CS7015: Deep Learning (for Computer Vision)

Name: Anoubhav Agarwaal
Roll no: BE16B002

Part-A

The best performing model from Assignment 1A has the network architecture, as shown in Fig 1.

Parameters tested:

- **Learning rate** = [0.01, 0.001, 0.0001]
- **Weight_decay** = [0.3, 0.1, 0.03, 0.01, 0.003]

The best performing model had given us an 88.7% test accuracy in training mode. And, 90% test accuracy in evaluation mode. The train and test accuracy presented below are those calculated in training mode.

Along with weight decay, the learning rate was also varied. As we are **fine-tuning the previous network**, we do not want large gradient updates to replace the well-performing trained parameters by something undesirable.

All permutations of above learning rate and weight decay are tried i.e., 3*5 = 15 runs in total. Each run was trained for **30 epochs.**

The learning rate is the same for all the layers in the network.

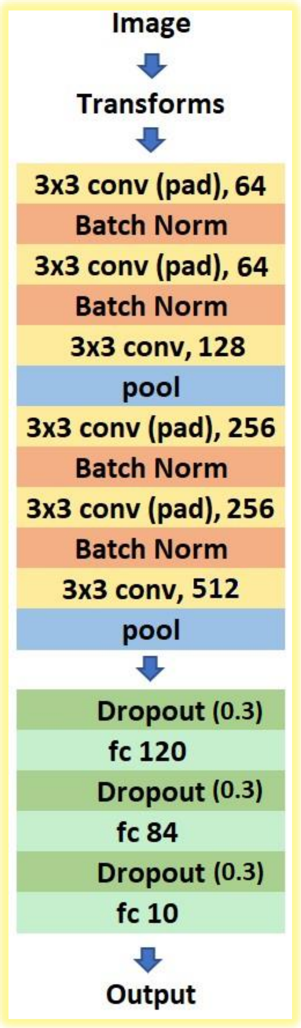


Fig 1. Best network architecture

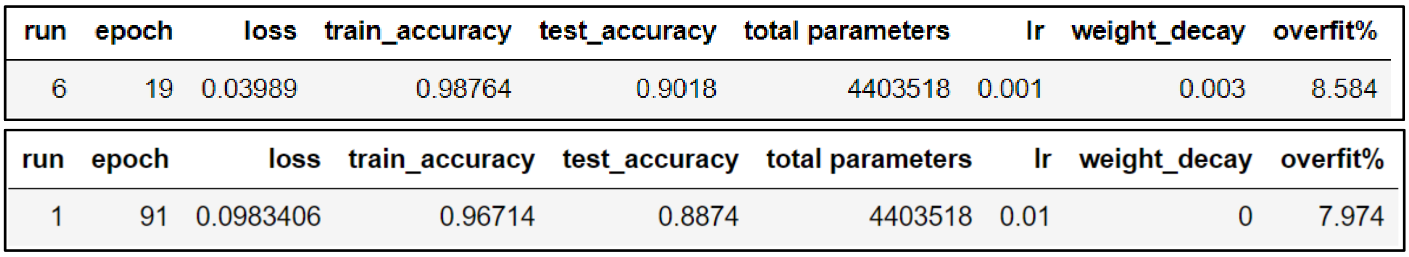
Analysis of results

1) **Best network configuration:**

Test accuracy of **90.18 %** using a learning rate of 0.001 and weight decay of 0.003.

The overfit% for the best network configuration is **8.6%** and has slightly increased compared to last time. The test accuracy has increased by **1.44 %**

From Fig 2., we see that the optimal learning rate (0.001) while fine-tuning the previous best performing network is ten times lower.



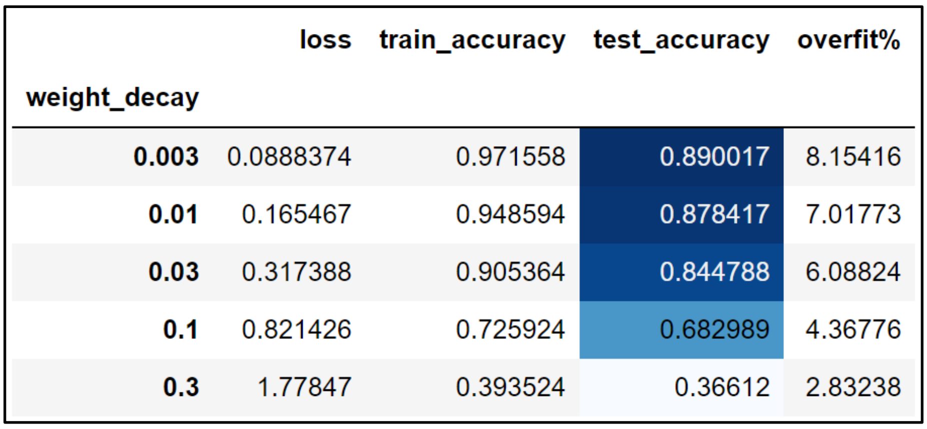
run	epoch	loss	train_accuracy	test_accuracy	total parameters	lr	weight_decay	overfit%
6	19	0.03989	0.98764	0.9018	4403518	0.001	0.003	8.584

run	epoch	loss	train_accuracy	test_accuracy	total parameters	lr	weight_decay	overfit%
1	91	0.0983406	0.96714	0.8874	4403518	0.01	0	7.974

Fig 2. The **top box** shows the parameter configurations for the best test accuracy obtained **after using weight decay**. The **bottom box** shows the best results of the network with **no weight decay**. We started training using the learned parameters from this run of the network.

2) **Weight decay:**

From Fig 3., we observe that on increasing the weight decay parameter, the **mean overfit% drops** significantly. Also, there is a **drop in both (mean) train and test accuracy**. This is because more importance is being given to keeping the magnitude of the weights low instead of classifying the input correctly. Thus, on increasing the weight_decay parameter, there exists a **trade-off between the mean accuracy of the model and the magnitude of the weights**. For our model, it is observed that keeping the weight decay low, about 0.003 is optimal. The model starts significantly underfitting on increasing the weight_decay beyond 0.03.



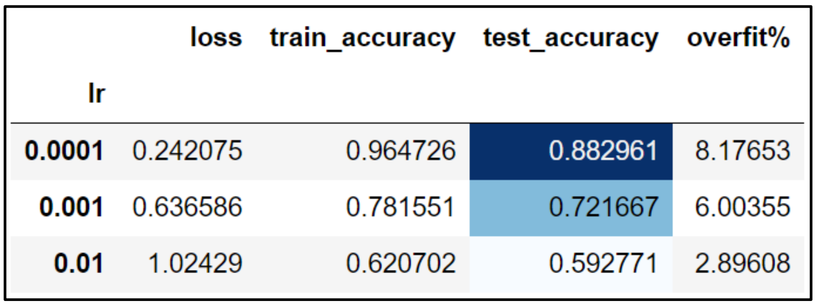
weight_decay	loss	train_accuracy	test_accuracy	overfit%
0.003	0.0888374	0.971558	0.890017	8.15416
0.01	0.165467	0.948594	0.878417	7.01773
0.03	0.317388	0.905364	0.844788	6.08824
0.1	0.821426	0.725924	0.682989	4.36776
0.3	1.77847	0.393524	0.36612	2.83238

Fig 3. The variation of the **mean** loss, train and test accuracy and overfit % with weight_decay.

3) **Learning Rate:**

From Fig 4., we observe that when the learning rate is low (such as 0.0001), the gradient updates are minimal. Thus, the mean test accuracy of 88.3% is almost the same as the starting test accuracy of 88.74%.

When the learning rate is higher, the effect of the weight decay parameter on the performance is more significant. With large decay parameter, the higher learning rate allowed for large gradient steps which lead to underfitting. Thus, we observe a massive drop in mean test accuracy with an increase in learning rate.



lr	loss	train_accuracy	test_accuracy	overfit%
0.0001	0.242075	0.964726	0.882961	8.17653
0.001	0.636586	0.781551	0.721667	6.00355
0.01	1.02429	0.620702	0.592771	2.89608

Fig 4. The variation of the **mean** loss, train and test accuracy, and overfit % with learning rate.

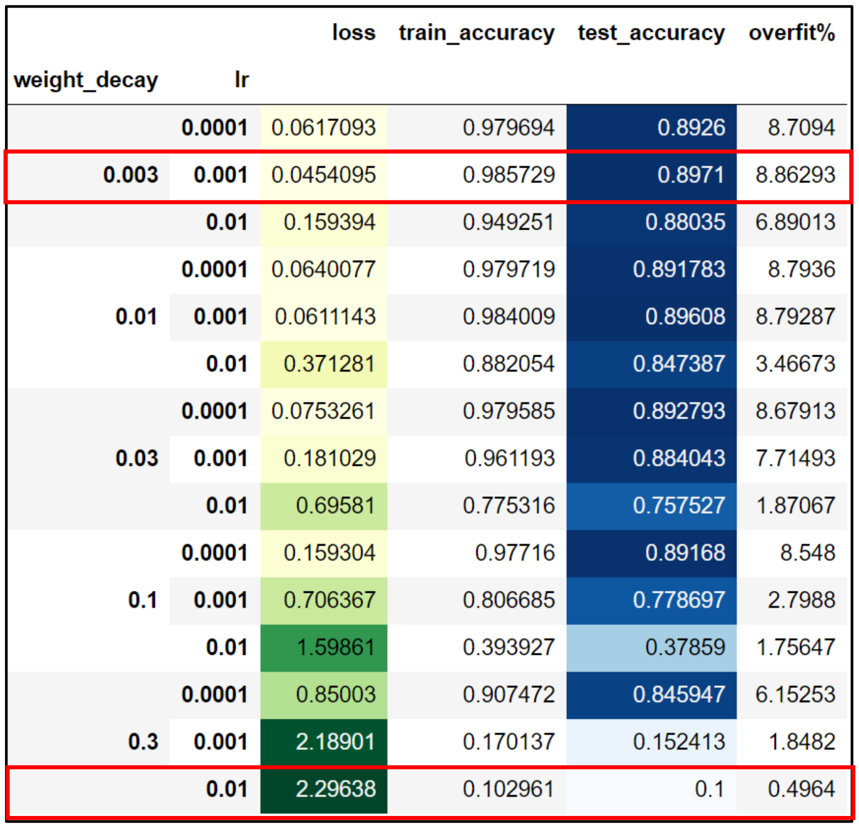
4) **Both weight decay and learning rate:**

From Fig 5., we observe that with the **increase in weight decay, there is an increase in loss and decrease in test accuracy.**

For weight_decay of 0.3 and learning rate of 0.1 (bottom row), we observe that the mean accuracy for both test and train drops to 10%. This is equivalent to guessing 1 of 10 classes.

The best network configurations were at a weight_decay = 0.003 and lr = 0.001. This field also has the highest mean test accuracy.

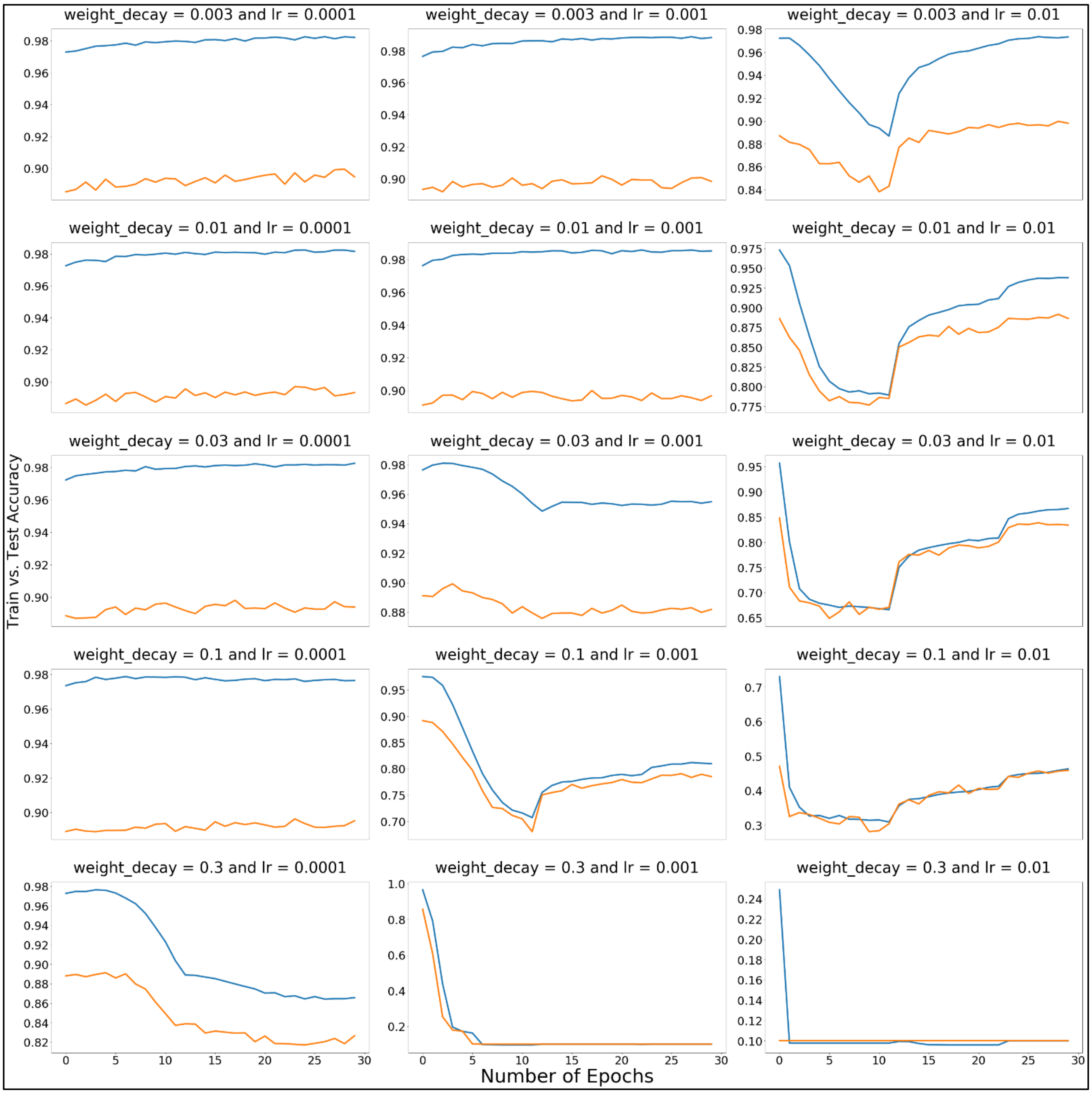
The table in Fig 5. is plotted below. We visualize the train and test accuracies as well as the loss for different weight decay and learning rate combinations.



weight_decay	lr	loss	train_accuracy	test_accuracy	overfit%
	0.0001	0.0617093	0.979694	0.8926	8.7094
0.003	**0.001**	0.0454095	0.985729	0.8971	8.86293
	0.01	0.159394	0.949251	0.88035	6.89013
	0.0001	0.0640077	0.979719	0.891783	8.7936
0.01	**0.001**	0.0611143	0.984009	0.89608	8.79287
	0.01	0.371281	0.882054	0.847387	3.46673
	0.0001	0.0753261	0.979585	0.892793	8.67913
0.03	**0.001**	0.181029	0.961193	0.884043	7.71493
	0.01	0.69581	0.775316	0.757527	1.87067
	0.0001	0.159304	0.97716	0.89168	8.548
0.1	**0.001**	0.706367	0.806685	0.778697	2.7988
	0.01	1.59861	0.393927	0.37859	1.75647
	0.0001	0.85003	0.907472	0.845947	6.15253
0.3	**0.001**	2.18901	0.170137	0.152413	1.8482
	0.01	2.29638	0.102961	0.1	0.4964

Fig 5. The variation of the **mean** loss, train and test accuracy, and overfit % with both learning rate and weight decay.

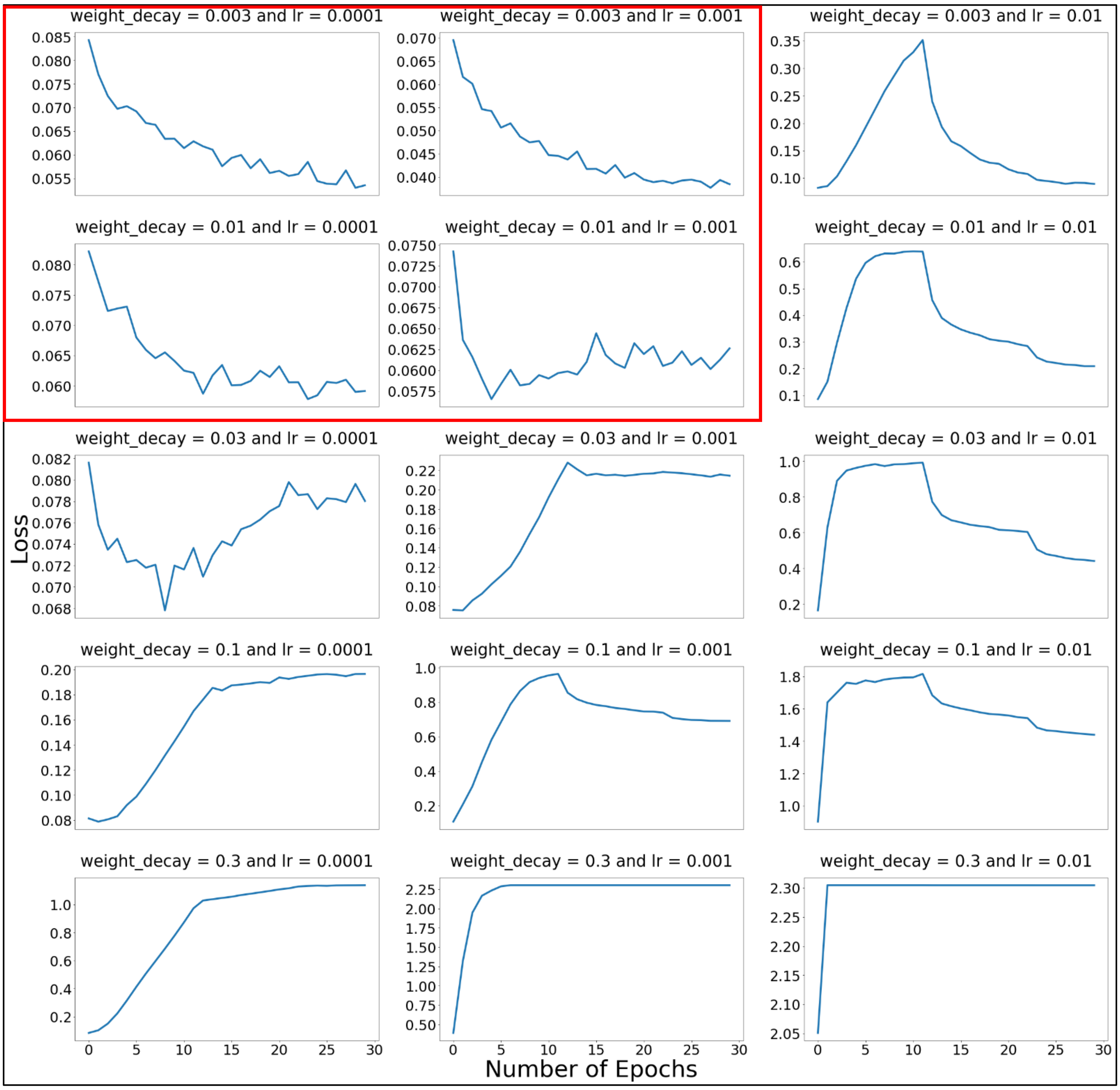
Train vs Test Accuracy Plots



Observations

- In the first column, i.e., lr = 0.0001, we observe that due to such a low learning rate, both accuracies have small fluctuations. However, for 0.3 weight decay, even such a low learning rate was not enough to stop a sudden drop in accuracy due to the large gradient in the loss function. For remaining weight decay values, there is a slow and gradual increase in both train and test accuracies.
- In the last column, i.e., lr = 0.01, we observe a sharp drop in the accuracy(both) initially, this is due to the large gradient updates. The extent to which the accuracy recovers depends on the weight_decay. After epoch 10, we see the accuracy start recovering. However, for weight_decay = 0.3, the model is trapped in a local minimum. Here, the magnitude of the weights is lowest, but, the classification accuracy is as good as random.
- In the middle column, i.e., lr = 0.001, we observe the effects seen in the first and last column. For weight decay of 0.1 and 0.3, we find the sharp drop (in accuracy) phenomena (seen in the last column). Whereas for weight decay of 0.003 and 0.01, we observe the slow and gradual increase in accuracies (seen in the first column). Lastly, weight decay of 0.03 (middle value) exhibits both of these phenomena (sharp drop and slow & gradual) in its plot.
- The overfit%, i.e., the vertical distance between the blue and orange line is lesser with larger weight_decay.

Visualization of Loss



Observations

- (Refer to the red box) We observe that the plots where the loss function continues to decrease steadily, give us the best results. The network parameters yielding a test accuracy of 90.18% has a weight_decay of 0.003 and lr of 0.001. The loss plot of which decreases steadily.
- In the last column, i.e., lr = 0.01, we observe a sharp increase in loss, after which it starts decreasing (i.e., it starts recovering). Let's compare this trend with the last column of the previous plots (train vs. test accuracy). There, we observed a sharp drop in accuracy phenomena up to epoch 10. Here, we observe a sharp increase in loss. Also, for the decay rate of 0.3, we observed previously that the accuracy plateaued at 10% and was stuck in local minima. This is evident in the loss plot, where the loss stagnates early on. The model quickly learns (due to large lr of 0.01) to minimize the weights irrespective of the classification accuracy, making it completely useless.
- On increasing weight decay, the loss usually tends to increase initially. This makes sense as we introduce weight decay for the sake of generalization. However, the increase in loss should be temporary after which it should start decreasing again. If this does not occur, the model will underfit, and the weight decay initially taken was too large.
- The plot at top left corner (weight decay = 0.003 and lr = 0.0001) v/s the plot at bottom right corner (weight decay = 0.3 and lr = 0.01) displays the stark effects of not picking a correct weight decay parameter and learning rate.