

LUT Computer Vision and Pattern Recognition Laboratory

BM20A6100 Advanced Data Analysis and Machine Learning

TASK:

Practical Assignment: Wood Species Classification

Date: 17-12-2020

Name: Tuo Yang (589715)

Suzan Yemane (592919)

1. Data description and applied preprocessing

The image dataset we used is the one including 41 wood species’ 2934 images, each image’s resolution is 3264\*2448, and image samples of each class are shown as below:

Fig 1.1 samples of database

Before we start sending samples into the network for training, three phases we need to finish: Data augmentation, Data preprocessing and datasets splitting and building:

**Data augmentation**: The intention of data augmentation is to generate more data samples used for training the CNN model, the specific methods include random cropping, color jittering, noise (generally would be gaussian noise or PCA jittering) adding, image spinning and shifting, image scaling and image flipping, the purpose of these methods is to improve the robustness of this model to make sure the image could be correctly classified or objects inside could be precisely recognized by the model when the image is scaled, resized, cropped, flipped and so on.

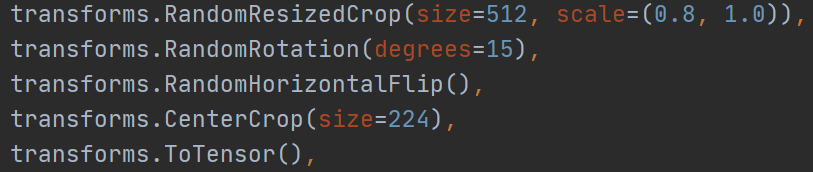
Because we used the pytorch to realize this task, for data augmentation, we need to define transformer object first as follows:

Fig 1.2 data augmentation operations on datasets

From the figure 1.2 four augmentation, resized cropping (make the shorter side length of the image decreased to 512 pixels), randomly rotation (rotate image to the left and the right about 15 degrees), random center crop (randomly crop 224 by 224 pixels’ area from baselined on the center of the image), To tensor means normalizing the image to the range within [0,1].

**Data preprocessing**: only one data processing method we used here, do some normalization(standardization) to each image samples sent into the network, every channel’s pixel values get deducted by its mean value first, then divided by its standard deviation, because each dimension of image follows the same distribution, features and difference will be highlighted by doing this operation.

**Datasets splitting and building:** we respectively take 50% of data samples for the training and testing dataset, among which we take 15% of training samples as the validation sets to evaluate the performance change during training, but sampling way is to take 50% training and testing dataset samples respectively from each class samples first, then combine them together just for making sure that all wood species samples will be used for training.

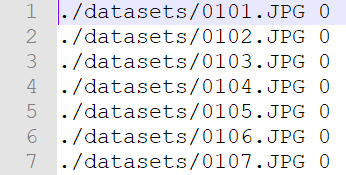
After splitting datasets from samples, we need to find one way to make connection between data samples with labels, according to pytorch syntax, we put all training, testing and validation samples location info inside three text files (ground truth files), each row represents one data sample with its corresponding label. The results are shown as below:

Fig 1.4 one small part of training samples

1. Description of chosen architectures

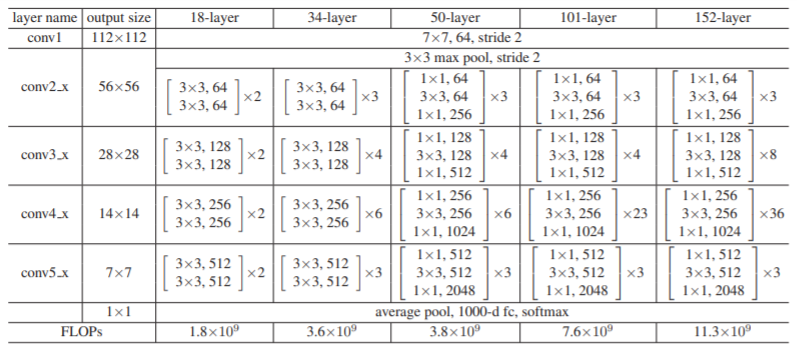
If we want to get good performance on classification, deeper-layer CNN model is the best choice, but more deeper CNN model will cause more bigger calculations and longer training time, especially for images with the high solution 3264\*2448, which will include 7990272 feature vectors as inputs of network, and deeper CNN model will be more helpful to learn more complex relations between inputs and outputs. VGG, ResNet and DenseNet all are best options here, here we take Resnet50 as our CNN model, which has 50 layers in total, and it’s based on Bottleneck, its architecture and comparison with other networks are shown as below:

Fig 2.1 architecture comparison among different layer ResNet

1. Chosen optimizer

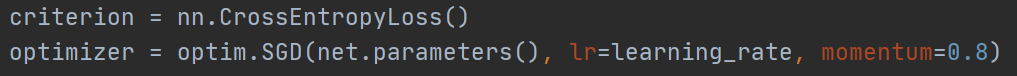
We have chosen Stochastic Gradient Descent (SGD) to optimize our objective function which calculate the cost by using common cross entropy, and its definition inside the code is shown as below:

Fig 3.1 the syntax of defining cross entropy and SGD function in python

From fig 3.1 we could know that the SGD we used has the momentum, which will automatically adjust the learning rate during the whole process. For example, during the gradient descent process, if the gradient is very high(the slope is very steep), we can increase the learning rate to speed up the dropping rate of the cost, otherwise when the cost(point) is getting close to the convergence, we can decrease the learning rate to decrease the step length to make sure the objective function will get convergence to the global minimum. We tried other optimizers like SGD with Nesterov Acceleration and Adam as well, and their performance(refer to classification accuracy is not better than SGD with momentum).

1. All relevant hyperparameters and methods used to improve the convergence

**optimized parameters**:

learning rate:0.009

epochs:30

batch size:24

optimizer: SGD (Stochastic Gradient Descent) with momentum 0.8

**network parameters**:

As the length limit, all activation functions(basically ReLU) and num of hidden layers and hidden layer units can be found in the folder testing\_results/output\_results.txt

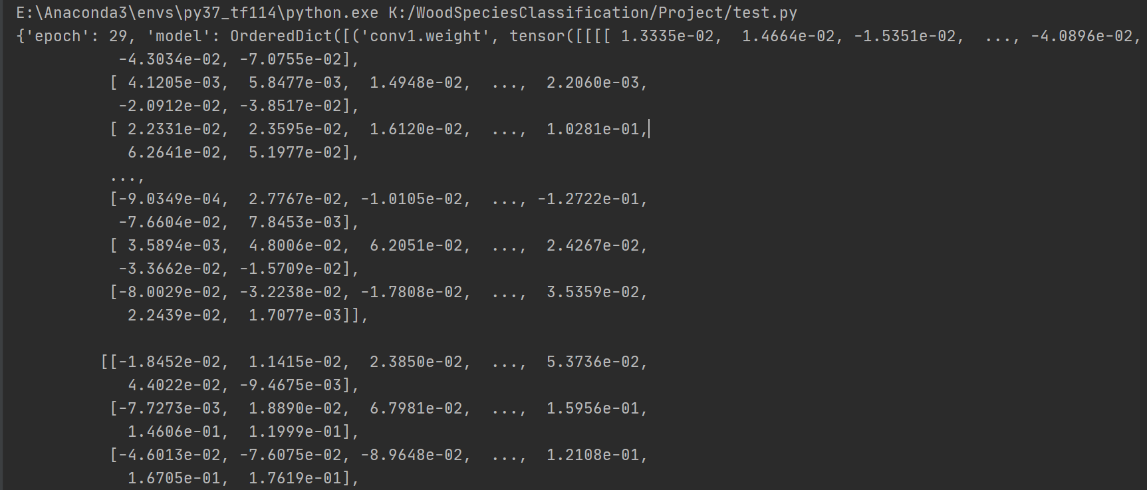
 During training we used pretrained Resnet50 model to train our own weights for this classification task, and partial related weights(because of the length limit) are shown as below:

Fig 4.1 partial weights from we trained ResNet50 model

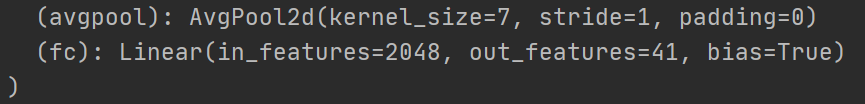
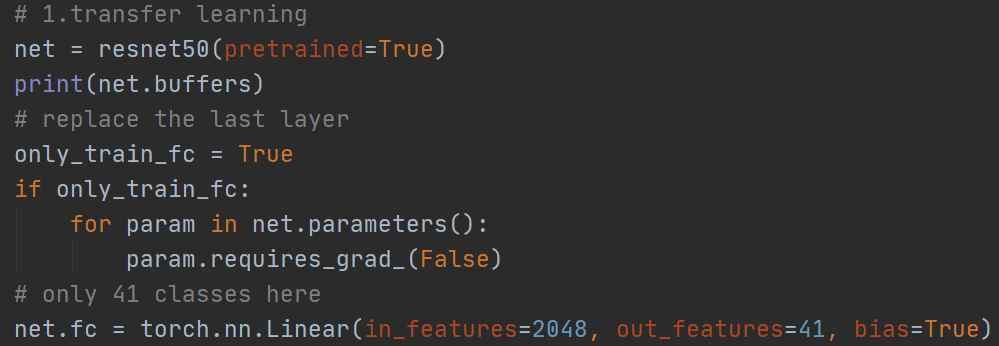
 The truth proves that using transfer learning is more time-consuming than we directly build one network from the start for training. According to our findings, each epoch time spent on non-pretrained model will be approximately 20 minutes, but on pretrained model will be about 6 minutes, that’s because the ResNet pretrained model has already been trained by ImageNet to be able to recognize 1000 classes’ samples. If we want to use pretrained model for this task’s classification task, we need to make little modifications for the last layer by modifying the output features from 1000 outputs to 41 outputs just as figure 4.1 and figure 4.2 shown as below:

Figure 4.1 the last layer of pretrained Resnet50 model

Figure 4.2 modification codes for the last layer in python

1. Results evaluation

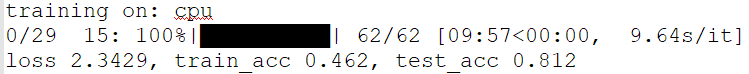
During the training we use three criteria to evaluate the performance of the trained model: cross entropy loss, training accuracy and validation accuracy (each epoch classification accuracy on training and validation datasets), and their values at the epoch 0 are:

Figure 5.1 three criteria values at epoch 0

values at the epoch 29 are:

Figure 5.2 three criteria values at epoch 29

Both results from figure 5.1 and 5.2 could be found in the folder training\_results/outputs\_in\_the\_terminal.txt.

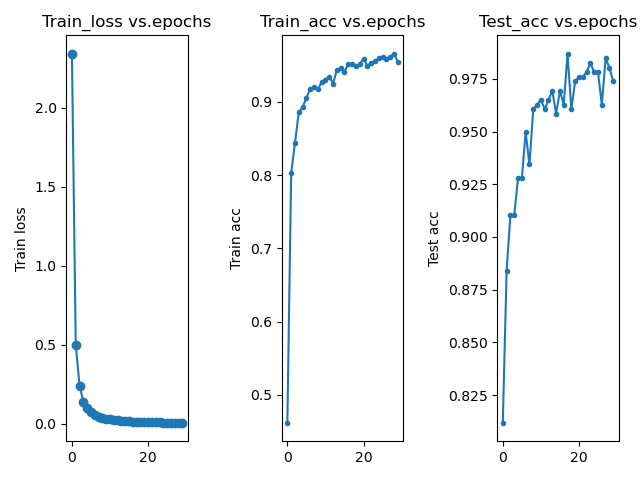
 Visualizing these results in plots, the drawn figure is shown as figure 5.3 below:

Figure 5.3 visualized results of three criteria’s changing trend during training

One thing needs to mention that this test\_acc refers to the classification accuracy on the validation dataset rather than testing dataset, and from the figure 5.3 we can see the loss changing from the one greater than 2 to the one getting to zero, and accuracies on training and testing dataset are increasing to over 90%, which represents that the model’s performance are becoming better after training.

For testing, we take 50% samples of the whole dataset (1414 samples) as testing one, then use the trained model to make prediction by using all testing dataset samples, the classification accuracy of the whole testing dataset is:

Figure 5.4 classification accuracy on testing dataset

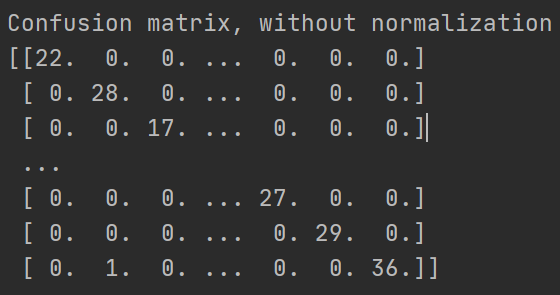
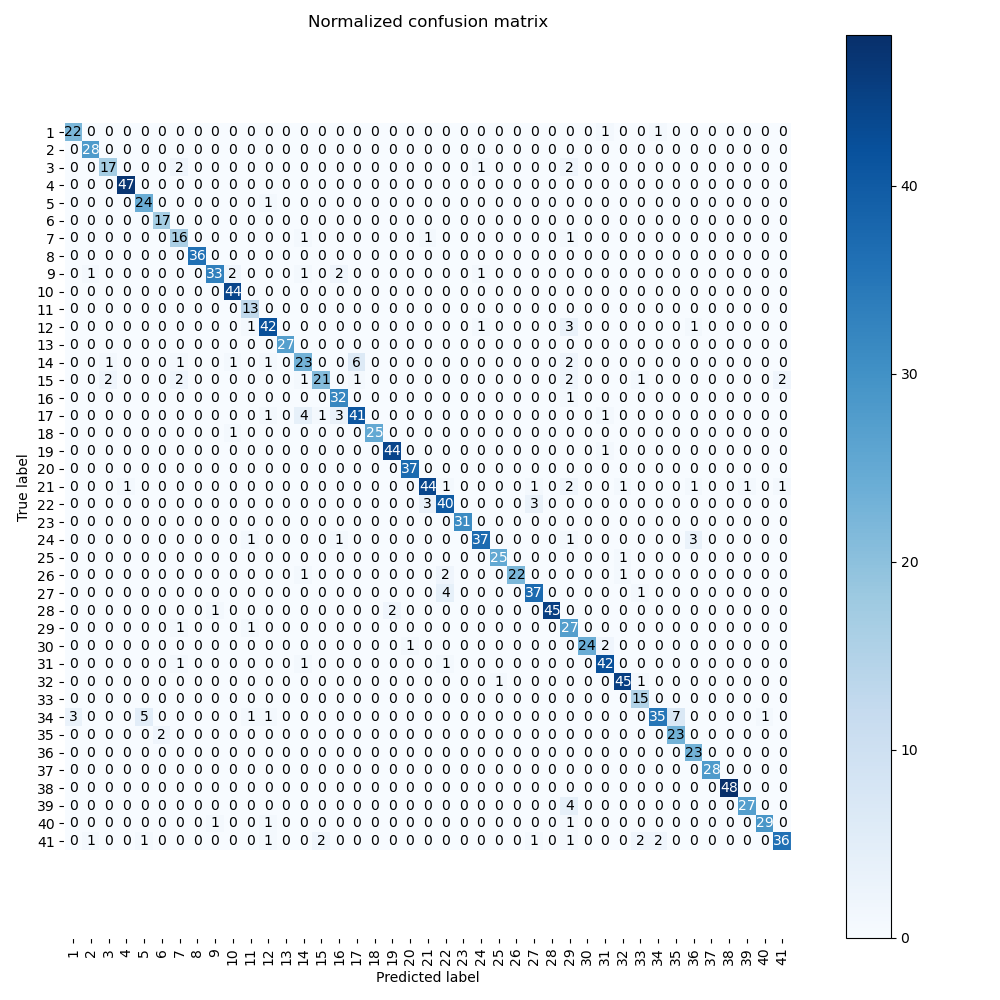
and use the confusion matrix to analyze each class samples’ misclassified conditions, and the confusion matrix during the prediction process is shown as figure 5.5 below:

Figure 5.5 multi-class classification confusion matrix on testing dataset

Visualizing this matrix, we can the figure 5.6 shown as below:

Figure 5.6 the visualized results of confusion matrix

From figure 5.6, all elements on the main matrix diagonal whose true labels are equal to their predicted labels, so values of these elements represent how many data samples are correctly classified for each class, and do the summation for these values, it totally got 1258 samples correctly classified for 1414 testing data samples.

And all elements off the main diagonal presents misclassified samples, for example, we take one element whose true label is 15, predicted label is 7, and this element value is 2, which means that 2 class 15’s samples with have been misclassified as class 7. For other elements off the main diagonal, the analysis method is the same.

1. Conclusion

For better training the network for multi-classes classification task, first we need to consider which architecture of CNN model is suitable for this job, generally more deeper CNN network will bring us good results, but much deeper model means more training time and more complicated model, so we need to balance the relationship between calculation amount and the depth of the CNN model, then we need to consider do some data processing and data augmentation operations to datasets to generate more data samples to train this model to make it become more robust. The last thing we need to consider is the optimizer’s choosing and hyperparameters’ setting, actually we need to try to use various optimizers to make sure which one is suitable for this task, and learning rate should not be set very high from the start, because if you use SGD with momentum or acceleration, the learning rate will be automatically adjusted later, batch size for image classification task could be increased appropriately, too big will bring us more training time, but might will cause the decreased generation capability of the model, too small will contribute to more training time, generally the batch size should be set as 32 or less for the image.