Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

LAB 03 : OPTIMIZING AND TRAINING DEEP NEUTRAL NETWORKS WITH KERAS

Presented By : FELLAH Abdelnour,Abbou Riyadh.

Date: March 13, 2024

1 Introduction

Neural networks have become a cornerstone in the field of machine learning, offering powerful tools for solving complex problems. However, training these networks can be challenging, especially when dealing with issues such as overfitting. To address this challenge, various techniques have been developed, including the use of different optimizers and overfitting avoidance techniques such as dropout layers, batch normalization, and L2 regularization.

In this lab, we aim to explore the impact of different optimizers and overfitting avoidance techniques on the training and optimization of neural networks.

2 Part One : Training and MLP with keras

2.1 Discussing The training time of the models

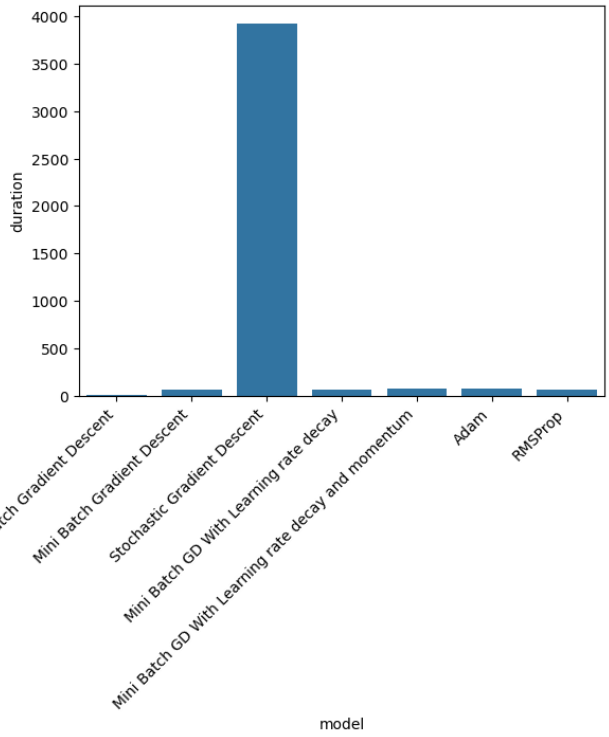

Figure 1: The histogram of the training time for each model

Neutral networks usually takes advantage of a type of machine instructions called SIMD (single instruction multiple data), available on GPUs and even modern CPUs,which are a type of instructions that apply the same operation to multiple data in parallel,making tasks like adding tow arrays or matrix multiplications very fast.

This is way it is always preferable to process data in batches,the larger the batch the more we take advantage of the parallel nature of SIMD instructions,this is why Stochastic gradient descent takes the longer to train because we process on data point at a time we take no advantage of these parallel instructions and Batch gradient descent takes the less to train because we by processing the whole train set at a time we are taking full advantage of the parallelism.

2.2 DISCUSSING THE PERFORMANCES OF THE DIFFERENT MODELS

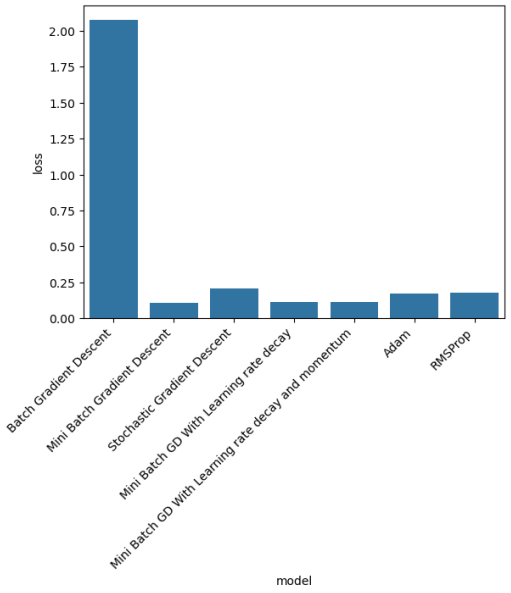

Figure 2: The loss histogram for different models on the validation set

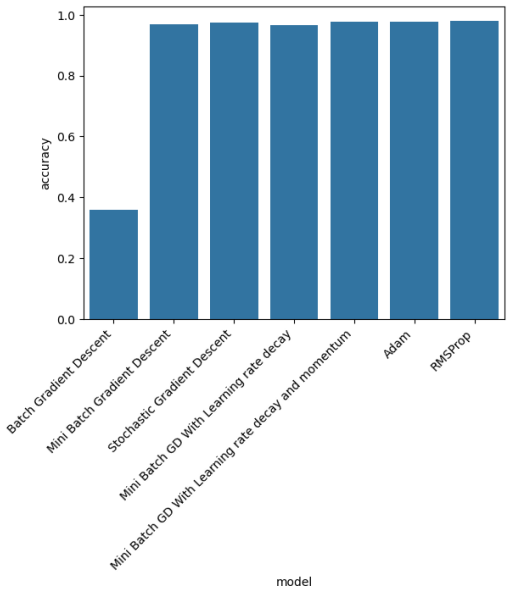

Figure 3: The accuracy histogram for different models on the validation set

We notice overall that the Batch Gradient descent model performs the worst because it only performed less number of iterations,since the batch gradient descent processes the entire training set at once then $num_iterations = num_epochs$,mean while a Mini Batch Gradient descent with batch size equals to 32 and training set that contains 48,000 data points for example performs $num_iterations = num\_epochs * (training\_set\_size/batch\_size) = 50 * (48000/32) = 75000$ iterations.

this doesn't mean we always have to go for smaller batches as it slows the training and introduces more noise which explains why the Stochastic Gradient Descent had relatively higher loss (ignoring the Batch Gradient Descent).

2.3 COMPARING THE LEARNING GRAPHS FOR THE DIFFERENT MODELS

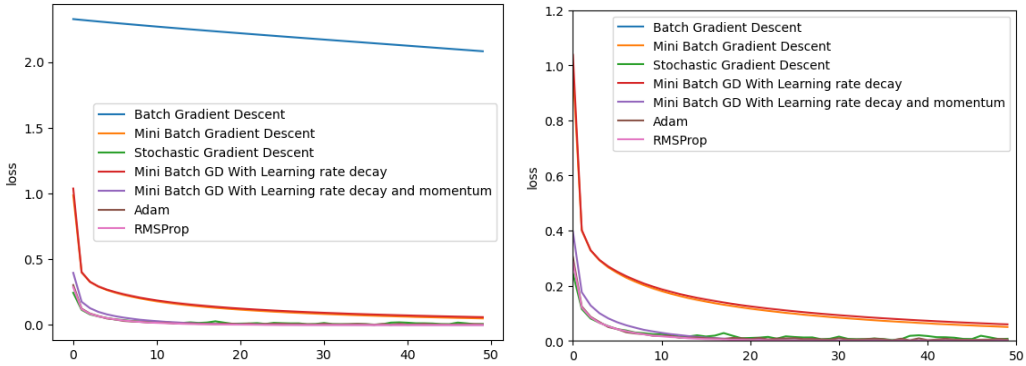

Figure 4: The training loss for the different models

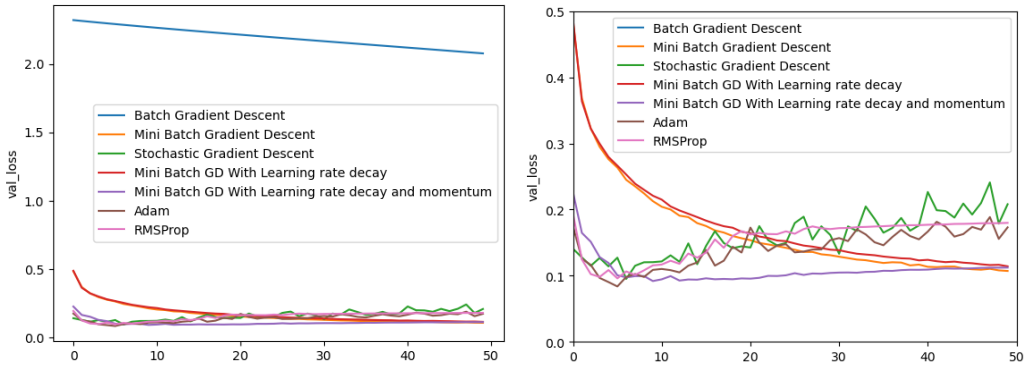

Figure 5: The validation loss for the different models

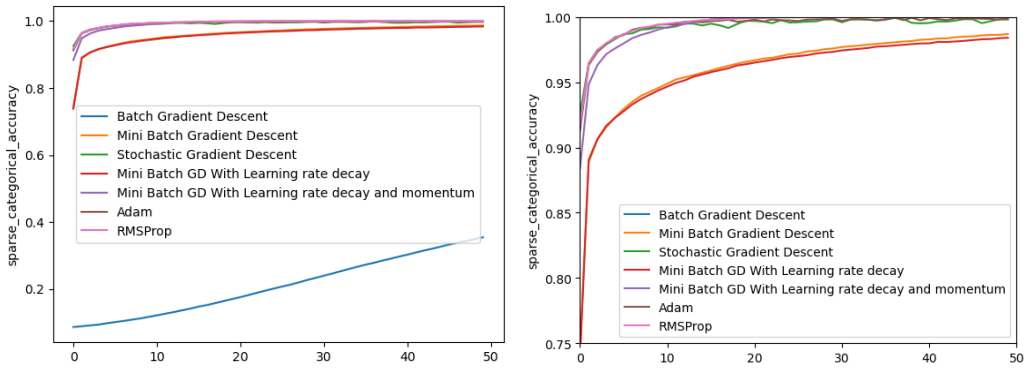

Figure 6: The training accuracy for the different models

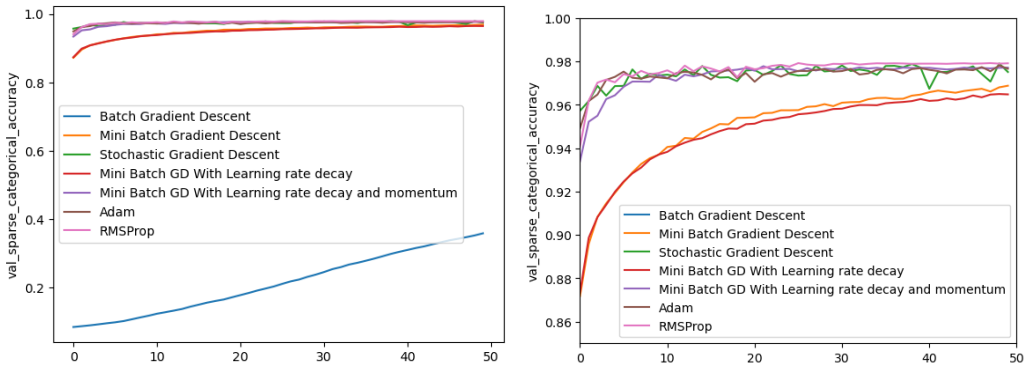

Figure 7: The validation accuracy for the different models

We notice that Adam,RMSProp and Mini Batch Gradient Descent with momentum converges faster than Mini Batch Gradient Descent,because they combine the current gradient with the previous gradients to overcome oscillations and noisy gradients thus preventing taking large steps in the wrong direction.

We also notice that Adam,RMSProp loss starts becoming less stable towards the last epochs due to the fact that the gradients becomes smaller at that point which can introduce instability a potential solution maybe reducing the learning rate.

2.4 THE CONVERGENCE GRAPHS FOR EACH MODEL

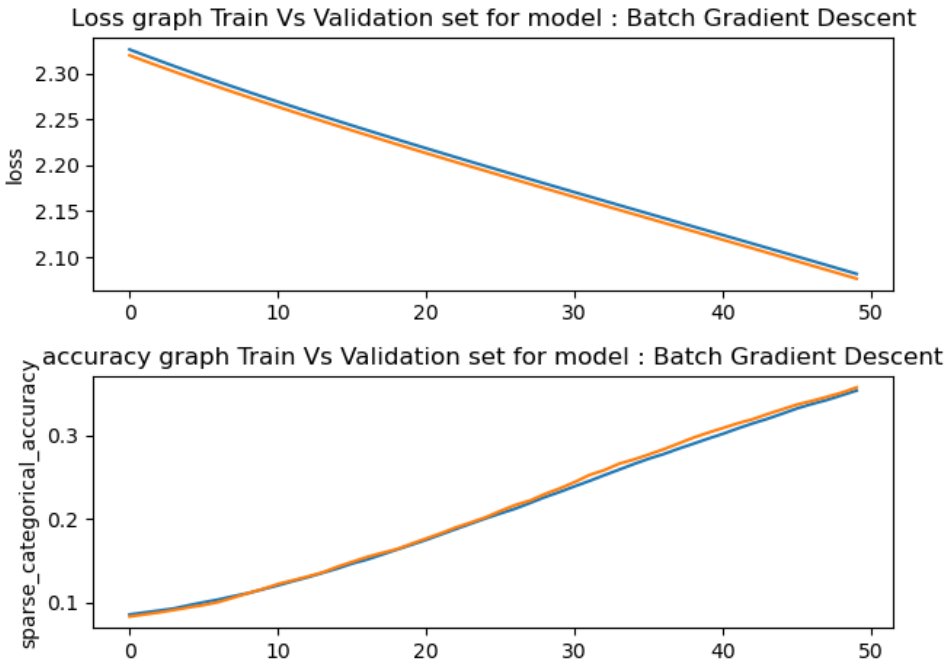

Figure 8: The convergence graph for the model : Batch Gradient Descent

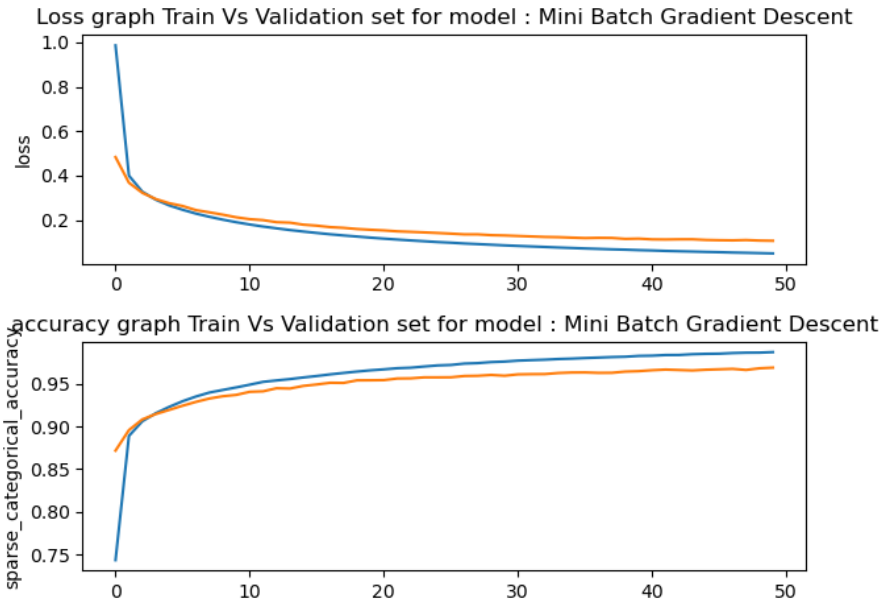

Figure 9: The convergence graph for the model : Mini Batch Gradient Descent

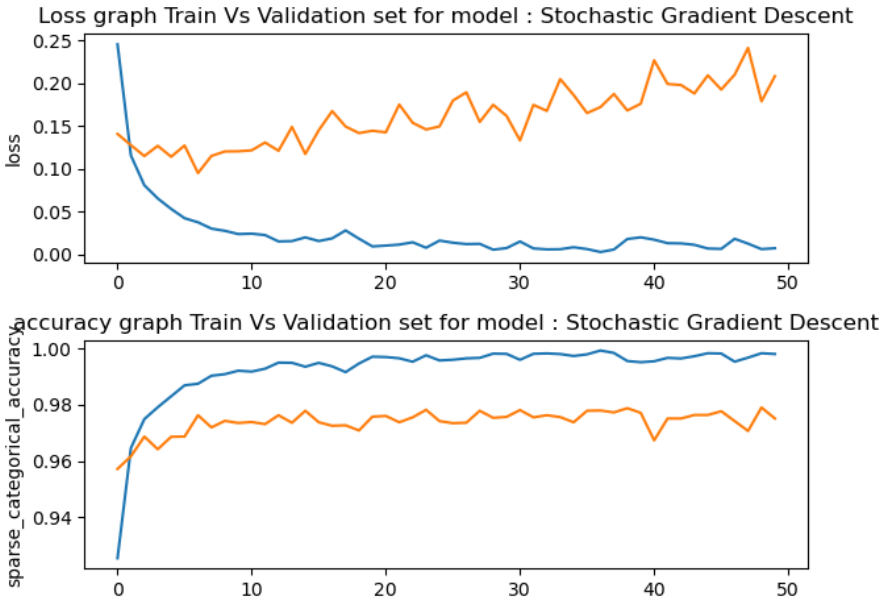

Figure 10: The convergence graph for the model : Stochastic Gradient Descent

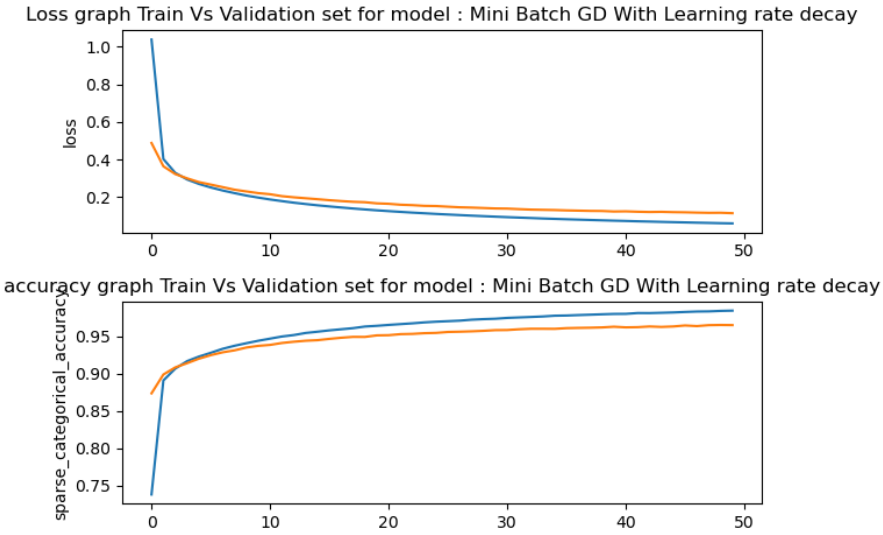

Figure 11: The convergence graph for the model : Mini Batch Gradient Descent With Learning rate decay

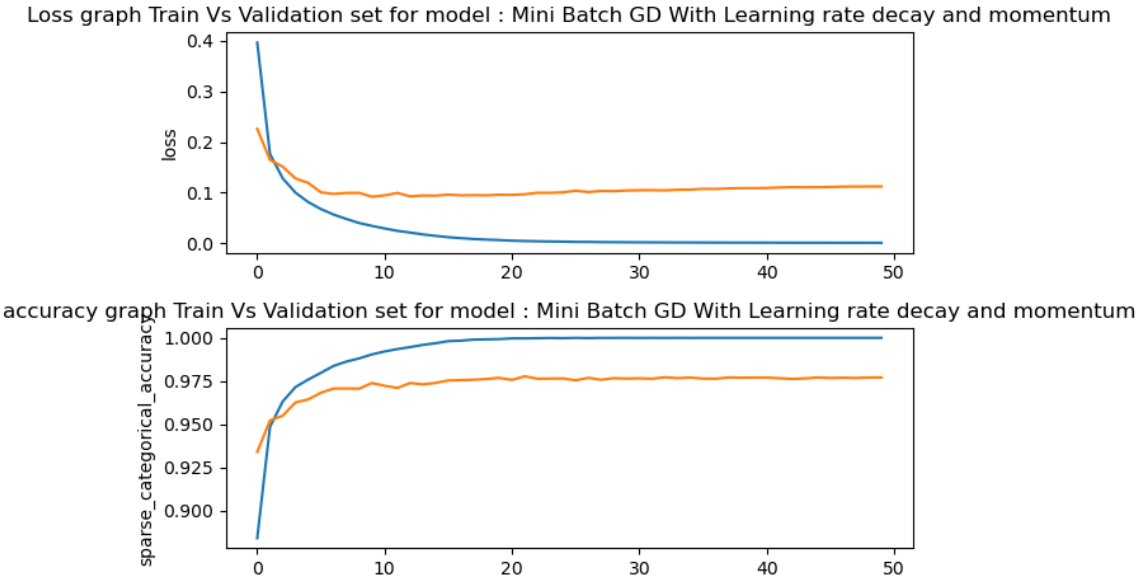

Figure 12: The convergence graph for the model : Mini Batch Gradient Descent With Learning rate decay and momentum

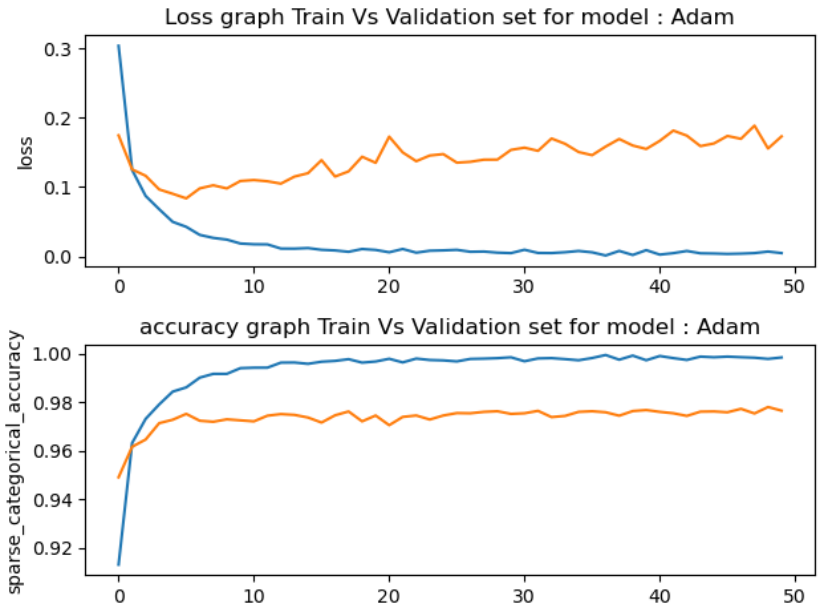

Figure 13: The convergence graph for the model : Adam

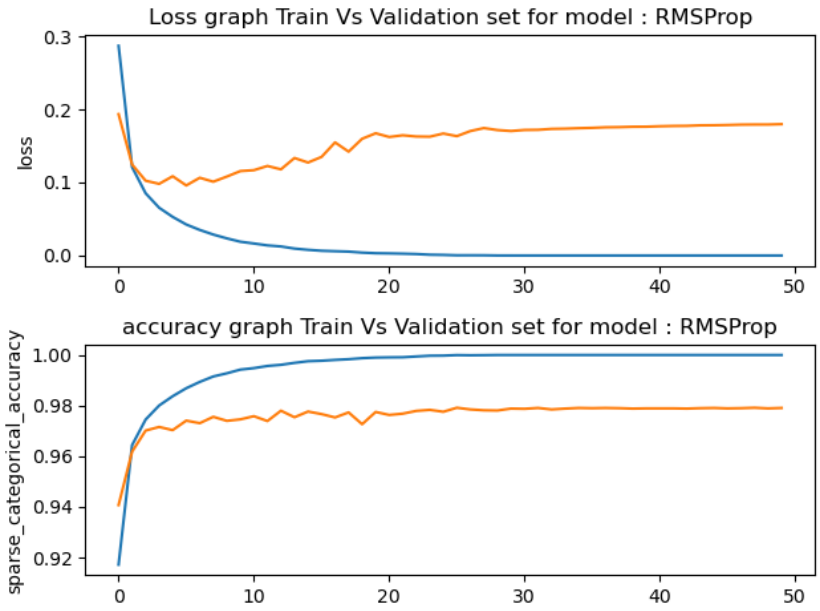

Figure 14: The convergence graph for the model : RMSProp

Even though Adam and RMSProp converges faster they can lead to overfitting if trained for longer than needed with a large value of the learning rate,the stochastic gradient descent however performs worst on the validation set due to the noise introduced during training (the bigger the batch size the noisier the training).

3 Part Tow : Optimizing hyperparameters

3.1 Discussing The training time of the models

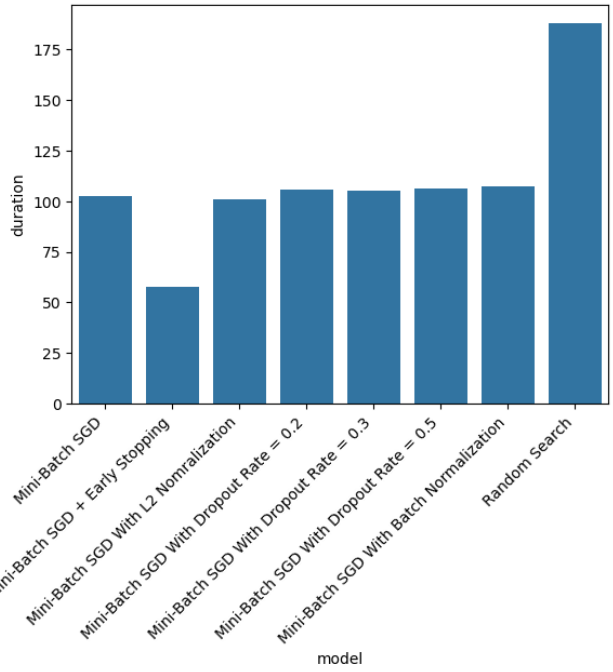

Figure 15: The histogram of the training time for each model

The training time for all the models is approximately the same except for the model that uses early stopping which has a significantly smaller training time because the training stopped at epoch 28 instead of 50,and the random search have the highest training time among all the models because the best batch size value that was found through random search is 16 which is less than the batch size that we used to train the other models and we know that using a smaller batch size leads to a longer training time.

3.2 Discussing The performances of the different models

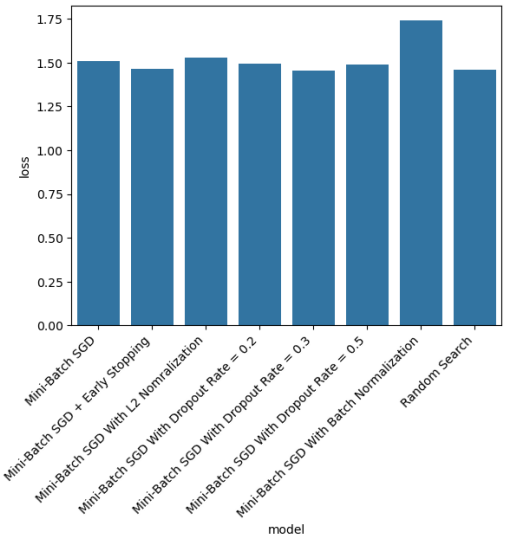

Figure 16: The loss histogram for different models on the validation set

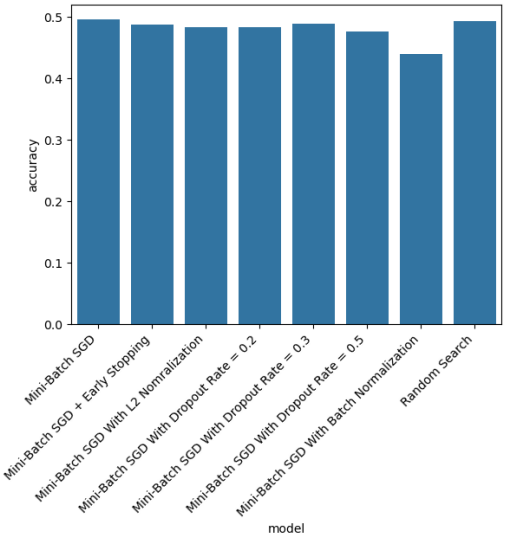

Figure 17: The accuracy histogram for different models on the validation set

The performance of the models is very similar, with the model trained using Mini-Batch GD and Mini-Batch GD with a dropout rate of 0.3. Of course, the model trained on the best hyperparameters found through random search is slightly better. Additionally, we can observe that the model trained with batch normalization on the second hidden layer has the worst accuracy and the highest loss. One reason for this could be the small batch size, which may lead to inaccurate estimations of the statistical parameters μ and σ, as well as the learned parameters α and β.

3.3 Comparing the learning graphs for the different models

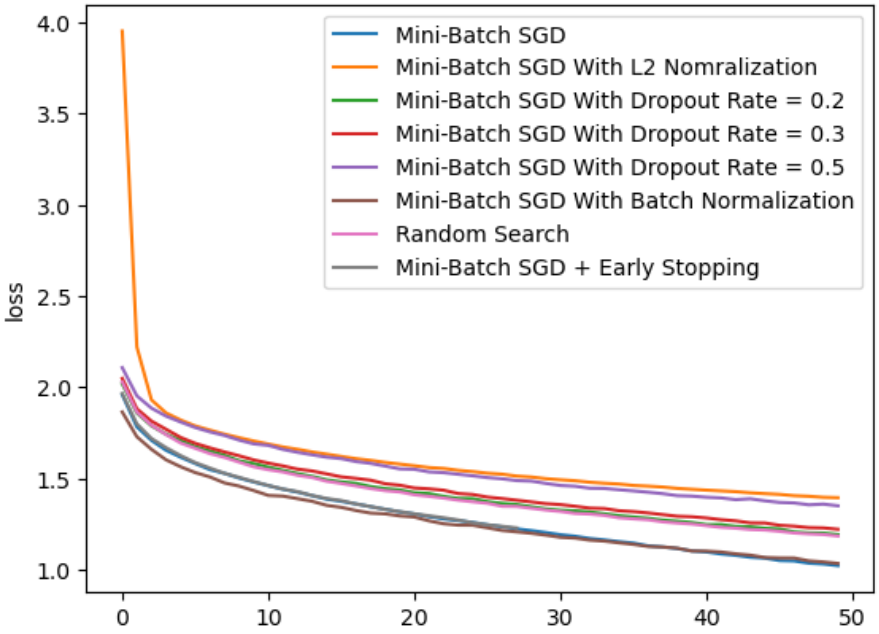

Figure 18: Caption

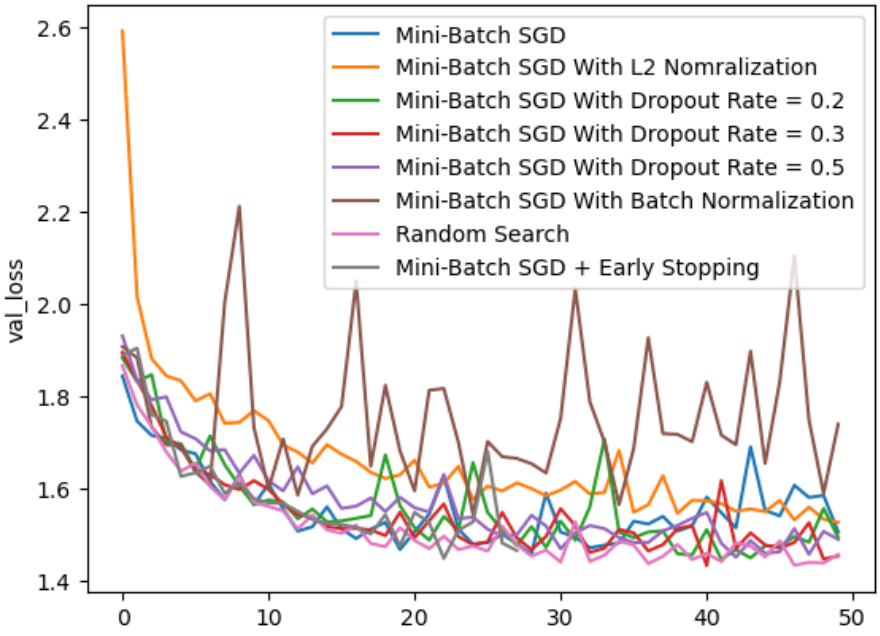

Figure 19: Caption

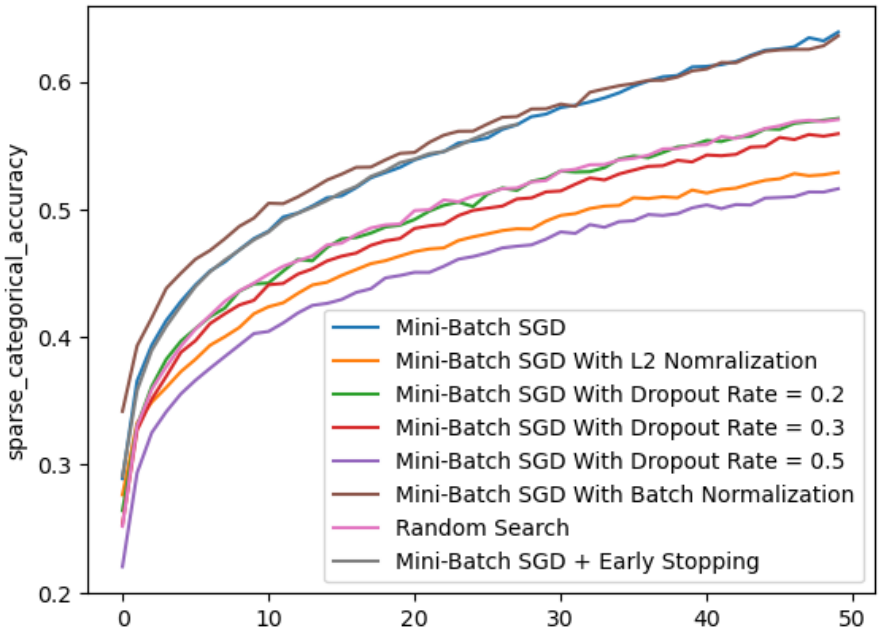

Figure 20: Caption

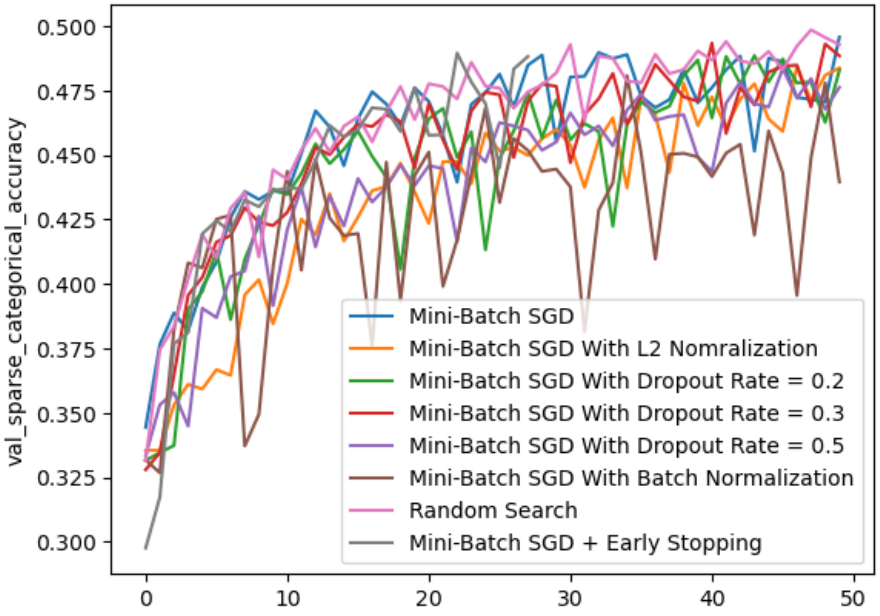

Figure 21: Caption

we notice that training graphs don't show much oscillations and the loss is deceasing steadily,with the model trained with L2 regularization taking a few more epochs for it loss to be in comparable range with the other model,it's also the model with highest loss which maybe due to a large regularization parameter for this particular problem (which may cause the model to even underfit),the model trained with batch normalization have the lowest training loss.

but the validation graphs don't tell the same story,there is so much oscillations specially in the case of the mode trained with batch normalization which maybe due to the batch being small and the network not being very deep as it is recommended to use batch normalization in the case of very deep neutral networks.

3.4 THE CONVERGENCE GRAPHS FOR EACH MODEL

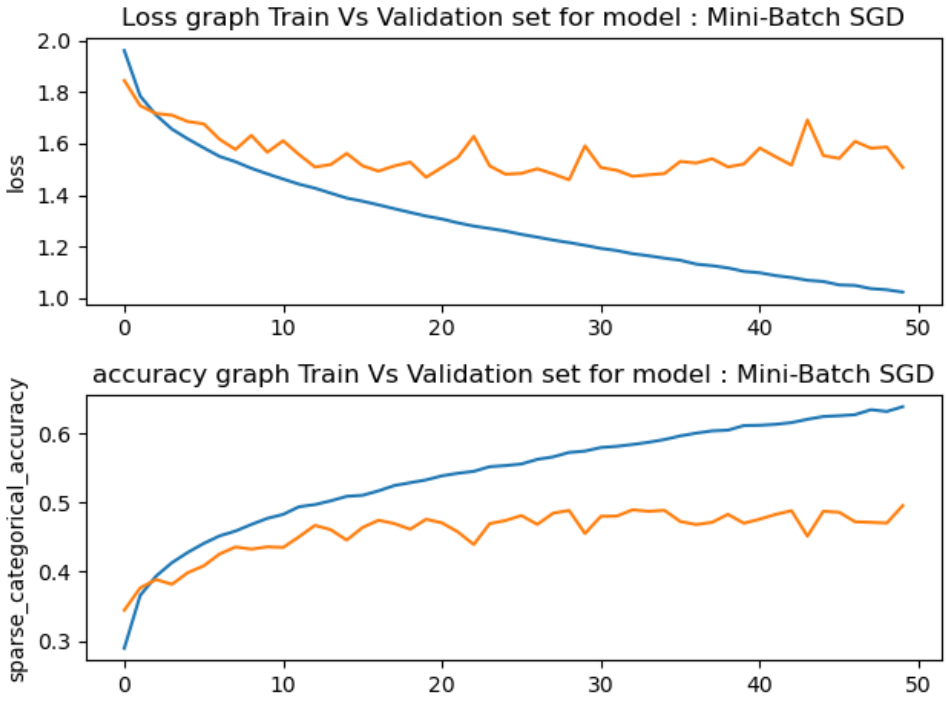

Figure 22: The convergence graph for the model : Mini-Batch GD

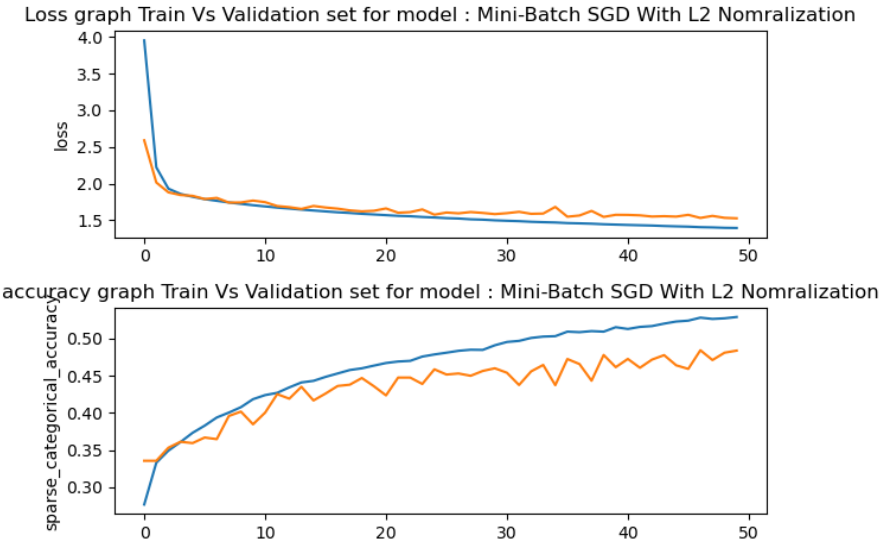

Figure 23: The convergence graph for the model : Mini Batch GD + L2 regularization

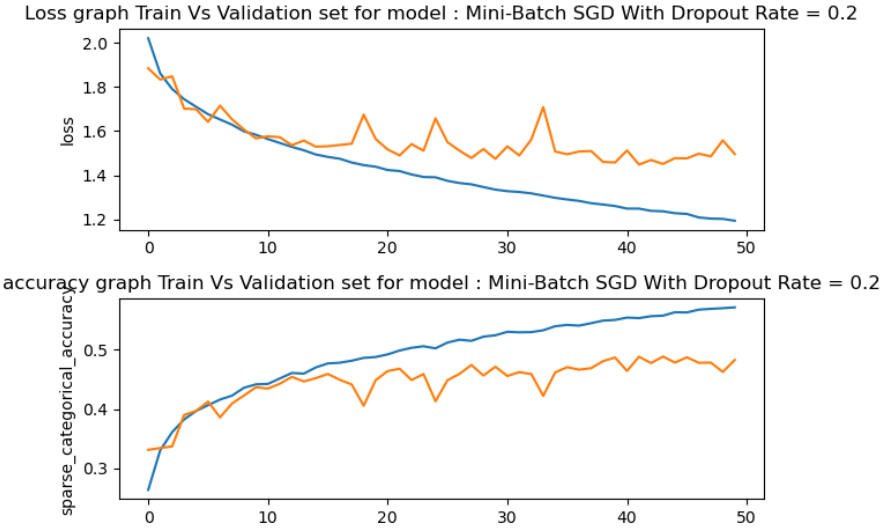

Figure 24: The convergence graph for the model : Mini-Batch GD with dropout rate = 0.2

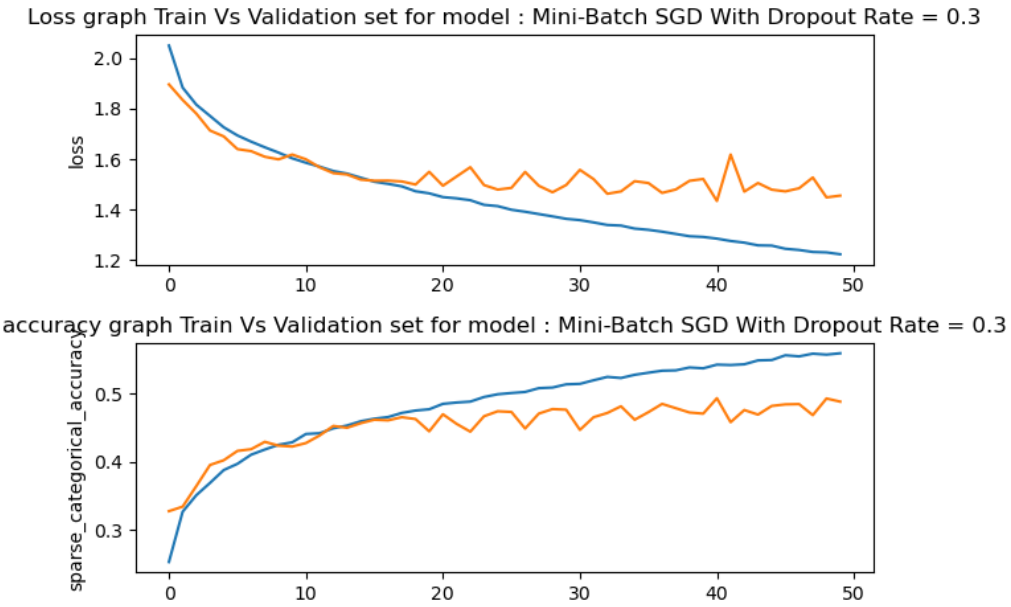

Figure 25: The convergence graph for the model : Mini-Batch GD with dropout rate = 0.3

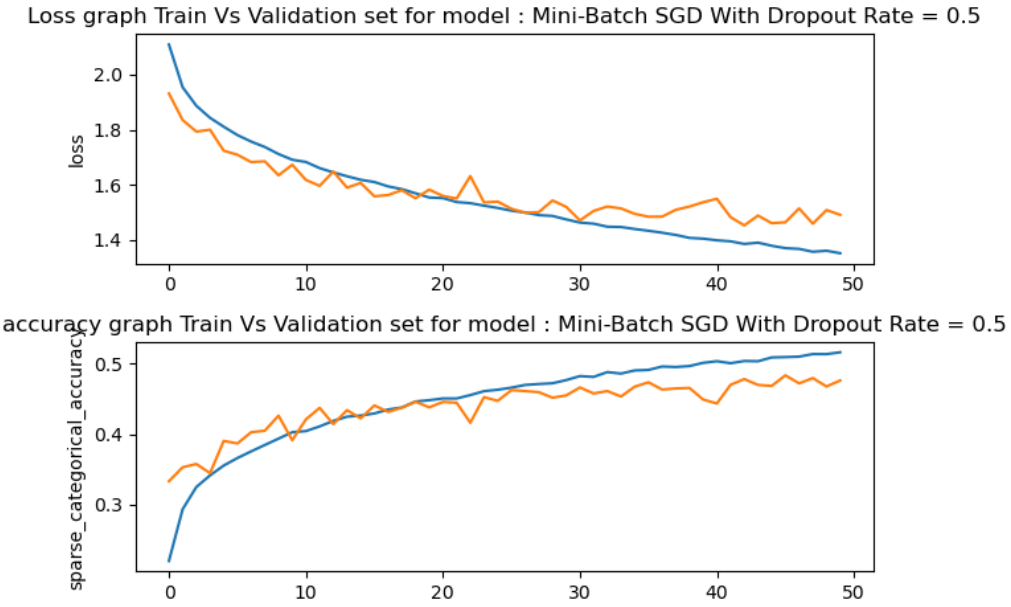

Figure 26: The convergence graph for the model : Mini-Batch GD with dropout rate = 0.5

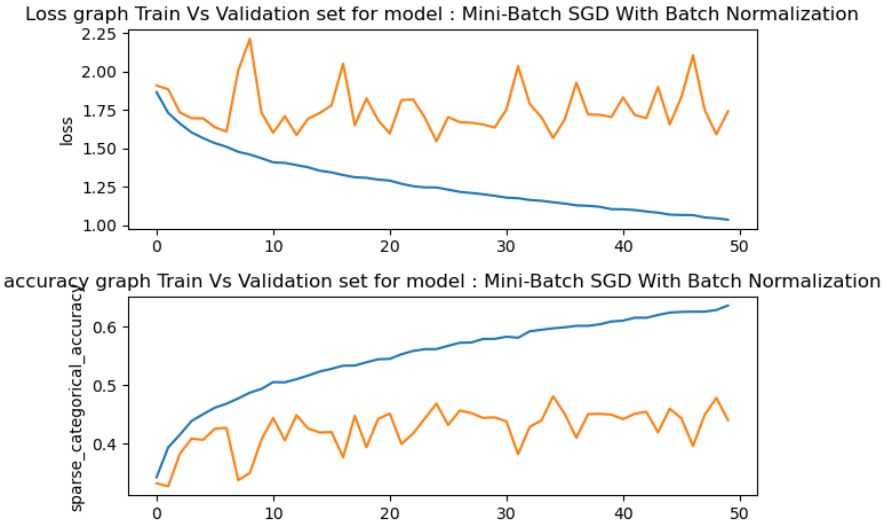

Figure 27: The convergence graph for the model : Adam

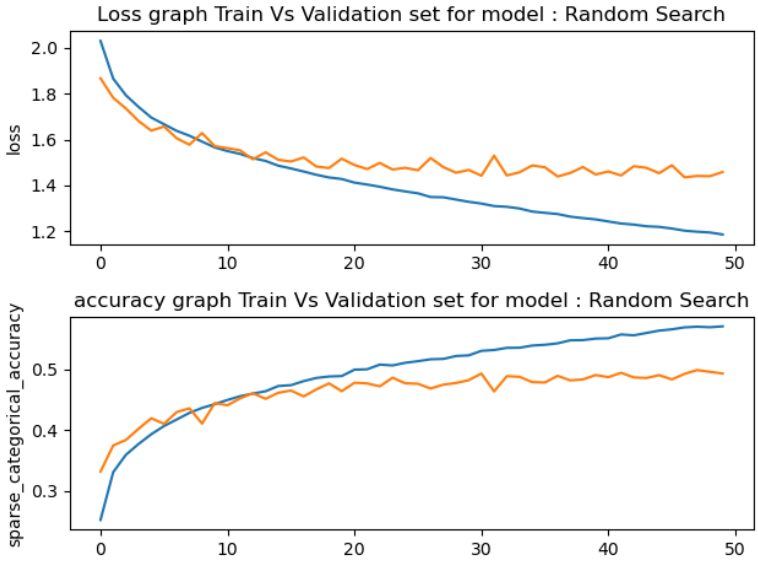

Figure 28: The convergence graph for the model : RMSProp

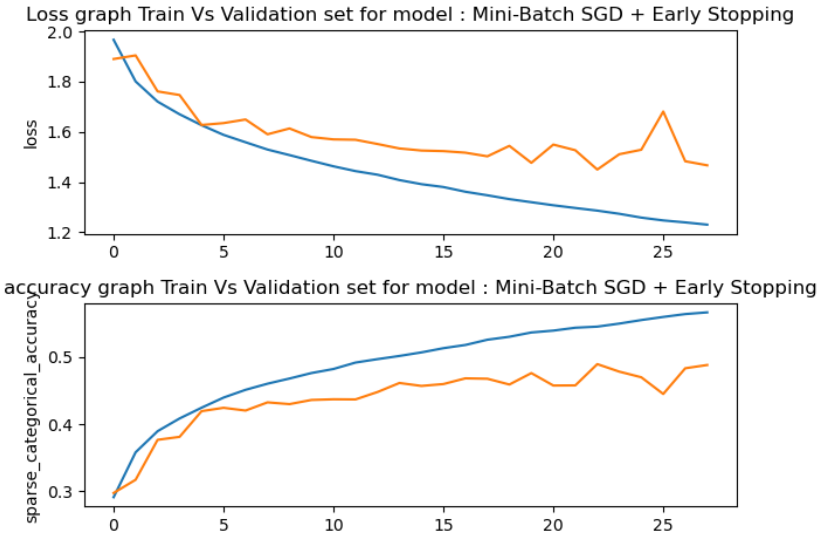

Figure 29: The convergence graph for the model : Mini-Batch GD + Early stopping

From the convergence graphs we can clearly visualize how can dropout and L2 regularization significantly reduce overfitting but also how a miss-configured batch normalization can lead to even worst results.

4 Conclusion

In conclusion, our experiments have provided valuable insights into the training and optimization of neural networks using different optimizers and overfitting avoidance techniques. We found that the choice of optimizer can significantly impact the performance of the model, with Adam generally outperforming SGD and RMSprop in terms of convergence speed and final accuracy. Additionally, we observed that using overfitting avoidance techniques such as dropout layers, batch normalization, and L2 regularization can effectively improve the generalization performance of the model, reducing overfitting and improving performance on unseen data.

Overall, our findings highlight the importance of carefully selecting and tuning these components when training neural networks. By understanding how different optimizers and overfitting avoidance techniques affect the training process, ultimately leading to improved performance and generalization capabilities.