Project2 Report

Rongsheng Qian 301449387

Team: 11111

Names of group members: Rongsheng Qian, Isaac Ding, Yuwen Jia SFU

Best accuracy: 69.6%

Part 1

1.1 Architecture Table & Describe

Layer No.	Layer type	Kernel size	stride	Padding	Input|output dimension	Input|output channels
1.	Conv2d	3	1	1	32|32	3|64
2.	BatchNorm2d	-	-	-	32|32	-
3.	ReLU	-	-	-	32|32	-
4.	Conv2d	3	1	1	32|32	64|128
5.	BatchNorm2d	-	-	-	32|32	-
6.	ReLU	-	-	-	32|32	-
7.	Conv2d	3	1	1	32|32	128|128
8.	BatchNorm2d	-	-	-	32|32	-
9.	ReLU	-	-	-	32|32	-
10.	MaxPool2d	2	2	-	32|16	-
11.	Conv2d	3	1	1	16|16	128|256
12.	BatchNorm2d	-	-	-	16|16	-
13.	ReLU	-	-	-	16|16	-
14.	Conv2d	3	1	1	16|16	256|256
15.	BatchNorm2d	-	-	-	16|16	-
16.	ReLU	-	-	-	16|16	-
17.	MaxPool2d	2	2	-	16|8	-
11.	Conv2d	3	1	1	8|8	256|512
12.	BatchNorm2d	-	-	-	8|8	-
13.	ReLU	-	-	-	8|8	-
14.	Conv2d	3	1	1	8|8	512|512
15.	BatchNorm2d	-	-	-	8|8	-
16.	ReLU	-	-	-	8|8	-
17.	MaxPool2d	2	2	-	8|4	-
18.	Linear	-	-	-	512*4*4| 512*4*4	-
19.	BatchNorm2d	-	-	-	512*4*4|512*4*4	-
20.	ReLU	-	-	-	512*4*4|512*4*4	-
21.	Linear	-	-	-	512*4*4| 100	-

Increasing channels layers using conv2d: (3,64), (64,128), (128,256), (256,512)

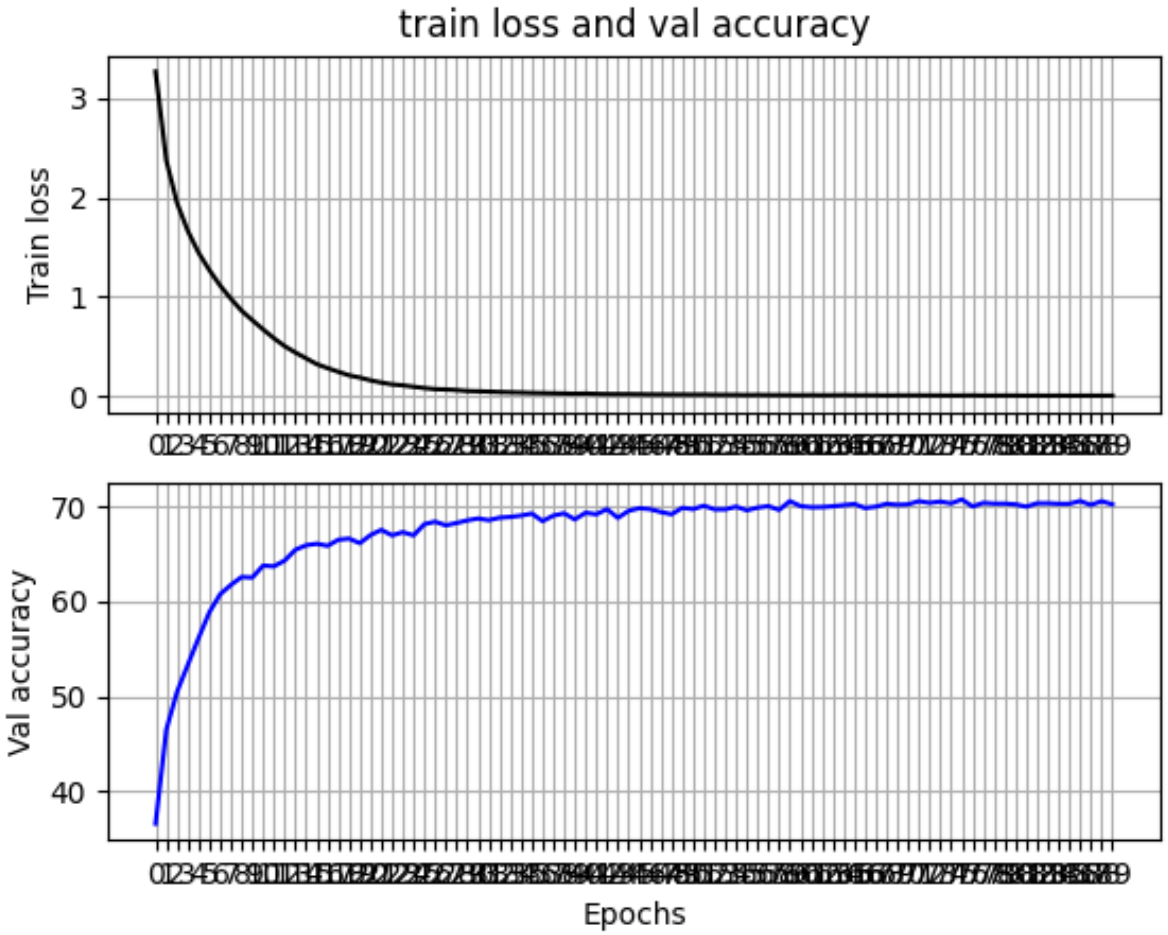
Repeated layers (in_channel == out_channel) using conv2d after (64,128), (128,256), (256,512), there isn't repeated layer after (3,64)

Add (2,2) max pooling 2d after each repeated layer.

BatchNorm2d,ReLU is added after each conv layer.

FC layer get idea from VGG fc layer, which repeated layer of fc layers (layer have the same dimension with the prev layer in fc)

1.2 Training loss and the Validation Accuracy



Validation Accuracy increased quickly from 30% to 60% which cooresponds the trainning loss from 4 to 1

After loss below 1.0 and validation Accuracy above 60%, accuracy increased slowly.

There were huge fluctuations between 61% and 63%.

When the loss dropped to around 0.01, best accuracy achieved 69.6%.

1.3 Ablation Study & Performance Improvement

1.3.1 Deleting Totall_num//2 fc layer, repeating layer of fc layers (layer have the same dimension with the prev layer in fc), and increase channels from 3 to 64 directly.

This change can make network learn more complex features.

The repeated neuron hidden layers can help network learn abstract and higher-level feature representations.

It can increase the intricate and nonlinear mappings from the input data to the output, which can help the network learn complexity tasks.

Before modify:

```
[1] loss: 3.900
Accuracy of the network on the val images: 12 %
[2] loss: 3.640
Accuracy of the network on the val images: 16 %
[3] loss: 3.463
Accuracy of the network on the val images: 13 %
[4] loss: 3.358
Accuracy of the network on the val images: 18 %
[5] loss: 3.279
Accuracy of the network on the val images: 21 %
[6] loss: 3.221
Accuracy of the network on the val images: 21 %
```

After modify:

```
[1] loss: 4.023
Accuracy of the network on the val images: 16 %
[2] loss: 3.315
Accuracy of the network on the val images: 24 %
[3] loss: 2.975
Accuracy of the network on the val images: 28 %
[4] loss: 2.730
Accuracy of the network on the val images: 29 %
[5] loss: 2.520
Accuracy of the network on the val images: 32 %
[6] loss: 2.334
Accuracy of the network on the val images: 32 %
```

1.3.2 Repeating the CONV with the same number of input channels and output channels

Each repeating convolutional layer can capture different features, which can let network learn more complex features.

It works when the train data is limited, which can mitigate overfitting issues

It can increase the depth of the network and add more non-linear part in network.

Adding (512,512) CONV after increasing the channels (256,512) CONV:

```
self.layer3 = make_block(256,512,2)
```

No CONV Repeat after increasing the channels (256,512) CONV:

```
self.layer3 = make_block(256,512,1)
```

Compare:

Can increase 2% by adding one such layer.

1.3.3 Padding when RandomCrop

This increase randomness and diversity in the training data, which can prevent features near the edges of the input may be underrepresented. This can make the accuracy increases more stable and fast.

Without padding:

The loss will go down quickly and the val accuracy will go up slowly.

```
[1] loss: 3.380
Accuracy of the network on the val images: 37 %
[2] loss: 2.203
Accuracy of the network on the val images: 47 %
[3] loss: 1.772
Accuracy of the network on the val images: 50 %
[4] loss: 1.508
Accuracy of the network on the val images: 54 %
[5] loss: 1.241
Accuracy of the network on the val images: 54 %
[6] loss: 1.033
Accuracy of the network on the val images: 55 %
```

With padding:

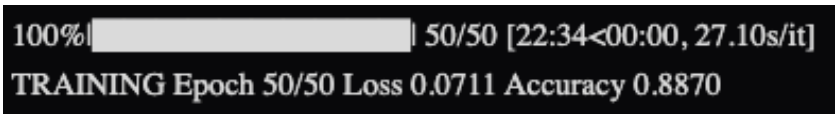
The loss will go down slower than above and the val accuracy will go up quicker than above.

```
[1] loss: 3.608
Accuracy of the network on the val images: 34 %
[2] loss: 2.543
Accuracy of the network on the val images: 43 %
[3] loss: 2.133
Accuracy of the network on the val images: 49 %
[4] loss: 1.884
Accuracy of the network on the val images: 53 %
[5] loss: 1.666
Accuracy of the network on the val images: 55 %
[6] loss: 1.507
Accuracy of the network on the val images: 58 %
```

Part 2

2.1 Train and Test accuracy

2.1.1 fine-tuning the whole network (with same hyperparameter)



Test accuracy: 56.58%

Train Accuracy: 88.7%

2.1.2 fixed feature extractor (with same hyperparameter)

```
 88%|████████▊ | 44/50 [22:00<02:32, 25.35s/it]TRAINING Epoch 44/50 Loss 0.2274 Accuracy 0.6433
 90%|█████████ | 45/50 [22:25<02:06, 25.39s/it]TRAINING Epoch 45/50 Loss 0.2291 Accuracy 0.6450
 92%|█████████▏| 46/50 [22:53<01:43, 25.98s/it]TRAINING Epoch 46/50 Loss 0.2271 Accuracy 0.6340
 94%|█████████▍| 47/50 [23:16<01:15, 25.21s/it]TRAINING Epoch 47/50 Loss 0.2251 Accuracy 0.6417
 96%|█████████▌| 48/50 [23:42<00:50, 25.34s/it]TRAINING Epoch 48/50 Loss 0.2181 Accuracy 0.6610
 98%|█████████▊| 49/50 [24:08<00:25, 25.45s/it]TRAINING Epoch 49/50 Loss 0.2195 Accuracy 0.6527
100%|██████████| 50/50 [24:31<00:00, 29.43s/it]TRAINING Epoch 50/50 Loss 0.2211 Accuracy 0.6467
Finished Training
```

```
✓ 3m [51] test(model, criterion)

      Test Loss: 0.2720 Test Accuracy 0.4530
```

Test accuracy: 45.3%

Train Accuracy: 64.67%

2.1.3 Compare

Fine-tuning the whole network is better than fixed feature extractor, which can have higher test accuracy and train Accuracy

2.2 Hyperparameter

2.2.1 fine-tuning the whole network

```
1 NUM_EPOCHS = 50
2 LEARNING_RATE = 0.0005
3 BATCH_SIZE = 8
4 RESNET_LAST_ONLY = False
5 mean = [0,0,0]
6 std = [1,1,1]
7 data_transforms = {
```

```
    'train': transforms.Compose([
        transforms.Resize(256),
        transforms.RandomResizedCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean,std)
    ]),
    'test': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize(mean,std)
    ]),
```

fixed feature extractor have the same hyperparameter except the `RESNET_LAST_ONLY = True`

2.2.2 Comment:

lr = 0.005 works well in epoch 50.

If lr = 0.001 in epoch 50. The train accuracy will increase up to 90% and test accuracy will decrease below 55%, which is overfit.

If we need increase lr, we also need to decrease the epoch to guarantee the test accuracy is not below 55%.(Overfit)