Report of Project 2

He Yajing 20459312

November 16, 2017

1 Project Description

In this project, the task is to train a deep learning model to classify 5 kinds of flowers efficiently and use validation dataset to evaluate the efficiency of model. And then use this model to predict the kinds of images in the test dataset. When it comes to images classification, tensorflow is usually taken into consideration. By using tensorflow, I can construct a CNN model to train with training dataset and adjust the layers and parameters. However, the accuracy of the model using tensorflow is just about 60%. Then I try to use pytorch to build a transfer learning classification model and it is much more efficient with accuracy over 90%. In the following chapter, I will introduce the details of my process of this project.

2 Some Concepts of Projects

2.1 Tensorflow

TensorFlow is an open-source library developed by Google company and applied for data-flow programming. Nowadays, it is widely used for machine learning applications like neural networks. Also, tensorflow layers module offers a high-level API to make it simple to construct a neural network model which can handle image recognition tasks properly and accurately.

2.2 CNN Model

CNN(Convolution Neural Network) is a class of deep, feed-forward artificial neural network which is successfully applied to images analytics nowadays. During the operation of CNN model, it applies convolution concept and uses pooling and other methods to convolute images for a number of times to format plenty of images of attributes. And then the input is tiled into a fully connected multi-layer neural network and determined the picture classification by the output of softmax. CNN tries to use a variation of multilayer perceptions designed to require minimal preprocessing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered.

2.3 Pytorch

PyTorch is a python package released recently which provides two high-level features: one is tensor computation (like numpy) with strong GPU acceleration, the other is Deep Neural Networks built on a tape-based auto-grad system. It is more efficient than tensorflow in term of image analytics in some degree.

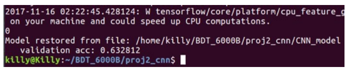
2.4 Transfer Learning

Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to deal with a new learning task with different distribution of dataset. Therefore, transfer learning does not make the same assumptions as traditional machine learning. There two major transfer learning scenarios looks: One is called fine-tuning the convnet which is to initialize the network with a pretrained network instead of random initializaion and rest of the training looks as usual, the other is ConvNet as fixed feature extractor in which the weights for all of the network except that of the final fully connected layer will be frozen and this last fully connected layer is replaced with a new one with random weights and only this layer is trained.

3 Process of Project

3.1 CNN Model Using Tensorflow

For this project, the first task is to read all the training and validation images and resize them to a uniform size to train and validate the model. I use transform method in skimage package to resize the images into 100*100 form. And then, I use tensorflow to build a CNN model. In this model, I firstly read uniform-sized images and print all the paths. Then it is a important step that I shuffle the training data into a random set in order that I can train the model accurately and efficiently. After data is ready, I construct a CNN model with four layers. I set the number of epochs into 200 and adjust the different parameters and the training accuracy is over 90% but the validation accuracy is much lower.



```
2017-11-16 02:22:45.428124: W tensorflow/core/platform/cpu_feature_g
 on your machine and could speed up CPU computations.
0
Model restored from file: /home/killy/BDT_6000B/proj2_cnn/CNN_model
   validation acc: 0.632812
killy@Killy:~/BDT_6000B/proj2_cnn$
```

Figure 1: The validation accuracy of CNN model

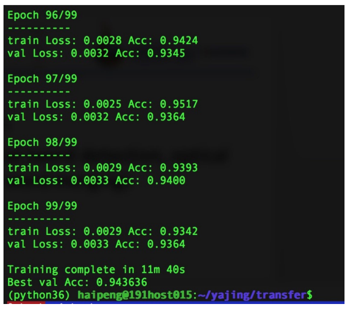
3.2 Transfer Learning Model Using Pytorch

By using tensorflow, the validation accuracy is only about 60%. Therefore, I find another way to build model using transfer learning with pytorch. In practice, it is hard to train an entire Convolutional Network from scratch because the dataset is of insufficient size. Instead, pretrained ConvNet on a very large ImageNet which contains 1.2 million images with 1000 categories is offered, so I use this Convnet to initialize the network which is called finetuning method.

For image preprocessing, I use resize method in skimage to resize all the images into 224*224 size. One important thing is the data augmentation and normalization for training in order to get more different images to train the model and just normalization for validation. And then I use torchvision and torch.utils.data packages for loading the data.

When datasets are all ready, I write a general function to train a model where parameter *scheduler* is an LR scheduler object from $torch.optim.lr_scheduler$. Then I load a pretrained model and reset final fully connected layer.After that, I put the training data and validation data into the model to train and evaluate the model on GPU rather than CPU due to the heavy computation cost. Finally, I save the whole model using *torch.save* for prediction.

In prediction part, the data preprocessing of test images is similar with that of validation images. The only different is that the test images have null labels. By using *torch.load*, I load the trained model and input the transfered test images. The output is their number of classes which is written into a text file. In addition, my prediction codes also run in GPU.



```
Epoch 96/99
----------
train Loss: 0.0028 Acc: 0.9424
val Loss: 0.0032 Acc: 0.9345

Epoch 97/99
----------
train Loss: 0.0025 Acc: 0.9517
val Loss: 0.0032 Acc: 0.9364

Epoch 98/99
----------
train Loss: 0.0029 Acc: 0.9393
val Loss: 0.0033 Acc: 0.9400

Epoch 99/99
----------
train Loss: 0.0029 Acc: 0.9342
val Loss: 0.0033 Acc: 0.9364

Training complete in 11m 40s
Best val Acc: 0.943636
(python36) haipeng@191host015:~/yajing/transfer$
```

Figure 2: The validation accuracy of transfer learning model