2 Batch size

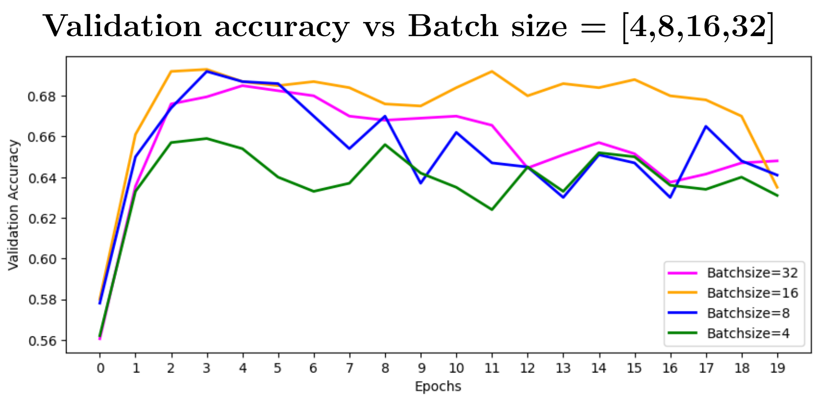


Figure 17: Validation accuracy curves vs epochs for differnet batch sizes

Overall Validation Accuracy

Batch size	Accuracy
32	0.6890
16	0.6790
8	0.6170
4	0.6530

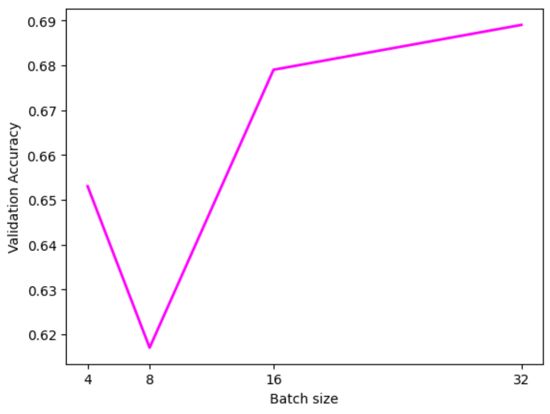


Figure 18: Final validation accuracy vs Batch size

Analysis:

- Decreasing batch size sometimes increases the accuracy, as the model takes numerous smaller steps towards the gradient and learns the data well. This can be visible from the accuracy plot (Figure 17), where the accuracy of batch size 16 is constantly greater than batch size = 32
- But due to the more number of classes = 10, small batch size doesnt help the model to learn the complexity of the distribution, and thus batch size 4 and 8 doesnt work better.

2.2.3 Effect of SGD Optimizer:

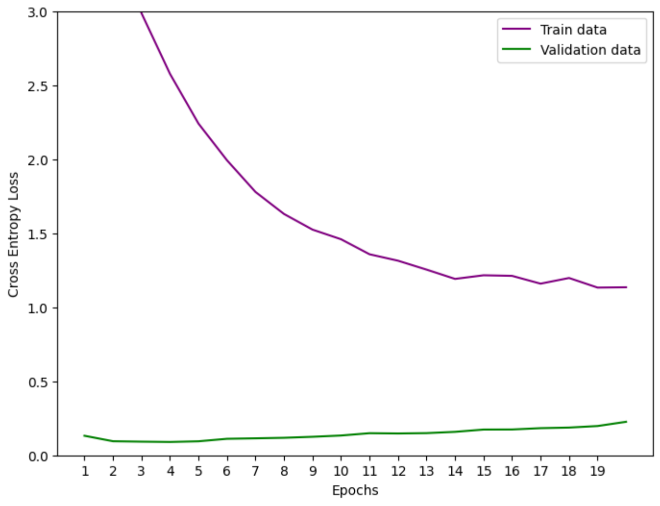


Figure 19: Loss curves for Learning Rate = 0.001

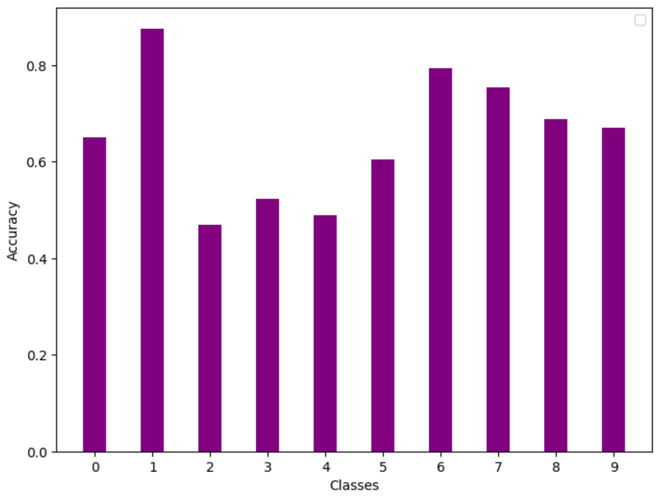


Figure 20: Classswise accuracy for Learning Rate = 0.001

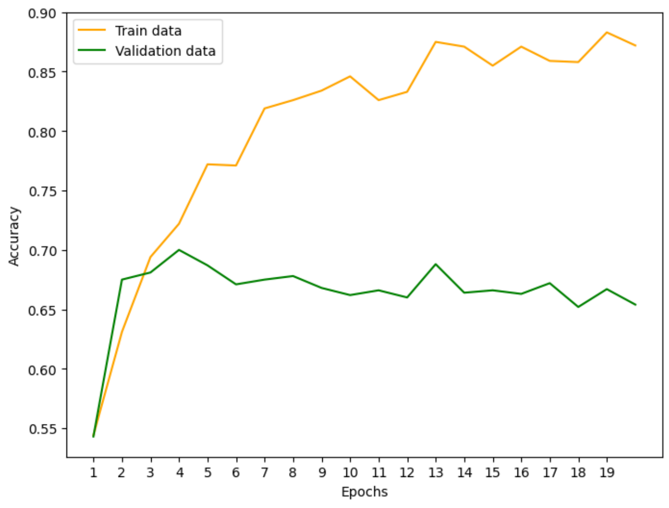


Figure 21: Accuracy for SDG Optimizer

Comparision with Adam optimizer:

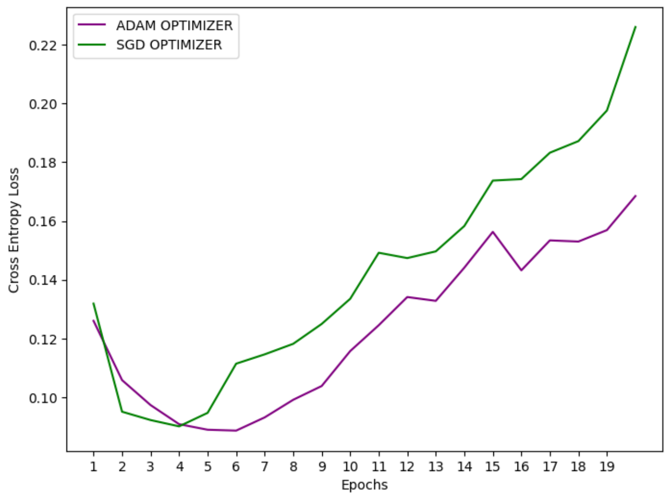


Figure 22: Loss curves for Validation Data

Analysis:

- Adam Optimizer uses many other factors than just gradient (as in Stochastic Gradient Descent).
- This factors include momentum and changing learning rate. Changing learning rate adjusts the learning rate in a way as to move adequate step towards the optimum, according to the gradient value. This results in faster convergence. This is obsevred from the graphs (Fig 21 and 22)
- It also uses momentum, which helps in the smoother convergence of loss functions.

2.3 Effect of Loss function:

2.3.1 KL Divergence Loss:

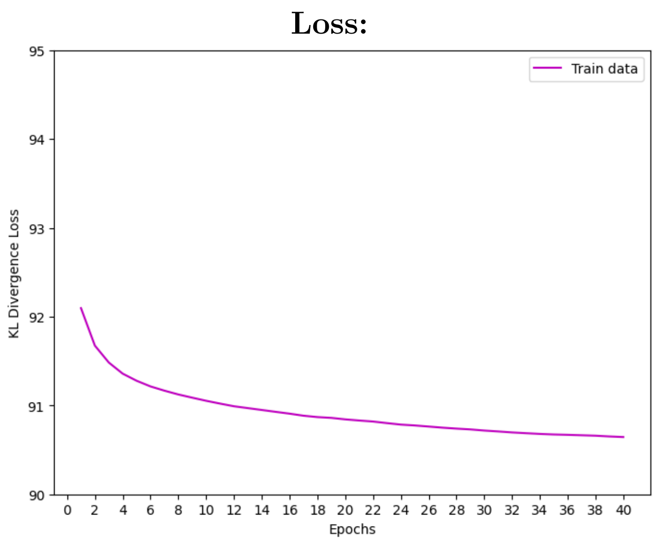


Figure 23: K-Divergence Loss curves (Train data) for Learning Rate = 0.001

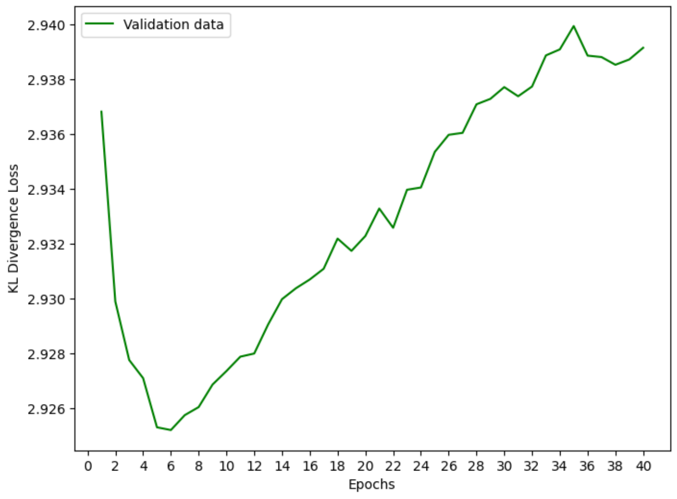


Figure 24: K-Divergence Loss curves (Validation data) for Learning Rate = 0.001

Classwise accuracy:

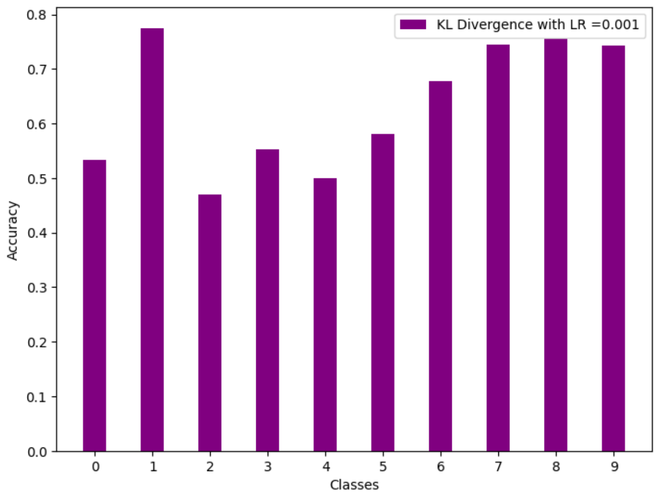


Figure 25: Class wise accuracy on Learning Rate = 0.001

Analysis:

- The KDivergence loss is less stable in terms of loss gradients as can be seen from the graph.
- But KDivergence loss has a regulariser component too, thus it prevents overfitting. The model will be penalised if it deviates too far from the real distribution, because this is how KDivergence loss measures it.

2.4 Without Data Augmentation:

Loss curves and Classwise Accuracy:

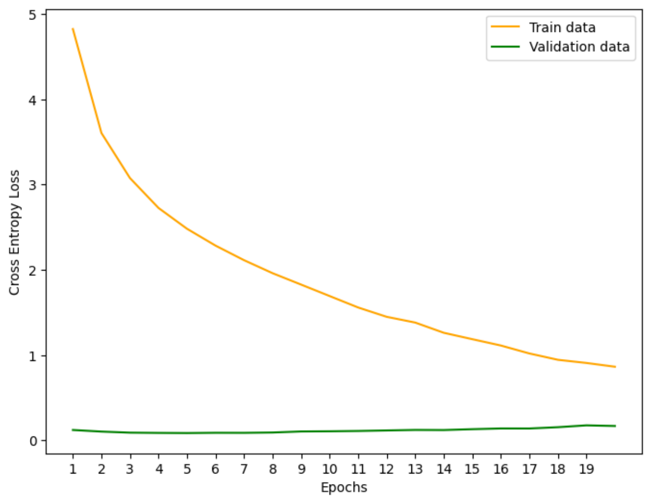


Figure 26: Cross Entropy Loss curves for Learning Rate = 0.001, without Data augmentation

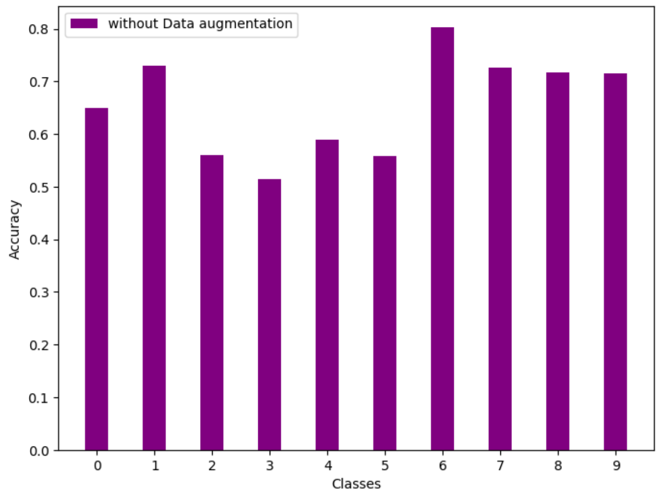


Figure 27: Classwise accuracy graph for Learning Rate = 0.001, without Data augmentation

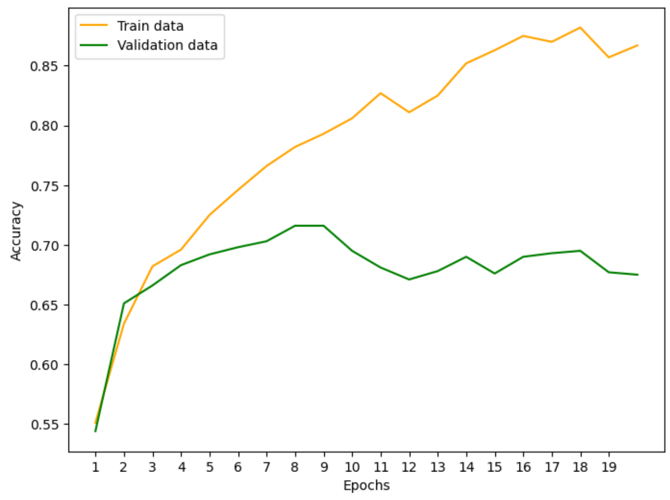


Figure 28: Accuracy curves for Learning Rate = 0.001, without Data augmentation

Comparision with Data Augmentation = ON:

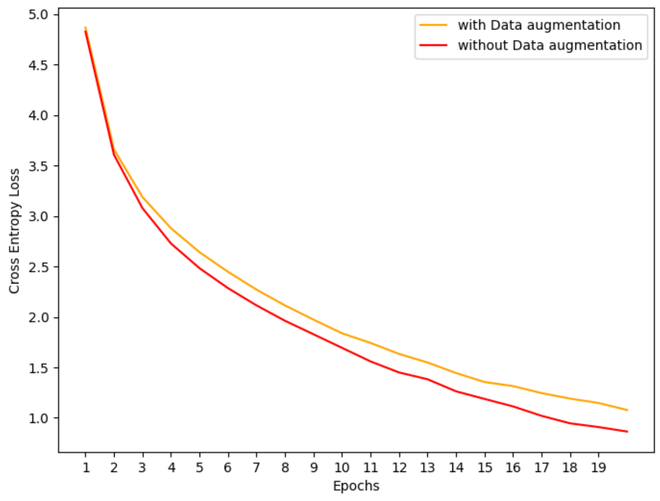


Figure 29: Cross Entropy Loss curves for Learning Rate = 0.001 on Train data, for both types of Data

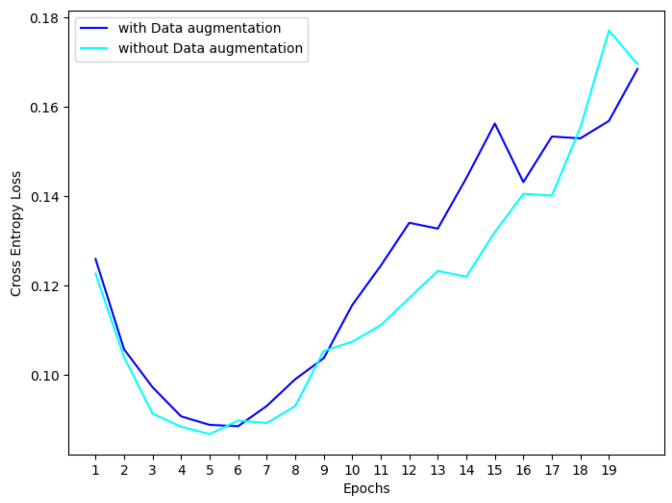


Figure 30: Cross Entropy Loss curves for Learning Rate = 0.001 on Validation data, for both types of data

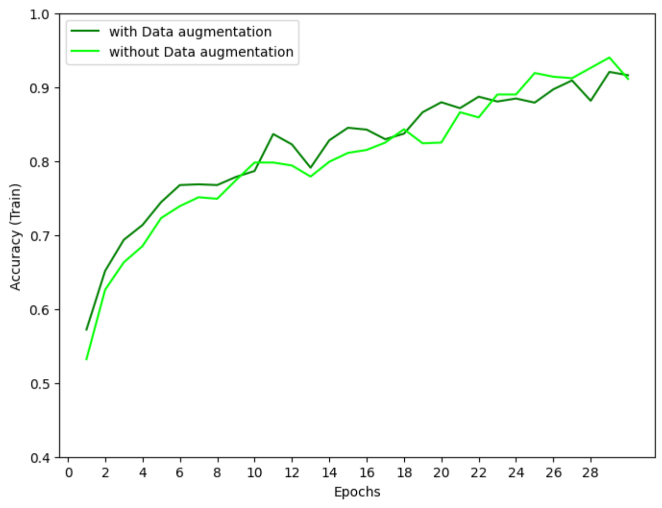


Figure 31: Accuracy curves for Learning Rate = 0.001 on Train data, for both types of Data

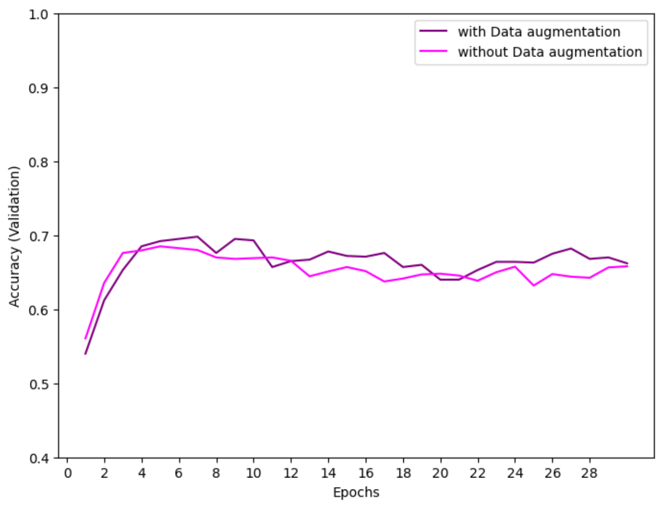


Figure 32: Accuracy curves for Learning Rate = 0.001 on Validation data, for both types of data

Analysis:

- Data augmentation technique increases the variability of the model, and thus increases the generalisability of the CNN.
- Due to augmentation, effects of certain biases like position or background, or direction are removed and thus the model learns the classification, rather than just memorising the limited training set.
- Thus, data augmentation increases the validation accuracy, as visible from Fig. 32.
- But as the complexity(variability) of the model increases on augmenting data, so the convergence of training loss delays. Thus, without data augmentation the train loss converges faster.
- The classwise accuracy is more even in case of augmentation, because of reducing class imbalance.

3 Improving the CNN model

To improve the accuracy of test set on CNN, I tried to include the following things in the model:

1. **Increasing the number of out-channels on Convolutional layers**
   The number of out channels is associated to how many feature maps would be generated after the convolutional layer computation. More number of feature maps would lead to better learning of the feature (more number of nodes), even intricate

and complex features would be learnt due to this, which will increase the overall accuracy.

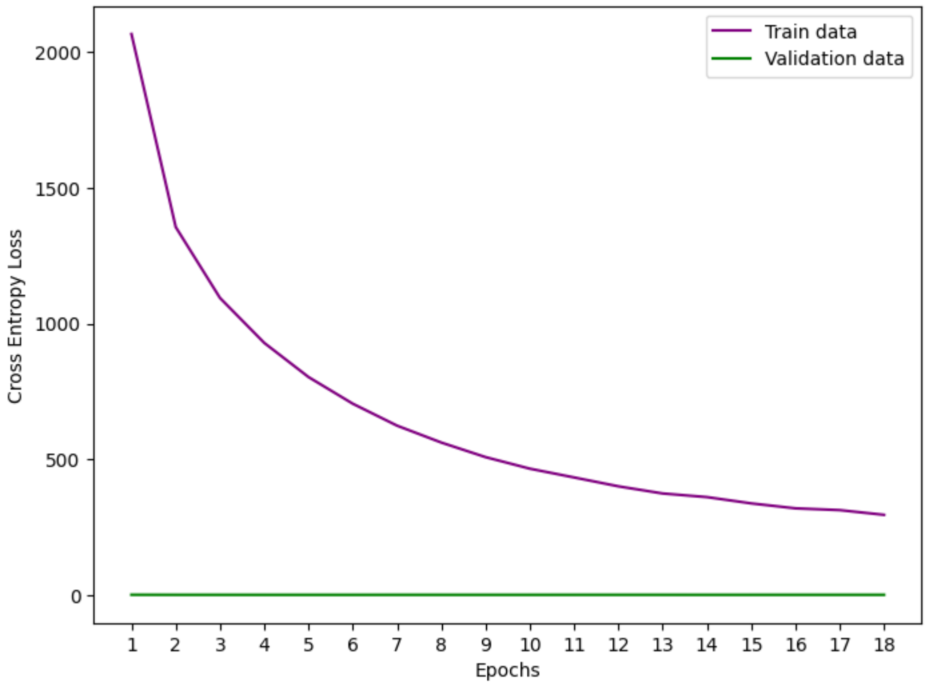
2. **Adding Batch Normalisation**
   This handles the small variations in the input distributions, and normalised the inputs which prevent overfitting and increases generalisability of the model. It prevents the model being overfitted on the input that is given for trainsize, or the model learning features belonging to some specific pattern (like position), to make the model more genralisable and to learn the general features belonging to all the classes.

3. **Adding Weight Decay in the optimizer for *Regularisation***
   This will decrease the overall effect of larger weights and thus prevent overfitting. This will be a commonly used technique for regularisation, which helps in preventing overfitting due to large weights.

4. **Adding Dropouts at some layers** Fully connected layers having large number of weights are more probe to overfitting. So we need dropouts in these layers, I have set the probability of a neuron being dropped as 0.5. I have also included dropout post maxpooling layer with 1/4th probability of being dropped out. This decreases overfitting probability of the model. As we have used higher number of channels in the convolutional layers, so adding dropout layer is necessary to prevent overfitting.

Final Overall Accuracy on Test set: 0.7935
Final Overall Accuracy on Train set: 0.9565

Loss curves:



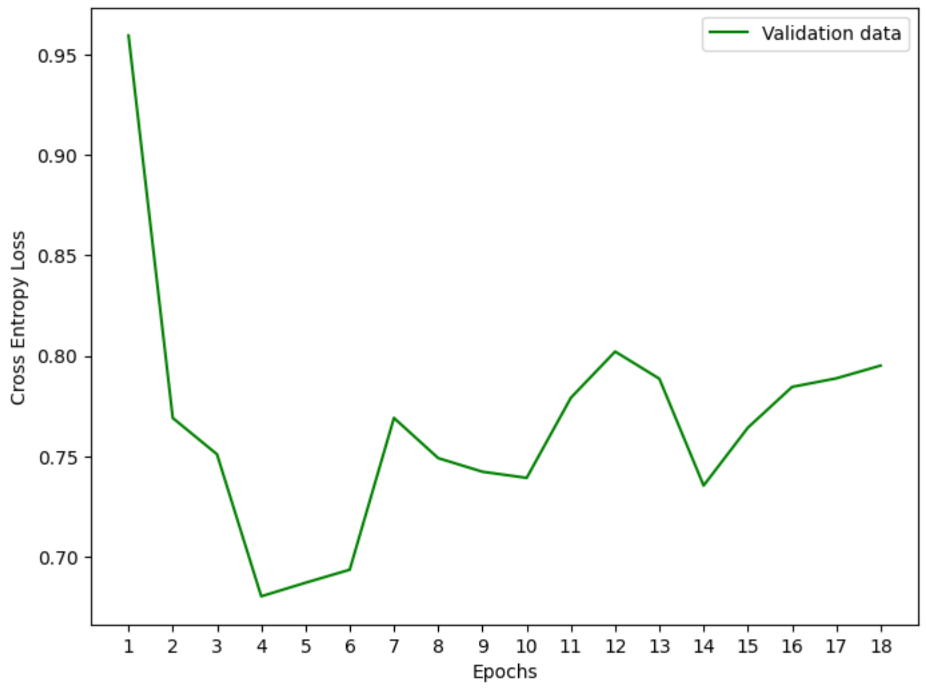


Figure 33: Cross Entropy Loss curves for Learning Rate = 0.001

Accuracy graphs:

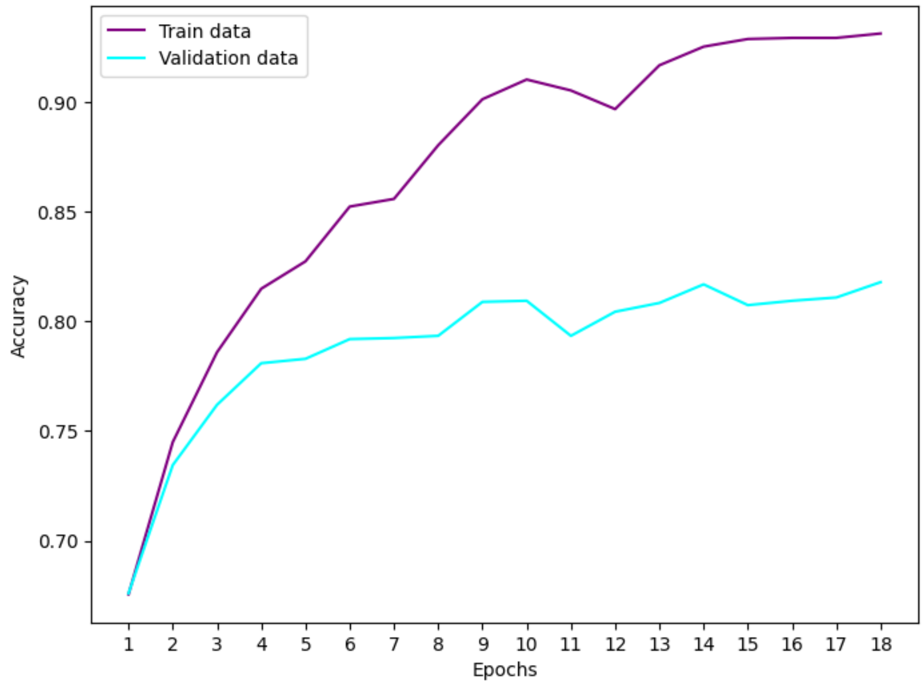


Figure 34: Accuracy curve for Learning Rate = 0.001

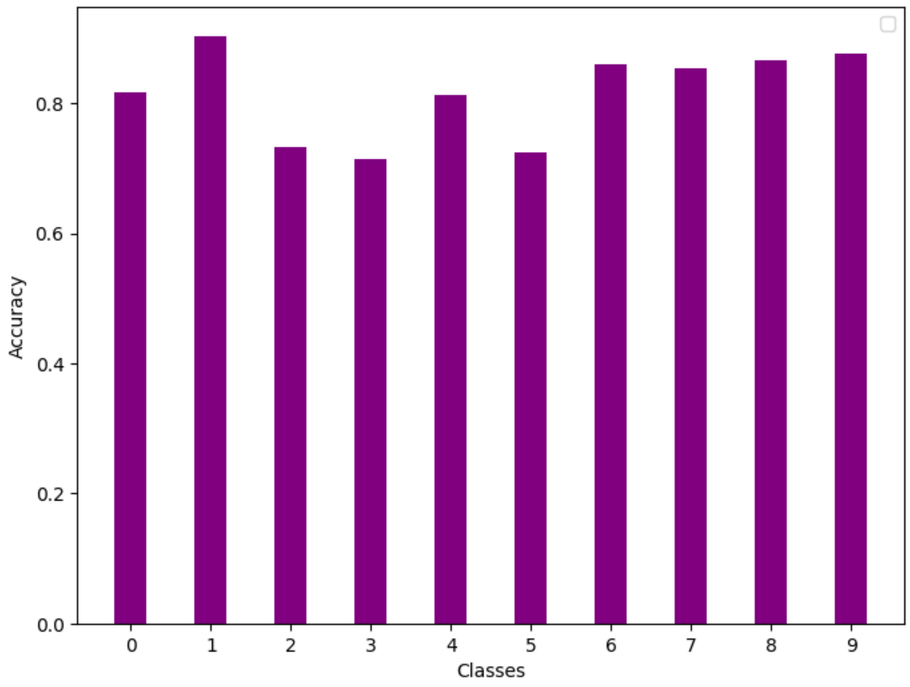


Figure 35: Classwise accuracy for Learning Rate = 0.001

4 References and Citations: