DLCV HW1 Report

111550098 楊宗儒

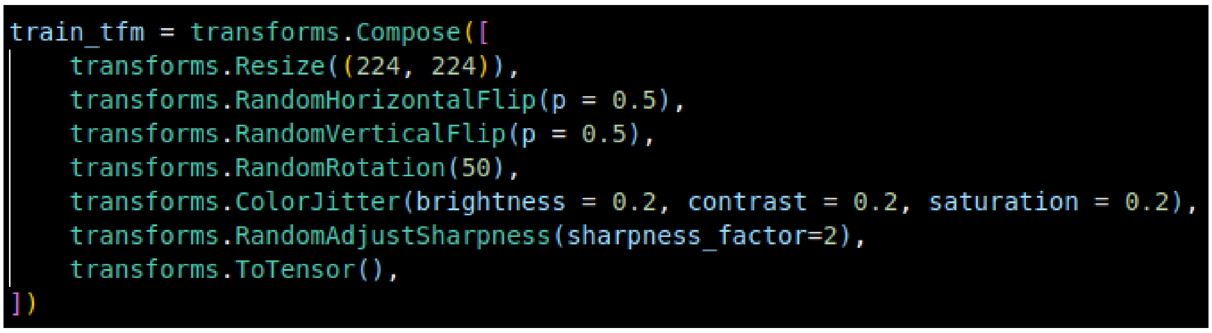
Introduction

In this homework, we are going to train a ResNet-based model to perform an image classification task consisting of 100 classes. The entire dataset has about 25,000 images of plants and animals etc.

The main idea of our approach is to leverage the pretrained weights on ImageNet datasets, and to use an improved version of ResNet (ResNext)[1] as our backbone.

Method

1. Preprocessing
   In the preprocessing step, in addition to the required resizing, I used the horizontal and vertical flip, random rotation of 50 degree, color jittering and random saturation.



```python
train_tfm = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(p = 0.5),
    transforms.RandomVerticalFlip(p = 0.5),
    transforms.RandomRotation(50),
    transforms.ColorJitter(brightness = 0.2, contrast = 0.2, saturation = 0.2),
    transforms.RandomAdjustSharpness(sharpness_factor=2),
    transforms.ToTensor(),
])
```

Figure 1. The transform of the preprocessing.

2. Model architecture

In our model, we leverage an improved version of ResNet: ResNeXt [1]. The difference between them is that for every residual block, instead of using a large number of channels like ResNet, ResNeXt first uses multiple convolution blocks (the number corresponding to the cardinality of the model) of a small number of channels then aggregates the output.

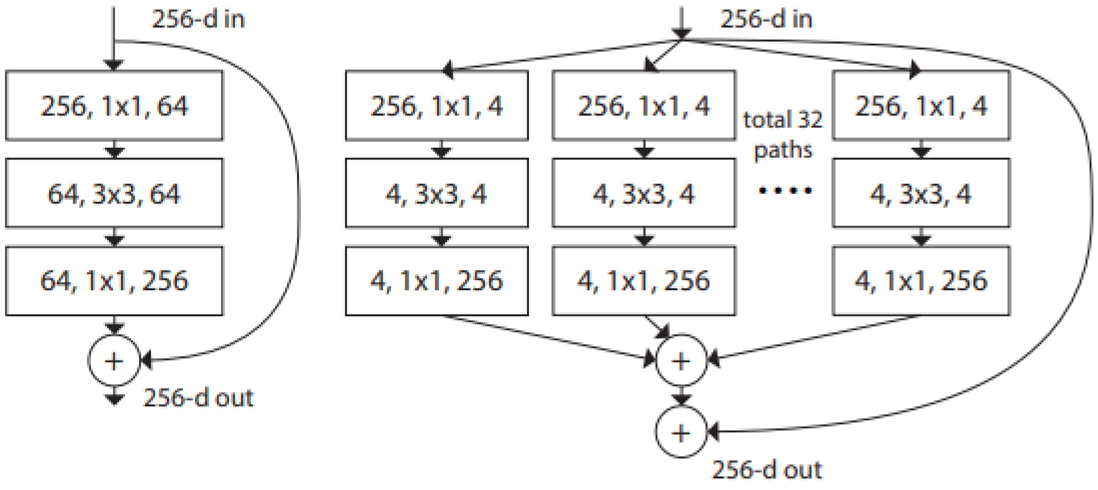
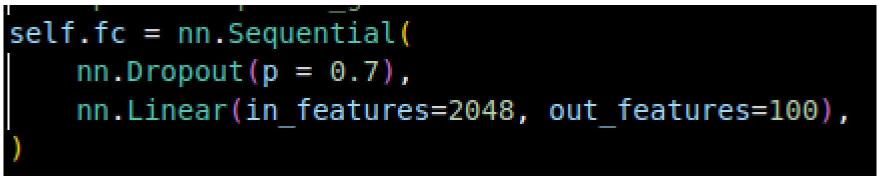


Figure 2. Demonstration of ResNext. The left is the original residual block from ResNet, the right is the modified version of the residual block in ResNeXt.

In our model, we use the pretrained weights of resnext50_32x4d provided by pytorch, and replace only the fully connected layer with the dropout layer and a linear layer.



```
self.fc = nn.Sequential(
    nn.Dropout(p = 0.7),
    nn.Linear(in_features=2048, out_features=100),
)
```

Figure 3. The modification on the fully connected layer

3. Hyperparameters
   During training, I used the Adam optimizer with learning rate 3e-5. The loss is the regular cross entropy loss. The batch size is 64. The model was trained with 50 epochs. The final weights would be the one with the best valid accuracy(instead of the final weights).

Result

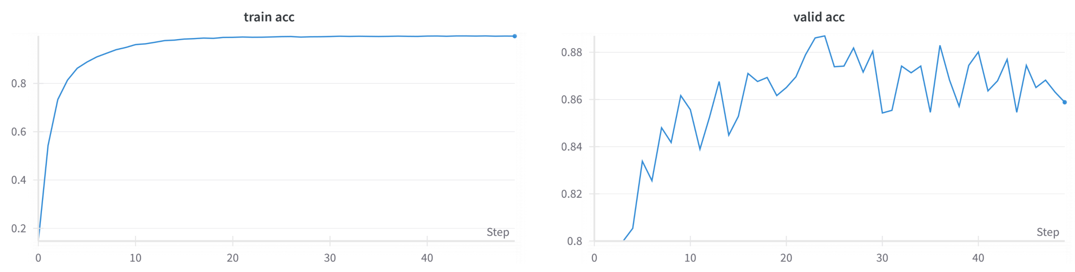


Figure 4. The training curve of the model

The validation accuracy of the model was 88.7%, and the public score of the model is 93%.
As shown in Figure 4, even with data augmentation, the model still suffers from some overfitting, thus the best model appears around the 30th epoch.
However, even with the smallest ResNet model, overfitting still exists, and the final performance is approximately correlated with the model size (see additional experiments), therefore even with the overfitting, a bigger model still gives a better generalization power.

Additional Experiment

1. ResNet vs ResNeXt

As stated in [1], with the same amount of parameters, the ResNeXt model performs better than the original ResNet model.

Table 1. The Validation Accuracy of ResNet50 and ResNeXt50

model	#parameter	validation accuracy
resnet50	25M	0.872
resnext50_32x4d	25M	0.883

The improvement matched the claim in the paper. The additional cardinality improves the model's generalized ability.

2. Scaling

As a rule of thumb in deep learning, a larger model often leads to a better performance. In these experiments I tried using different sizes of the ResNeXt, I expect to have better performance for the larger model.

Table 2. The Validation Accuracy of Different Sizes of ResNeXt Models

model	# parameter	validation accuracy
resnext50_32x4d	25M	0.883
resnext101_32x8d	89M	0.887

The larger model has a slightly better accuracy. The reason the improvement isn't significant might be because the model capacity is far more than the available data, resulting in little difference in accuracy.

3. Contrastive Loss

The goal of adding contrastive loss is to minimize the distance of the feature vector in the same class while maximizing the distance between different classes.
The contrastive loss is defined as following: for every image, randomly sample a image of same class or different class with probability 50%, after calculating the feature through the backbone model, calculate the cosine similarity between the two image feature times the corresponding factor depending on whether the two image are in the same class or not.
The reason I originally thought that it would work is because it can guide the backbone model to produce features that can be more easily separated by the linear layer, which might be a good regulation.

Table 3: The Validation Accuracy of Model with Contrastive Loss and Without It

	validation accuracy
w/o contrastive loss	0.887
w/ contrastive loss	0.876

As shown in Table 3, the performance didn't improve. My guess is that even without contrastive loss, the model can already produce the best feature encoding, which ended up overfitting the data. After adding contrastive loss, the value of contrastive loss eventually came close to zero, which might support my argument.

4. Positional Encoding

Positional encoding often appears in the transformer based language model to inject the position data into the embedding. However, the other function of it is to expand the input into frequency domain, which facilitates the training since the MLP tends to learn the low frequency signal much faster than the high frequency one [2].
My original expectation is that by expanding the feature into multiple frequencies, the model might learn better as stated in [2].

The positional encoding I use uses L = 3.

Table 4. The Validation Accuracy of Model with positional encoding and Without It

	validation accuracy
w/o positional encoding	0.887
w/ positional encoding	0.863

As shown in Table 4, the model didn’t improve. My guess is that since the model would eventually overfit, the increase of convergence speed from positional encoding wouldn’t help as the model capacity was already maxed out.

References

[1] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
[2] Rahaman, N., Baratin, A., Arpit, D., Dräxler, F., Lin, M., Hamprecht, F.A., Bengio, Y., Courville, A.C. On the spectral bias of neural networks. In ICML (2018).

GitHub Link

https://github.com/s0n9Yu/DLCV-hw1#