Project 2

Yang Yi

Problem:

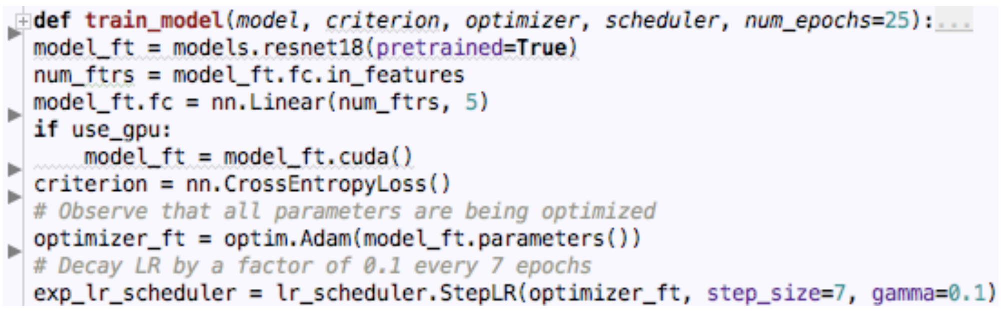
1. Required to train a deep model
2. Test on a test set including flowers.

(at first we know those files including the train, validation)

Data Preprocessing

Just like the project 1, we need to input data and change the data form. But project 2 are based on the IMAGE dataset. So I use the skimage library 1with imread() and resize() function to prepocess those labeled pictures. In this part, I also use numpy,os and pytorch library. After research some CNN paper and blog, I choose the GPU pytorch and resize those picture to 224*224.1

Download the pytorch library, to construct the CNN model2 and find the a best dropout and loss function by reading websites documentations. First use the use compose(), randomsize(), randomhorizontaflip() and totensor() function to transform the Training data and Test data. Second, construct a datasets and dict dataloaders, which data structure are for do input and output in the training_model. Thirdly, set the scheduler and optimizer, criterion by the loss function(crossentropyloss) and dropout function(adam) in the hidden layers and choose the resnet in the last layer. In this project, I just use two hidden layers. In this section, construct the model, I set the epoch is 100 or 40 base on GPU and AWS server. Last, after running the input = training data, output = test data(validation data). when this model has 93 validation accuracy. I store this model. And use this model to analysis the test dataset.



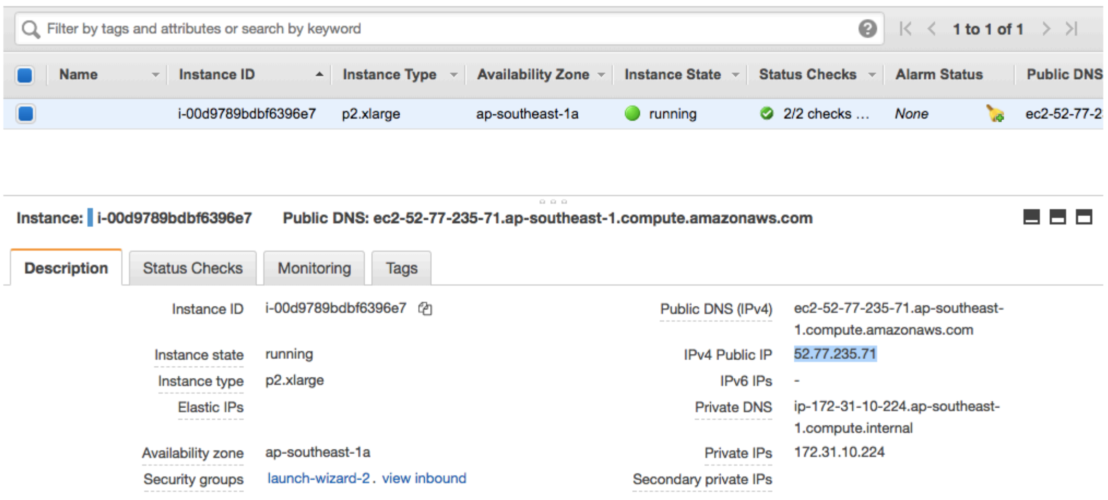
```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):...
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, 5)
if use_gpu:
    model_ft = model_ft.cuda()
criterion = nn.CrossEntropyLoss()
# Observe that all parameters are being optimized
optimizer_ft = optim.Adam(model_ft.parameters())
# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
```

Transfer Learning Model

After step one, I got a accuracy 93% CNN.pkl model. Because this project's dataset is too small, and I though the research finds that transfer learning, it can taking learned knowledge and model in another problem. It can prove the highest model base on the big dataset. 3Which model can be use load() function to directly invoking. It's very popular in those small dataset analysis4. So I choose the SCRATCH and pre-training CNN at large data in AWS server. And using the transfer learning method: Fine-Tuning. Firstly, this model has been trained by another, I can use it with retraining some parameters, this step will be practice many times. secondly, finding the beset parameters to fit this project and setting the threshold. (using the shuffle() in this step to improve accuracy)

Process & Result

That my AWS server with GPU



Filter by tags and attributes or search by keyword — 1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-00d9789bdbf6396e7	p2.xlarge	ap-southeast-1a	running	2/2 checks ...	None	ec2-52-77-2

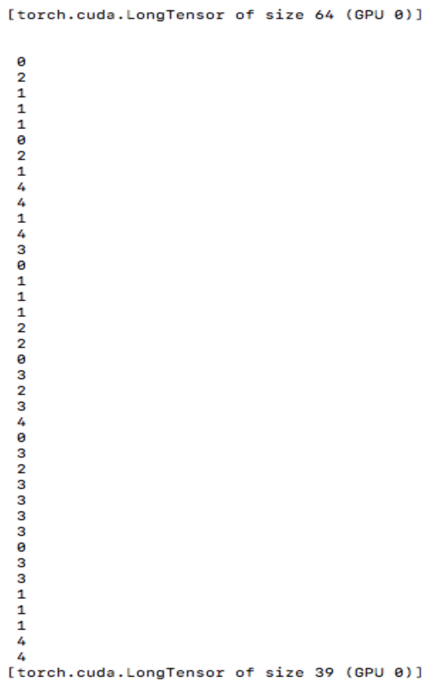
Instance: i-00d9789bdbf6396e7   Public DNS: ec2-52-77-235-71.ap-southeast-1.compute.amazonaws.com

Description | Status Checks | Monitoring | Tags

Instance ID	i-00d9789bdbf6396e7	Public DNS (IPv4)	ec2-52-77-235-71.ap-southeast-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	52.77.235.71
Instance type	p2.xlarge	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-10-224.ap-southeast-1.compute.internal
Availability zone	ap-southeast-1a	Private IPs	172.31.10.224
Security groups	launch-wizard-2. view inbound	Secondary private IPs	

Left feature is running processing, the right feature is result in .txt file.

```
Epoch 27/39
----------
train Loss: 0.0026 Acc: 0.9478
val Loss: 0.0027 Acc: 0.9418

Epoch 28/39
----------
train Loss: 0.0024 Acc: 0.9498
val Loss: 0.0029 Acc: 0.9364

Epoch 29/39
----------
train Loss: 0.0025 Acc: 0.9475
val Loss: 0.0028 Acc: 0.9364

Epoch 30/39
----------
train Loss: 0.0026 Acc: 0.9475
val Loss: 0.0029 Acc: 0.9382

Epoch 31/39
----------
train Loss: 0.0028 Acc: 0.9393
val Loss: 0.0028 Acc: 0.9436

Epoch 32/39
----------
train Loss: 0.0025 Acc: 0.9424
val Loss: 0.0029 Acc: 0.9345

Epoch 33/39
----------
train Loss: 0.0024 Acc: 0.9455
val Loss: 0.0029 Acc: 0.9382

Epoch 34/39
----------
train Loss: 0.0024 Acc: 0.9482
val Loss: 0.0029 Acc: 0.9400

Epoch 35/39
----------
train Loss: 0.0030 Acc: 0.9397
val Loss: 0.0029 Acc: 0.9364

Epoch 36/39
----------
train Loss: 0.0025 Acc: 0.9439
val Loss: 0.0030 Acc: 0.9400

Epoch 37/39
----------
train Loss: 0.0025 Acc: 0.9451
val Loss: 0.0029 Acc: 0.9400

Epoch 38/39
----------
train Loss: 0.0029 Acc: 0.9416
val Loss: 0.0028 Acc: 0.9382

Epoch 39/39
----------
```



```
[torch.cuda.LongTensor of size 64 (GPU 0)]


 0
 2
 1
 1
 1
 0
 2
 1
 4
 4
 1
 4
 3
 0
 1
 1
 1
 2
 2
 0
 3
 2
 3
 4
 0
 3
 2
 3
 3
 3
 3
 0
 3
 3
 1
 1
 1
 4
 4
[torch.cuda.LongTensor of size 39 (GPU 0)]
```

Refences

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
2. Lempitskii, V. S. Introduction to convolutional neural networks.
3. Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In *Advances in neural information processing systems* (pp. 3320-3328).
4. Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, *35*(5), 1285-1298.