NCTU Pattern Recognition, Homework 5

- 309553018 林孟學

Result:

```
Accuracy of my model on test-set:  0.8129
```

Code:

```python
def test_model(model, test_data):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    y_pred = []
    for i, (inputs, _) in enumerate(test_data):
        if i % 5 == 0:
            print(f'Batch: {i+1}/20')
        inputs = inputs.to(device)
        # labels = labels.to(device)

        optimizer.zero_grad()

        outputs = model(inputs)
        ret, predictions = torch.max(outputs.data, 1)
        y_pred += [predictions.cpu().numpy()]

    y_pred = np.array(y_pred)
    y_pred = y_pred.reshape((-1))


    # ## DO NOT MODIFY CODE BELOW!
    # please screen shot your results and post it on your report

    assert y_pred.shape == (10000,)

    y_test = np.load("dataset/y_test.npy")
    print("Accuracy of my model on test-set: ", accuracy_score(y_test, y_pred))


# Train or Test
'''
trained_model, history = train_model( ···
)
'''
test_model(resnet50, TestDataLoader)
```

Pretrained Model:

Use pretrained model resnet50.

Modify the last layer of resnet50, and only train these two layers.

```python
    resnet50 = models.resnet50(pretrained=True)
    for param in resnet50.parameters():
        param.requires_grad = False

    fc_inputs = resnet50.fc.in_features
    resnet50.fc = nn.Sequential(
        nn.Linear(fc_inputs, 256),
        nn.ReLU(),
        nn.Dropout(0.4),
        nn.Linear(256, 10),
        nn.LogSoftmax(dim=1)
    )
```

Hyper Parameters:

```
BatchSize = 500
LearningRate = 0.001
EpochCounts = 10
```

Optimizer:

```
optimizer = optim.Adam(resnet50.parameters(), lr=LearningRate)
```

Loss function:

```
loss_function = nn.NLLLoss(),
```

loss(input, class) = -input[class]

The property we guess of the correct label(input[class]) must be as high as possible →

add minus symbol → as low as possible

Preprocessing:

Just resize and normalize it.

No other transforms such as rotate centerCrop…

```
preprocess = transforms.Compose([
    transforms.ToTensor(),
    # transforms.Resize(256),
    # transforms.CenterCrop(224),
    transforms.Resize(224),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

Train:

Train model 4 times on the SAME model (Load model after save previous model).

```
'''
    train 1 → Epochs: 5, BatchSize: 500
    train 2 → Epochs: 5, BatchSize: 500
    train 3 → Epochs: 10, BatchSize: 500
    train 4 → Epochs: 10, BatchSize: 500
'''
```

GPU:

Use GPU for speed up.

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```