CS543 Assignment 4

Your Name: Zong Fan
Your NetId: zongfan2

Part 1: Improving BaseNet on CIFAR100

The name on Kaggle
Zong Fan

Best accuracy
0.449

Network architecture

Layer No.	Layer type	Kernel size (for conv)	Stride (for conv)	Padding (for conv)	Input | output dimension	Input | output channels
1	conv2d	3	1	1	32|32	3|6
2	bn	-	-	-	32|32	-
3	relu	-	-	-	32|32	-
4	maxpool	2	-	-	16|16	-
5	conv2d	3	2	1	16|8	6|16
6	bn	-	-	-	8|8	-
7	relu	-	-	-	8|8	-
8	res_block	3	2	1	8|4	16|64
9	res_block	3	1	0	4|4	64|128
10	linear	-	-	-	2048|512	-
11	bn	-	-	-	512|512	-
12	relu	-	-	-	512|512	-
13	linear	-	-	-	512|100	-

Architecture of res_block with given stride=s, input | output dimension=d1|d2, input | output channels=c1|c2

Layer No.	Layer type (branch1)	Layer type (branch2)	Kernel size (for conv)	Stride (for conv)	Padding (for conv)	Input | output dimension	Input | output channels
1	conv2d	-	3	s	1	d1 | d2	c1 | c2
2	bn	-	-	-	-	d2 | d2	-
3	relu	-	-	-	-	d2 | d2	-
4	conv2d	-	3	1	1	d2 | d2	c2 | c2
5	bn	-	-	-	-	d2 | d2	-
6	-	conv2d	1	s	0	d1 | d2	c1 | c2
7	-	bn	-	-	-	d2 | d2	-
8	add		-	-	-	d2 | d2	c2 | c2
9	relu		-	-	-	d2 | d2	-

Ablation studies for investigating factors that helped improve your model performance. Explain each factor in 2-3 lines.
Since there is submission limits for test dataset, here we mainly discuss the factors that show improvement on **validation** dataset.

The baseline performance achieves accuracy of 23% when the input image size is 32×32.
1. + data normalization.
When normalizing the input data to value [-1, 1], the model's accuracy was 26%, achieving 3% improvement.

2. + data augmentation.
It's an effective way to enlarge the training dataset. When random resized crop and random horizontal flip were employed, the accuracy went up to 27%.

3. Increase the depth of network

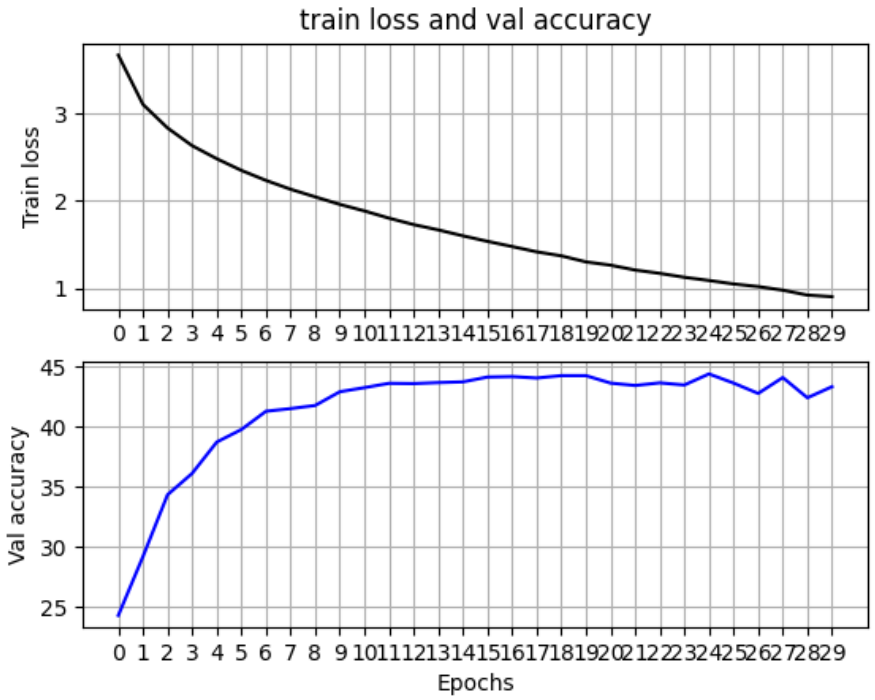
On top of the convolution (conv) layers of the BaseNet, another 2 convolution followed by ReLU activation was added into the network. Their input/output dimensions were kept the same as 5×5. Their input/output channels were 16/32 and 32/64, respectively. By doing so, the model accuracy increased to 28%.

4. + batchnormalization (BN)
When adding BN layers before each ReLU activation layer, the accuracy improved to 39%, which is the more significant improvement with a single trick.

5. Replace the 2 more conv layers with 2 residual blocks (res_block)
The architecture of res_block is shown in the table previously. It has two branches: one branch has 2 conv layers followed by BN and ReLU, another is called skip connection which contains 1 conv layer followed by BN. Then the feature maps of the two branches are added element-wisely. By doing so, the model's accuracy increased to 42% (42.5% on test dataset).

6. Adjust kernel size, employ stride on conv layer, and increase the number of training epochs.
Based on previous architecture in experiment 5, the kernel size of conv layers was reduced to 3 and padding was employed. As shown in the previous table, the second maxpooling was removed. I used conv with stride=2 instead for downsampling. Also, the first res_block also performed downsampling and the output feature of conv layers was 128×4×4. By doing so, the model's accuracy was improved to 44.5% on test dataset.

Final architecture's plot for training loss and validation accuracy.
As we can see, the model performance almost saturates at epoch 10, which means elongating the training under this situation doesn't improve the performance.



Explanation of any extra credit features if attempted.
1.Use SE-block
Squeeze-and-excitation (SE) module was introduced into the residual block after the last BN layer in the downsampling branch. SE module is a kind of attention mechanism that adaptively recalibrates channel-wise feature responses by modeling interdependencies between channels. However, in out experiment, the model's accuracy was still 42%, no improvement compared to the use of res_block individually. We suppose this is because the network is too shallow and the number of channels is not very high. These factors limit the capability of modeling channel interdependency information.

Reference: Hu, Jie et al. "Squeeze-and-Excitation Networks." *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018): 7132-7141.
Repository: https://github.com/moskomule/senet.pytorch

Part 1: Improving BaseNet on CIFAR100

The name on Kaggle
Zong Fan

Best accuracy

0.64

Report the train and test accuracy achieved by using the ResNet as a fixed feature extractor vs. fine-tuning the whole network.
When using learning rate=0.01, batch size=12, epochs=20
Fixed feature extractor:
Training accuracy: 0.9967
Validation accuracy: 0.375
Test accuracy: no submission

Tuning whole network:
Training accuracy: 0.9950
Validation accuracy: 0.5100
Test accuracy: no submission

When using learning rate=0.001, batch size=12, epochs=30
Fixed feature extractor:
Training accuracy: 0.9948
Validation accuracy: 0.4421
Test accuracy: no submission

Tuning whole network:
Training accuracy: 0.9961
Validation accuracy: 0.6333
Test accuracy: 0.5572

Report the hyperparameter settings you used and explain how you came up with these settings(batch_size, learning_rate, resnet_last_only, num_epochs).

Batch size:
When using learning rate=0.001, epochs=30, tuning whole network
When the batch size was 24, accuracy was 0.6167 on validation dataset, and 0.5457 on test dataset.
When the batch size was 12, accuracy was 0.6333 on validation dataset, and 0.5572 on test dataset.
It's interesting that using larger batch size decreases the model performance. We suppose this may be caused by insufficient training.

Learning rate:

When using batch size=12, epochs=30, tuning whole network
When learning rate=0.001, accuracy was 0.6333 on validation dataset and 0.5572 on the test dataset.
When learning rate=0.01, accuracy was 0.5512 on validation dataset (no submission).
When learning rate=0.0001, accuracy was 0.1537 on validation dataset (no submission).

Using a proper learning rate (LR) is essential to train a good model. Using too large LR may cause model convergence unstable, while using too small LR may cause convergence too slow.

ResNet_last_only:
As same as previous section.

Number of epochs:
When using learning rate=0.001, batch size=12, tuning whole network
When epochs=20, accuracy was 0.6112 on validation dataset (no submission).
When epochs=30, accuracy was 0.6333 on validation dataset and 0.5572 on test dataset.
Increasing the training epochs is usually able to improve the model performance by guaranteeing the convergence of the model.

Explain the transformations you used on the train set. Discuss how including/excluding each of them made a difference in the accuracy.
When using learning rate=0.001, batch size=12, tuning whole network, training epochs=30
1. Only normalize the data
Accuracy: 0.5867 on validation dataset

2. + random horizontal flip
Accuracy: 0.6133 on validation dataset

3. + random resized crop
Accuracy: 0.6333 on validation dataset

We can see that including both data augmentation methods could improve the model performance by 2%-3%.

Explanation of any extra credit features if attempted.

1. Increase the network depth of ResNet

Usually, the deeper network has higher feature extraction capability. To test this assumption, we employed ResNet50 instead of ResNet18 for classification on this dataset. With other model hyperparameters being the same, the model's accuracy increased from 0.6333 to 0.7217 on validation dataset and 0.5572 to 0.6406 on test dataset. This experiment shows the effectiveness of using deep network.

2. Using SE module for ResNet50

As part 1, we further employed the SE module into the ResNet50 as the attention module. The trained weights were provided by Hataya Ryuichiro (https://github.com/moskomule/senet.pytorch). Similarly, the model's accuracy only increased to 0.6833 on validation dataset and 0.5865 on test dataset when number of training epochs was 30. We suppose more training epochs is necessary to guarantee the convergence of this network architecture.