MSBD 6000B Project2

Pictures Preprocessing

In this project, I designed independent function to preprocess the labeled pictures. In this part, the library I imported mainly is the skimage. I resized each picture from original size into 224*224.

```
def process(file_):
    im = imread(file_)
    image = resize(im, (224, 224))
    output = file_.split("/")
    fname = os.path.join("val", output[2], output[3])
    imsave(fname,image)
```

I used the pytorch to construct my model. In the pytorch, I also normalized the data of each picture. The detail is the following:

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomSizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Scale(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

Transfer Learning Model

For the deep learning, I applied the transfer learning method here, Fine-Tuning, which really responded very good accuracy on the validation data set. Prior to this, I have tried the usual CNN models, but CNN models didn't work well. The accuracy on the test data is approximate 65% (100 epochs).

Transfer learning:

```
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, 5)
```

I applied the model well pre-trained by other people, named resnet18. Directly downloaded from website and tried to retrain new parameters to fit this flower

classification problem.

For the data input, I applied batch method. 64 samples for a definite batch. And set the shuffle equals True, which means randomly select 64 samples a time. Set 4 readers a time to read the data paralleled, which really improves the speed of the reading process.

```
dataloders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=64,
                                             shuffle=True, num_workers=4)
              for x in ['train', 'val']}
```

Loss Function: nn.CrossEntropyLoss()

```
criterion = nn.CrossEntropyLoss()
```

Optimizing function: Adam()

```
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)
```

The training epoch: 40 times

Training Process and Result

```
Epoch 0/39
----------
train Loss: 0.0125 Acc: 0.7073
val Loss: 0.0472 Acc: 0.5818

Epoch 1/39
----------
train Loss: 0.0093 Acc: 0.7855
val Loss: 0.0110 Acc: 0.8000

……

Epoch 39/39
----------
train Loss: 0.0027 Acc: 0.9393
val Loss: 0.0033 Acc: 0.9218

Training complete in 16m 19s
Best val Acc: 0.927273
```

I trained this model on the AWS cloud computing platform and used the GPU for accelerate.