Method description 404410089廖薏萍

what are your reference codes?

使用助教提供的程式碼

參考AlexNet、VGG16的架構<https://medium.com/%E9%9B%9E%E9%9B%9E%E8%88%87%E5%85%94%E5%85%94%E7%9A%84%E5%B7%A5%E7%A8%8B%E4%B8%96%E7%95%8C/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-ml-note-cnn%E6%BC%94%E5%8C%96%E5%8F%B2-alexnet-vgg-inception-resnet-keras-coding-668f74879306>

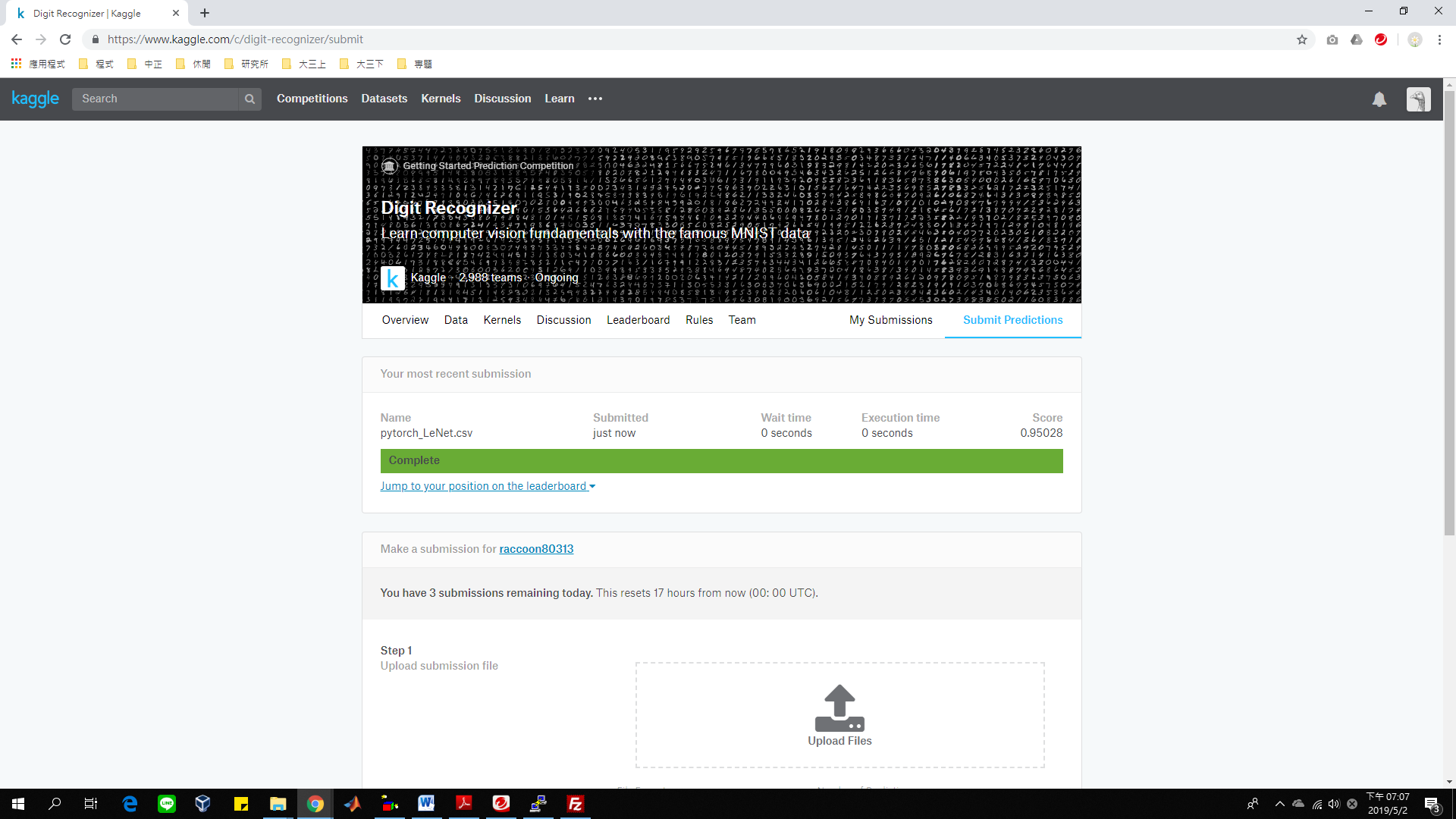
How to run your test?

在CONDA環境中，開啟jupyter notebook

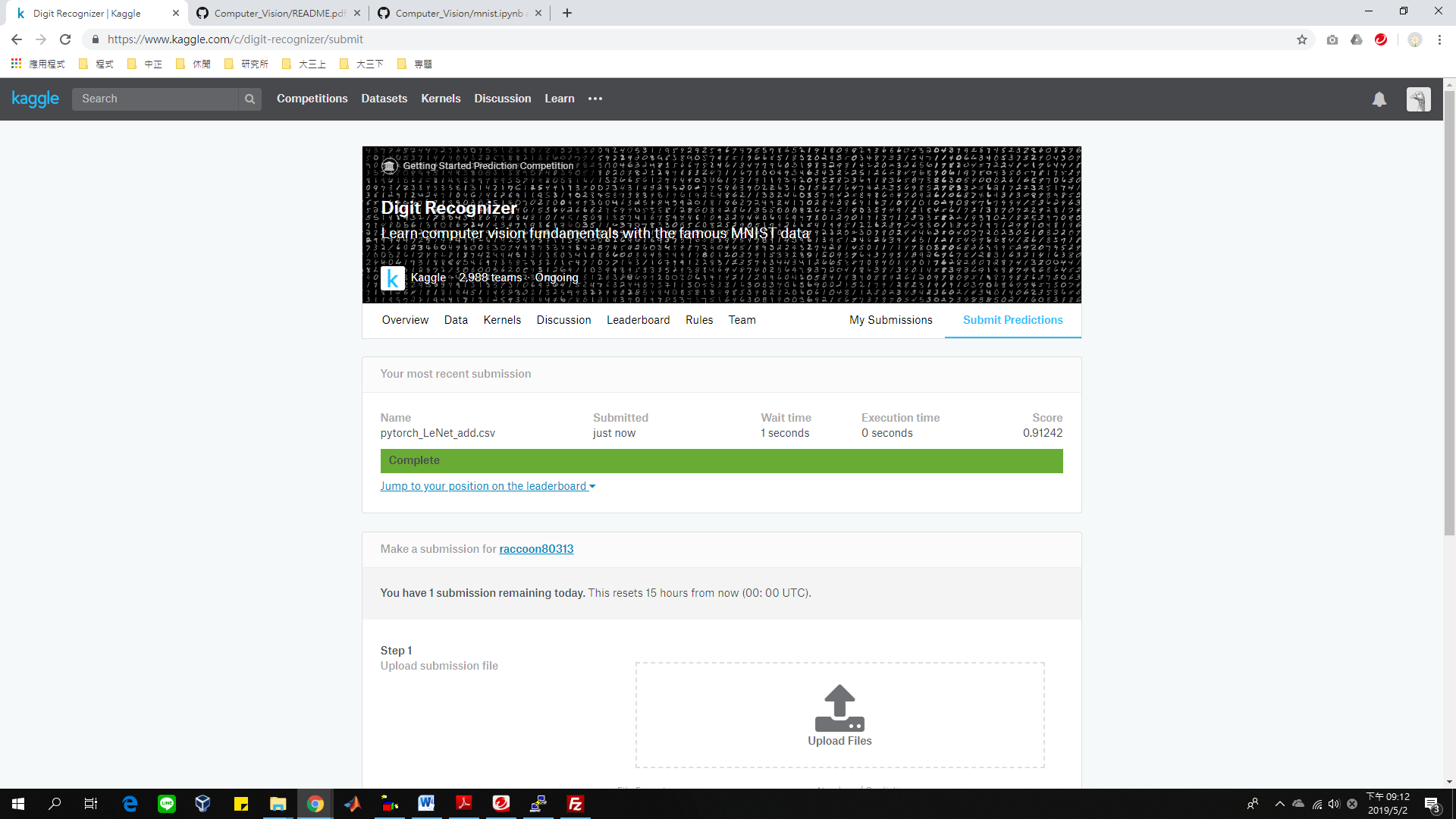
執行mnist.ipynb、mnist-add.ipynb、mnist-new0.96357.ipynb，各別得到pytorch\_LeNet.csv、pytorch\_LeNet\_add.csv、pytorch\_new.csv檔案。

Experimental results

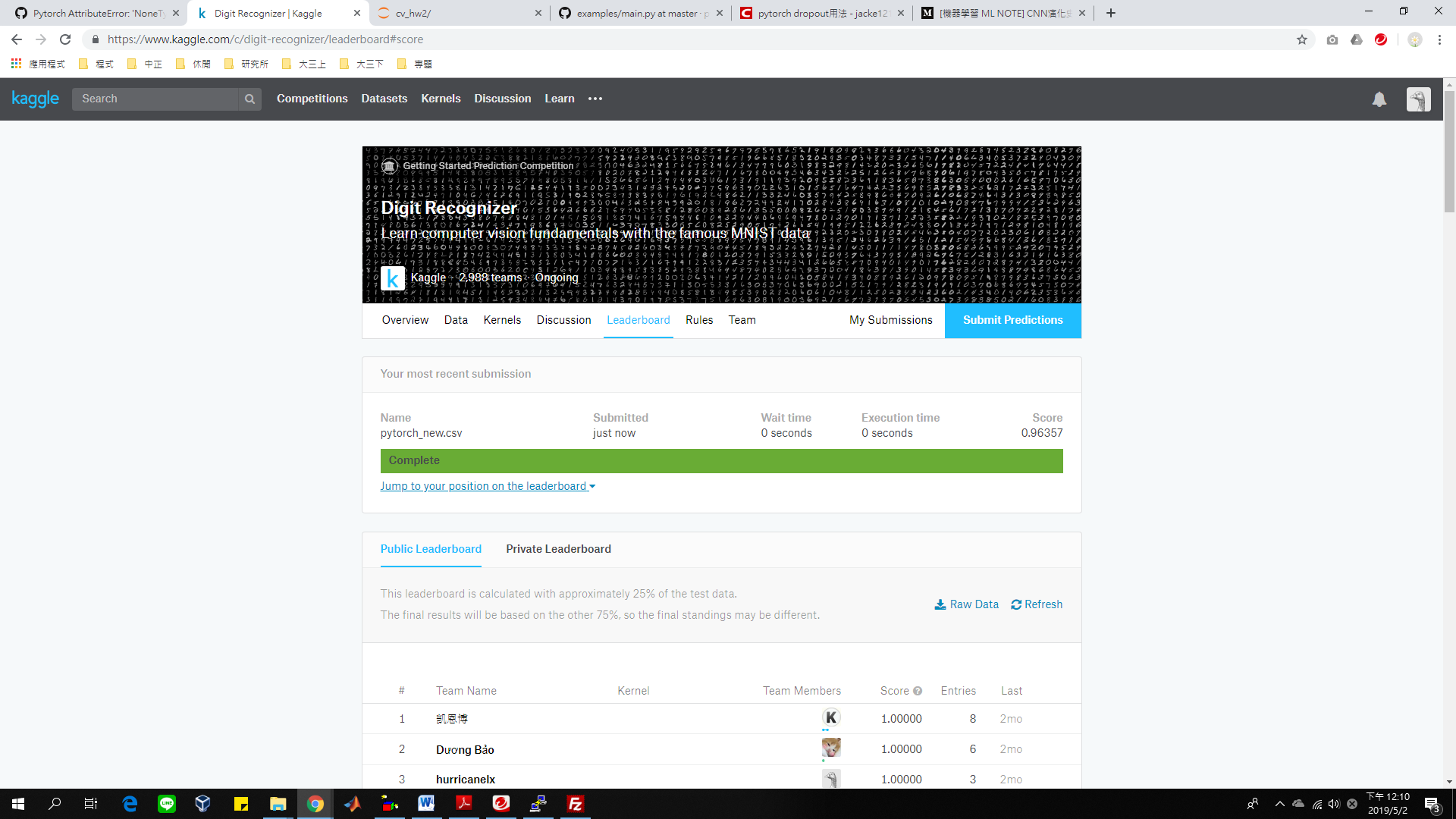
mnist.ipynb執行後產生的pytorch\_LeNet.csv



mnist-add.ipynb執行後產生的pytorch\_LeNet\_add.csv



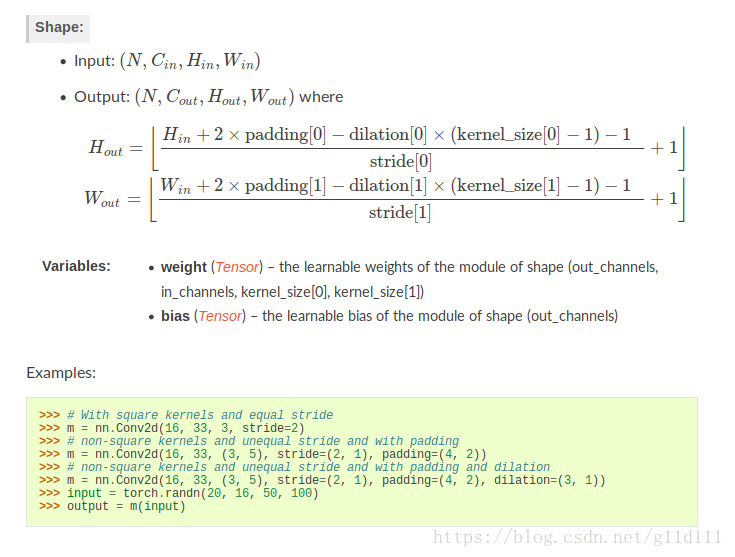
mnist-new.ipynb執行後產生的pytorch\_new.csv



在kaggle上的排名依序為2469、2731、2329

Disscussion

1. mnist.ipynb

在原本的程式碼中，self.conv1 = nn.Conv2d(1, 6, (5,5), padding=2)代表從原本的長寬為28\*28的一張圖，轉變成6張28\*28的圖，轉變計算方式如下

再經過x = F.max\_pool2d(F.relu(self.conv1(x)), (2,2))後，由6\*28\*28轉變成6\*14\*14。

再來self.conv2 = nn.Conv2d(6, 16, (5,5))使6\*14\*14轉變成16\*10\*10。

再來x = F.max\_pool2d(F.relu(self.conv2(x)), (2,2))使16\*10\*10轉變成16\*5\*5。

再經過x = x.view(-1, self.num\_flat\_features(x))將16\*5\*5flatten成400\*1的陣列。

再經過 self.fc1 = nn.Linear(16\*5\*5, 120)

self.fc2 = nn.Linear(120, 84)

self.fc3 = nn.Linear(84, 10) fully connected layer從400個node依序轉變為120個、84個、10個，模型設計結束，因為資料集有10種數字。

1. mnist-add.ipynb

在原本的程式碼再加上一層conv、fc後，準確率變差了。因此我嘗試修改fc1、fc2、fc3、fc4的參數，但是發現嘗試過很多組數字，四層的fc依然不如三層的好。

同時，我也嘗試修改conv1、conv2、conv3的參數，這次的模型設計如下:

1\*28\*28經過self.conv1 = nn.Conv2d(1, 128, (5,5), padding=2)轉變成128\*28\*28

再經過x = F.max\_pool2d(F.relu(self.conv1(x)), (2,2))轉變成128\*14\*14

再經過self.conv2 = nn.Conv2d(128, 256, (3,3))轉變成256\*12\*12

再經過x = F.max\_pool2d(F.relu(self.conv2(x)), (2,2))轉變成256\*6\*6

再經過self.conv3 = nn.Conv2d(256, 512, (3,3))轉變成512\*4\*4

再經過x = F.max\_pool2d(F.relu(self.conv3(x)), (2,2))轉變成512\*2\*2

再經過x = x.view(-1, self.num\_flat\_features(x))將512\*2\*2flatten成2048\*1的陣列。

再經過 self.fc1 = nn.Linear(512\*2\*2, 256)

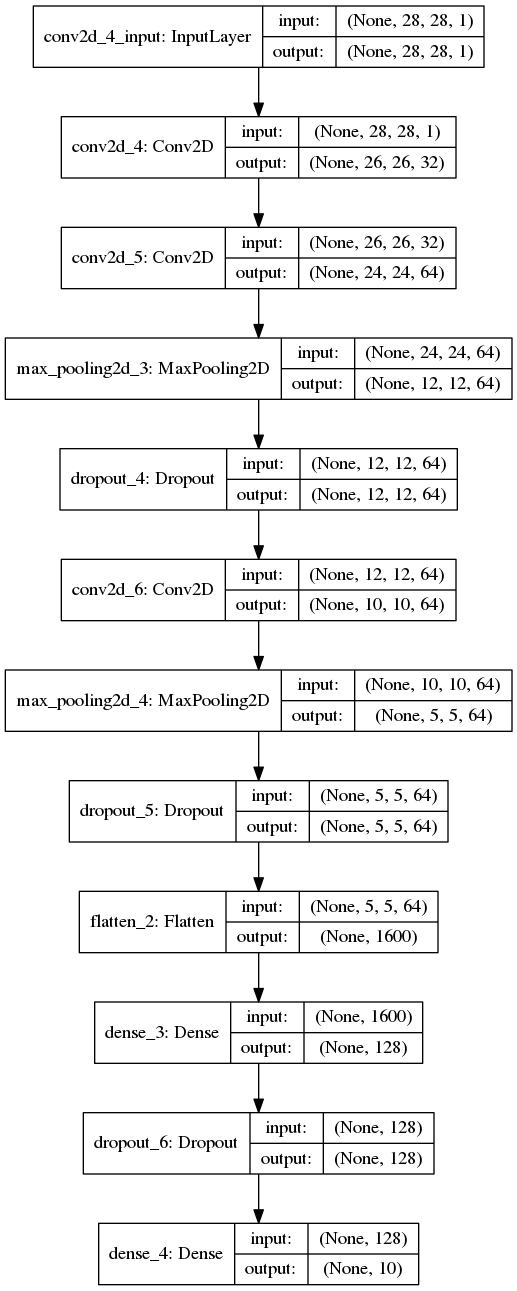
self.fc2 = nn.Linear(256,128)

self.fc3 = nn.Linear(128,64)

self.fc4 = nn.Linear(64, 10) fully connected layer從2048個node依序轉變為256個、128個、64個、10個，模型設計結束，因為資料集有10種數字。

1. mnist-new.ipynb

因為修改後的準確率沒有提升，所以我參考了下圖的模型設計。

與原始的LeNet-5較大的不同在於有使用Dropout，還有加了一層conv、少了一層fc。

1\*28\*28經過self.conv1 = nn.Conv2d(1, 32, (3,3))轉變成32\*26\*26

再經過self.conv2 = nn.Conv2d(32, 64, (3,3))轉變成64\*24\*24

再經過x = F.max\_pool2d(x, 2, 2)轉變成64\*12\*12

經過x = F.dropout(x, p=0.25, training=self.training)後

再經過self.conv3 = nn.Conv2d(64, 64, (3,3))轉變成64\*10\*10

再經過x = F.max\_pool2d(x, 2, 2)轉變成64\*5\*5

經過x = F.dropout(x, p=0.35, training=self.training)後

再經過x = x.view(-1, self.num\_flat\_features(x))將64\*5\*5flatten成1600\*1的陣列。

再經過 self.fc1 = nn.Linear(1600, 128)

self.fc2 = nn.Linear(128,10)

fully connected layer從1600個node依序轉變為128個、10個，模型設計結束，因為資料集有10種數字。

在mnist-new.ipynb中，採用了dropout，使模型較不容易發生overfitting。準確率在這三個實驗中為最高的。

Problem and difficulties

在與助教給的程式碼相同的前提下，只增加一層conv、一層fc，其準確率竟然會下降。我推測可能是因為這次的圖片大小沒有很大(28\*28)，想要增加layer的話，每層layer擷取的特徵數會更細部化，導致增加了太多特徵細節，無法更準確的辨識。