1 INTRODUCTION

The purpose of this report is to discuss the training of a Convolutional Neural Network model having modified LeNet topology, i.e., without zero padding, on the CIFAR-10 dataset.
From the results of Case I (detailed in Section *Simulations and Results*), the obvious hyperparameter choice was **Learning Rate**.
Further results from Case II suggest that an important hyperparameter would be **Regularization** to curb the overfitting issue. However, it is identified that the **Stride** of (2,2) in the Pooling Layer may result in some information loss. Hence, the stride is reduced so that more information can be retained.
The rest of the cases are with different combinations of **Learning Rate**, **Stride**, **Regularization,** and changes in **Number of Epochs** at a later stage.

2 SIMULATIONS AND RESULTS

- **Case I:** After observing the training, validation, and test results [Train Accuracy: 91%; Test Accuracy: 55%] for the modified LeNet topology, it is understood that the model is overfitting the training data after ~2 epochs. Also, from the loss curve, it can be estimated that the learning rate is too high.

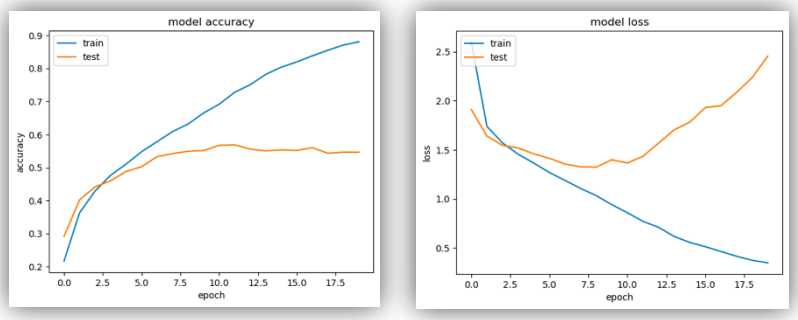

Figure 1: Accuracy and Loss on Modified LeNet

- **Case II:** Tweaking the Learning Rate to 0.01 gave an average train and test accuracy of 9.9% and 10% respectively.

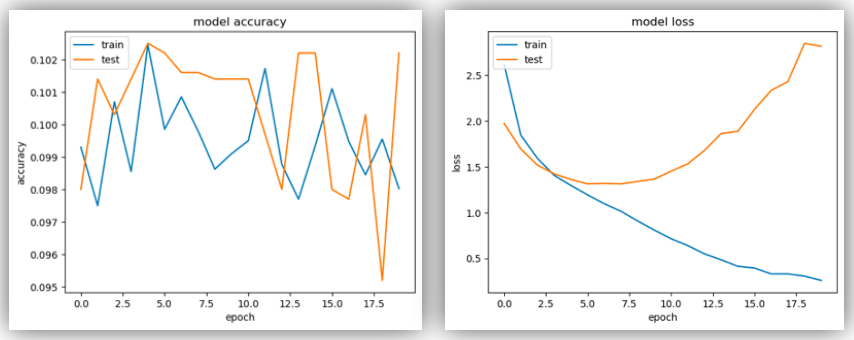

Figure 2:(Left) Accuracy after a change in Learning Rate; (Right) Loss after further reduction in Learning Rate

The oscillating training and validation accuracy suggests that there could be saturated neurons killing the gradients, which possibly indicates that the Learning Rate is still high.

Hence, the Learning Rate is further reduced to 0.001 *(Figure 2, Right)*. However, the overfitting problem reappears resulting in an average train and test accuracy of 94% and 55.4% respectively.

Appendix 2 shows the accuracy and loss curves from Cases III to VIII showing the effects of hyperparameters.

- **Case III:** The Stride, is adjusted to (1,1) in Pooling Layers to retain information. However, the overfitting issue continues with an average train and test accuracy of 76% and 40.5% respectively.

- **Case IV:** Regularization technique, Batch Normalization is introduced to penalise the model complexity, thereby reducing overfitting. But the problem of overfitting continues with average train and test accuracy of 100% and 74% respectively, indicating that the learning rate is still on the higher end.

- **Case V:** Keeping the learning rate the same, i.e., 0.001, introducing the Regularization technique, Dropout instead of Batch Normalization to observe if it gives better results. The average train and test accuracy of 70% and 47% respectively alludes to Dropout working slightly better with overfitting problems for CIFAR-10, however, reports far lower test accuracy.

- **Case VI:** Keeping the learning rate the same, i.e., 0.001, implementing both Regularization techniques, i.e., Batch Normalization and Dropout at once. The average train and test accuracies are 98% and 74% respectively. As the overfitting problem continues, the possible cause could be the high learning rates.

- **Case VII:** Further decreasing the learning rate to 1e-4, which leads to training and testing accuracy of 95% and 72% respectively impacting overfitting minutely.

- **Case VIII:** The stride in the Pooling Layer is increased to (2,2) to check if that helps reduce overfitting, leading to average train and test accuracy of 74.5% and 73% respectively and a heavy reduction in overfitting – however indicates slight underfitting.

- **Case IX:** Thus, to enhance learning, the training instances are increased by incrementing the number of the Epochs to 50 and 100 sequentially, to improve the testing accuracy and loss. Although after 100 epochs, the model shows a good fit, the average accuracy and loss values after learning for 50 and 100 epochs are not very evident.

No. of Epochs	Training Accuracy	Training Loss	Testing Accuracy	Testing Loss
50	**89.8%**	**0.29**	**78%**	**0.67**
100	**90.4%**	**0.27**	**78%**	**0.69**

Table 1: Accuracy & Loss after increasing Epochs.

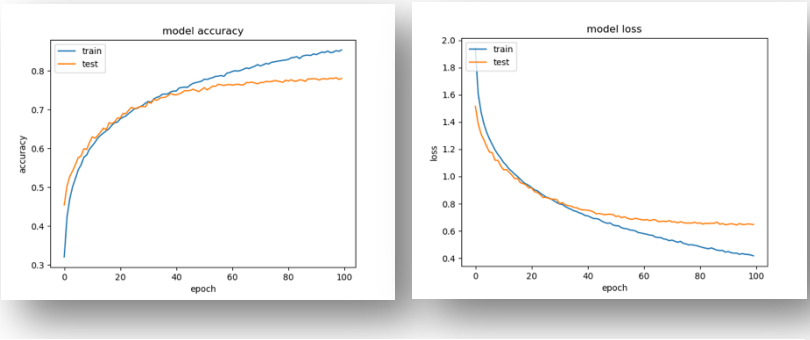

Figure 3: Accuracy and Loss after final run

3 CONCLUSION

The table in Appendix 1 signifies that a lower learning rate (1e-4) with adequate regularization on LeNet can lead to an accuracy of 78% and a loss of 0.68 providing a good fit.

Appendix 1 illustrates all the combinations of hyperparameters tested and the corresponding accuracy and loss.

4 RECOMMENDATIONS

i. Since there are obvious overfitting issues, Data Augmentation is one of the possible routes to improve accuracy and loss, as it will increase training data and introduce noise to improve generalisation.
ii. Learning rate decay is another way to improve generalization and optimization.

iii. Hyperparameter tuning using Randomized Search will help explore the range of choices for the given set of hyperparameters to determine the possible solutions for that model.
iv. Increasing model complexity will help with the underfitting issues cropping up after regularization in Case VIII.

Appendix 1: Table showing Accuracy and Loss for every hyperparameter combination based on the above-mentioned cases.

						Accuracy			Loss		
Case	**Sl. No.**	**Learning Rate**	**Stride**	**Regularization**	**Epoch**	**Train**	**Validation**	**Test**	**Train**	**Validation**	**Test**
2	1	0.01	(2,2)	Nil	20	0.1024	0.0977	0.1	2.3033	2.3034	2.3033
2	2	0.01	(2,2)	Nil	20	0.0981	0.0997	0.1	2.3033	2.3033	2.3032
2	3	0.01	(2,2)	Nil	20	0.098	0.1022	0.1	2.3034	2.3032	2.3031
2	4	0.001	(2,2)	Nil	20	0.9115	0.5621	0.5581	0.2589	2.8161	2.812
2	5	0.001	(2,2)	Nil	20	0.9508	0.5525	0.5499	0.1813	6.0064	5.8901
2	6	0.001	(2,2)	Nil	20	0.958	0.5554	0.5549	0.1965	8.1976	7.9484
3	1	0.001	(1,1)	Nil	20	0.8957	0.5442	0.5441	0.302	2.7242	2.7345
3	2	0.001	(1,1)	Nil	20	0.9557	0.5459	0.5429	0.1484	4.9269	5.0984
3	3	0.001	(1,1)	Nil	20	0.9523	0.5421	0.5474	0.2086	7.4916	7.3926
3	4	0.001	(1,1)	Nil	20	0.1004	0.0977	0.1	2.3027	2.3027	2.3026
3	5	0.001	(1,1)	Nil	20	0.8124	0.3468	0.3484	0.5261	3.9695	3.9262
3	6	0.001	(1,1)	Nil	20	0.8565	0.3483	0.3505	0.4288	8.7748	8.659
4	1	0.001	(1,1)	BatchNormalization	20	1	0.745	0.7382	0.0012	1.1748	1.1873
4	2	0.001	(1,1)	BatchNormalization	20	1	0.7461	0.7399	0.0014	1.187	1.2067
4	3	0.001	(1,1)	BatchNormalization	20	1	0.7454	0.7373	0.0019	1.1952	1.223
4	4	0.001	(1,1)	BatchNormalization	20	1	0.75	0.7436	8.44E-04	1.165	1.1798
5	1	0.001	(1,1)	Dropout	20	0.6561	0.4671	0.4611	0.9976	1.7179	1.7054
5	2	0.001	(1,1)	Dropout	20	0.6333	0.4821	0.4737	1.0475	1.5598	1.5799
5	3	0.001	(1,1)	Dropout	20	0.8209	0.4827	0.4679	0.563	0.4827	2.2756
6	1	0.001	(1,1)	BatchNormalization & Dropout	20	0.965	0.7435	0.743	0.1074	0.9707	0.9909
6	2	0.001	(1,1)	BatchNormalization & Dropout	20	0.9851	0.7407	0.7321	0.0476	1.211	1.2672
6	3	0.001	(1,1)	BatchNormalization & Dropout	20	0.9904	0.7467	0.7456	0.029	1.2386	1.2683
7	1	1.00E-04	(1,1)	BatchNormalization & Dropout	20	0.8955	0.7036	0.7004	0.3325	0.9134	0.9461
7	2	1.00E-04	(1,1)	BatchNormalization & Dropout	20	0.9686	0.732	0.7245	0.1156	0.899	0.9445
7	3	1.00E-04	(1,1)	BatchNormalization & Dropout	20	0.9851	0.737	0.7304	0.0582	0.9473	1.0082
8	1	1.00E-04	(2,2)	BatchNormalization & Dropout	20	0.6822	0.6848	0.6783	0.9096	0.9009	0.9009
8	2	1.00E-04	(2,2)	BatchNormalization & Dropout	20	0.7569	0.7456	0.743	0.6911	0.7278	0.738
8	3	1.00E-04	(2,2)	BatchNormalization & Dropout	20	0.7985	0.7682	0.7586	0.5689	0.6752	0.6903
9	1	1.00E-04	(2,2)	BatchNormalization & Dropout	50	0.8667	0.7843	0.778	0.3776	0.6331	0.6505
9	2	1.00E-04	(2,2)	BatchNormalization & Dropout	50	0.9042	0.7922	0.7861	0.2736	0.6487	0.6654
9	3	1.00E-04	(2,2)	BatchNormalization & Dropout	50	0.9232	0.7911	0.786	0.2198	0.6761	0.6867
9	4	1.00E-04	(2,2)	BatchNormalization & Dropout	100	0.854	0.7803	0.7772	0.4171	0.6468	0.6594

9	5	1.00E-04	(2,2)	BatchNormalization & Dropout	100	0.9187	0.7913	0.7827	0.2321	0.674	0.6867
9	6	1.00E-04	(2,2)	BatchNormalization & Dropout	100	0.9421	0.7901	0.7834	0.1669	0.7142	0.7255

Appendix 2: Training and Validation Accuracy and Loss presented sequentially for Cases III to VIII.