Selected Topics in Visual Recognition using Deep Learning
HW1 Report: Image Classification

310706004 Hsin-Ying Lien

I. Introduction

In this assignment, I tried some famous CNN model architectures for image classification. Finally, I implemented a pretrained model based on ResNeXt-101 and fine-tune it to accomplish the image classification on the given bird dataset. It reached a final accuracy rate about 0.669 on the test dataset.

Code GitHub link: https://github.com/angelalien/VRDL-HW1
Dataset: 6,033 bird images belonging to 200 bird species (training: 3,000, test: 3,033)

II. Methodology

1. Data Pre-process

I first loaded the train dataset from the training_images file, and split them into training data and validation data at a proportion of 8:2. Then I performed data augmentation with torchvision modules toward the data. There are two preprocessing function setting for training and testing data shown below.

Training data preprocess steps:

1) Resizes the image into 256*256 pixels size.
2) Crops the image at the center into 224*224 pixels size.
3) Horizontally flip the image randomly.
4) Rotate the image by angle in the range of (-15,15).
5) Convert the PIL Image to tensor.
6) Normalize the float tensor image with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] (three channels).

Testing data preprocess steps:

1) Resizes the image into 256*256 pixels size.
2) Crops the image at the center into 224*224 pixels size.
3) Convert the PIL Image to tensor.
4) Normalize the float tensor image with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] (three channels).

2. Model architecture

The final model I used is based on ResNeXt-101 structure and pretrained on ImageNet. ResNeXt is composed of ResNet and Inception's structure (split-transform-merge). I modified the last fully connected layer, changing the output to 200 (number of image classes). Model architectures of ResNet-50 and ResNeXt-50 are shown below. (I could only find the structure figure of 50 layers.)

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

3. Hyperparameters

batch size = 32
epochs = 8
loss function: cross entropy
optimizer: SGD, momentum=0.9, weight_decay=1e-6
initial learning rate=0.01
learning scheduler: StepLR, step_size=5, gamma=0.1

III. Experimental Results and Observations

I have tried different composition of data preprocessing method and model architecture. For data augmentation, I found that it is better to resize and crop training images to a given size, rather than cropping a random portion of image and resize it. I think it is because the second method may not crop the bird area. Also, randomly horizontally flipping, rotating and normalizing could help increase the image information and decrease noises. While it doesn't help much to randomly change the brightness, contrast, and saturation of an image, possibly due to the little color difference between images.

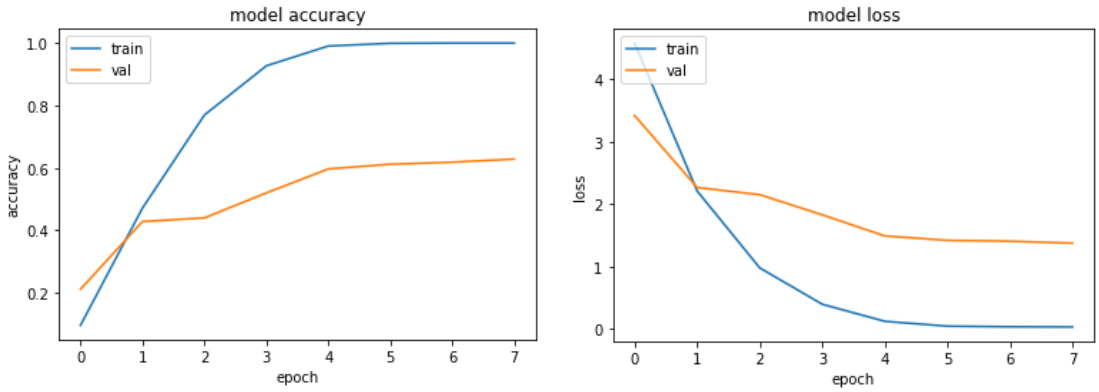
For the pretrained model, I have tried models based on AlexNet, VGG, ResNet and ResNeXt architecture. The result shows that ResNeXt, which is the most modern model among them, outperforms the other ones. This verifies that ResNeXt assemble advantages of ResNet and Inception. The following table is the validation accuracy rate of different models with the hyperparameter setting specified before.

model	ResNeXt	ResNet	VGG	AlexNet
validation accuracy rate	0.7200	0.6517	0.4600	0.3083

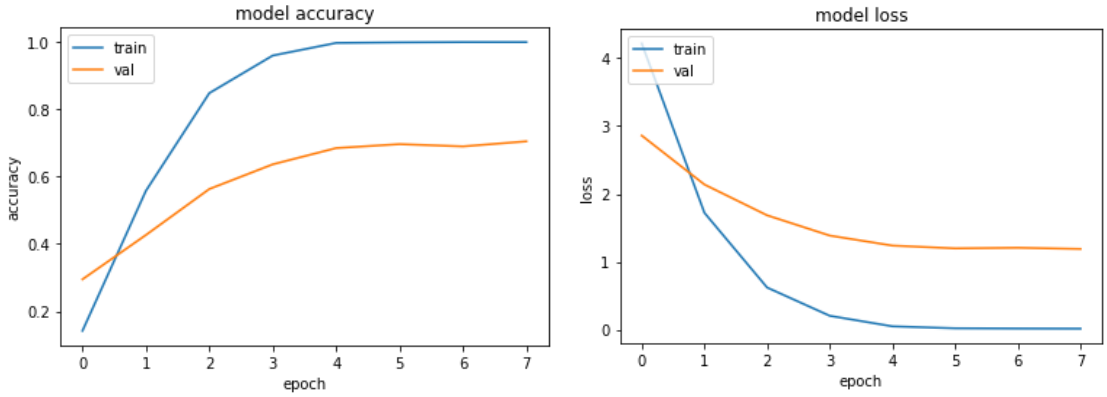
During the training process, the largest problem I met is overfitting, due to few training images. Besides adjusting the data augmentation method, I also tried to modify the last model layer by adding some nonlinear and dropout layer. However, the final performance doesn't improve although it mitigated the overfitting problem at the beginning. I think it's because the model is too complex. Then I freeze all the earlier layers (except for my modified layers), but the final accuracy rate is lower even though I have tried different hyperparameter setting.

The following plots show the loss and accuracy rate changing process of ResNet and ResNeXt with the best hyperparameter setting.

ResNet



ResNeXt



IV. Summary

Nowadays, it's easy for us to implement pretrained models and fine-tune it to deal with image classification tsks. However, we may easily meet the overfitting problem with little training data. At this moment, proper data augmentation methods play an important role in avoiding this situation if we couldn't increase training data. Besides, with the experimental results, ResNeXt could reach relatively high accuracy rate with proper hyperparameter setting. As a result, data preprocessing and model selection are important for training performance.

V. Reference

Transfer Learning For Computer Vision Tutorial
https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Aggregated Residual Transformations for Deep Neural Networks (I started to try ResNeXt with the paper result.)
https://arxiv.org/abs/1611.05431