# Project 6 Deep learning by PyTorch

## Part 1: Improving BaseNet on CIFAR100

### 1. Name on Kaggle and Best accuracy

The name under which I submitted on Kaggle is **Songyi Huang**. The best accuracy is **59.8%**.

### 2. Table defining the final architecture

The following table defines my final architecture.

| Layer No. | Layer Type | Kernel Size (for conv layers) | Input \| Output dimension | Input \| Output Channels (for conv layers) |
|---|---|---|---|---|
| 1 | Conv2d Padding = 1 | 3 | 32 \| 32 | 3 \| 64 |
| 2 | BatchNorm2d | - | 32 \| 32 | - |
| 3 | Relu | - | 32 \| 32 | - |
| 4 | Conv2d Padding = 1 | 3 | 32 \| 32 | 64 \| 64 |
| 5 | BatchNorm2d | - | 32 \| 32 | - |
| 6 | Relu | - | 32 \| 32 | - |
| 7 | MaxPool2d | 2 | 32 \| 16 | - |
| 8 | Conv2d Padding = 1 | 3 | 16 \| 16 | 64 \| 128 |
| 9 | BatchNorm2d | - | 16 \| 16 | - |
| 10 | Relu | - | 16 \| 16 | - |
| 11 | Conv2d Padding = 1 | 3 | 16 \| 16 | 128 \| 128 |
| 12 | BatchNorm2d | - | 16 \| 16 | - |
| 13 | Relu | - | 16 \| 16 | - |
| 14 | MaxPool2d | 2 | 16 \| 8 | - |
| 15 | Conv2d Padding = 1 | 3 | 8 \| 8 | 128 \| 256 |
| 16 | BatchNorm2d | - | 8 \| 8 | - |
| 17 | Relu | - | 8 \| 8 | - |
| 18 | Conv2d Padding = 1 | 3 | 8 \| 8 | 256 \| 256 |
| 19 | BatchNorm2d | - | 8 \| 8 | - |
| 20 | Relu | - | 8 \| 8 | - |
| 21 | MaxPool2d | 2 | 8 \| 4 | - |
| 22 | Conv2d Padding = 1 | 3 | 4 \| 4 | 256 \| 512 |
| 23 | BatchNorm2d | - | 4 \| 4 | - |
| 24 | Relu | - | 4 \| 4 | - |
| 25 | Conv2d Padding = 1 | 3 | 4 \| 4 | 512 \| 512 |
| 26 | BatchNorm2d | - | 4 \| 4 | - |

| | | | | |
|---|---|---|---|---|
| 27 | Relu | - | 4 \| 4 | - |
| 28 | MaxPool2d | 2 | 4 \| 2 | - |
| 29 | Conv2d<br>Padding = 1 | 3 | 2 \| 2 | 512 \| 512 |
| 30 | BatchNorm2d | - | 2 \| 2 | - |
| 31 | Relu | - | 2 \| 2 | - |
| 32 | Conv2d<br>Padding = 1 | 3 | 2 \| 2 | 512 \| 512 |
| 33 | BatchNorm2d | - | 2 \| 2 | - |
| 34 | Relu | - | 2 \| 2 | - |
| 35 | Linear | - | 2048 \| 4096 | - |
| 36 | BatchNorm1d | - | 4096 \| 4096 | - |
| 37 | Relu | - | 4096 \| 4096 | - |
| 38 | Linear | - | 4096 \| 2048 | - |
| 39 | BatchNorm1d | - | 2048 \| 2048 | - |
| 40 | Relu | - | 2048 \| 2048 | - |
| 41 | Linear | - | 2048 \| 1024 | - |
| 42 | BatchNorm1d | - | 1024 \| 1024 | - |
| 43 | Relu | - | 1024 \| 1024 | - |
| 44 | Linear | - | 1024 \| 100 | - |

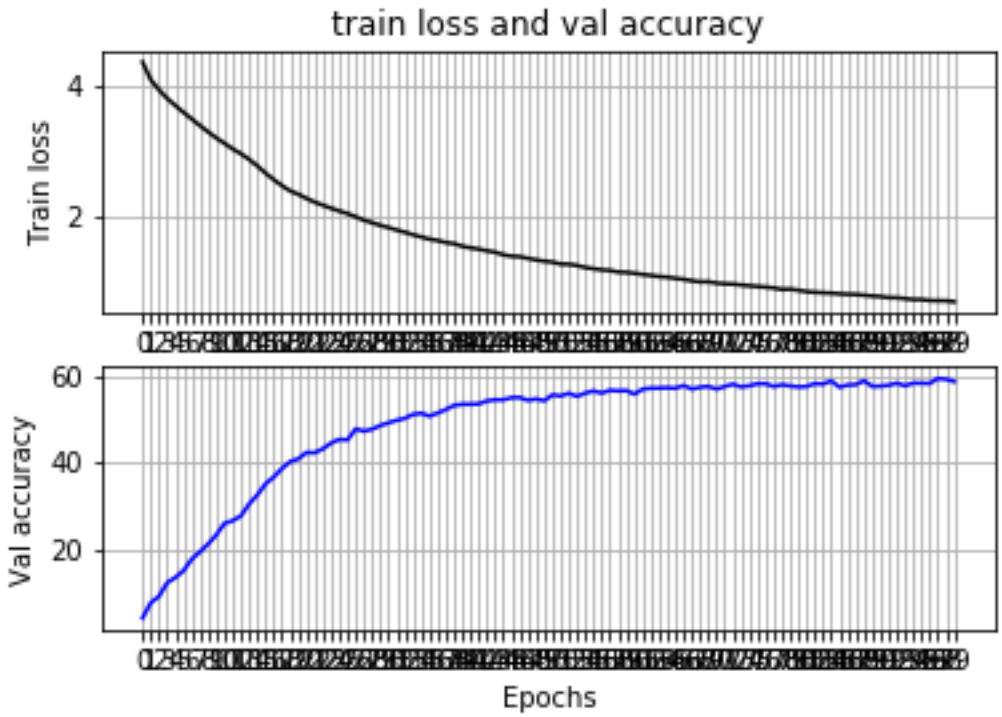
## 3. Factors which help improve the model

The factors which helped improve my model performance are as follow:

1. **Data normalization**. I normalized train and test dataset to zero mean and standard deviation with sigma = 1 to help me gain a robust and easier training process.
2. **Data augmentation**. To have more training data, I did data augmentation with ColorJitter, RandomHorizontalFlip, RandomRotation and RandomCrop.
3. **Deeper network**. I modified the BaseNet class, added 8 more Conv2d layers, 2 more MaxPool2d layer and 2 more Linear layers to make the network gain larger learning capacity.
4. **Normalization layers**. I added BatchNorm2d layer after every Conv2d layer and BatchNorm1d layer after every Linear layer (but not the last Linear layer) to help reduce overfitting and make the training converge faster.
5. **Early stopping**. After doing experiments with different epochs, I found that training the network for 100 epochs is a good choice for my final network structure, with which the validation acc converges while the model hasn't overfitting the training dataset.
6. **Change optimizer**. The default optimizer SGD is replaced by Adam with learning rate set to 0.005.

## 4. Final architecture's plot for training loss and validation accuracy

The following image is the plot for training loss and validation accuracy.




### 5. Ablation study

Making the network deeper leads to the most significant accuracy improvement. Before making the network deeper, I have done data normalization, data augmentation and added normalization layers. The accuracy on Kaggle for the model with these modifications is 22.6%. But after changing the network structure to make it deeper, the accuracy on Kaggle is boosted to 55.9%.

## Part 2: Transfer Learning

### 1. Report the train and test accuracy achieved by using the ResNet as a fixed feature extractor vs. fine-tuning the whole network

The accuracy on the ResNet as a fixed feature extractor is 64.20% on the train dataset and 45.27% on the test dataset.

The accuracy on the ResNet with fine-tuning the whole network is 81.67% on the train dataset and 57.63% on the test dataset.

### 2. Report any hyperparameter settings you used (batch_size, learning_rate, resnet_last_only, num_epochs)

To reduce the gap between training and testing accuracy, I added an extra dropout layer before the last fully connected layer.

The hyperparameter settings for the ResNet as a fixed feature extractor are batch_size 32, learning_rate 0.003, resnet_last_only True, num_epochs 30.

The hyperparameter settings for the ResNet with fine-tuning the whole network are batch_size 32, learning_rate 0.001, resnet_last_only False, num_epochs 20.