# Normalizations in Deep Learning

Harman Singh Shubham Mittal

## Image Classification using Residual Network

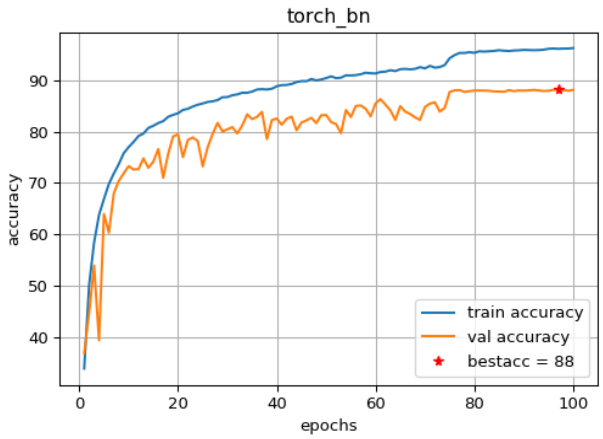
For this part a 14 layer ResNet model with batchnorm layers (of pytorch) is trained and validated on 40k and 10k samples from CIFAR10 train dataset respectively.

1. The model is trained for 100 epochs, starting with a learning rate of 0.1 which is decreased by a factor of 10 at 75th and 95th epoch.
2. Cross entropy loss and SGD optimizer is used with a batch size of 128.
3. For data augmentation on training data, random crop and random horizontal flip is used (as per the paper)

**Statistics on Train, Val and Test splits :**

| | Accuracy | Micro F1 | Macro F1 |
|---|---|---|---|
| **Train** | 0.970 | 0.970 | 0.970 |
| **Validation** | 0.881 | 0.881 | 0.880 |
| **Test** | 0.879 | 0.879 | 0.879 |

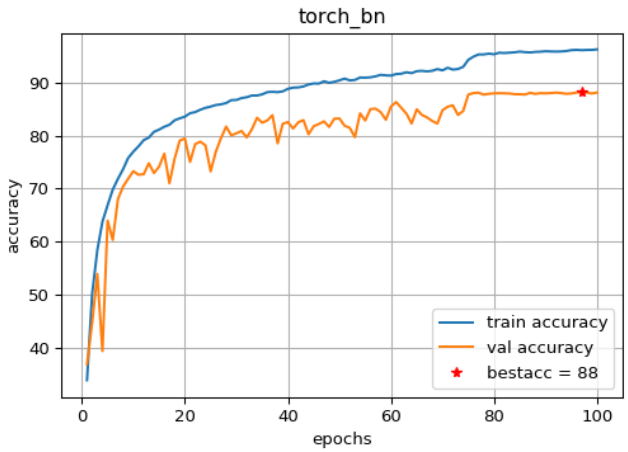
**Accuracy Curves for train and val data**

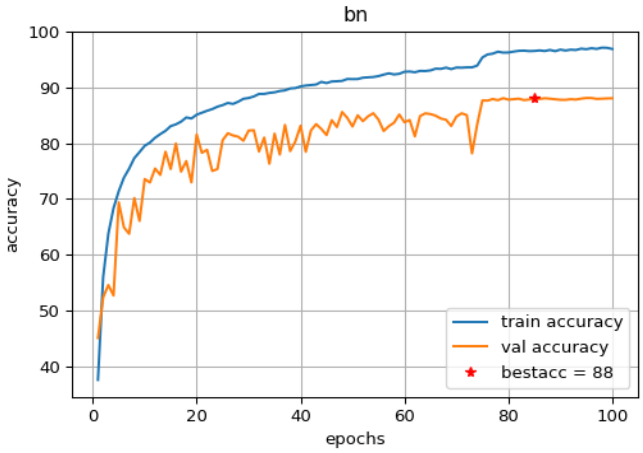



## Impact of Normalization

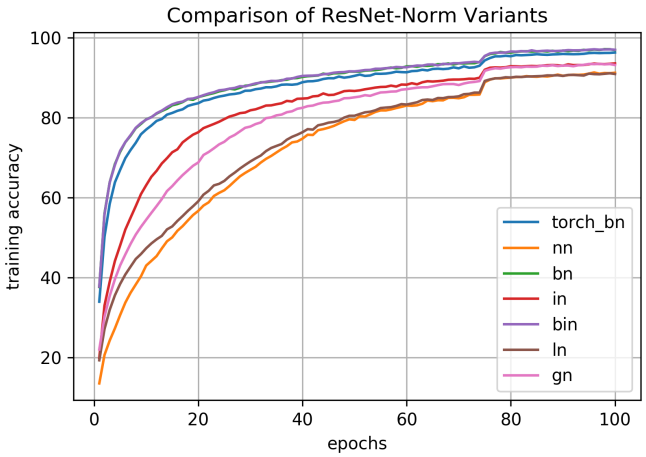
In this part, various normalization schemes are used (implemented from scratch in pytorch). The experimental details are the same (learning rate schedule, optimizer, loss function, batch size, etc) as above.

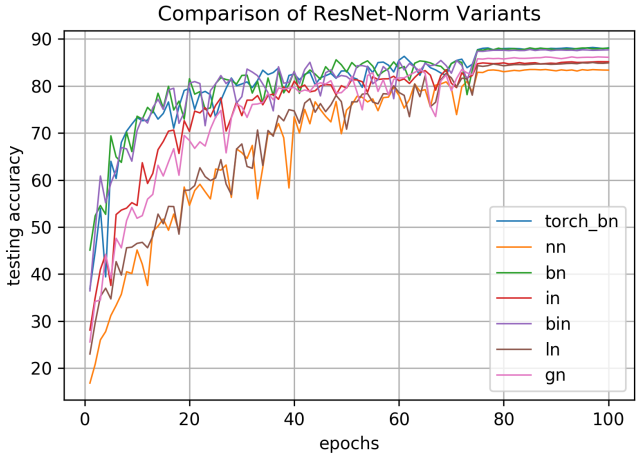
1. **Comparison of torch Batch Norm and our Batch Norm** is shown below. We can see the plots are almost identical, also the validation accuracies are the same. The performance statistics of both are given along with other norm-variants.




2. **Comparing all 7 variants of ResNet Models** : torch batch norm, no norm, batch norm, instance norm, batch-instance norm, layer norm, and group norm.




| Variant | Train | | | Validation | | | Test | | |
|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | MicroF1 | MacroF1 | Accuracy | MicroF1 | MacroF1 | Accuracy | MicroF1 | MacroF1 |
| Torch_BN | 0.970 | 0.970 | 0.970 | 0.881 | 0.881 | 0.880 | 0.879 | 0.879 | 0.879 |
| NN | 0.915 | 0.915 | 0.915 | 0.833 | 0.833 | 0.849 | 0.849 | 0.849 | 0.848 |
| BN | 0.977 | 0.977 | 0.977 | 0.880 | 0.880 | 0.880 | 0.878 | 0.878 | 0.877 |
| IN | 0.923 | 0.923 | 0.923 | 0.851 | 0.851 | 0.851 | 0.847 | 0.847 | 0.847 |
| BIN | 0.977 | 0.977 | 0.977 | 0.876 | 0.876 | 0.877 | **0.881** | **0.881** | **0.881** |
| LN | 0.911 | 0.911 | 0.911 | 0.849 | 0.849 | 0.849 | 0.849 | 0.849 | 0.848 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| GN | 0.937 | 0.937 | 0.937 | 0.861 | 0.861 | 0.861 | 0.858 | 0.858 | 0.858 |

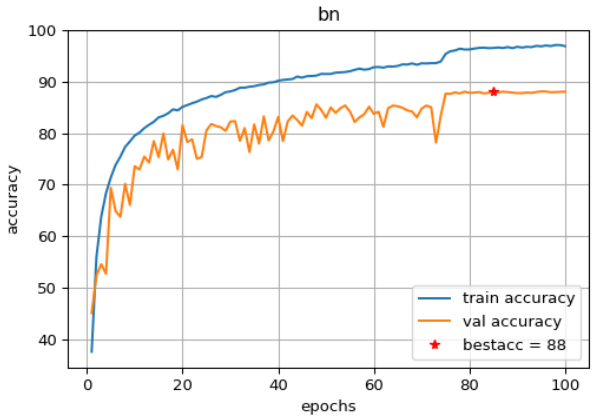
**Experimental detail:** Group norm variant requires another hyperparameter of defining the number of groups. We choose the number of groups in a norm layer, accepting *n* number of channels, as *n/4*. For example - for norm layer taking input of 16 channels, it uses 4 groups for normalisation.

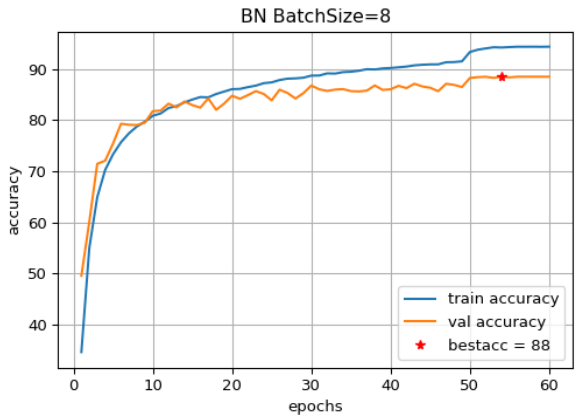
**Inferences:**

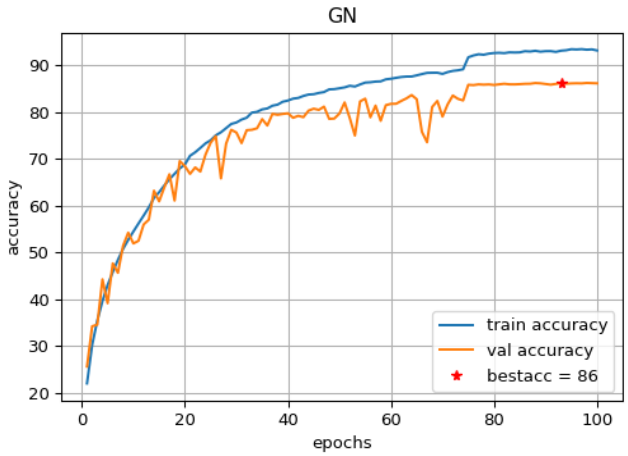
1. The accuracy plots of torch_BN and BN variants of ResNet are overlapping. Also the performance statistics are almost same. Hence the sanity check (1.2.4) is done and verified.
2. From the plots, we can see that BIN learns most rapidly i.e. in about 10 epochs it has attained about 80% train acc and 70% val acc, whereas in same number of epochs the NN variant is at 40% train and val acc.
3. The performance statistics also conclude that the BIN variant performs the best.

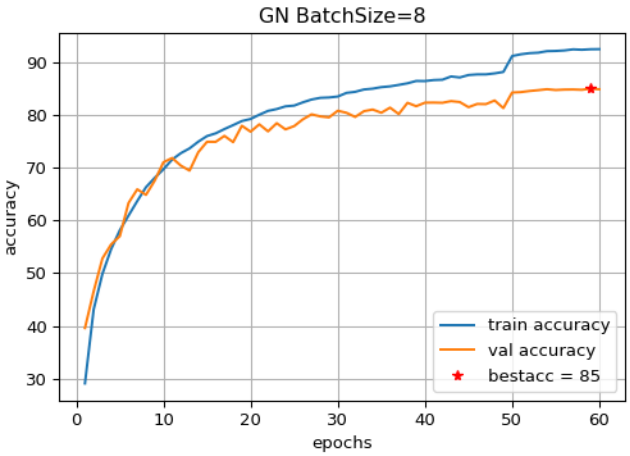
## 3. Impact of Batch Size

Batch normalisation and group normalisation are compared: left- batch size = 128, right- batch size=8



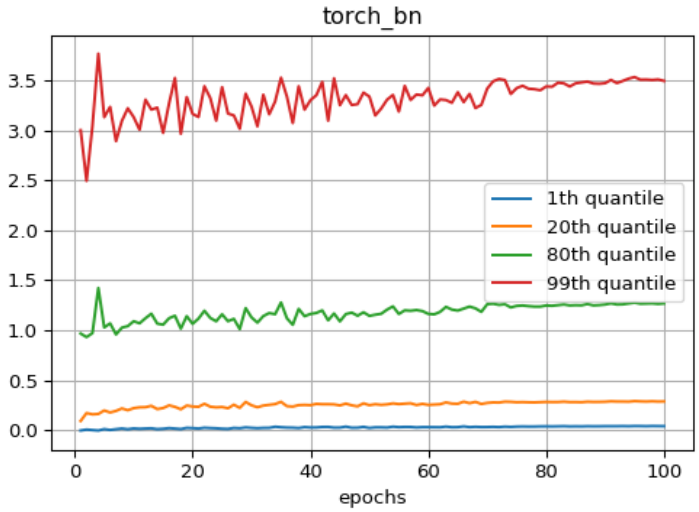



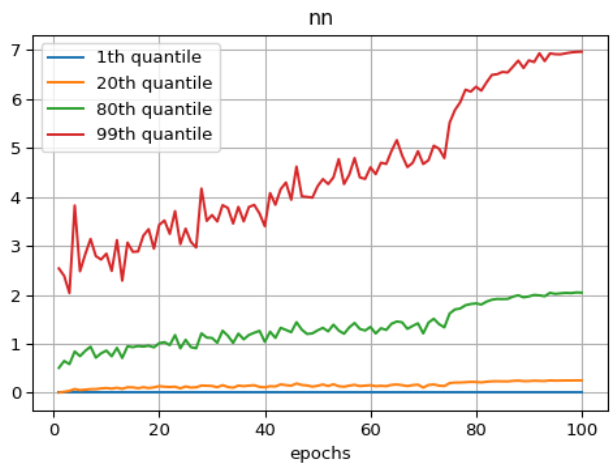

**Inferences/Comments:**

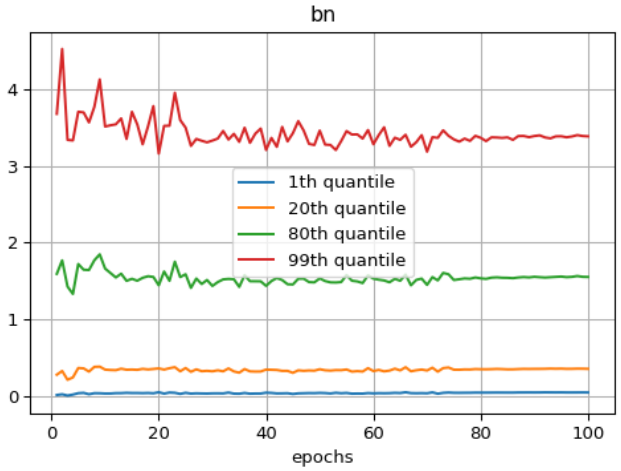
1. The training time substantially increased by almost 6 times when batch size reduced from 128 to 8.

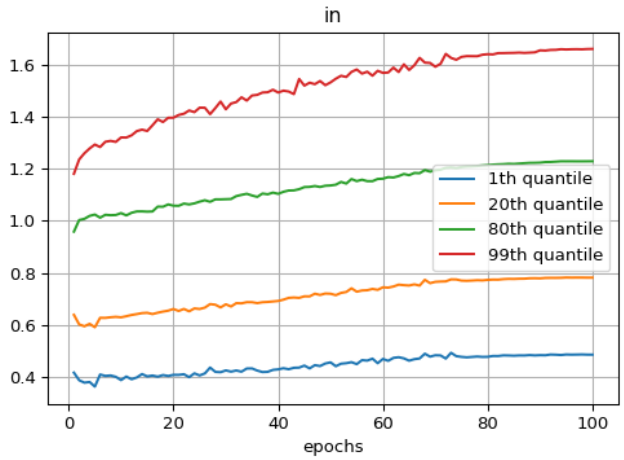
2. To save time, we early stop at the 60th epoch in right plots.
3. The learning curves clearly show that Group norm is almost invariant to batch size. To our surprise, the batchnorm is also showing no degradation in performance even when batch size is reduced. This need further investigation but our intuitions says that when the batch size is less, the learning will take more time, but the dataset is probably such that the convergence has taken place even with small batch size, and this convergence may have taken place may have been at a similar position in the loss space giving similar accuracy. This needs further investigation

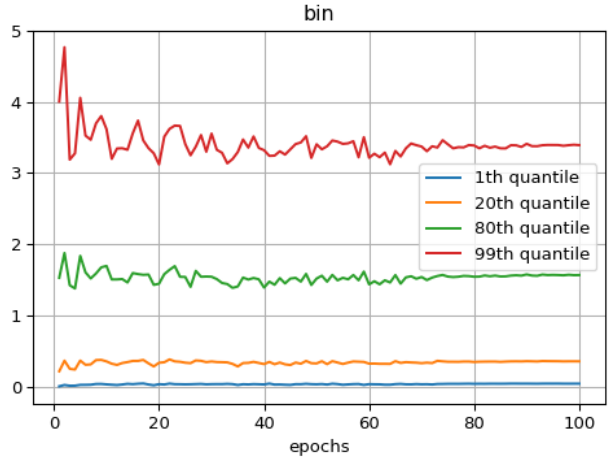
4. **Evolution of feature distributions**

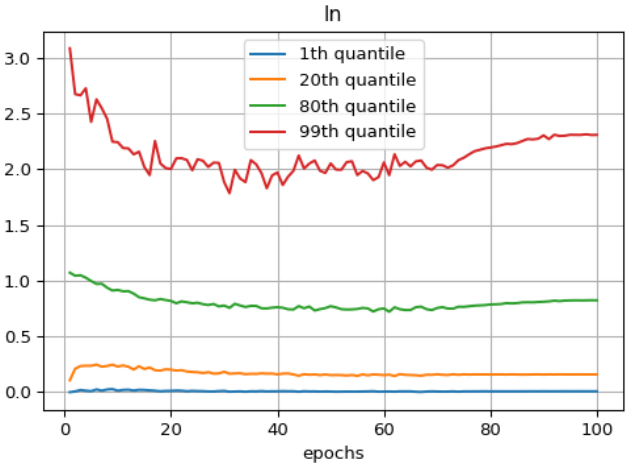


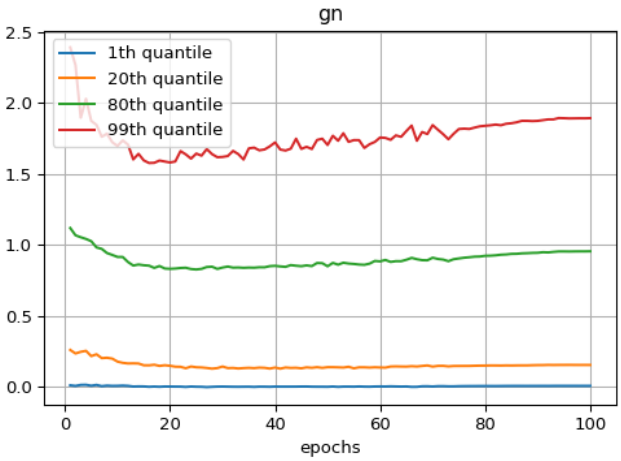






**Inferences**

1. From the above plots, we can infer/see how the models can be trained “healthily” using norm layers.
2. NN variant features are diverging with training whereas torch_bn, BN, BIN variants are stable with time.
3. Thus, we conclude that performing normalization is also helpful in controlling the distribution of features (as discussed in the group normalization paper).