Project 2

Deep learning by PyTorch

Wenxiang He

301417521

Due date: 23:59 10/10th (2023)

(1 Free late-day used, total 4 days remain)

Team name: TeamEB

Team member: Eric chan, Wenxiang He, Borui Li

All undergraduate

Kaggle submission ID: Wenxiang He

Part 1

Final network structure

BaseNet Structure

Layer NO	Layer Type	Kernel size (For Conv layers)	Input | Output dimension	Input | Output channels (For Conv layers)
1	Conv2d	3	32|32	64
2	BatchNorm	-	32|32	-
3	ReLU	-	32|32	-
4	Residual Block	3	32|16	128
5	Residual Block	3	16|16	128
6	Residual Block	3	16|8	256
7	Residual Block	3	8|8	256
8	Residual Block	3	8|4	512
9	Residual Block	3	4|4	512
10	Linear	-	512*4*4|1024	-
11	Dropout	-	1024|1024	-
12	BatchNorm	-	1024|1024	-
13	ReLU	-	1024|1024	-
14	Linear	-	1024|512	-
15	Dropout	-	512|512	-
16	BatchNorm1d	-	512|512	-
17	ReLU	-	512|512	-
18	Linear	-	512|100	-

ResBlock Structure

let the width of the input denote by W, stride denote by S

Layer NO	Layer Type	Kernel size (For Conv layers)	Input | Output dimension	Input | Output channels (For Conv layers)
1	Conv2d	3	W | W/S	IN|OUT channels
2	BatchNorm	-	W/S | W/S	-
3	Dropout	-	W/S | W/S	-
4	ReLU	-	W/S | W/S	-
5	Conv2d	3	W/S | W/S	IN|OUT channels
6	BatchNorm	-	W/S | W/S	-
7	Dropout	-	W/S | W/S	-
8	SkipConnection	-	W | W/S	-
9	ReLU		W/S | W/S	-

Description:

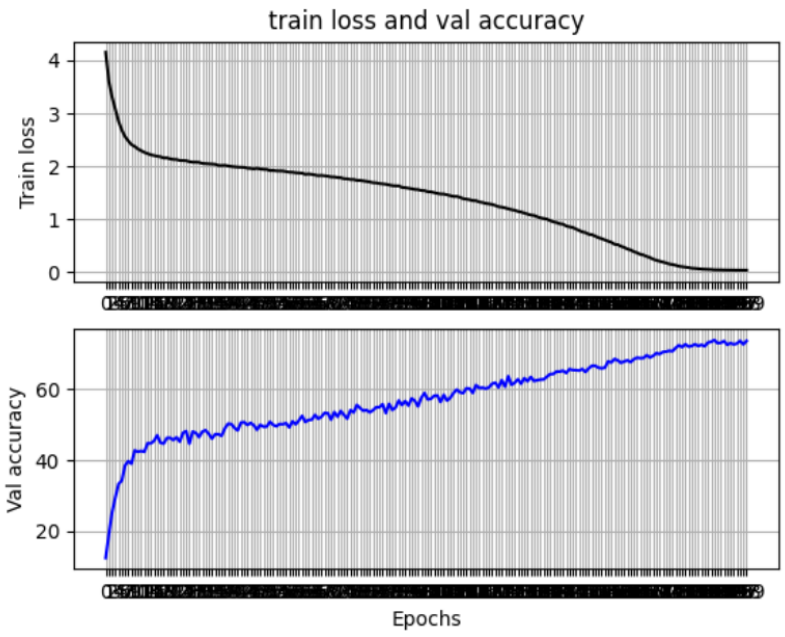
The model starts with an initial Convolutional layer using a 3x3 kernel to transform the input's channel count from 3 to 64. This is followed by Batch Normalization and a ReLU activation for normalization and non-linearity, respectively.

In the residual block, the input undergoes convolution, Batch Normalization, Dropout, and ReLU activation, potentially altering its spatial and channel dimensions. After a second similar sequence, a "shortcut" from the initial input is added to the processed feature map, which then passes through a final ReLU activation.

For the fully connected section, the feature map is flattened and processed through a Linear layer, followed by Dropout, Batch Normalization, and ReLU layers. The architecture further refines the feature vector through two Linear layers, narrowing down to 512 dimensions and finally 100, matching the number of classification classes.

Training loss and the validation accuracy

The best accuracy submitted to Kaggle is **78.3%**



As is shown in the plot, with the decrease of loss, the accuracy rises steadily until it approximately approaches 80%.

Ablation Study:

Initially, we applied data augmentation techniques like Normalize, RandomCrop, and HorizontalFlip to our model, boosting accuracy to 35% with mean and std set to 0 and 1. By restructuring the network to a sequence of Conv -> BatchNorm -> RELU operations, the accuracy improved to 45%. Fine-tuning hyperparameters like learning rate, momentum, and weight decay further elevated the accuracy to 51%.

Incorporating the lr_scheduler.ReduceLROnPlateau, which reduces the learning rate upon plateauing of loss, brought accuracy to 54%. Recognizing overfitting in deeper architectures, we integrated the ResNet model with a single residual block, referencing this guide. This adaptation significantly raised accuracy to 61%.

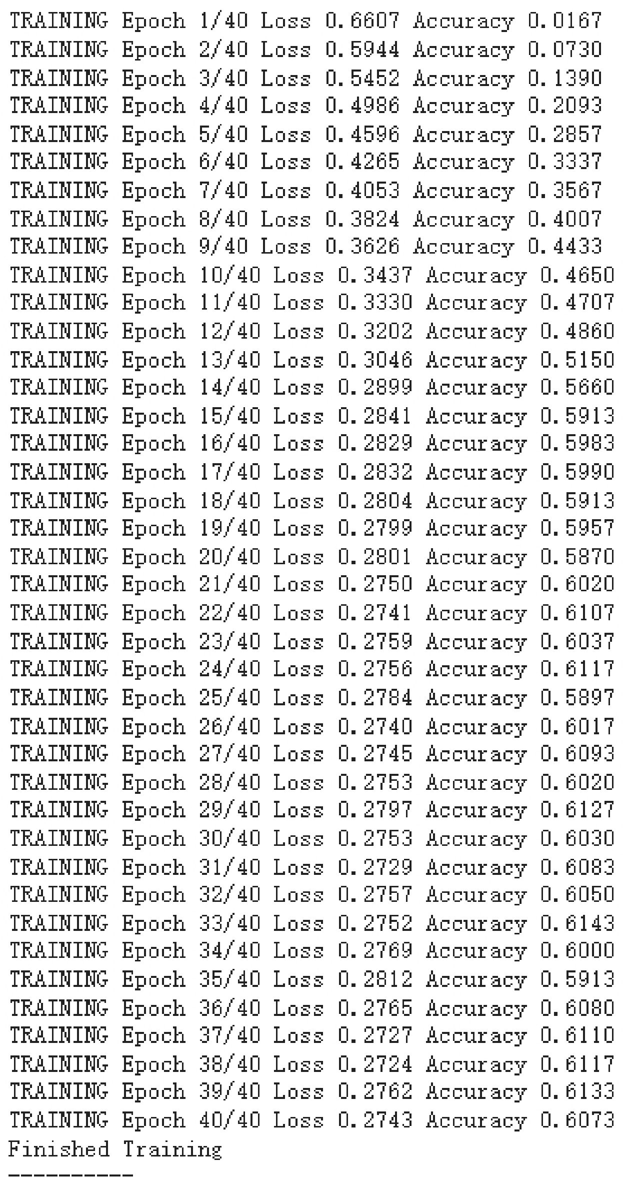
Switching to lr_scheduler.CosineAnnealingLR, which decreases the learning rate in a cosine manner throughout training, we observed a jump to 65% accuracy. With deeper residual blocks, CosineAnnealingLR, and extended training (from 30 to 200 epochs), the model's accuracy progressively increased, ultimately achieving 75% at 200 epochs.

Part 2

Train and test accuracy

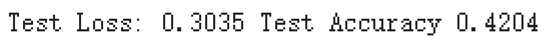
At first only fine tune the fc layer

Obtained 60.7% accuracy on the train set in 40 epochs.



```
TRAINING Epoch 1/40 Loss 0.6607 Accuracy 0.0167
TRAINING Epoch 2/40 Loss 0.5944 Accuracy 0.0730
TRAINING Epoch 3/40 Loss 0.5452 Accuracy 0.1390
TRAINING Epoch 4/40 Loss 0.4986 Accuracy 0.2093
TRAINING Epoch 5/40 Loss 0.4596 Accuracy 0.2857
TRAINING Epoch 6/40 Loss 0.4265 Accuracy 0.3337
TRAINING Epoch 7/40 Loss 0.4053 Accuracy 0.3567
TRAINING Epoch 8/40 Loss 0.3824 Accuracy 0.4007
TRAINING Epoch 9/40 Loss 0.3626 Accuracy 0.4433
TRAINING Epoch 10/40 Loss 0.3437 Accuracy 0.4650
TRAINING Epoch 11/40 Loss 0.3330 Accuracy 0.4707
TRAINING Epoch 12/40 Loss 0.3202 Accuracy 0.4860
TRAINING Epoch 13/40 Loss 0.3046 Accuracy 0.5150
TRAINING Epoch 14/40 Loss 0.2899 Accuracy 0.5660
TRAINING Epoch 15/40 Loss 0.2841 Accuracy 0.5913
TRAINING Epoch 16/40 Loss 0.2829 Accuracy 0.5983
TRAINING Epoch 17/40 Loss 0.2832 Accuracy 0.5990
TRAINING Epoch 18/40 Loss 0.2804 Accuracy 0.5913
TRAINING Epoch 19/40 Loss 0.2799 Accuracy 0.5957
TRAINING Epoch 20/40 Loss 0.2801 Accuracy 0.5870
TRAINING Epoch 21/40 Loss 0.2750 Accuracy 0.6020
TRAINING Epoch 22/40 Loss 0.2741 Accuracy 0.6107
TRAINING Epoch 23/40 Loss 0.2759 Accuracy 0.6037
TRAINING Epoch 24/40 Loss 0.2756 Accuracy 0.6117
TRAINING Epoch 25/40 Loss 0.2784 Accuracy 0.5897
TRAINING Epoch 26/40 Loss 0.2740 Accuracy 0.6017
TRAINING Epoch 27/40 Loss 0.2745 Accuracy 0.6093
TRAINING Epoch 28/40 Loss 0.2753 Accuracy 0.6020
TRAINING Epoch 29/40 Loss 0.2797 Accuracy 0.6127
TRAINING Epoch 30/40 Loss 0.2753 Accuracy 0.6030
TRAINING Epoch 31/40 Loss 0.2729 Accuracy 0.6083
TRAINING Epoch 32/40 Loss 0.2757 Accuracy 0.6050
TRAINING Epoch 33/40 Loss 0.2752 Accuracy 0.6143
TRAINING Epoch 34/40 Loss 0.2769 Accuracy 0.6000
TRAINING Epoch 35/40 Loss 0.2812 Accuracy 0.5913
TRAINING Epoch 36/40 Loss 0.2765 Accuracy 0.6080
TRAINING Epoch 37/40 Loss 0.2727 Accuracy 0.6110
TRAINING Epoch 38/40 Loss 0.2724 Accuracy 0.6117
TRAINING Epoch 39/40 Loss 0.2762 Accuracy 0.6133
TRAINING Epoch 40/40 Loss 0.2743 Accuracy 0.6073
Finished Training
----------
```

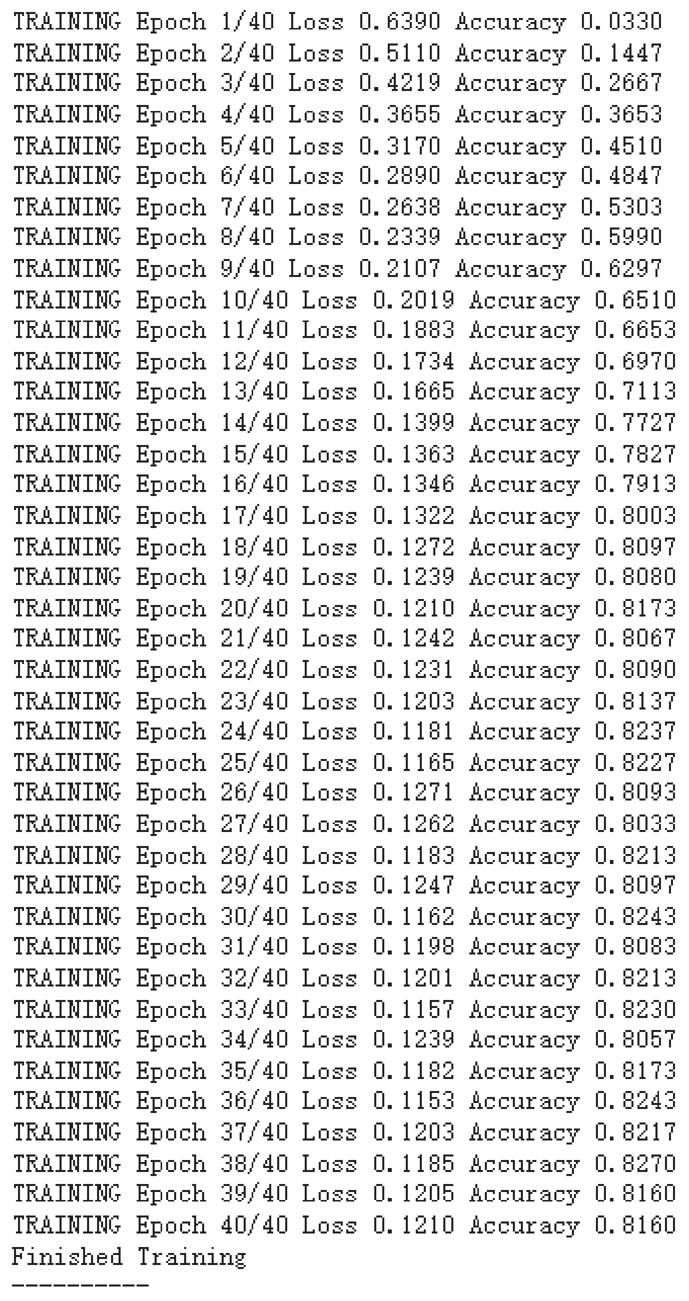
Obtained 42.04% accuracy on the test set.



```
Test Loss: 0.3035 Test Accuracy 0.4204
```

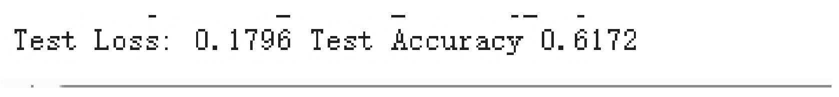
Then fine tune the entire network.

Obtained 81.6% accuracy on the train set in 40 epochs.



```
TRAINING Epoch 1/40 Loss 0.6390 Accuracy 0.0330
TRAINING Epoch 2/40 Loss 0.5110 Accuracy 0.1447
TRAINING Epoch 3/40 Loss 0.4219 Accuracy 0.2667
TRAINING Epoch 4/40 Loss 0.3655 Accuracy 0.3653
TRAINING Epoch 5/40 Loss 0.3170 Accuracy 0.4510
TRAINING Epoch 6/40 Loss 0.2890 Accuracy 0.4847
TRAINING Epoch 7/40 Loss 0.2638 Accuracy 0.5303
TRAINING Epoch 8/40 Loss 0.2339 Accuracy 0.5990
TRAINING Epoch 9/40 Loss 0.2107 Accuracy 0.6297
TRAINING Epoch 10/40 Loss 0.2019 Accuracy 0.6510
TRAINING Epoch 11/40 Loss 0.1883 Accuracy 0.6653
TRAINING Epoch 12/40 Loss 0.1734 Accuracy 0.6970
TRAINING Epoch 13/40 Loss 0.1665 Accuracy 0.7113
TRAINING Epoch 14/40 Loss 0.1399 Accuracy 0.7727
TRAINING Epoch 15/40 Loss 0.1363 Accuracy 0.7827
TRAINING Epoch 16/40 Loss 0.1346 Accuracy 0.7913
TRAINING Epoch 17/40 Loss 0.1322 Accuracy 0.8003
TRAINING Epoch 18/40 Loss 0.1272 Accuracy 0.8097
TRAINING Epoch 19/40 Loss 0.1239 Accuracy 0.8080
TRAINING Epoch 20/40 Loss 0.1210 Accuracy 0.8173
TRAINING Epoch 21/40 Loss 0.1242 Accuracy 0.8067
TRAINING Epoch 22/40 Loss 0.1231 Accuracy 0.8090
TRAINING Epoch 23/40 Loss 0.1203 Accuracy 0.8137
TRAINING Epoch 24/40 Loss 0.1181 Accuracy 0.8237
TRAINING Epoch 25/40 Loss 0.1165 Accuracy 0.8227
TRAINING Epoch 26/40 Loss 0.1271 Accuracy 0.8093
TRAINING Epoch 27/40 Loss 0.1262 Accuracy 0.8033
TRAINING Epoch 28/40 Loss 0.1183 Accuracy 0.8213
TRAINING Epoch 29/40 Loss 0.1247 Accuracy 0.8097
TRAINING Epoch 30/40 Loss 0.1162 Accuracy 0.8243
TRAINING Epoch 31/40 Loss 0.1198 Accuracy 0.8083
TRAINING Epoch 32/40 Loss 0.1201 Accuracy 0.8213
TRAINING Epoch 33/40 Loss 0.1157 Accuracy 0.8230
TRAINING Epoch 34/40 Loss 0.1239 Accuracy 0.8057
TRAINING Epoch 35/40 Loss 0.1182 Accuracy 0.8173
TRAINING Epoch 36/40 Loss 0.1153 Accuracy 0.8243
TRAINING Epoch 37/40 Loss 0.1203 Accuracy 0.8217
TRAINING Epoch 38/40 Loss 0.1185 Accuracy 0.8270
TRAINING Epoch 39/40 Loss 0.1205 Accuracy 0.8160
TRAINING Epoch 40/40 Loss 0.1210 Accuracy 0.8160
Finished Training
----------
```

Obtained 61.72% accuracy on the test set.



```
Test Loss: 0.1796 Test Accuracy 0.6172
```

Hyperparameter settings

NUM_EPOCHS change to 40

```
NUM_EPOCHS = 40
```

LEARNING_RATE initialized as 0.001

```
LEARNING_RATE = 0.001
```

Use

```
from torch.optim.lr_scheduler import MultiStepLR as MultiStepLR
scheduler = MultiStepLR(optimizer, [13,20,30], gamma=0.1)
scheduler.step()
```

To set the learning rate to be reduced to 0.1 times in the 13th, 20th and 30th epochs respectively

Keep the BATCH_SIZE

```
BATCH_SIZE = 8
```

Keep using the pre-trained Resnet18, and fine tune the entire network.

```
RESNET_LAST_ONLY = False
```

For data augmentation, in the train set, replace CenterCrop with RandomResizedCrop, and add RandomHorizontalFlip, Normalize.

```
    'train': transforms.Compose([
        transforms.Resize(256),
        # transforms.CenterCrop(224),
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        #TODO: Transforms.RandomResizedCrop() instead of CenterCrop(), 
RandomRoate() and Horizontal Flip()
        transforms.ToTensor(),
        #TODO: Transforms.Normalize()
        transforms.Normalize((0,0,0), (1,1,1)),
    ]),
```

In the test set, add Normalize.

```
'test': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        #TODO: Transforms.Normalize()
        transforms.Normalize((0,0,0), (1,1,1)),
    ]),
```