Part 1:

In this part, we needed the modified baseline to train the data. In this regard, first the data has augmented using the following functions:

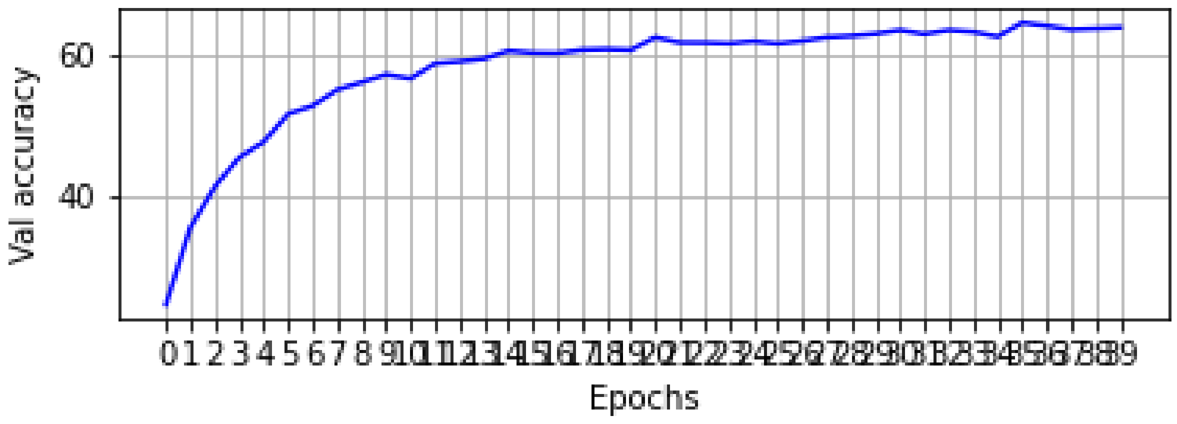
```
  transforms.RandomCrop(32),
transforms.RandomHorizontalFlip(),transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5)).
```

Then the network designed as a follow which has inspired from Alexnet work.:

Layer number	layer type	kernel size	Input|Output dimension	Input|Output Channels
1	Conv2d	5	32|32	3|32
2	Norm2d	-	32|32	-
3	Relu	-	32|32	-
4	Conv2d	5	32|32	32|64
5	Norm2d	-	32|32	-
6	Relu	-	32|32	-
7	Maxpooling	2	32|16	-
8	Conv2d	5	16|16	64|128
9	norm2d	-	16|16	-
10	relu	-	16|16	-
11	conv2d	5	16|16	128|256
12	Norm2d	-	16|16	-
13	relu	-	16|16	-
14	pooling	2	16|8	-
15	conv2d	5	8|8	256|512
16	norm2d	-	8|8	-
17	Relu	-	8|8	-
18	Conv2d	5	8|8	512|1024
19	Norm2d	-	8|8	-
20	Relu	-	8|8	-
21	MaxPooling	2	8|4	-
22	Flatten	-	-	-
23	Linear	-	10384|4096	-
24	Norm1d	-	4096|4096	-
25	Relu	-	4096|4096	-
26	Linear	-	4096|1024	-
27	Norm1d	-	1024|1024	-
28	Linear	-	1024|512	-
29	Norm1d	-	512|512	-
30	Relu	-	512|512	-
31	Linear	-	512|100	-

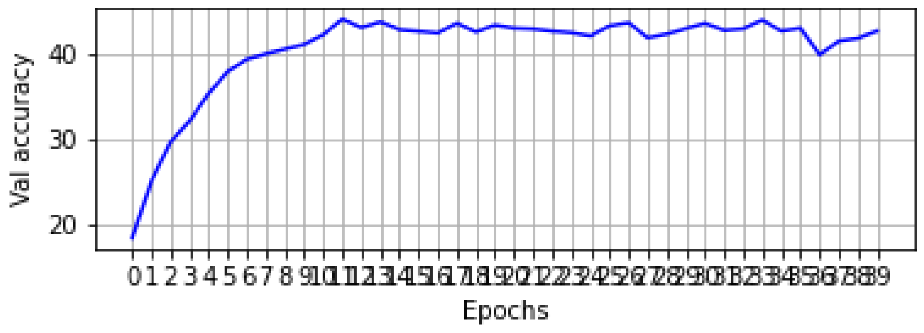
Moreover, the L1norm regularization has added to the model to avoid overfitting, however, it has been realized that this term reduced the Validation accuracy, so it removed. The epoch has increased to 40, the learning rate set to 0.001 and the momentum didn't change since 0.9 is the best value for momentum.

The obtained accuracy result for the validation accuracy over 15 epochs is 60% and over 40 epochs is 64%.



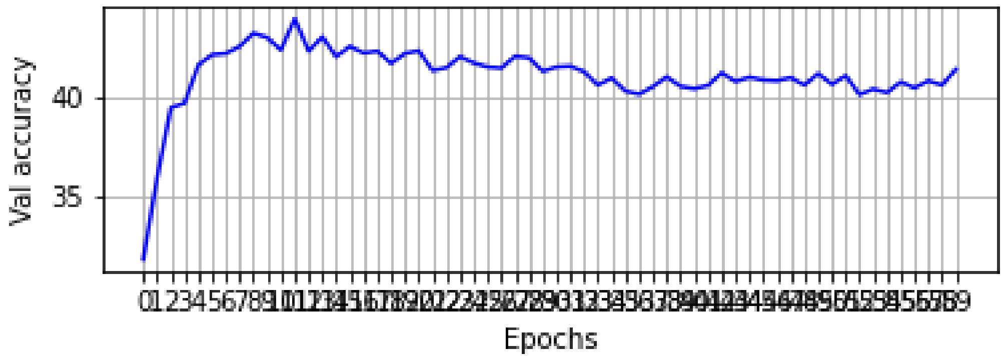
Ablation Study:

The first thing I tried was making a new dataset with the combination of the augmented dataset and actual dataset to have more data for the training but I did not see any improvement so I removed this part. Then I tried different Augmentation methods like rotation, random-cropping and normalization, figuring out that by using the rotation, the accuracy decreases while cropping and flipping helps improving the validation accuracy. Also, I tried to normalize data with the Mean and Std of the Train data which decreased the accuracy, so I chose the other values ((0.5,0.5,0.5) as a mean and STD of normalized data. In addition, my pervious design was without padding and the best accuracy I could get with different architecture was 45% on Validation data and 42% on test data:



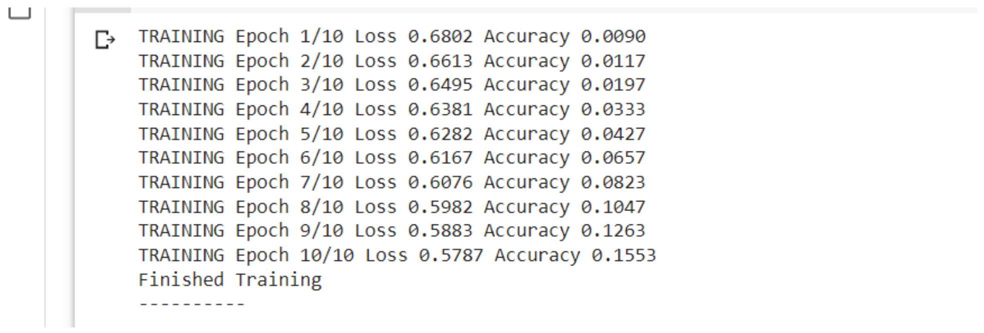
After adding padding layer, I could get over 64% accuracy on Validation and 62.2% accuracy on Test data. The other things I changed were learning rate, Batch numbers, and the kernel size.

I also tried to use the dept-wise 2d convolution, inspiring from the EffNet structure which has shown the best performance for image classification [1]. By doing so, although at first the accuracies were so great, after a while no improvement has seen and the test accuracy was 40% with this structure. I'm still working on this structure to make an improvement as the initial accuracy is so good and it seems that the loss function has trapped in the local minimum and I need to solve this issue with regularization or changing the initial values. The better calculated accuracies will upload on the Kaggle competition if I got the better results. For now, the calculated accuracy with my method is 62% on Test data which has uploaded to the Kaggle competition (with name **Mahdie Ghane**). The following figure shows the accuracy calculated using the dept-wise 2d convolution layer:

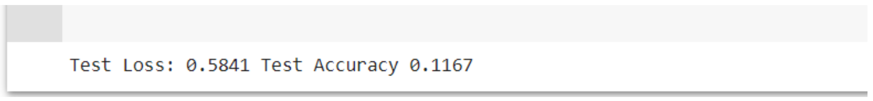


Part 2:

First, the resnet pertained model has loaded to the code and the last layer of it has changed. As it has written in the instruction, in this step, we don't need to change the weight of the pre-trained model. The obtained accuracy on Training data and Test data in this step is 15.53 % and 11.67% respectively.

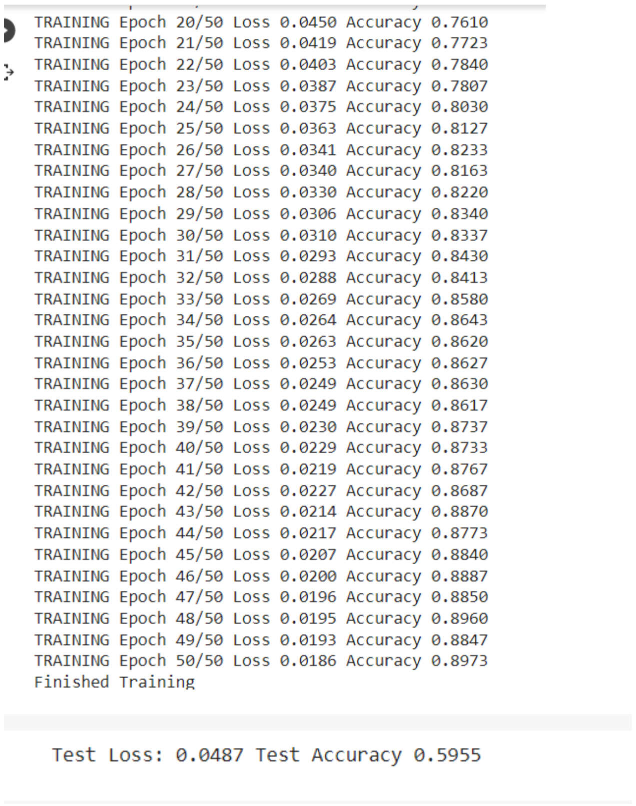


```
TRAINING Epoch 1/10 Loss 0.6802 Accuracy 0.0090
TRAINING Epoch 2/10 Loss 0.6613 Accuracy 0.0117
TRAINING Epoch 3/10 Loss 0.6495 Accuracy 0.0197
TRAINING Epoch 4/10 Loss 0.6381 Accuracy 0.0333
TRAINING Epoch 5/10 Loss 0.6282 Accuracy 0.0427
TRAINING Epoch 6/10 Loss 0.6167 Accuracy 0.0657
TRAINING Epoch 7/10 Loss 0.6076 Accuracy 0.0823
TRAINING Epoch 8/10 Loss 0.5982 Accuracy 0.1047
TRAINING Epoch 9/10 Loss 0.5883 Accuracy 0.1263
TRAINING Epoch 10/10 Loss 0.5787 Accuracy 0.1553
Finished Training
----------
```



```
Test Loss: 0.5841 Test Accuracy 0.1167
```

In the next step, I changed the learning rate to be 0.001, increased the epoch to 50,and set the Batch size to be 32. I also did the augmentation (Normalization, Crop and HorizotalFlip) and L1 regularization to avoid overfitting but the L1 regularization caused accuracy decrement, so I removed it. The Normalization values are: Mean: `(0.48, 0.456, 0.406)`, Std: `(0.229, 0.224, 0.225)` based on the papers reported best performance of implementation Resnet18 on this data. Moreover, the Resnet_Last_only parameter changed to False to be able to adjust pertained weights. By doing so, the training accuracy reached 89.73% but the test accuracy is still 59.55%:



```
TRAINING Epoch 20/50 Loss 0.0450 Accuracy 0.7610
TRAINING Epoch 21/50 Loss 0.0419 Accuracy 0.7723
TRAINING Epoch 22/50 Loss 0.0403 Accuracy 0.7840
TRAINING Epoch 23/50 Loss 0.0387 Accuracy 0.7807
TRAINING Epoch 24/50 Loss 0.0375 Accuracy 0.8030
TRAINING Epoch 25/50 Loss 0.0363 Accuracy 0.8127
TRAINING Epoch 26/50 Loss 0.0341 Accuracy 0.8233
TRAINING Epoch 27/50 Loss 0.0340 Accuracy 0.8163
TRAINING Epoch 28/50 Loss 0.0330 Accuracy 0.8220
TRAINING Epoch 29/50 Loss 0.0306 Accuracy 0.8340
TRAINING Epoch 30/50 Loss 0.0310 Accuracy 0.8337
TRAINING Epoch 31/50 Loss 0.0293 Accuracy 0.8430
TRAINING Epoch 32/50 Loss 0.0288 Accuracy 0.8413
TRAINING Epoch 33/50 Loss 0.0269 Accuracy 0.8580
TRAINING Epoch 34/50 Loss 0.0264 Accuracy 0.8643
TRAINING Epoch 35/50 Loss 0.0263 Accuracy 0.8620
TRAINING Epoch 36/50 Loss 0.0253 Accuracy 0.8627
TRAINING Epoch 37/50 Loss 0.0249 Accuracy 0.8630
TRAINING Epoch 38/50 Loss 0.0249 Accuracy 0.8617
TRAINING Epoch 39/50 Loss 0.0230 Accuracy 0.8737
TRAINING Epoch 40/50 Loss 0.0229 Accuracy 0.8733
TRAINING Epoch 41/50 Loss 0.0219 Accuracy 0.8767
TRAINING Epoch 42/50 Loss 0.0227 Accuracy 0.8687
TRAINING Epoch 43/50 Loss 0.0214 Accuracy 0.8870
TRAINING Epoch 44/50 Loss 0.0217 Accuracy 0.8773
TRAINING Epoch 45/50 Loss 0.0207 Accuracy 0.8840
TRAINING Epoch 46/50 Loss 0.0200 Accuracy 0.8887
TRAINING Epoch 47/50 Loss 0.0196 Accuracy 0.8850
TRAINING Epoch 48/50 Loss 0.0195 Accuracy 0.8960
TRAINING Epoch 49/50 Loss 0.0193 Accuracy 0.8847
TRAINING Epoch 50/50 Loss 0.0186 Accuracy 0.8973
Finished Training
```

```
Test Loss: 0.0487 Test Accuracy 0.5955
```

Also some prediction for these values are:

```
class: 073.Blue_Jay predicted: 073.Blue_Jay
```

```
class: 183.Northern_Waterthrush predicted: 022.Chuck_will_Widow
```

```
class: 040.Olive_sided_Flycatcher predicted: 115.Brewer_Sparrow
```

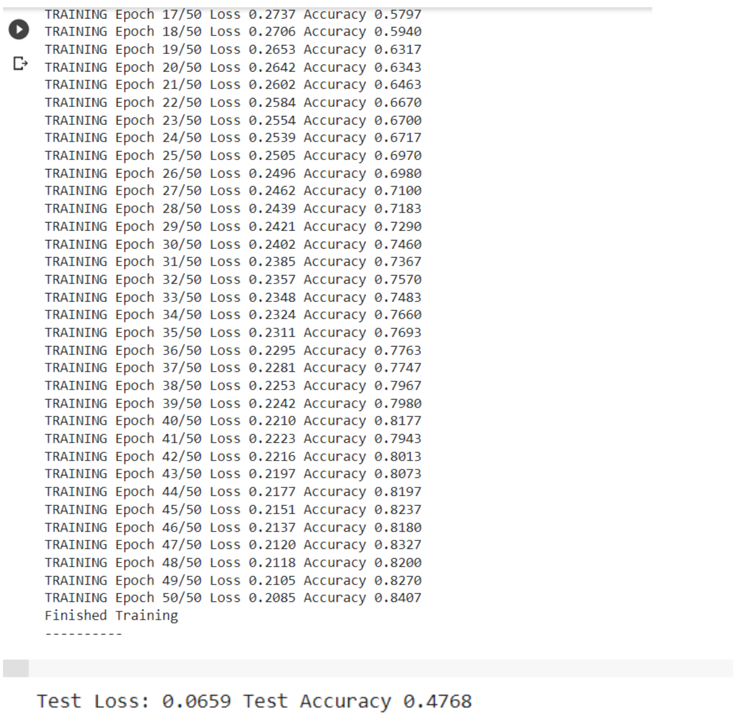
```
class: 099.Ovenbird predicted: 099.Ovenbird
```

```
class: 028.Brown_Creeper predicted: 028.Brown_Creeper
```

```
class: 025.Pelagic_Cormorant predicted: 025.Pelagic_Cormorant
```

```
class: 110.Geococcyx predicted: 110.Geococcyx
```

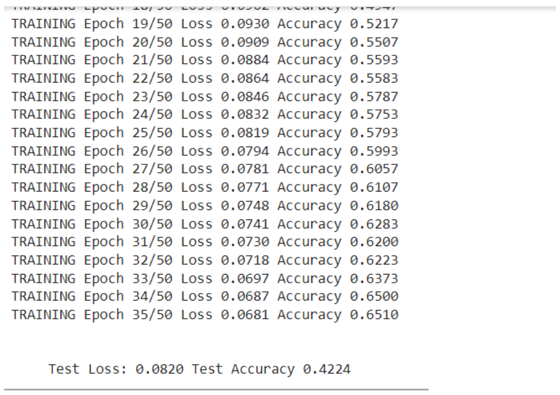
However, by setting the Resnet_Last_only to be True, which is fixed feature extractor setting, the calculated accuracies are 84% and 47% for train and test respectively:



```
TRAINING Epoch 17/50 Loss 0.2737 Accuracy 0.5797
TRAINING Epoch 18/50 Loss 0.2706 Accuracy 0.5940
TRAINING Epoch 19/50 Loss 0.2653 Accuracy 0.6317
TRAINING Epoch 20/50 Loss 0.2642 Accuracy 0.6343
TRAINING Epoch 21/50 Loss 0.2602 Accuracy 0.6463
TRAINING Epoch 22/50 Loss 0.2584 Accuracy 0.6670
TRAINING Epoch 23/50 Loss 0.2554 Accuracy 0.6700
TRAINING Epoch 24/50 Loss 0.2539 Accuracy 0.6717
TRAINING Epoch 25/50 Loss 0.2505 Accuracy 0.6970
TRAINING Epoch 26/50 Loss 0.2496 Accuracy 0.6980
TRAINING Epoch 27/50 Loss 0.2462 Accuracy 0.7100
TRAINING Epoch 28/50 Loss 0.2439 Accuracy 0.7183
TRAINING Epoch 29/50 Loss 0.2421 Accuracy 0.7290
TRAINING Epoch 30/50 Loss 0.2402 Accuracy 0.7460
TRAINING Epoch 31/50 Loss 0.2385 Accuracy 0.7367
TRAINING Epoch 32/50 Loss 0.2357 Accuracy 0.7570
TRAINING Epoch 33/50 Loss 0.2348 Accuracy 0.7483
TRAINING Epoch 34/50 Loss 0.2324 Accuracy 0.7660
TRAINING Epoch 35/50 Loss 0.2311 Accuracy 0.7693
TRAINING Epoch 36/50 Loss 0.2295 Accuracy 0.7763
TRAINING Epoch 37/50 Loss 0.2281 Accuracy 0.7747
TRAINING Epoch 38/50 Loss 0.2253 Accuracy 0.7967
TRAINING Epoch 39/50 Loss 0.2242 Accuracy 0.7980
TRAINING Epoch 40/50 Loss 0.2210 Accuracy 0.8177
TRAINING Epoch 41/50 Loss 0.2223 Accuracy 0.7943
TRAINING Epoch 42/50 Loss 0.2216 Accuracy 0.8013
TRAINING Epoch 43/50 Loss 0.2197 Accuracy 0.8073
TRAINING Epoch 44/50 Loss 0.2177 Accuracy 0.8197
TRAINING Epoch 45/50 Loss 0.2151 Accuracy 0.8237
TRAINING Epoch 46/50 Loss 0.2137 Accuracy 0.8180
TRAINING Epoch 47/50 Loss 0.2120 Accuracy 0.8327
TRAINING Epoch 48/50 Loss 0.2118 Accuracy 0.8200
TRAINING Epoch 49/50 Loss 0.2105 Accuracy 0.8270
TRAINING Epoch 50/50 Loss 0.2085 Accuracy 0.8407
Finished Training
----------
```

```
Test Loss: 0.0659 Test Accuracy 0.4768
```

and with 35 epochs, the calculated accuracies are 65% and 42% respectively:



```
TRAINING Epoch 19/50 Loss 0.0930 Accuracy 0.5217
TRAINING Epoch 20/50 Loss 0.0909 Accuracy 0.5507
TRAINING Epoch 21/50 Loss 0.0884 Accuracy 0.5593
TRAINING Epoch 22/50 Loss 0.0864 Accuracy 0.5583
TRAINING Epoch 23/50 Loss 0.0846 Accuracy 0.5787
TRAINING Epoch 24/50 Loss 0.0832 Accuracy 0.5753
TRAINING Epoch 25/50 Loss 0.0819 Accuracy 0.5793
TRAINING Epoch 26/50 Loss 0.0794 Accuracy 0.5993
TRAINING Epoch 27/50 Loss 0.0781 Accuracy 0.6057
TRAINING Epoch 28/50 Loss 0.0771 Accuracy 0.6107
TRAINING Epoch 29/50 Loss 0.0748 Accuracy 0.6180
TRAINING Epoch 30/50 Loss 0.0741 Accuracy 0.6283
TRAINING Epoch 31/50 Loss 0.0730 Accuracy 0.6200
TRAINING Epoch 32/50 Loss 0.0718 Accuracy 0.6223
TRAINING Epoch 33/50 Loss 0.0697 Accuracy 0.6373
TRAINING Epoch 34/50 Loss 0.0687 Accuracy 0.6500
TRAINING Epoch 35/50 Loss 0.0681 Accuracy 0.6510

     Test Loss: 0.0820 Test Accuracy 0.4224
```

Reference:

1. https://paperswithcode.com/sota/image-classification-on-cifar-100

Code:
https://colab.research.google.com/drive/1AQcnhqT3qhJWDlrc8u5dbP2GpFKIf_SL?usp=sharing