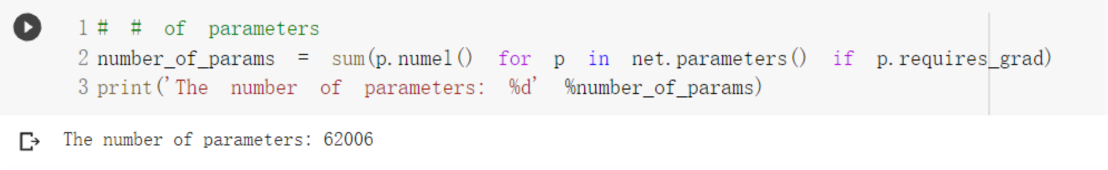
0712534 陳永承

1. Experiment Setup (details of your model, # of parameters)

number of parameters: 62006

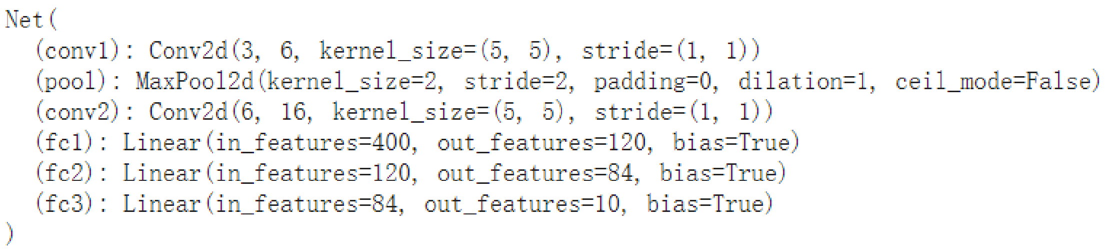


```
# # of parameters
number_of_params = sum(p.numel() for p in net.parameters() if p.requires_grad)
print('The number of parameters: %d' %number_of_params)
```

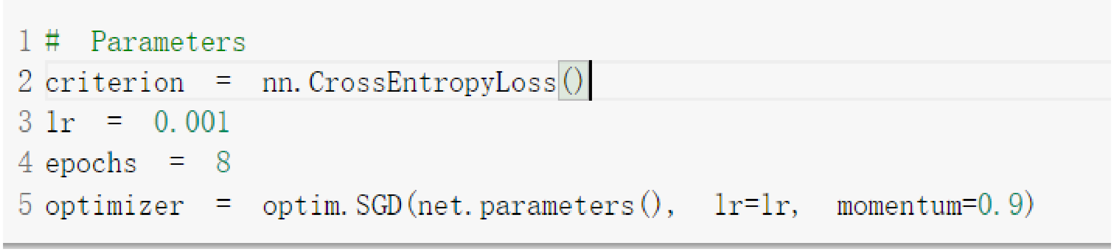
```
The number of parameters: 62006
```

Experiment: colab https://colab.research.google.com/drive/13Ps3B0u53Kl2ccBub3OIj1gXTMAsVIPZ

model:



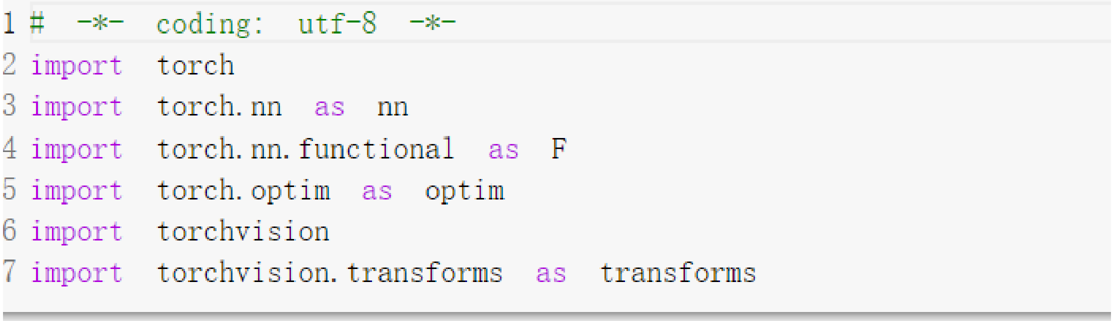
```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```



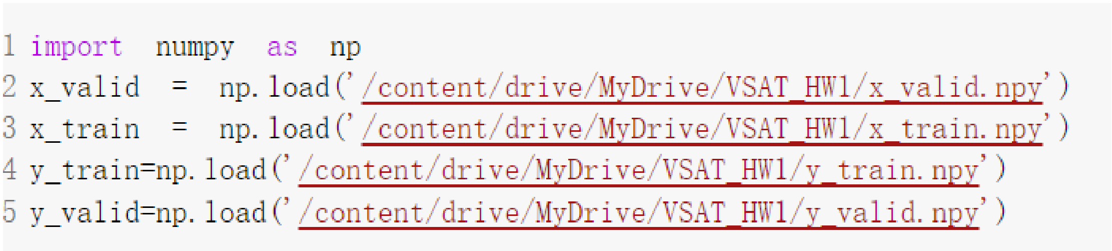
```
# Parameters
criterion = nn.CrossEntropyLoss()
lr = 0.001
epochs = 8
optimizer = optim.SGD(net.parameters(), lr=lr, momentum=0.9)
```

2. Screenshot and explain your code

import 並且讀取資料

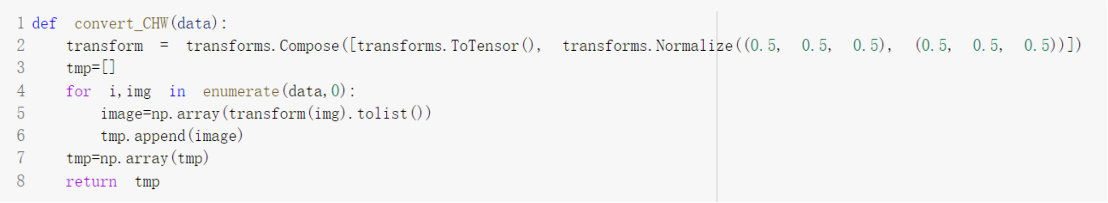


```
# -*- coding: utf-8 -*-
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
```

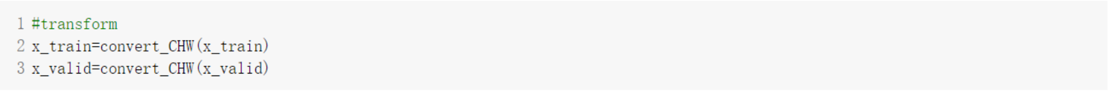


```
import numpy as np
x_valid = np.load('/content/drive/MyDrive/VSAT_HW1/x_valid.npy')
x_train = np.load('/content/drive/MyDrive/VSAT_HW1/x_train.npy')
y_train=np.load('/content/drive/MyDrive/VSAT_HW1/y_train.npy')
y_valid=np.load('/content/drive/MyDrive/VSAT_HW1/y_valid.npy')
```

將資料從 HWC 轉換成 CHW 並且 normalize

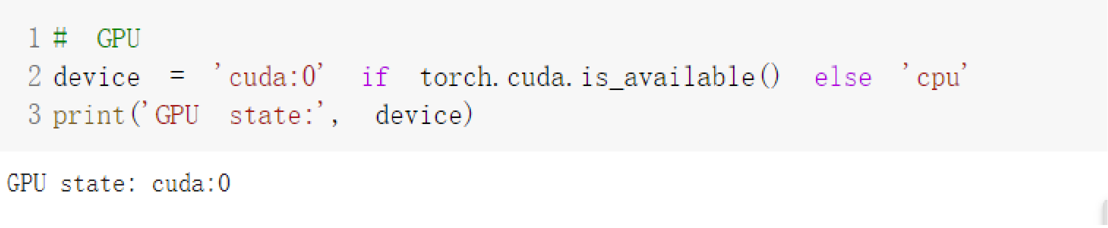


```python
def  convert_CHW(data):
    transform  =  transforms.Compose([transforms.ToTensor(),  transforms.Normalize((0.5,  0.5,  0.5),  (0.5,  0.5,  0.5))])
    tmp=[]
    for  i,img  in  enumerate(data,0):
        image=np.array(transform(img).tolist())
        tmp.append(image)
    tmp=np.array(tmp)
    return  tmp
```



```python
#transform
x_train=convert_CHW(x_train)
x_valid=convert_CHW(x_valid)
```

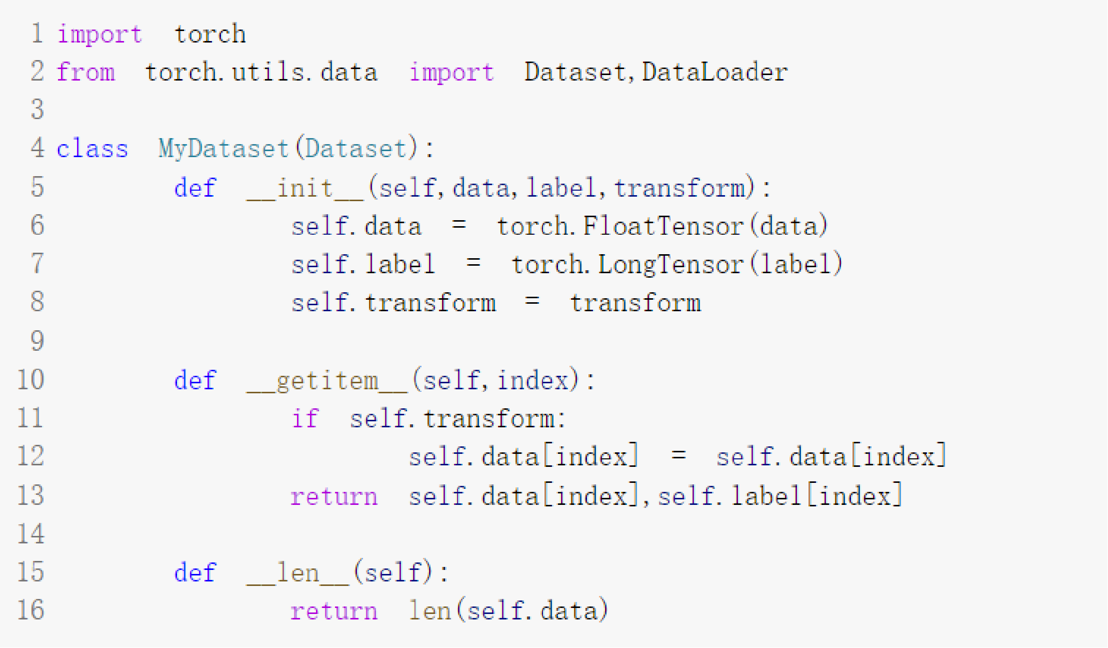
用 GPU 計算



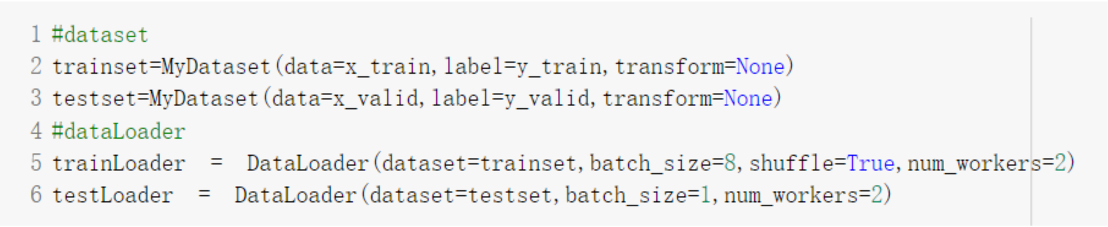
```python
#  GPU
device  =  'cuda:0'  if  torch.cuda.is_available()  else  'cpu'
print('GPU  state:',  device)
```

```
GPU state: cuda:0
```

自訂 dataset 並且將 data 轉換成 data loader 方便 pytorch 訓練



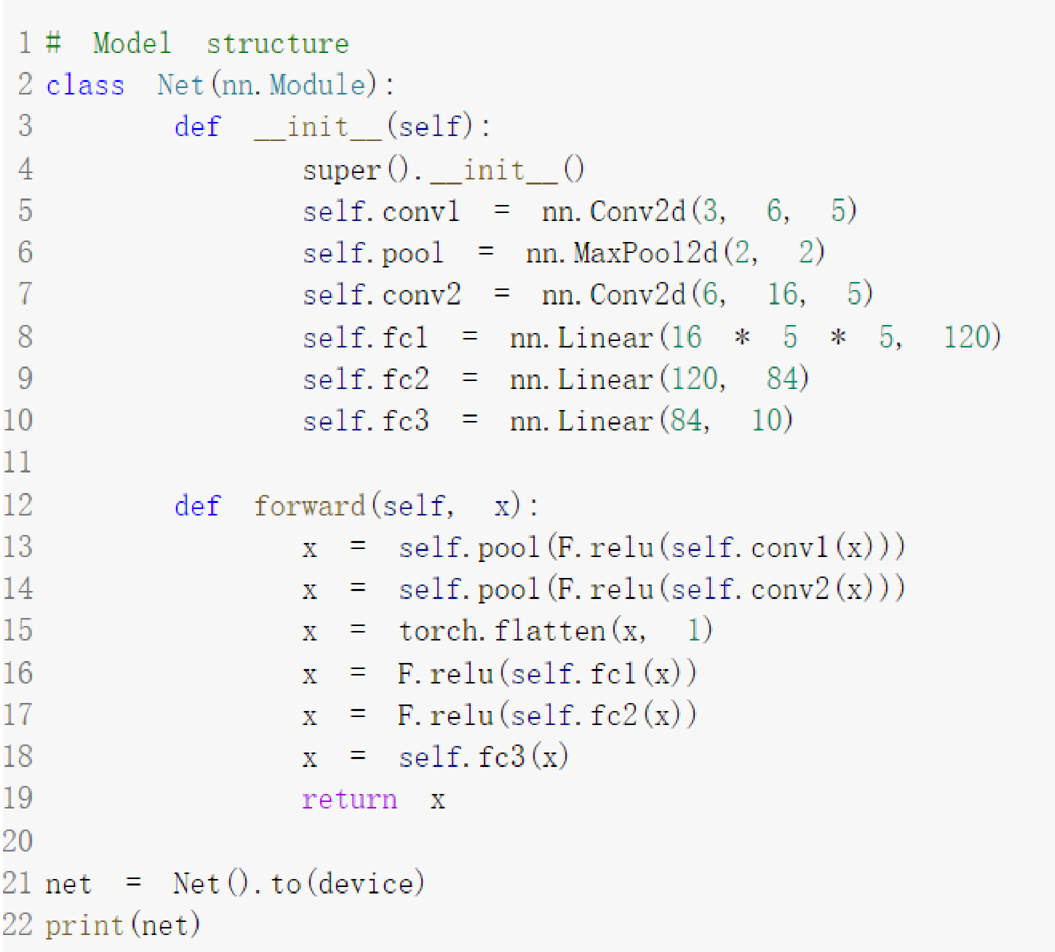
```python
import  torch
from  torch.utils.data  import  Dataset,DataLoader

class  MyDataset(Dataset):
        def  __init__(self,data,label,transform):
                self.data  =  torch.FloatTensor(data)
                self.label  =  torch.LongTensor(label)
                self.transform  =  transform

        def  __getitem__(self,index):
                if  self.transform:
                        self.data[index]  =  self.data[index]
                return  self.data[index],self.label[index]

        def  __len__(self):
                return  len(self.data)
```



```python
#dataset
trainset=MyDataset(data=x_train,label=y_train,transform=None)
testset=MyDataset(data=x_valid,label=y_valid,transform=None)
#dataLoader
trainLoader  =  DataLoader(dataset=trainset,batch_size=8,shuffle=True,num_workers=2)
testLoader  =  DataLoader(dataset=testset,batch_size=1,num_workers=2)
```

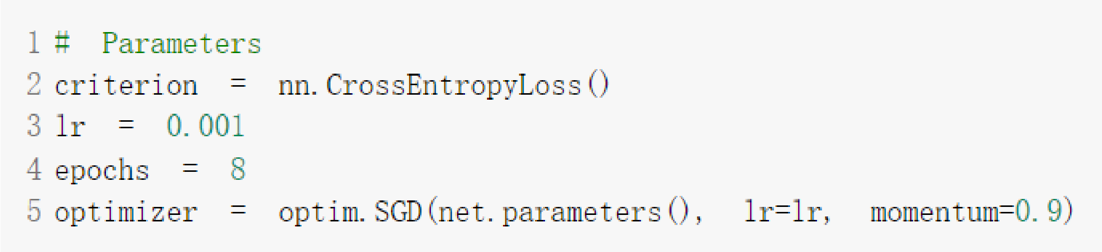
定義 CNN model，定義每個 layer 與 forward，比較重要的是最後一層 output 要是 10 (分 10 類)，第一層 input 是 3。其他層 input 與 output 對到。

reference: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html



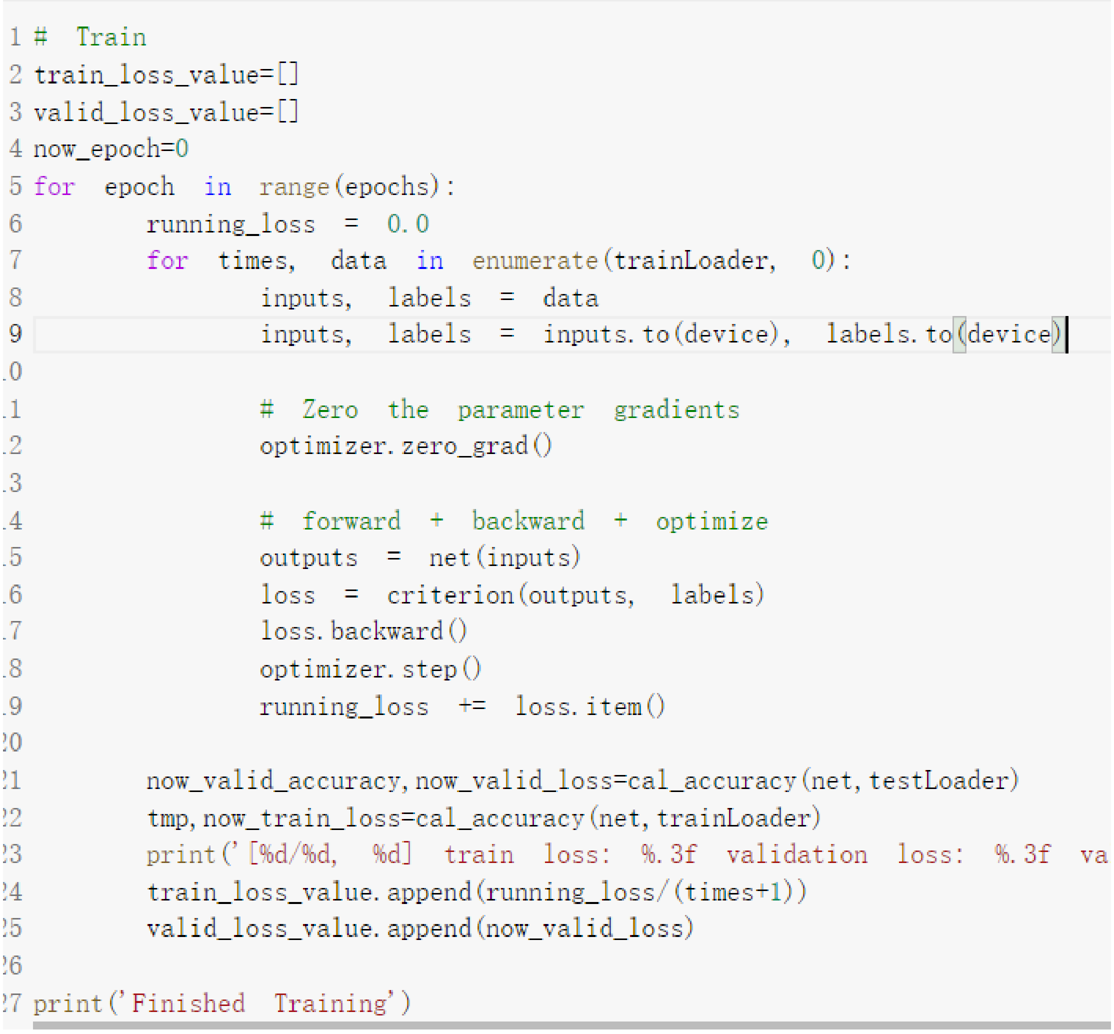
```
# Model structure
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net().to(device)
print(net)
```

設定 learning rate, epochs 等參數，並且以 cross entropy 當作 loss 的計算方式



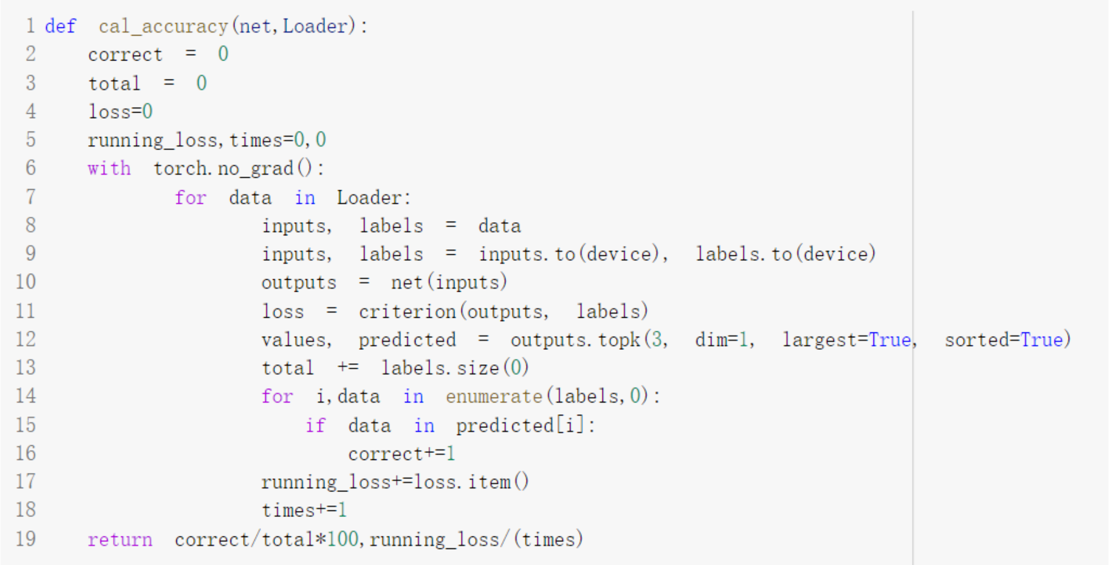
```
# Parameters
criterion = nn.CrossEntropyLoss()
lr = 0.001
epochs = 8
optimizer = optim.SGD(net.parameters(), lr=lr, momentum=0.9)
```

training model 並且將每次的 loss 紀錄，方便後續畫圖，to(device)是將資料利用 GPU 計算，每次都先清掉先前梯度，將當前資料預測完後用 criterion(cross entropy)計算 loss，並且反向傳播與計算梯度，最後 update model

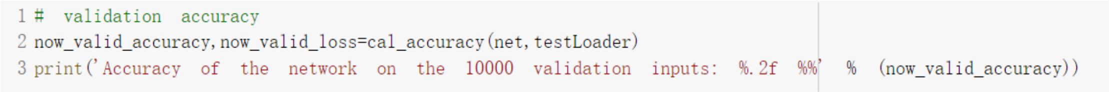


```
# Train
train_loss_value=[]
valid_loss_value=[]
now_epoch=0
for epoch in range(epochs):
    running_loss = 0.0
    for times, data in enumerate(trainLoader, 0):
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    now_valid_accuracy,now_valid_loss=cal_accuracy(net,testLoader)
    tmp,now_train_loss=cal_accuracy(net,trainLoader)
    print('[%d/%d, %d] train loss: %.3f validation loss: %.3f va
    train_loss_value.append(running_loss/(times+1))
    valid_loss_value.append(now_valid_loss)

print('Finished Training')
```

計算 validation data 的 top 3 accuracy，torch.no_grad()是為了不計算梯度，to(device)利用 GPU 計算，outputs.topk()以計算 top k 的分類，最後當 label 在 top 3 的 predicted 裡時表示 correct，反之 incorrect

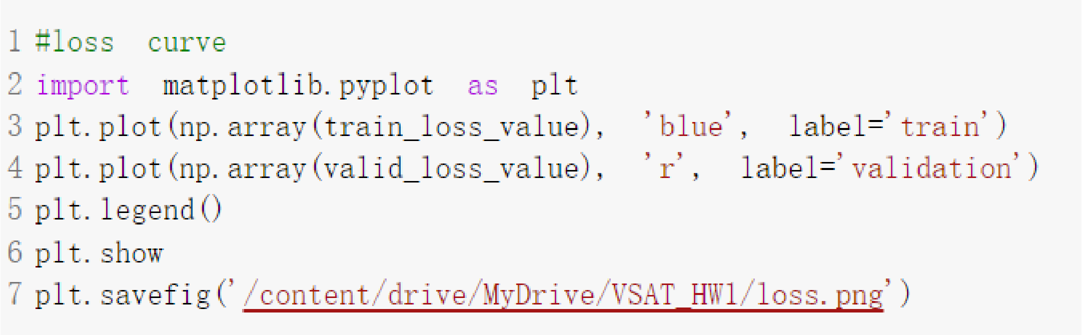


```python
def cal_accuracy(net,Loader):
    correct = 0
    total = 0
    loss=0
    running_loss,times=0,0
    with torch.no_grad():
        for data in Loader:
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = net(inputs)
            loss = criterion(outputs, labels)
            values, predicted = outputs.topk(3, dim=1, largest=True, sorted=True)
            total += labels.size(0)
            for i,data in enumerate(labels,0):
                if data in predicted[i]:
                    correct+=1
            running_loss+=loss.item()
            times+=1
    return correct/total*100,running_loss/(times)
```



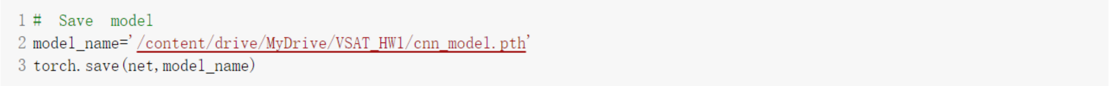
```python
# validation accuracy
now_valid_accuracy,now_valid_loss=cal_accuracy(net,testLoader)
print('Accuracy of the network on the 10000 validation inputs: %.2f %%' % (now_valid_accuracy))
```

畫出 loss 的圖

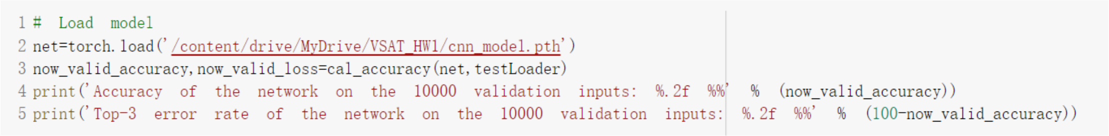


```python
#loss curve
import matplotlib.pyplot as plt
plt.plot(np.array(train_loss_value), 'blue', label='train')
plt.plot(np.array(valid_loss_value), 'r', label='validation')
plt.legend()
plt.show
plt.savefig('/content/drive/MyDrive/VSAT_HW1/loss.png')
```

Save model and load model

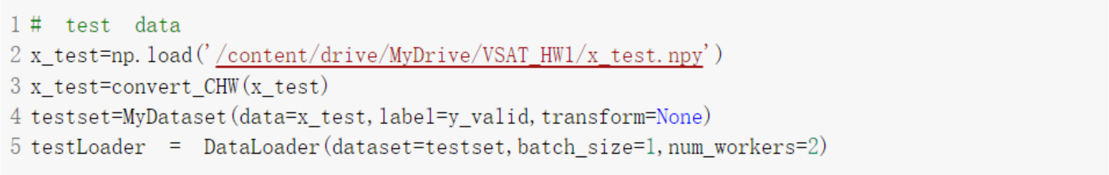


```
# Save model
model_name='/content/drive/MyDrive/VSAT_HW1/cnn_model.pth'
torch.save(net,model_name)
```

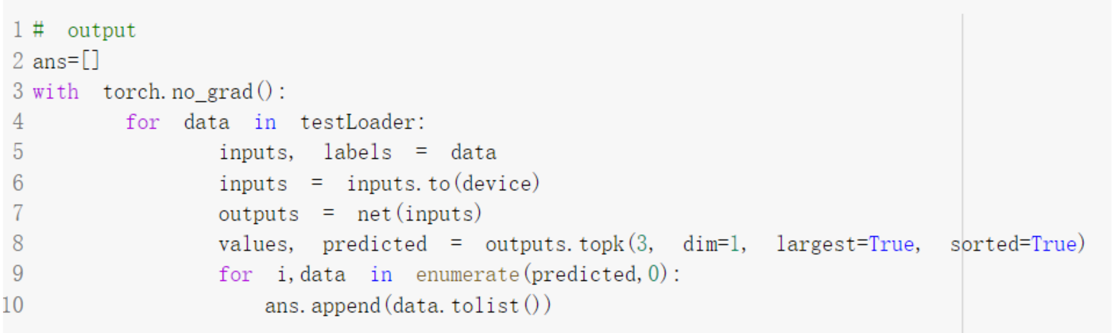


```
# Load model
net=torch.load('/content/drive/MyDrive/VSAT_HW1/cnn_model.pth')
now_valid_accuracy,now_valid_loss=cal_accuracy(net,testLoader)
print('Accuracy of the network on the 10000 validation inputs: %.2f %%' % (now_valid_accuracy))
print('Top-3 error rate of the network on the 10000 validation inputs: %.2f %%' % (100-now_valid_accuracy))
```

讀 test data，轉換成 CHW 後標準化，在轉成 data loader，把每筆資料 top 3 預測結果轉成 list 存下來。



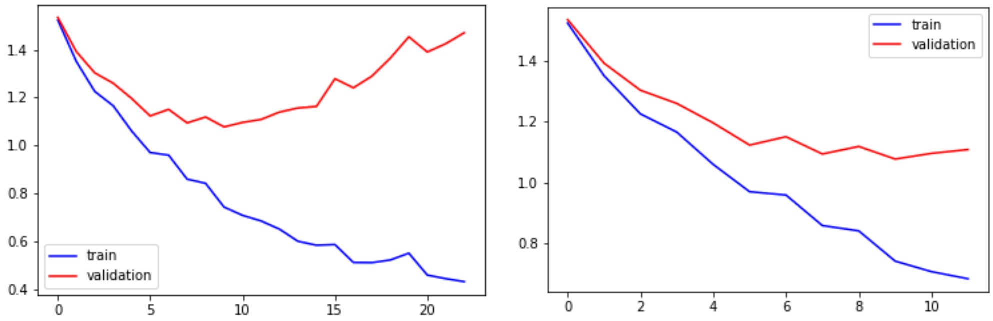
```
# test data
x_test=np.load('/content/drive/MyDrive/VSAT_HW1/x_test.npy')
x_test=convert_CHW(x_test)
testset=MyDataset(data=x_test,label=y_valid,transform=None)
testLoader = DataLoader(dataset=testset,batch_size=1,num_workers=2)
```



```
# output
ans=[]
with torch.no_grad():
    for data in testLoader:
        inputs, labels = data
        inputs = inputs.to(device)
        outputs = net(inputs)
        values, predicted = outputs.topk(3, dim=1, largest=True, sorted=True)
        for i,data in enumerate(predicted,0):
            ans.append(data.tolist())
```

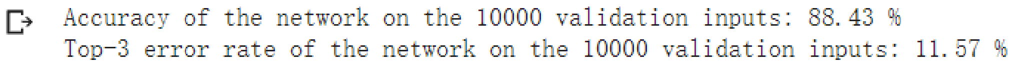
3. Result

training and validation loss curve



橫軸是 epoch，縱軸是 loss，因為在 epoch 超過 12 之後在 validation 的 loss 會大幅上升，推測是因為 training data 的 overfit 導致，因此再截一個 x 軸只有到 12 的圖。

validation data top-3 error rate: 11.57



```
Accuracy of the network on the 10000 validation inputs: 88.43 %
Top-3 error rate of the network on the 10000 validation inputs: 11.57 %
```

4. Problems encountered and discussion

pytorch 相當的便利，即使是第一次寫 CNN model 也不會花到太多的時間，只要搞懂參數設定並且將資料處理好就會自動訓練了，最耗時的部分是將資料寫成 data set，要把圖片轉成 CHW 的形式等。剩下的是調一調 batch size, epochs, learning rate 等參數看哪樣訓練出的結果比較好。但 model 內層數調動與增加似乎不太會讓表現變好，有時反而更差，在 pytorch 的 tutorial 中似乎是很不錯的簡易模型，表現要提高似乎要用更深的模型或對圖片做更多的前處理。