COL775:Deep Learning

Assignment 1.1: ResNet over Convolutional Networks and different Normalization schemes

Name: Bogam Sai Prabhath Entry Number:2023AIB2079

---

1. Image Classification using Residual Network

Link to the trained model is hear

implemented ResNet architecture from scratch in PyTorch. Assume that the total number of layers in the network is given by 6n+2.the first convolution layer which is independent of 6n+2 There are n Residual blocks and Each residual block has 2 layers and the number of filters are used as mentioned in the problem statement the last layer is fully connected layer.downsampling is done after two skip connections and also the number of filters are changed as 16,32,64 respectively.the global mean pooling is done after convolution operations and it is feeded into output layer.

Results of Resnet base model for 25 class classifications :

Optimizer: SGD
Hyperparameters:

1. Batch size : 32
2. Epochs: 50
3. Starting Learning rate: 0.0001
4. n=2 (residual block layers)
5. r=25 (classes)

Criterion: Cross Entropy Loss

Normalization: PyTorch Batch Norm
The results are as follows :

Data set	Accuracy	Micro F1	Macro F1
Train Data	0.508141	0.50814	0.50526
Validation Data	0.514085	0.51408	0.51257
Test Data	0.4322	0.4322	0.4239

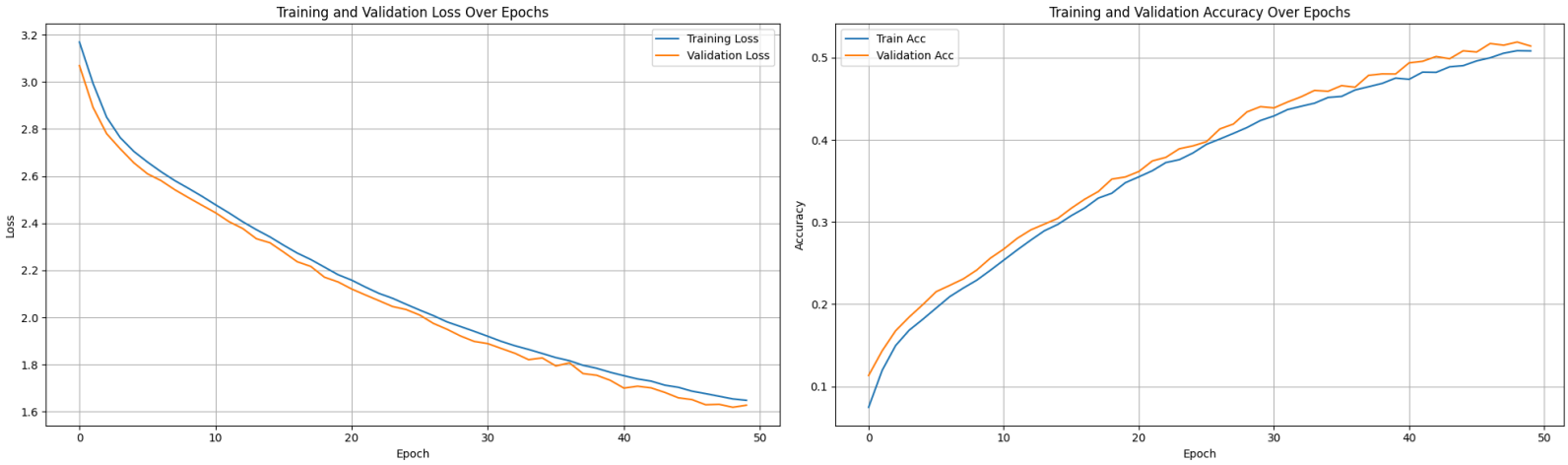


Figure 1:Train and Val (a) loss and (b) accuracy curve for PyTorch batch normalization using SGD

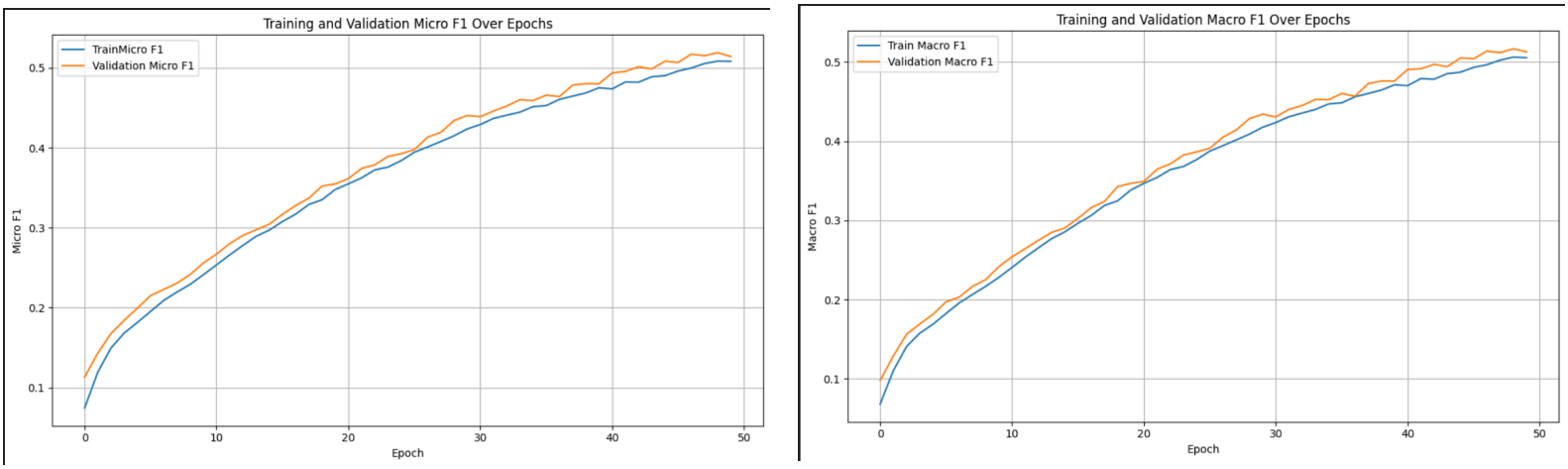


Figure 2:Train and Val (a) Micro F1 and (b) Macro F1 curve for PyTorch batch normalization using SGD

Plot Observations:

1. The model's training and validation losses decreased substantially early on, with the loss curves remaining close, indicating no overfitting and effective initial learning.
2. Accuracy for both training and validation improved consistently across all 50 epochs, with no plateaus, suggesting potential for further gains with more training.
3. The absence of a gap between training and validation performance metrics demonstrates the model's good generalization ability, supported by effective use of Batch Normalization and Cross-Entropy Loss.
4. Although showing signs of convergence, the model did not fully converge within the 50 epochs, indicating room for improvement either through more epochs or an adjusted learning rate strategy.

As the above model is inaccurate, i have changed the **initial learning to 0.001 without using the scheduler** the observations are as follows:

Data set	**Accuracy**	**Micro F1**	**Macro F1**
Train Data	0.8864	0.8864	0.8869
Validation Data	0.8898	0.889	0.8900
Test Data	0.8015	0.8015	0.8026

Plot Observations:

1. Both training and validation accuracy rise quickly and remain closely aligned throughout training, which indicates the higher learning rate of 0.001 is initially effective but may benefit from a decrease as training progresses to prevent overfitting.
2. Training and validation loss decrease sharply at first, with validation loss exhibiting some fluctuation in later epochs.
3. The presence of slight oscillations in the validation loss towards the later epochs could hint at the model starting to overfit.

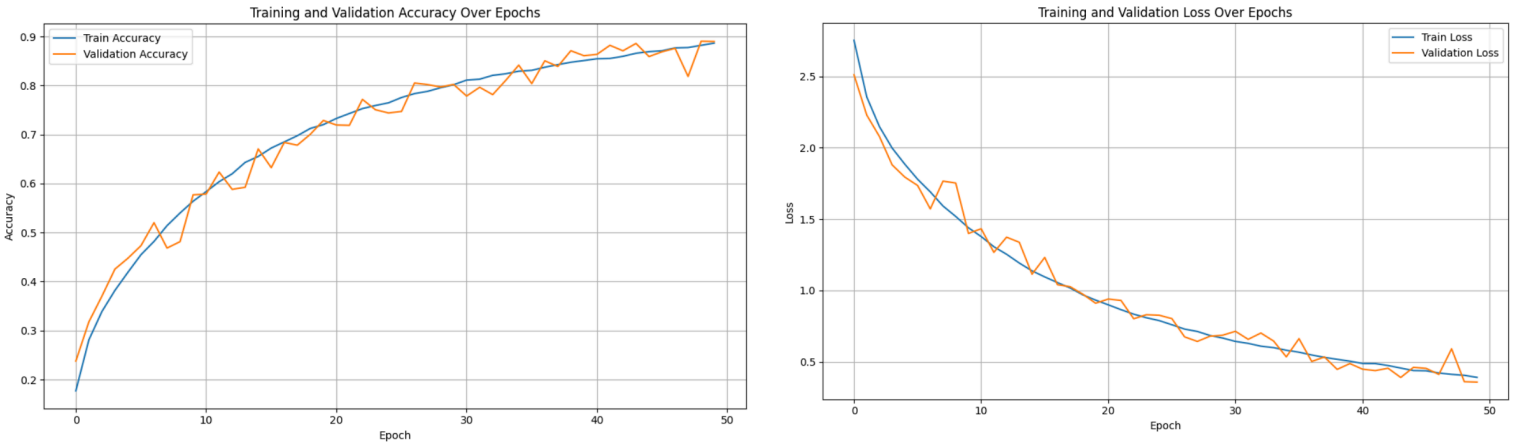


Figure 3:Train and Val (a) loss and (b) accuracy curve for PyTorch batch normalization using SGD Lr=0.001

2. Impact of Normalization

2.1 Comparison of curves for all 7 Normalizations including PyTorch:

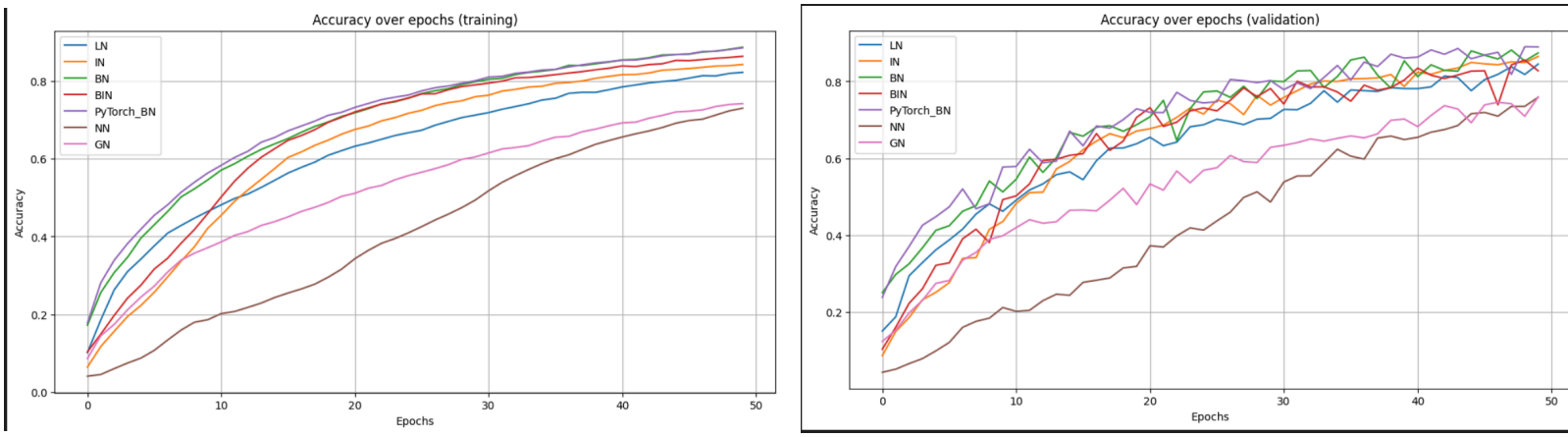


Figure 4(a) illustrates the decay of Accuracy values over the validation set and Figure 4 (b) over the Train set for all 7 implementations. a.Batch Normalization (BN) b.Instance Normalization (IN) c.Batch-Instance Normalization (BIN) d.Layer Normalization (LN) e.Group Normalization (GN) f.No Normalization (NN) g.PyTorch Normalization (PN)

The Figure 4 is the comparison of train and Validation Accuracy v/s epochs for all 7 Normalizations

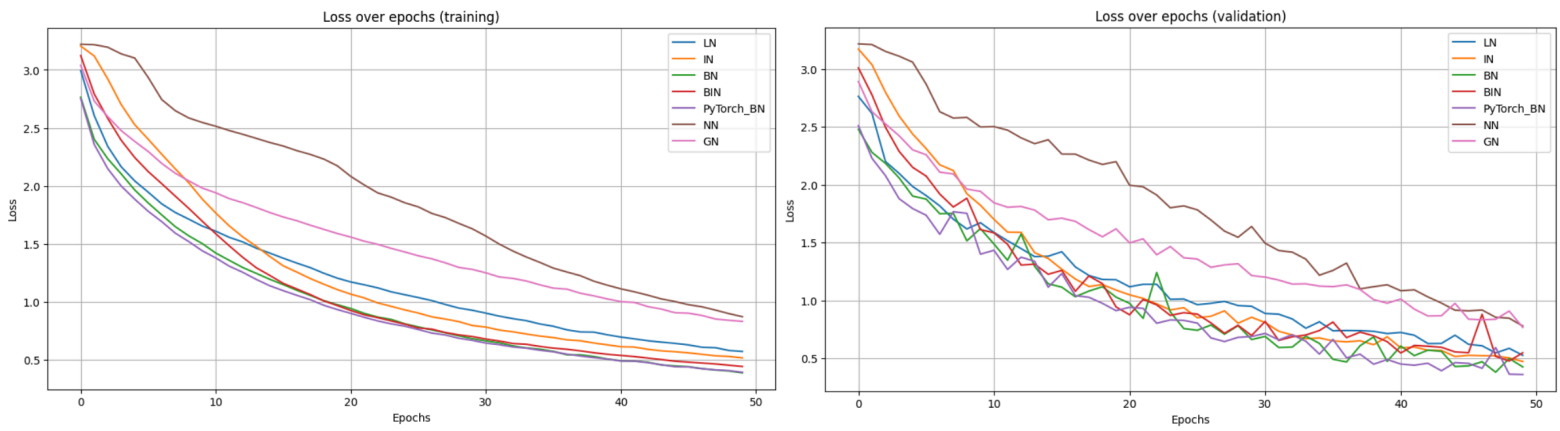


Figure 5 is the comparison of train and Validation Loss v/s epochs for all 7 Normalizations

Observation and Inference:

Accuracy of Validation Data:

1. BN and BIN perform the best, showing the highest accuracy throughout the training process.
2. IN, LN and PyTorch BN show similar performance but slightly lower accuracy than BN and BIN.
3. GN performance is moderate, not as high as BN/BIN but better than NN.
4. NN has the lowest accuracy, indicating the effectiveness of normalization techniques in improving model performance.

Loss of Validation Data:

1. BN and BIN show the lowest loss, indicating a good fit on the validation data.
2. IN, LN and PyTorch BN again show similar loss values, slightly higher than those of BN and BIN.
3. GN has moderately lower loss values than NN but doesn't perform as well as BN/BIN.
4. NN shows the highest loss, reinforcing the importance of normalization in neural networks.

Accuracy of Training Data:

1. All normalization techniques, except NN, show similar trends with high accuracy, suggesting that normalization helps in better and faster learning.

2. NN significantly lags behind, indicating that the model is less efficient in learning patterns from the training data without normalization.

Loss of Training Data:

1. BN and BIN lead to the lowest loss, stabilizing the learning process.
2. IN, LN, and PyTorch BN also help reduce loss but are less effective than BN/BIN.
3. GN is better than no normalization but doesn't achieve as low a loss as the other techniques.
4. NN results in the highest loss across all epochs, showcasing the challenges in training models without normalization.

Inference and Comparison:

1. Batch Normalization (BN) and Batch-Instance Normalization (BIN) are the most effective in both the training and validation phases, showing high accuracy and low loss.
2. Layer Normalization (LN) and Instance Normalization (IN) are effective alternatives, with performance slightly behind BN and BIN.
3. Group Normalization (GN) provides a middle ground, outperforming no normalization but not matching the effectiveness of BN and BIN.
4. No Normalization (NN) results in the poorest performance, highlighting the necessity of normalization techniques to improve training stability and model performance.
5. PyTorch's built-in normalization (PyTorch BN) competes closely with BN's custom implementations, showing that standard library functions are quite efficient.

2.2 Comparison of performance metrics for all 6 Normalizations:

Norms	Train (%)			Val (%)			Test (%)		
	Accuracy	Micro F1	Macro F1	Accuracy	Micro F1	Macro F1	Accuracy	Micro F1	Macro F1
Torch	0.886	0.886	0.886	0.889	0.889	0.890	0.801	0.801	0.802
BN	0.887	0.887	0.888	0.874	0.874	0.877	0.828	0.828	0.833
BIN	0.864	0.864	0.865	0.827	0.827	0.831	0.734	0.734	0.732
GN	0.742	0.742	0.743	0.759	0.759	0.759	0.684	0.684	0.681
IN	0.843	0.843	0.844	0.864	0.864	0.866	0.778	0.778	0.776
LN	0.823	0.823	0.823	0.845	0.845	0.843	0.798	0.798	0.801
NN	0.730	0.730	0.731	0.758	0.758	0.761	0.646	0.646	0.649

Table 1: Performance results of Resnet with Pytorch inbuilt and all implementations of normalization schemes.

2.3 sanity check of PyTorch Libray and implemented BN :

From Table 1 we can infer that the PyTorch Library BN and the Implemented BN gives almost the same Accuracy and other Metrics.

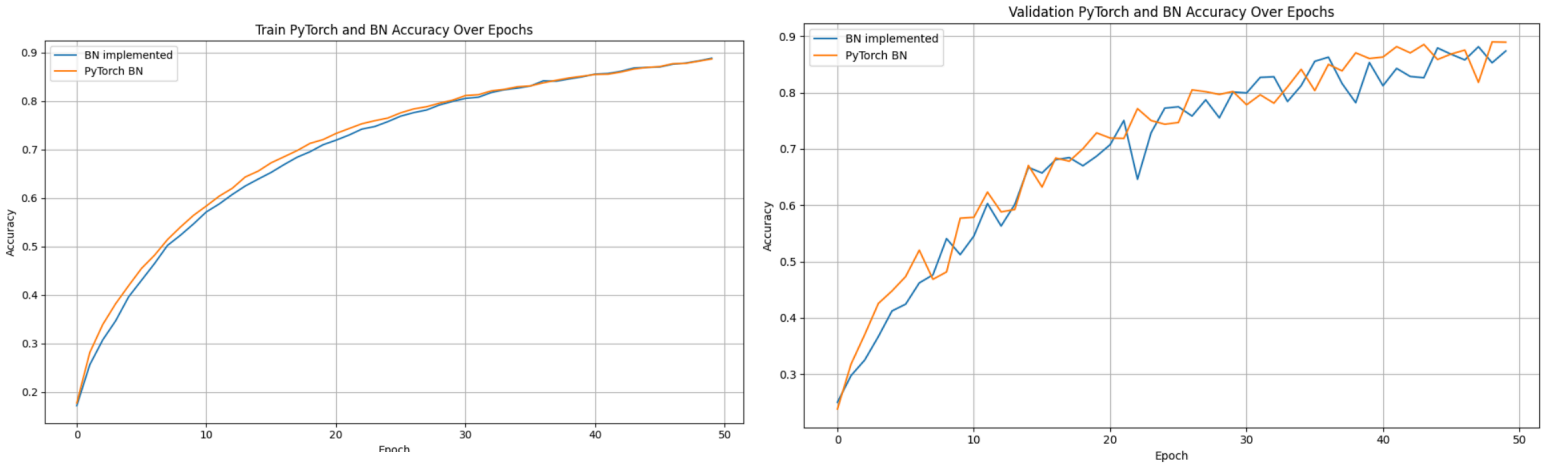


Figure 6 is the comparison of a)train and b) Validation Accuracy v/s epochs for PyTorch and BN

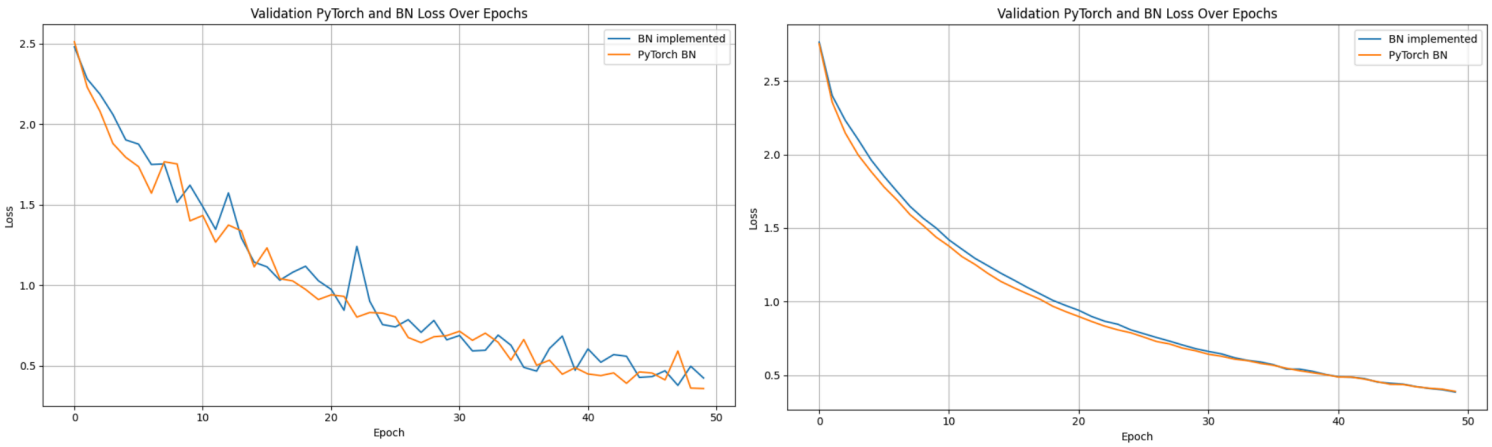


Figure 7 is the comparison of a)train and b) Validation Loss v/s epochs for PyTorch and BN

2.4 Impact of Batch Size:

The BN and the GN are trained with a varying batch size of 8 and 128 , and the following observations were made and compared with the same varient of batch size 32.

2.4.1 Comparision of BN:

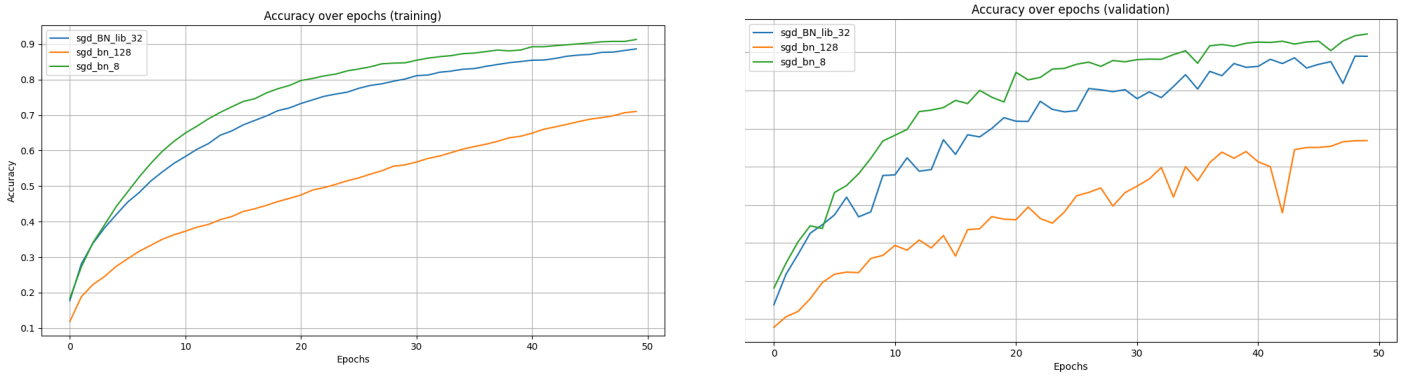
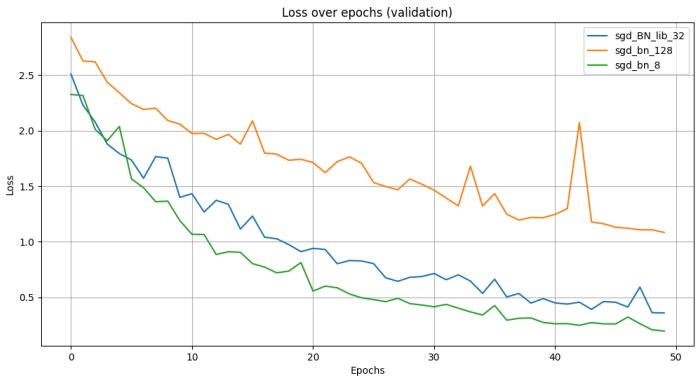


Figure 8 is the comparison of a)train and b) Validation Accuracy v/s epochs for BN with varying batch size



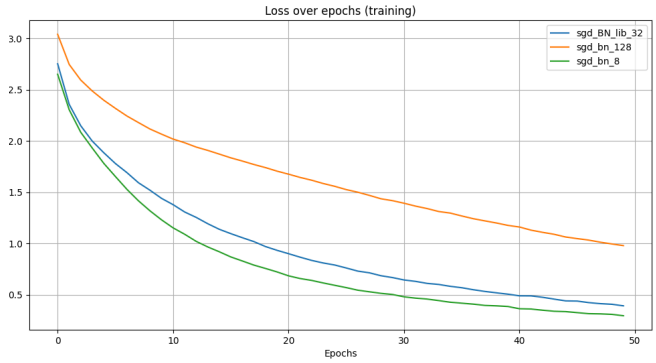


Figure 9 is the comparison of a)train and b) Validation Loss v/s epochs for BN with varying batch size

Observation and Inference:

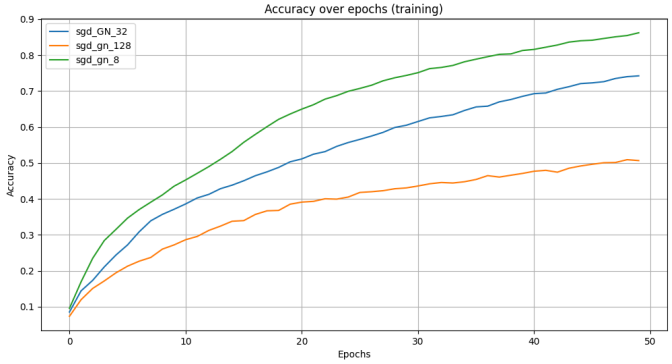
Accuracy over different Batch Size:

1. As the batch size decreases, the accuracy of the model increases by maintaining the same conditions.
2. We can confirm this as the number of updations of the weights are higher compared to others.
3. We can infer that BN with Batch size 8 has more accuracy than 32 batch and at last, 128 Batch stands.

Loss over different Batch Size:

1. Much lesser the batch size more the forward and backprop more the updation of weights this has a high chance of approaching the local or global minima .
2. 8 batch has lower loss compared with 32 and 128 and 32 performs better than 128.

2.4.2 Comparision of GN:



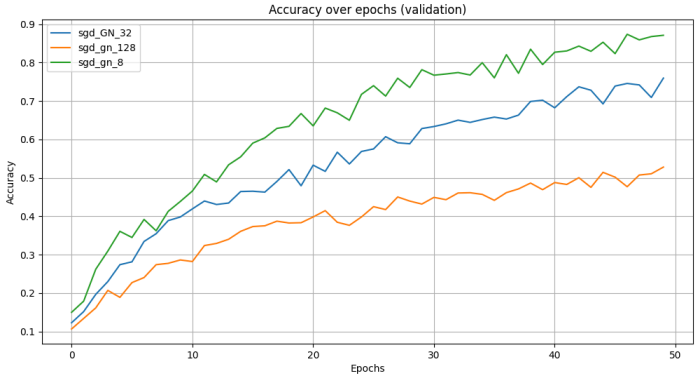


Figure 9 is the comparison of a)train and b) Validation Accuracy v/s epochs for GN with varying batch size

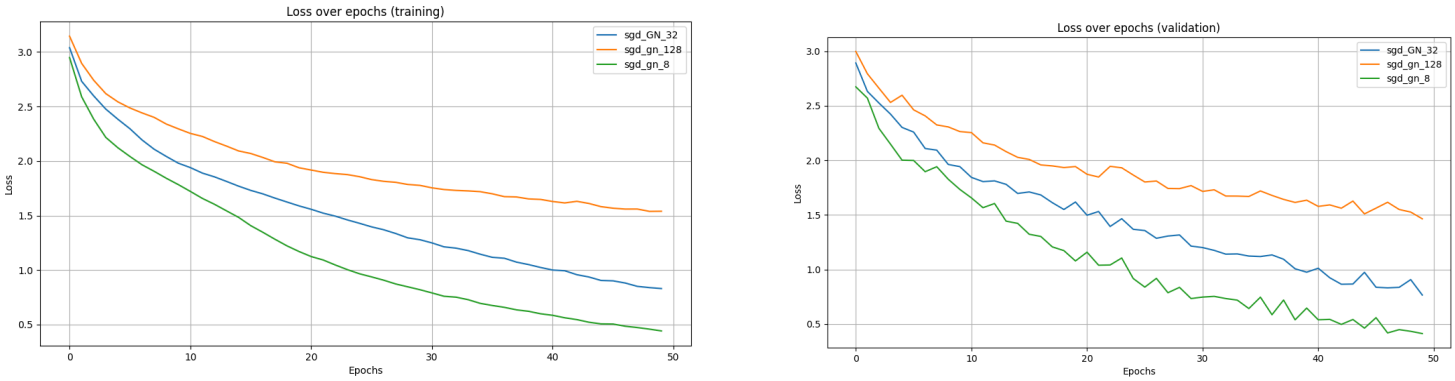


Figure 10 is the comparison of a)train and b) Validation Loss v/s epochs for GN with varying batch size

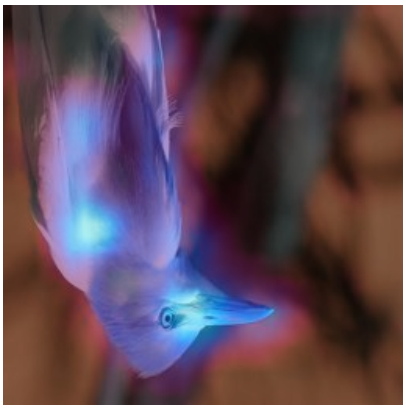
GN also performs in the same way as that of BN .

3. Visualize model's thinking :

As a part of visualization using the Gradcam Theory i have implemented the visual explanation of how the model is predicting the class which area or pixels are contributing for the class prediction this can be simply implemented by calculating the gradients of the final conv layer w.r.t to input.the classes which are stated to visualize for positive and negative class is as follows:

3.1 Cattle Egre

Positively classified:

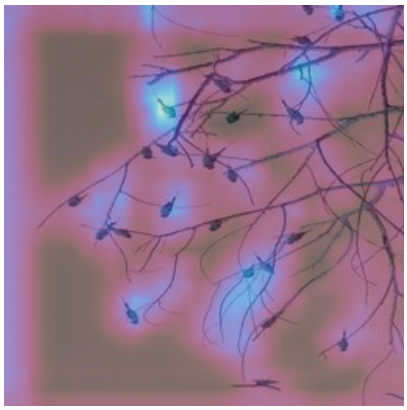


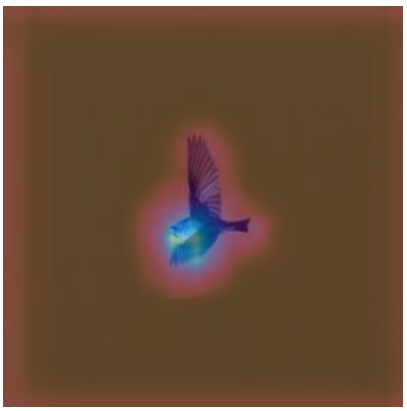
Negatively classified:

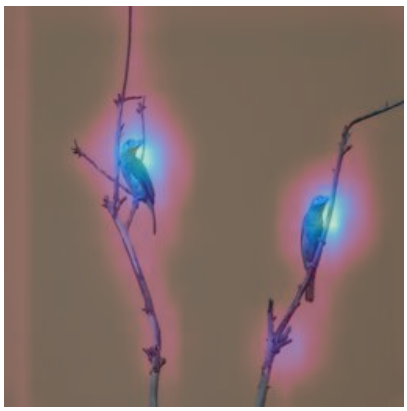
3.2

Positively classified:

Negatively classified:







3.3 Indian Peacock

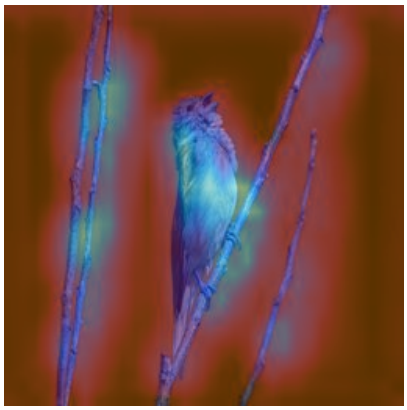
Positively classified:

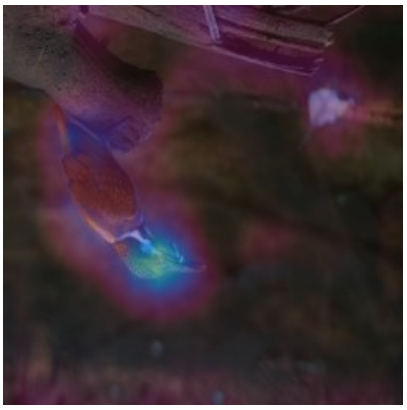




Negatively classified:

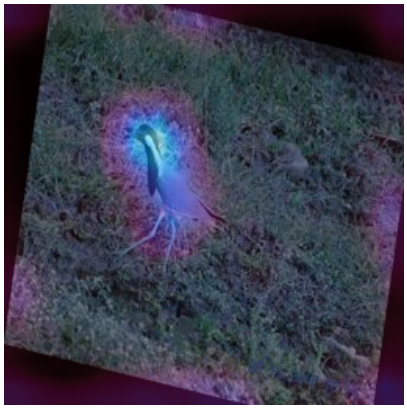


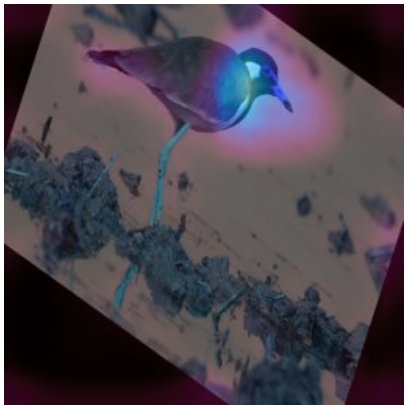


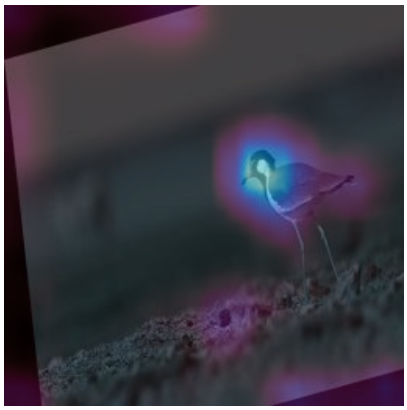


3.4 Red Wattled Lapwing

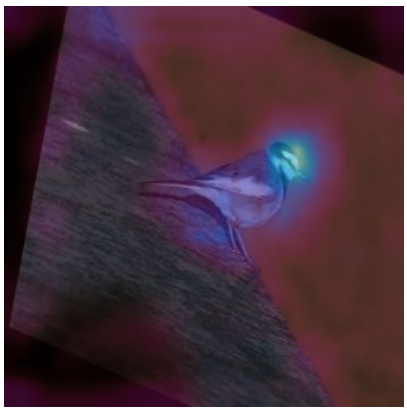
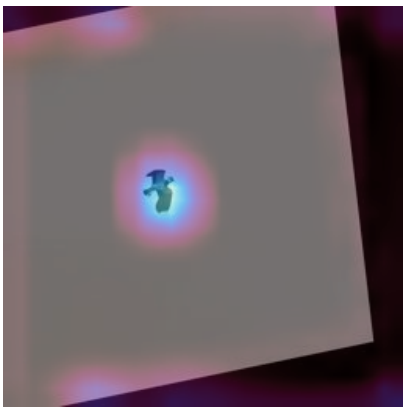
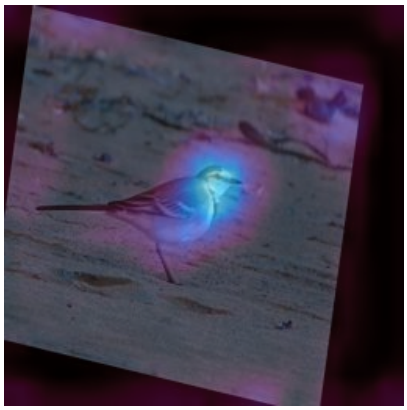
Positively classified:





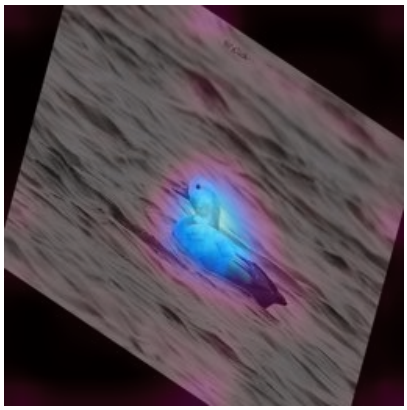
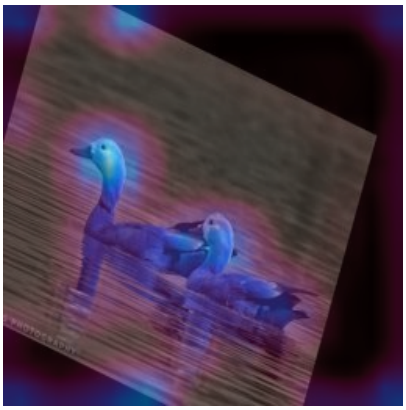


Negatively classified:

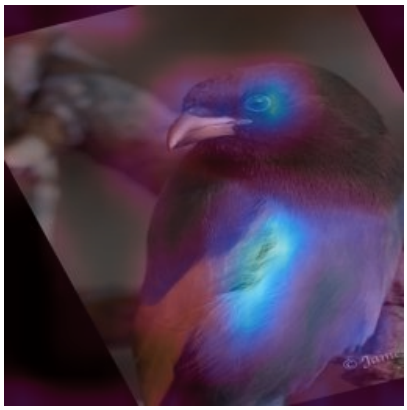




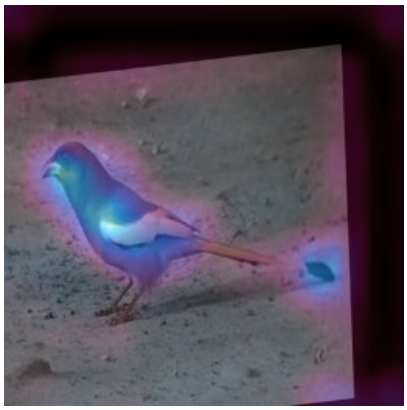
3.5 Ruddy Shel-duck

Positively classified:

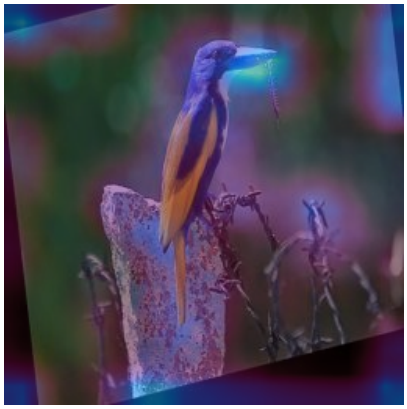
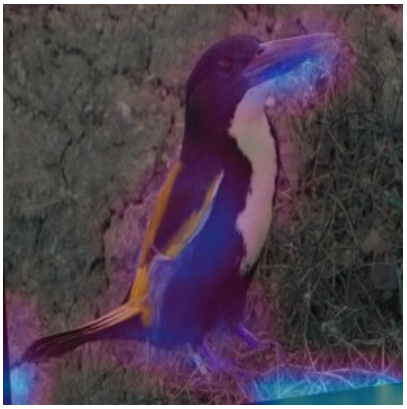

Negatively classified:

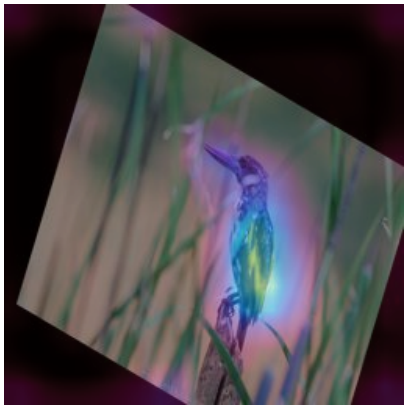




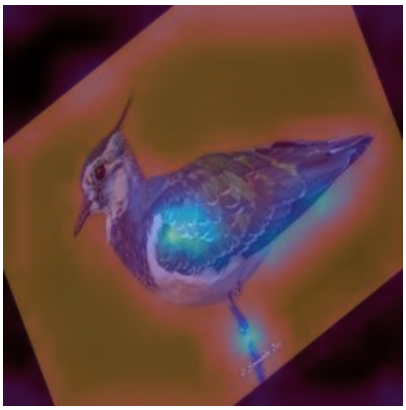
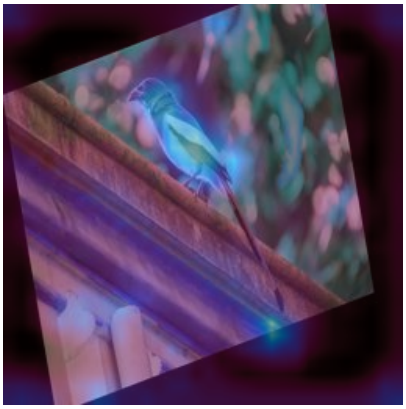
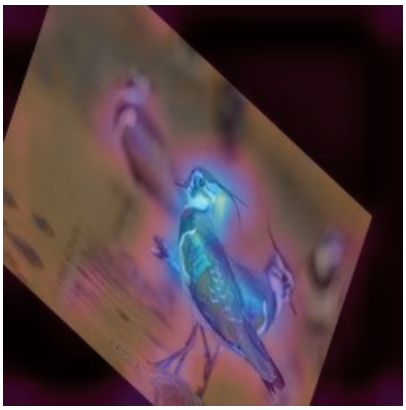
3.6 White Breasted Kingsher

Positively classified:

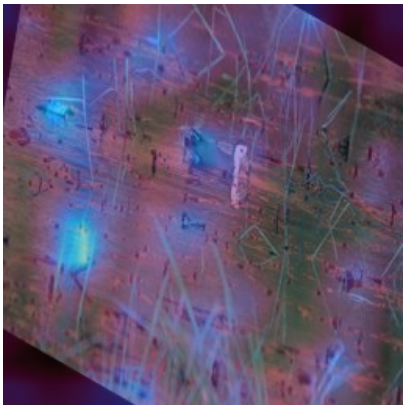
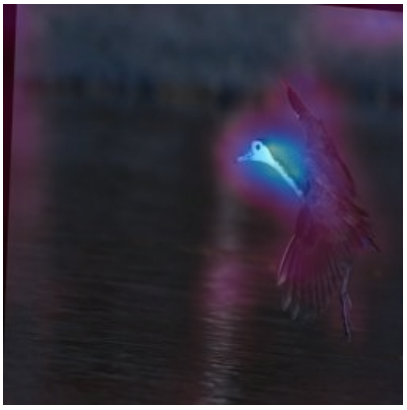
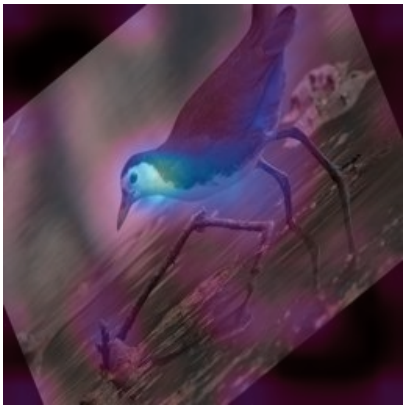
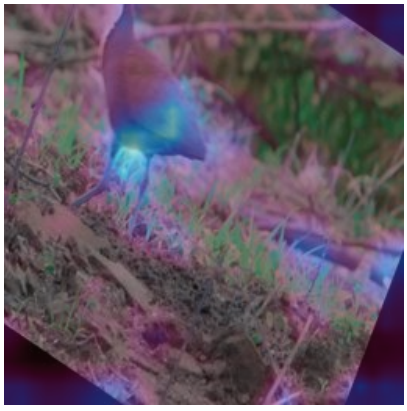



Negatively classified:

3.7 White Breasted Waterhen

Positively classified:


Negatively classified:



Observations:

From the Visualization of gradcam we can infer that the pixels which have the high gradients in the last layer of conv will effect th eoutput class.