COL775: Deep Learning

Assignment - 1 Part 1

ResNet over Convolutional Networks and different Normalization schemes

Piyush Chauhan

2021CS11010

April 5, 2024

Contents

§1 Overview

ResNet (formally Residual Network), is a deep learning architecture renowned for its ability to train very deep neural networks effectively. Given an input x, ResNet computes the output y, by explicitly fitting the residual mapping , $F(x) = H(x) - x$, where H(x) denotes the desired underlying mapping.
The skip connections help in mitigating the problem of vanishing gradients that can arise in deep networks. The skip connections enable information to travel directly between non-adjacent layers, allowing for more efficient training of deeper networks.

§2 Model Specifications

Below are the details of ResNet model used for image classification task

1. **Hyper-parameters** :
   - Batch Size : 32, Max Epochs : 50 , n : 2 , total layers $(6n + 2) = 14$, with skip connections in every 2 layer, r : 25
   - learning rate : $1e^{-4}$(for SGD and Adams) and 0.1 (for SGD)
   - weight dacay : $1e^{-4}$ for 0.1 LR and $1e^{-7}$ for $1e^{-4}$ LR.
2. **Criterion** : Cross Entropy Loss
3. **Optimizer** : Experimented with two optimizers SGD and Adams.
4. **Scheduler** : Tested two scheduler StepLR and CosineAnnealingLR. The latter was found to perform well on the default parameters given and thus was used for rest of the analysis.**Stopping Criterion** : Best Validation Accuracy.
5. **Augmentation** : Applied RandomCrop and HorizontalFlip and compared the performance with no augmented data.

§3 Results

§3.1 Effect of Data Augmentation on PyTorch inbuilt Batch normalization

Following data augmentation was performed only on train and val set on ResNet with SGD and learning rate: 0.1 using PyTorch inbuilt batch normalization:
$RandomCrop(256 \times 256) \rightarrow HorizontalFlip$.

Dataset	Micro F1(%)	Macro F1(%)	Accuracy(%)
Without Augmentation			
Train Set	99.53	99.53	99.53
Val Set	98.87	98.87	98.87
Test Set	89.56	89.64	89.56
With Augmentation			
Train Set	79.53	79.6	79.53
Val Set	79.46	79.51	79.46
Test Set	83.53	83.56	83.53

Table 1 Augmentation effect on ResNet with SGD and 0.1 LR

Inference from table

- **Overfitting** Without augmentation, overfitting is observed which is evident from high difference in train and test accuracies.
- Augmenting data has reduced the accuracy and F1 scored but it has helped reducing overfitting, as the gap between train and test accuracies is highly reduced.
- Before augmentation, the split between val and test set is approx 9% which is reduced to 4%. This shows that augmentation has made data more robust.
- There is also a drop in test set accuract of about 5%, there are multiple reasons for it, model might not have enough capacity, or after random cropping, the object might not bre present in crop.

As augmentation helps in reducing overfitting, all data and plots ahead are ran on model with aforementioned augmentation.

§3.2 Analysis of Train and Val Loss

To see the effect of various optimizer on base ResNet model, I used SGD and Adams with different learning rates.

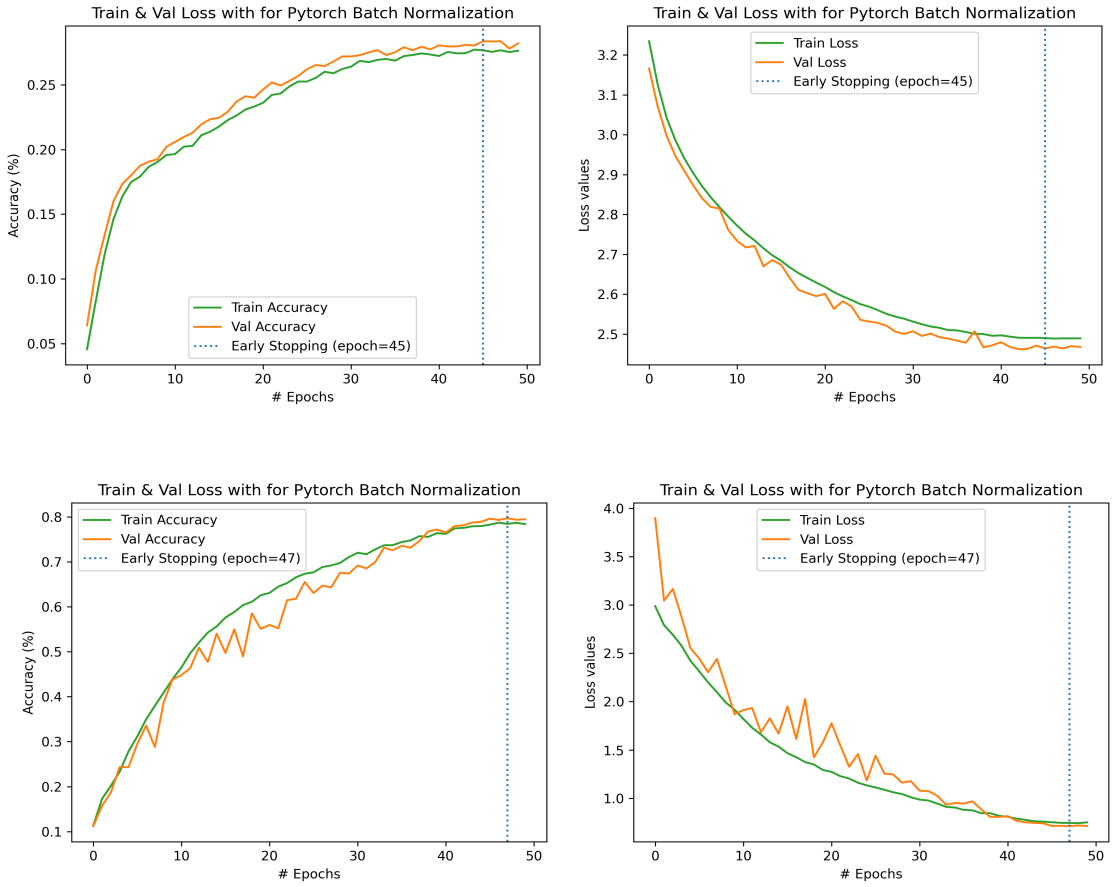

Figure 1: Train and Val accuracy(former) and loss(latter) for Pytorch Batch Normalizer with SGD and LR = 0.0001 (above) and LR=0.1 (below)

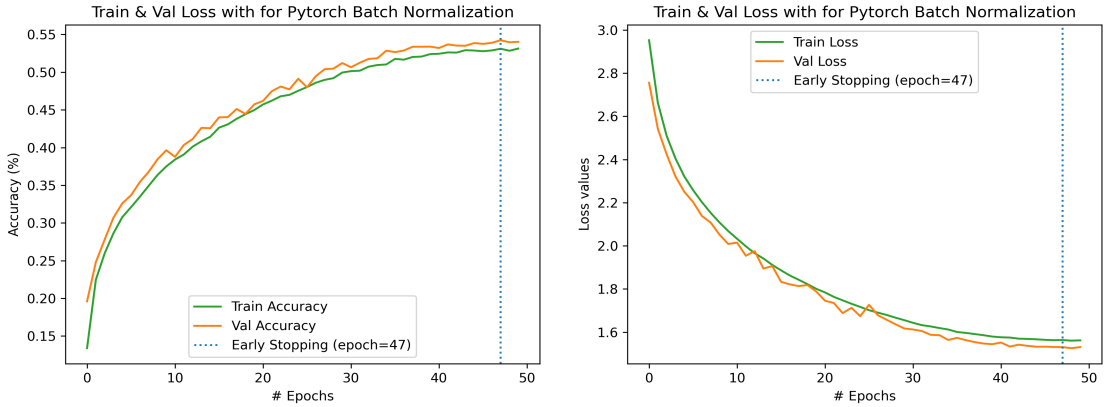

Figure 2: Train and Val accuracy(former) and loss(latter) for Pytorch Batch Normalizer with Adam and LR = 0.0001

Inference from plots :

- In all plots training loss curve decreases smoothly while val curve (especially in SGD with LR = 0.1) is irregular.
- For SGD, we observe that after 40^{th} epoch (LR = 0.0001) and after 30^{th} epoch (LR = 0.1) for SGD and after 40^{th} epoch in Adams, the val curve smoothen out indicating that model has reached the minima valley. Loss curve also flattens out after mentioned epochs, confirming the mentioned observation.
- More irregular curve is observed for higer LR in SGD
- Train and val loss continues to decrease and settles only near the end of epoches.
- Gap between train and val loss and accuracy in every case is minimal. Thus, model does not over-fit on training dataset.

NOTE :
While experimenting with different learning rate and optimiers , it was found the SGD with high learning rate (eg 0.1) performed much better than Adams or SGD with lower learning rate. Thus, all the data plotted from now on is based on SGD with LR = 0.1 (unless stated otherwise). Plot for SGD with LR = 0.0001 is in the Appendix Section.

§4 Impact Of Normalization

§4.1 Sanity check with comparison of Torch and Custom Batch Normalization

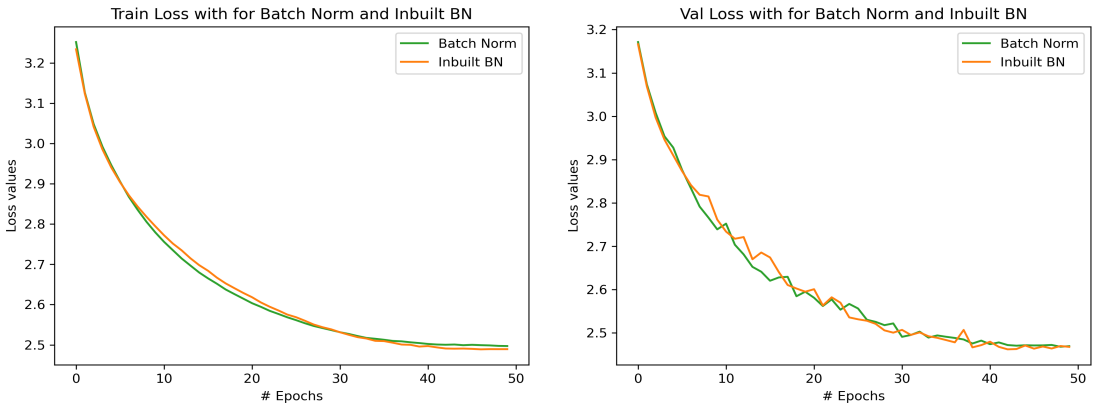

Figure 3: Comparison of train and val loss curves v/s epochs for custom batch normalization and inbuilt batch normalization for SGD.

Dataset	Accuracy(%)	Micro F1	Macro F1
Inbuilt BN			
Train Set	79.53	0.7953	0.796
Val Set	79.46	0.7946	0.7951
Test Set	83.53	0.8353	0.8363
Custom BN			
Train Set	79.98	0.7998	0.8012
Val Set	79.27	0.7927	0.7938
Test Set	84.31	0.8431	0.8442

Table 2 Augmentation effect on ResNet with SGD

Inference from table and Plot :

- the loss curves of both the implementations is very similar, however, training loss for the pytorch inbuilt normalizer is slightly lower than that of custom implementation. Similar trend is observed in val curve as well.

Therefore, we can conclude that the custom batch normalization implementation is correct and achieves similar loss curves and performance metrics as that of Pytorch inbuilt Batch Normalizer.

§4.2 Comparison of loss curves for all 6 Normalizations

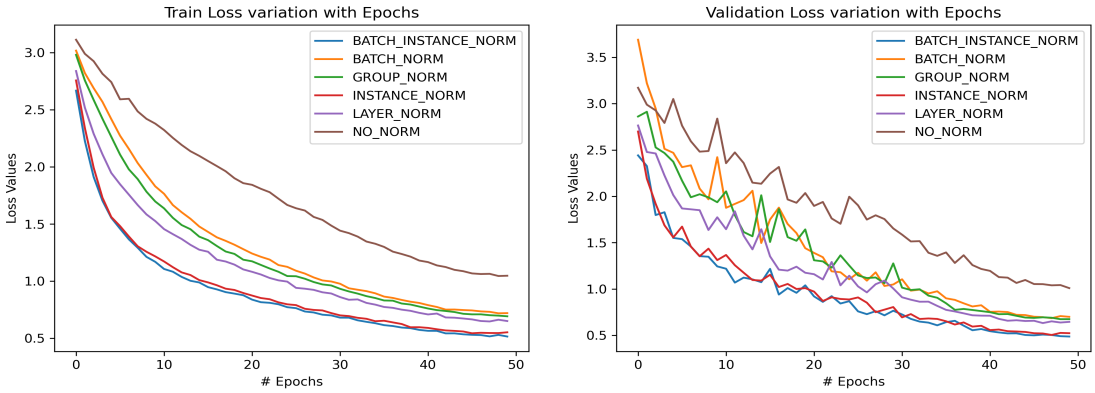

Figure 4: Comparison of train and val loss curves v/s epochs for all 6 normalization schemes.

Inference from Plot :

1. **Train Loss Comparison :**
   - For all the schemes train loss curves show varied behaviour of decay.
   - Initial rate of decay of train loss if faster for Instance Norm and Batch Instance Norm, followed by Layer Norm, Group Norm, Batch norm which have very similar curve
   - No Norm decays the worst in all others and have highest Loss value among all other norms respectively.
2. **Val Loss Comparison :**
   - For all implementations val loss curves fluctuates at beginning and then becomes smoother towards the end after 25-30 epochs. This shows model starts towards the bottom converge to a local minima valley.
   - Final loss values are highest for No Norm, highlighting the fact that normalization significantly improves training as compared to no normalization.
3. From curves it is evident that Instance Norm and Batch Instance Norm achieves the best performance and No Norm the least.

§4.3 Comparison of performance metrics for all 6 Normalizations

Table 3 Performance results of Resnet on Bird Dataser with Pytorch inbuilt batch normalization and all custom implementations.

Method	**Train (%)**			**Val (%)**			**Test (%)**		
	Accuracy	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**
Inbuilt Batch Norm	79.53	0.7953	0.796	79.46	0.7946	0.7951	83.53	0.8353	0.8363
No Norm	70.48	0.7048	0.7067	70.45	0.7045	0.7063	76.27	0.7627	0.764
Batch Norm	79.98	0.7998	0.8012	79.27	0.7927	0.7938	84.31	0.8431	0.8442
Batch Instance Norm	**85.85**	0.8585	0.859	**85.59**	0.8559	0.8564	**88.93**	0.8893	0.8902
Group Norm	80.21	0.8021	0.8033	79.97	0.7997	0.8013	84.12	0.8412	0.8431
Instance Norm	84.5	0.845	0.8454	85.38	0.8538	0.8543	88.67	0.8867	0.8873
Layer Norm	81.46	0.8146	0.8154	81.87	0.8187	0.8196	85.4	0.854	0.8549

Inference from Table :

- Batch Instance Norm performs better than every other model, followed by Instance Norm, Layer Norm, Batch Norm, Group Norm Inbuilt Batch norm and No Norm. These results are inline with the validation loss values discussed in the previous sub section.
- Validation performance are similar to that of test performance(under bound of 5%). This, after data augmentation, val set provides true reflection of prospective test performance.

§4.4 Impact of Batch Size in Batch Norm and Group norm

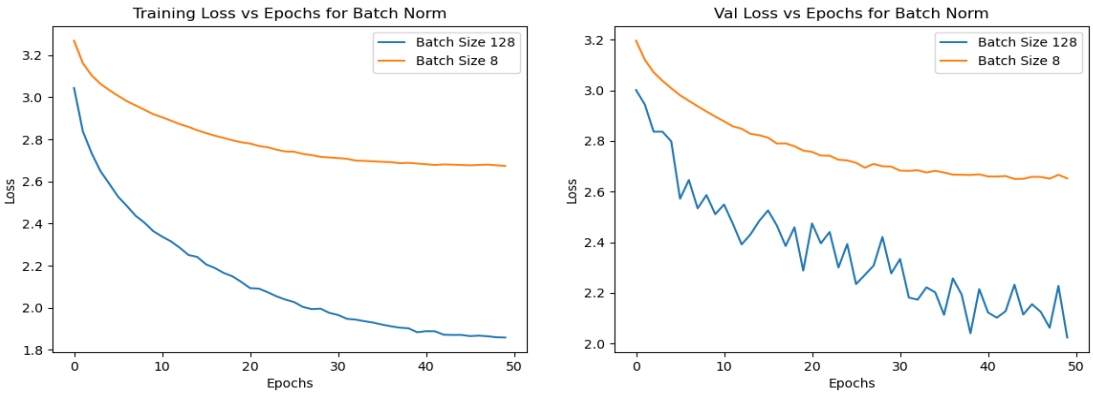

Figure 5: Impact of batch size on Batch Normalization, Train loss curve, and Val loss curve.

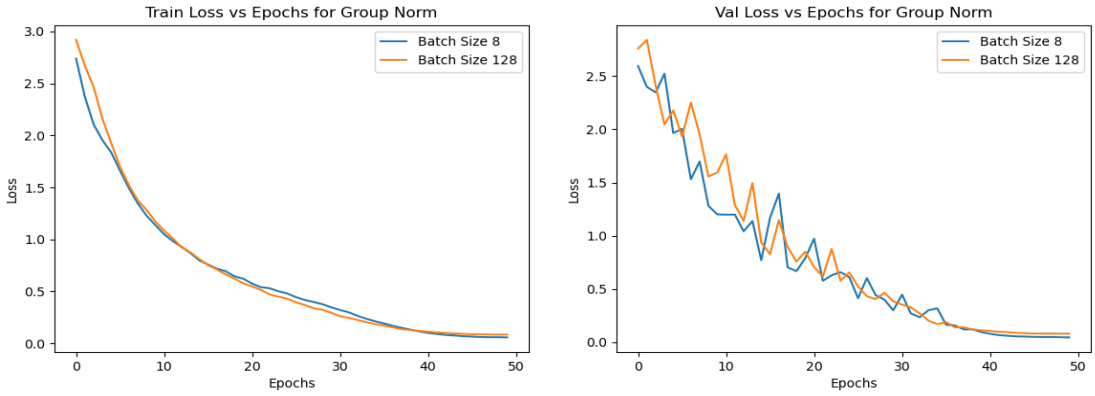

Figure 6: Impact of batch size on Group Normalization, Train loss curve, and Val loss curve.

Inference from plot

- For Batch Normalization, for batch size = 128 the train loss decreases at a faster rate as compared to batch size = 8, therefore the curve for batch size = 8 remains lower than batch size = 128. Val curve too reduces faster for batch size = 128.
- There is minimal impact of batch size on train and val loss curves on Group Normalization as claimed in the paper. This validates that the performance of GN remains similar for both the cases.

§4.5 GradCAM analysis

§5 Appendix A

§5.1 Adams with LR = 0.0001

Table 4 Performance results of Resnet on Bird Dataser with Pytorch inbuilt batch normalization and all custom implementations for Adam with LR =0.0001.

Method	**Train (%)**			**Val (%)**			**Test (%)**		
	Accuracy	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**
Inbuilt Batch Norm	53.84	0.5384	0.5364	54.24	0.5424	0.5398	51.2	0.512	0.5104
No Norm	52.28	0.5228	0.5197	52.24	0.5224	0.5196	49.21	0.4921	0.4886
Batch Norm	57.05	0.5705	0.5686	57.15	0.5715	0.5689	53.88	0.5388	0.5367
Batch Instance Norm	73.22	0.7322	0.7318	73.32	0.7332	0.7331	69.6	0.696	0.6952
Group Norm	55.47	0.5547	0.5531	55.64	0.5564	0.5553	52.96	0.5296	0.5283
Instance Norm	**77.42**	0.7742	0.7747	**77.18**	0.7718	0.7721	**72.32**	0.7232	0.7238
Layer Norm	59.64	0.5964	0.5952	59.59	0.5959	0.5958	56.56	0.5656	0.5646

§5.2 SGD with LR = 0.0001

Below are the results for SGD with LR = 0.0001.

Table 5 Performance results of Resnet on Bird Dataser with Pytorch inbuilt batch normalization and all custom implementations for SGD with LR =0.0001.

Method	**Train (%)**			**Val (%)**			**Test (%)**		
	Accuracy	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**	**Accuracy**	**Micro F1**	**Macro F1**
Inbuilt Batch Norm	28.05	0.2805	0.2622	28.37	0.2837	0.2669	27.81	0.2781	0.2609
No Norm	25.4	0.254	0.2371	26.2	0.262	0.2455	25.55	0.2555	0.2402
Batch Norm	25.29	0.2529	0.2379	26.49	0.2649	0.2487	25.08	0.2508	0.2368
Batch Instance Norm	27.25	0.2725	0.2423	27.66	0.2766	0.2486	26.21	0.2621	0.2302
Group Norm	24.25	0.2425	0.2205	24.75	0.2475	0.2275	23.89	0.2389	0.218
Instance Norm	26.13	0.2613	0.2279	26.3	0.263	0.2294	25.87	0.2587	0.2281
Layer Norm	**28.48**	0.2848	0.2647	**29.01**	0.2901	0.2717	**27.75**	0.2775	0.2588

§6 Model Link

MODELS