ECE549 Computer Vision

Assignment 4 Report

Zheng Liu (zliu93)

Part 1

a. The name under which you submitted on Kaggle.

User Name: nagisakaworuadam
Display Name: Zheng Liu

b. Best Accuracy: 0.63900

c. Table defining your final architecture similar to the image above.

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

28	Conv2d	3	4 | 4	256 | 512
29	Relu	-	4 | 4	-
30	BatchNorm2d	-	4 | 4	-
31	Conv2d	3	4 | 4	512 | 512
32	Relu	-	4 | 4	-
33	BatchNorm2d	-	4 | 4	-
34	Conv2d	3	4 | 4	512 | 512
35	Relu	-	4 | 4	-
36	BatchNorm2d	-	4 | 4	-
37	Conv2d	3	4 | 4	512 | 512
38	Relu	-	4 | 4	-
39	BatchNorm2d	-	4 | 4	-
40	MaxPool2d	2	4 | 2	-
41	Conv2d	3	2 | 2	512 | 512
42	Relu	-	2 | 2	-
43	BatchNorm2d	-	2 | 2	-
44	Conv2d	3	2 | 2	512 | 512
45	Relu	-	2 | 2	-
46	BatchNorm2d	-	2 | 2	-
47	Conv2d	3	2 | 2	512 | 512
48	Relu	-	2 | 2	-
49	BatchNorm2d	-	2 | 2	-
50	Conv2d	3	2 | 2	512 | 512
51	Relu	-	2 | 2	-
52	BatchNorm2d	-	2 | 2	-
53	Linear	-	2048 | 4096	-
54	Relu	-	4096 | 4096	-
55	Dropout	-	4096 | 4096	-
56	BatchNorm1d	-	4096 | 4096	-
57	Linear	-	4096 | 4096	-
58	Relu	-	4096 | 4096	-
69	Dropout	-	4096 | 4096	-
60	BatchNorm1d	-	4096 | 4096	-
61	Linear	-	4096 | 1000	-
62	Relu	-	1000 | 1000	-
63	Dropout	-	1000 | 1000	-
64	BatchNorm1d	-	1000 | 1000	-
65	Linear	-	1000 | 100	-

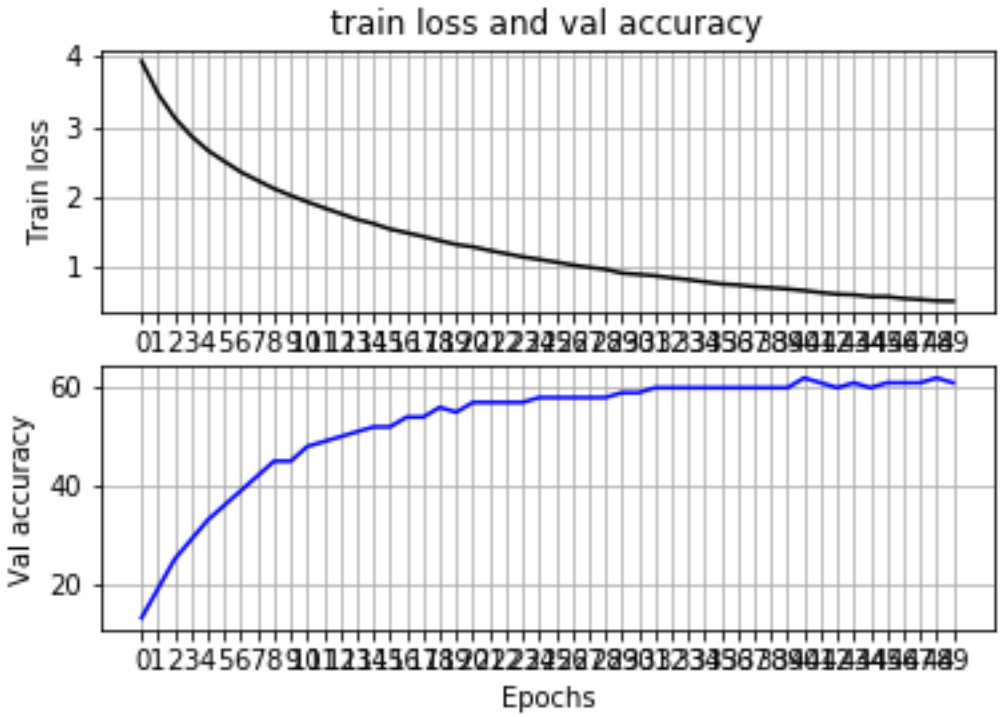
d. Factors which helped improve your model performance. Explain each factor in 2-3 lines.

- Normalize the data: The training data and the test data are normalized when being loading in. This is helpful because the normalization can reduce some unimportant effect in the data, for example, the intensity difference.
- Data augmentation: Only the training data processed by data augmentation. This method can help the classifier to learn more spatial relationship in the image. And this

can reduce the spatial similarity between different images in the original data set, for example, the original data set may tend to put the object in the center of the image, and this method can reduce such effect.

- Deeper Network: With deeper network, the model can learn the features with more different and stable combination so that the features can be understood better by the model.
- Larger epoch number: Larger epoch number can make the classifier learn the data more times so that the model can change the weights more times to get an optimal solution. But if the epoch number is too large, the overfitting effect can reduce the test accuracy significantly.

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



f. Ablation study to validate the above choices, i.e., a comparison of performance for two variants of a model, one with and one without a certain feature or implementation choices.

- Different epoch number: For one model, I used 55 epochs and for another model, I used 45 epochs. All other parameters remain same for two models. And the model with higher number of epochs can achieve 63% on training and 0.629 accuracy on test and the model with lower number of epochs can achieve 61% on training and 0.608 on test. So the one with higher number of epochs can perform better.
- Data augmentation and normalization: I tried one model without any augmentation and normalization technique, and it can achieve 0.528 accuracy on test data. Then I added the RandomHorizontalFlip and RandomRotation and Normalize, and this model could achieve 0.57 accuracy on the test data. All other parameters remain the same. So with some reasonable data augmentation, the accuracy can be higher.

- Deeper network: This is the first modification I tried on the given BaseNet. For the given BaseNet, the accuracy on the test data is only about 0.22. After I added more convolutional layers and more fully connected layers, the test accuracy can be easily higher.

Part 2

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

Fixed feature extractor:

Train Accuracy: 0.869
Test Accuracy: 0.4563

Fine-tuning the whole network:

Train Accuracy: 0.9377
Test Accuracy: 0.5978

b. Report any hyperparameter settings you used (batch_size, learning_rate, resnet_last_only, num_epoches)

For the fixed extractor:

I used learning rate 0.0005 and number of epochs 40, and the accuracy is not high as the final model. I think this is caused by the fact that if the learning rate is too small, then the model can not be optimized to a local optimal.

Final model:

Batch size: 20
Learning rate: 0.001
Resnet last only: True
Num epochs: 45

For the fine-tuning the whole network:

After training the model as the fixed extractor, I directly train the model as the fine-tuning, without changing the number of epochs. And I found that the model can be trained very fast to get a high training accuracy. Then after 40 epochs, the training accuracy can be 0.99, but the test accuracy is only ~0.45. I believe that this is caused by the overfitting. So I changed the number of epochs to be 15.

Final model:

Batch size: 20
Learning rate: 0.001
Resnet last only: False
Num epochs: 15