Deep Learning by PyTorch

CMPT 412 - Computational Vision

Fitzpatrick Laddaran

January 31, 2023

1 Part 1 - Improving BaseNet on CIFAR100

This section pertains to Part 1 of the project handout.

Kaggle ID: **Loading...**

Group members: **Nathan Nicholas Dsouza**, **Wahid Sanjan**

Best prediction accuracy is 71.10%, matching the results on (10% of the test data in) Kaggle. Figure 1 illustrates the prediction accuracy on both the training and validation sets.

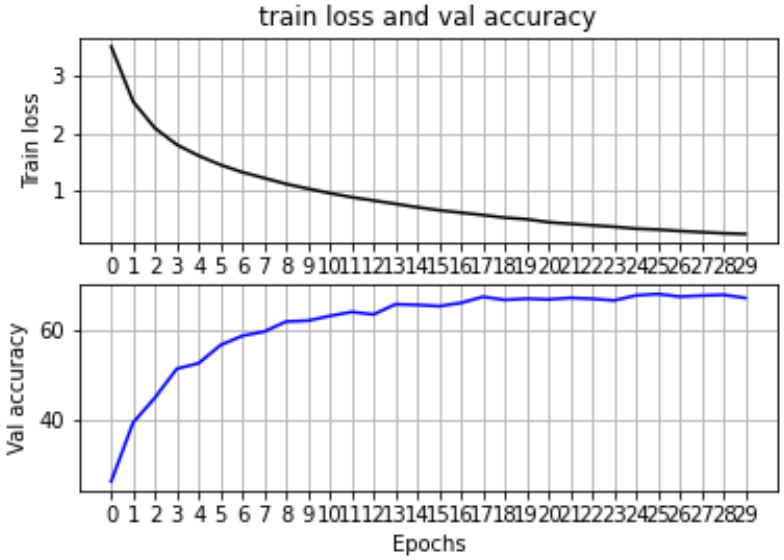

Figure 1: Prediction accuracy on the training and validation set.

Note that for visual purposes, we only show up to 30 epochs. In our competitive submission, we used 50 epochs to slightly increase our performance. The accuracy on both the training and validation set remain steady at the point represented on the figure above.

The table below illustrates the architecture of our final network. Layers 7 to 15 are consecutive repeats of layers 4-5-6 with the same input parameters. In layer 16, we double the number of channels. In layer 22 to 23, we introduce a residual layer `nn.sequential()` composed of a `conv2d()` and `batchnorm2d()` layer. This was an attempt to reduce the effects of the *vanishing gradient problem*[1]. After the residual layer, we introduced a max pooling layer with a kernel size and stride of 2.

[1] https://en.wikipedia.org/wiki/Vanishing_gradient_problem

We repeat the above segment multiple times: the second segment starts with a $16 \times 16 \times 128$ input, the third segment starts with a $8 \times 8 \times 256$ input, and the fourth segment starts with a $4 \times 4 \times 512$ input.

Finally, the image is passed through an average pooling layer, becoming a $1 \times 1 \times 1024$ image. This is converted into a 1D-tensor of size 1024, which is passed into a fully-connected layer, a relu layer, then finally another fully connected layer to which the output is of equivalent size to the number of classes.

Layer No.	Layer Type	Kernel Size	Input/Output Dimensions	Input/Output Channels
1	conv2d	3	32/32	3/64
2	batchnorm2d	-	32/32	-
3	relu	-	32/32	-
4	conv2d	3	32/32	64/64
5	batchnorm2d	-	32/32	-
6	relu	-	32/32	-
...	...	...	...	...
16	conv2d	3	32/32	64/128
17	batchnorm2d	-	32/32	-
18	relu	-	32/32	-
19	conv2d	3	32/32	128/128
20	batchnorm2d	-	32/32	-
21	relu	-	32/32	-
22	conv2d (residual)	1	32/32	64/128
23	batchnorm2d (residual)	-	32/32	-
24	maxpool2d	2	32/16	-
...	...	...	...	...
81	conv2d	3	4/4	512/1024
82	batchnorm2d	-	4/4	-
83	relu	-	4/4	-
84	conv2d	3	4/4	1024/1024
85	batchnorm2d	-	4/4	-
86	relu	-	4/4	-
87	conv2d (residual)	1	4/4	512/1024
88	batchnorm2d (residual)	-	4/4	-
89	avgpool2d	4	4/1	-
90	linear	-	1024/512	-
91	relu	-	512/512	-
92	linear	-	512/(total number of classes)	-

The network was constructed based on trial and error: we did not follow a specific architecture found in literature.

1.1 Ablation Study

The biggest improvement in our prediction accuracy was when we increased the number of layers. The base model provided ran with a prediction score of about 23% on the validation set. After adding all the layers mentioned above, our group increased our prediction score by about 35%, amounting to approximately 58%. The remainder of the improvements were made to the data input, that is: a permutation of transformations were tested, including data normalization. These operations yielded an additional 11%-12% improvement on predicting classes in the validation set.

2 Part 2 - Transfer Learning

This section pertains to Part 2 of the project handout.

Using the hyperparameters below (and setting `resnet_last_only` to false), I fine-tuned the entire ResNet. I obtained a prediction accuracy of 99.30% on the training set (as shown in Figure 2) and 59.85% on the test set (as shown in Figure 3).

```
TRAINING Epoch 1/10 Loss 0.5673 Accuracy 0.0807
TRAINING Epoch 2/10 Loss 0.3320 Accuracy 0.3657
TRAINING Epoch 3/10 Loss 0.2087 Accuracy 0.5893
TRAINING Epoch 4/10 Loss 0.1374 Accuracy 0.7413
TRAINING Epoch 5/10 Loss 0.0856 Accuracy 0.8503
TRAINING Epoch 6/10 Loss 0.0541 Accuracy 0.9217
TRAINING Epoch 7/10 Loss 0.0346 Accuracy 0.9563
TRAINING Epoch 8/10 Loss 0.0230 Accuracy 0.9793
TRAINING Epoch 9/10 Loss 0.0160 Accuracy 0.9860
TRAINING Epoch 10/10 Loss 0.0117 Accuracy 0.9930
Finished Training
```

Figure 2: Prediction accuracy on the training set after fine-tuning the entire ResNet.

```
Test Loss: 0.2024 Test Accuracy 0.5987
```

Figure 3: Prediction accuracy on the test set after fine-tuning the entire ResNet.

Using the hyperparameters below (and setting `resnet_last_only` to true), I trained only the last fully-connected layer of the architecture. This architecture gave slightly worse results compared to the first architecture. I obtained a prediction accuracy of 80.40% on the training set (as shown in Figure 4) and 42.66% on the test set (as shown in Figure 5).

Hyperparameters used are as follows:

- `batch_size`: 8
- `learning_rate`: 0.0025
- `resnet_last_only`: True, False
- `num_epochs`: 10

```
TRAINING Epoch 1/10 Loss 0.6303 Accuracy 0.0393
TRAINING Epoch 2/10 Loss 0.4588 Accuracy 0.2290
TRAINING Epoch 3/10 Loss 0.3522 Accuracy 0.4037
TRAINING Epoch 4/10 Loss 0.2811 Accuracy 0.5253
TRAINING Epoch 5/10 Loss 0.2363 Accuracy 0.6157
TRAINING Epoch 6/10 Loss 0.2014 Accuracy 0.6807
TRAINING Epoch 7/10 Loss 0.1770 Accuracy 0.7163
TRAINING Epoch 8/10 Loss 0.1591 Accuracy 0.7507
TRAINING Epoch 9/10 Loss 0.1414 Accuracy 0.7707
TRAINING Epoch 10/10 Loss 0.1278 Accuracy 0.8040
Finished Training
```

Figure 4: Prediction accuracy on the training set after training only the last fully-connected layer.

```
Test Loss: 0.2879 Test Accuracy 0.4266
```

Figure 5: Prediction accuracy on the test set after training only the last fully-connected layer.