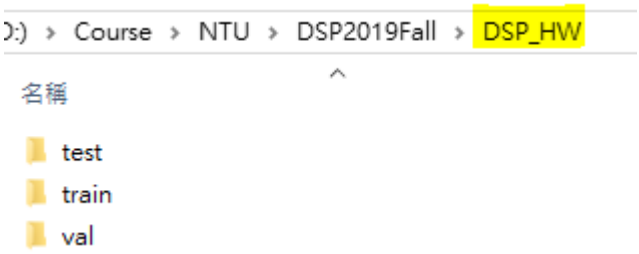
DSP2019_FP_r08921005 黃國郡

Explain how to execute program clearly

The all source code are in r08921005.ipynb,

TA can just open it, and realize what I do.



The important will be described as below.The architecture of my program in folder:

DSP_HW->train(*.jpg, *.npy), test(*.jpg, *.npy), val(*.jpg, *.npy) ()→is the content in folder

將 npy 轉到 spectrogram

Preprocess-change npy to image

```
import matplotlib.pyplot as plt
import numpy as np
import librosa
import librosa.display
import os
#can choose 'train' or 'val'
path = os.path.join('DSP_HW','val')
for filename in os.listdir(path):
    train_voice = os.path.join(path,filename)
    for each_npy_file in os.listdir(train_voice):
        npy_file = os.path.join(train_voice,each_npy_file)
        y = np.load(npy_file,allow_pickle=True)
        S= librosa.feature.melspectrogram(y)
        plt.figure(figsize=(10, 4))
        S_dB = librosa.power_to_db(S, ref=np.max)
        librosa.display.specshow(S_dB, x_axis='time',
                                 y_axis='mel',
                                 fmax=8000)
        plt.colorbar(format='%+2.0f dB')
        plt.title('Mel-frequency spectrogram')
        plt.tight_layout()
        name = each_npy_file.split(".", 1)[0]
        plt.savefig(os.path.join(train_voice,name+'.jpg'))
```

Take test for example,
the output will be as below

Load model

```
def load_model(model,filename):
    model.load_state_dict(torch.load(filename))
    return model
net = Net(num_classes=len(traindata.classes)) # initialize your network
net = load_model(net,"weight.pth")
# Whether to use GPU or not?
device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print("use",device,"now!")
net.to(device)
```

Evaluate on validation data

```
net.eval()
correct = 0
with torch.no_grad():
    for batch_idx, (data, target) in enumerate(valloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
    acc = correct.item() / len(valloader.dataset)
print("Validation Classification Accuracy: %f"%(acc))
```

In this part, we can load the model which we trained before, the upper code just take “weight.pth” for example. Then, we can load the model to validate the *jpg in val to output is validation accuracy.

Put result in dictionary type and write in csv file

In [ ]:
```
# Testing
net.eval()
result = {}
import collections
def sortedDictValues(adict):
    keys = adict.keys()
    keys.sort()
    return [dict[key] for key in keys]

with torch.no_grad():
    for idx, (data,) in enumerate(testloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred_idx = output.data.max(1, keepdim=True)[1]
        pred_class = idx_to_class[pred_idx.cpu().numpy()[0][0]]
        index = os.path.split(testlist[idx])[1][:-4]
        result[int(index)] = classes2label[pred_class]
        result =  dict(sorted(result.items()))
```

Write the result to csv

In [ ]:
```
import csv
with open('2019_12_07.csv', 'w') as csvfile:
    fieldnames = ['id', 'category']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames, lineterminator = '\n')
    writer.writeheader()
    for key in result.keys():
        csvfile.write("%s,%s\n"%(key,result[key]))
```

DSP2019_FP_r08921005 黃國郡

1 Settings for generating spectrograms (at least 2 settings)

1.1 利用 librosa.feature.melspectrogram python library 將個 npy 檔轉成 jpg 檔

librosa.feature.melspectrogram

`librosa.feature.melspectrogram(y=None, sr=22050, S=None, n_fft=2048, hop_length=512, win_length=None, window='hann', center=True, pad_mode='reflect', power=2.0, **kwargs)` [source]

總共做了兩次第一次都使用 default,第二次將 hop_length 調為 1024 使 Overlap 從 75%變為 50%

Overlap 主要被 hop_length parameter 控制

EX:The default frame length is 2048 (for STFT operations), and the default hop is 512, which results in 75% overlap.

- Exp 1. Generate a magnitude spectrum using the hanning window function with a FFT length of 2048 and overlapping of 75% between segments
- Exp 2. Generate a phase spectrum using the hanning window function with a FFT length of 2048 and overlapping of 50% between segments

在 network 不變的情況下只是單純 dataSource 下去 train 得到不同的結果

overlapping 50% 的 validation 效果比 overlapping 75%原本差

Exp 1	Exp 2

Evaluate on validation data
In [10]:
`net.eval()`
`correct = 0`
`with torch.no_grad():`
`  for batch_idx, (data, target) in enumerate(valloader):`
`  data = data.to(device)`
`  target = target.to(device)`
`  output = net(data)`
`  pred = output.data.max(1, keepdim=True)[1]`
`  correct += pred.eq(target.data.view_as(pred)).sum()`
`  acc = correct.item() / len(valloader.dataset)`
`print("Validation Classification Accuracy: %f"%(acc))`
Validation Classification Accuracy: 0.925129
	Evaluate on validation data
	In [11]:
	`net.eval()`
	`correct = 0`
	`with torch.no_grad():`
	`  for batch_idx, (data, target) in enumerate(valloader):`
	`  data = data.to(device)`
	`  target = target.to(device)`
	`  output = net(data)`
	`  pred = output.data.max(1, keepdim=True)[1]`
	`  correct += pred.eq(target.data.view_as(pred)).sum()`
	`  acc = correct.item() / len(valloader.dataset)`
	`print("Validation Classification Accuracy: %f"%(acc))`
	Validation Classification Accuracy: 0.835215

validation accuracy:0.925129	validation accuracy:0.835215

DSP2019_FP_r08921005 黃國郡

2 Settings for your neural network (at least 1 setting)

2.1 Please include implementation details like architecture(LeNet/VGGNet/...), optimizer(Adam/SGD/...), initialization, learning rate, etc.

2.2 全部的實驗都在 LeNet 架構下完成(在 validate 階段測其實都差不多的結果，唯一很特別是:SGD, learning rate :0.001 可能 overfitting 了!最後 accuracy 只有 0.088640)

Exp 1 optimizer:SGD, learning rate :0.05

```
net = Net(num_classes=len(traindata.classes)) # initialize your network
# Whether to use GPU or not?
device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print("use",device,"now!")
net.to(device)
optimizer = optim.SGD(net.parameters(), lr=0.05) # setup your optimizer
criterion = nn.CrossEntropyLoss() # setup your criterion
```

Evaluate on validation data

```
In [10]:
net.eval()
correct = 0
with torch.no_grad():
    for batch_idx, (data, target) in enumerate(valloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
    acc = correct.item() / len(valloader.dataset)
print("Validation Classification Accuracy: %f"%(acc))
```

```
Validation Classification Accuracy: 0.925129
```

Exp 2 optimizer:SGD, learning rate :0.001

```
net = Net(num_classes=len(traindata.classes)) # initialize your network
# Whether to use GPU or not?
device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print("use",device,"now!")
net.to(device)
optimizer = optim.SGD(net.parameters(), lr=0.001) # setup your optimizer
criterion = nn.CrossEntropyLoss() # setup your criterion
```

Evaluate on validation data

```
In [21]:
net.eval()
correct = 0
with torch.no_grad():
    for batch_idx, (data, target) in enumerate(valloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
    acc = correct.item() / len(valloader.dataset)
print("Validation Classification Accuracy: %f"%(acc))
```

```
Validation Classification Accuracy: 0.088640
```

Exp 3 optimizer: Adam, learning rate :0.05

```
In [6]:
net = Net(num_classes=len(traindata.classes)) # initialize your network
# Whether to use GPU or not?
device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print("use",device,"now!")
net.to(device)
optimizer = optim.Adam(net.parameters(), lr=0.05) # setup your optimizer
criterion = nn.CrossEntropyLoss() # setup your criterion
```

```
use cuda now!
```

Evaluate on validation data

```
In [21]:
net.eval()
correct = 0
with torch.no_grad():
    for batch_idx, (data, target) in enumerate(valloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
    acc = correct.item() / len(valloader.dataset)
print("Validation Classification Accuracy: %f"%(acc))
```

```
Validation Classification Accuracy: 0.895214
```

Exp 4 optimizer: Adam, learning rate :0.001

```
net = Net(num_classes=len(traindata.classes)) # initialize your network
# Whether to use GPU or not?
device = 'cpu'
if torch.cuda.is_available():
    device = 'cuda'
else:
    device = 'cpu'
print("use",device,"now!")
net.to(device)
optimizer = optim.Adam(net.parameters(), lr=0.001) # setup your optimizer
criterion = nn.CrossEntropyLoss() # setup your criterion
```

```
use cuda now!
```

```
In [10]:
net.eval()
correct = 0
with torch.no_grad():
    for batch_idx, (data, target) in enumerate(valloader):
        data = data.to(device)
        target = target.to(device)
        output = net(data)
        pred = output.data.max(1, keepdim=True)[1]
        correct += pred.eq(target.data.view_as(pred)).sum()
    acc = correct.item() / len(valloader.dataset)
print("Validation Classification Accuracy: %f"%(acc))
```

```
Validation Classification Accuracy: 0.947074
```

2.3 Report the validation accuracy based on different settings.

- 上個 part 已經對不同種 dataset 得出來的 validation accuracy 做過比較，此 part 不再描述，主要針對不同 optimizer，learning rate,做比較

3 What I have learned for this project.(Difficulties, interesting things, or special techniques)

我將分為以下幾點做感想整理

3.1 Signal vs. Spectrum

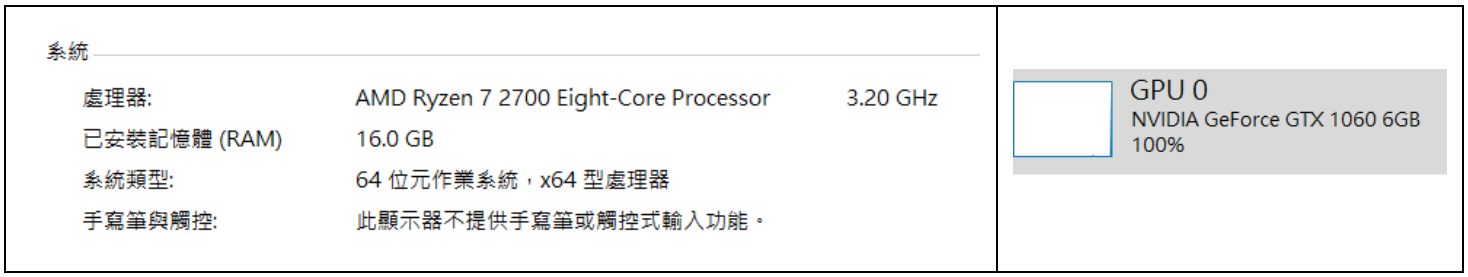
- 本次作業學到 signal 與 spectrum 之間的關係，實踐了課堂上所學所教，非常有感覺

3.2 Neural Networks

- 對於 Neural Networks、CNN 的深入理解和實作，也認識到各種不同的 Network 架構 EX: LeNet/VGGNet/...
- 因為本次作業有機會讓我去對神經網路有更深入研究，甚至去看了本次作業主要所採用的 Lenet 論文:http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

3.3 GPU vs. CPU

- 本次作業我採用 GPU 來 train 然後發現其實只要改一下.cuda() 或 .cpu()其實大多的東西不太用改，滿方便的
- 上網 survey 一下發現 GPU 比 CPU 的運算速度好以外，原來他 train 出來的 model 也會比較好，所以 validation accuracy 也相較好了一點，但就真的只有一點，有可能是因為本次作業運算量比較小。
- 看了一下我電腦規格發現其實有可能是我的 CPU 本身就已經屬於高規格，所以可能不會效能差太多



3.4 To be Improve:或許 dataset 的座標軸，label，colormap 可以拿掉使判斷更精準