深度學習HW4 LeNet-5

吳玫萱RE6091054

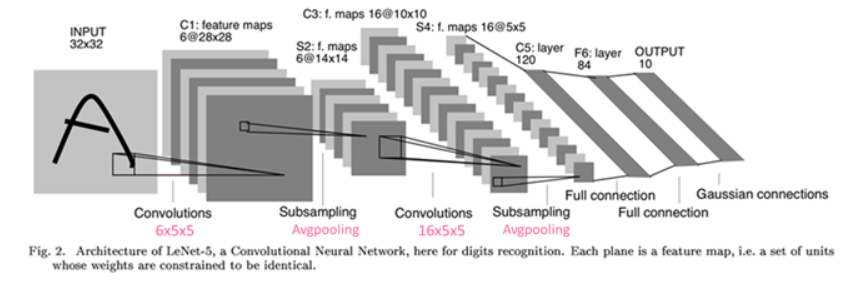
**作業大綱：**

分別使用Pytorch 和 Tensorflow架構LeNet模型，介紹實驗方法並展示分別試做的模型效果。

**LeNet-5模型介紹：**

LeNet-5 是由 Yann LeCun 團隊提出的網路架構，是卷積神經網路的始祖。其架構由兩個卷積層、池化層、全連接層以及最後一層 Gaussian 連接層所組成，早期用來辨識手寫數字圖像

圖一是原論文中模型架構展示圖，我們可以看到 LeNet-5 的網路架構共有七層：卷積層 (Convolutions, C1)、池化層 (Subsampling, S2)、卷積層 (C3)、池化層 (S4)、全連接卷積層 (C5)、全連接層 (F6)、Gaussian 連接層 (output)。



圖一：LeNet-5論文中模型架構圖

由圖一中我們可以看到原論文中輸入為32\*32\*1的圖象特徵輸入，而在此次作業中我將所有圖片重新制定大小為64\*64\*3的圖片作為輸入，故以下用此次實做的網路架構進行網路架構介紹。

Lenet-5除輸入層外由7層網路構成：

(1).卷積層Conv1：

接受64\*64\*3 的圖像輸入，本層包含6個大小為5\*5，步長為1\*1的卷積核，padding類型為'valid'(即沒有padding)。輸出神經元：60\*60\*6

(2).池化（pooling）層:

對上一層的輸出做2\*2的max pooling，輸出神經元形狀為30\*30\*6。

(3).卷積層Conv2：

接受30\*30\*6的輸入，本層有16個大小為5\*5，步長為1\*1的卷積核，同第一個卷積層，本層的padding類型為'valid’。輸出神經元為26\*26\*16。

(4).池化層：

對上層輸出做2\*2的max pooling，輸出神經元形狀為13\*13\*16。

(5).全連接層fc1：

本層將上層的13\*13\*16 = 2704神經元展開作為輸入，本層包含1024個神經元。

(6).全連接層fc2：

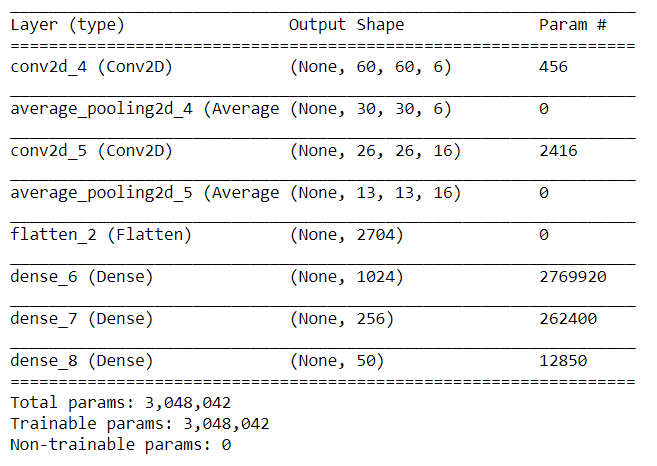
本層包含256個神經元。

(7).輸出層：

本層包含50個神經元，分別代表數字0到49。

(8).啟動函數

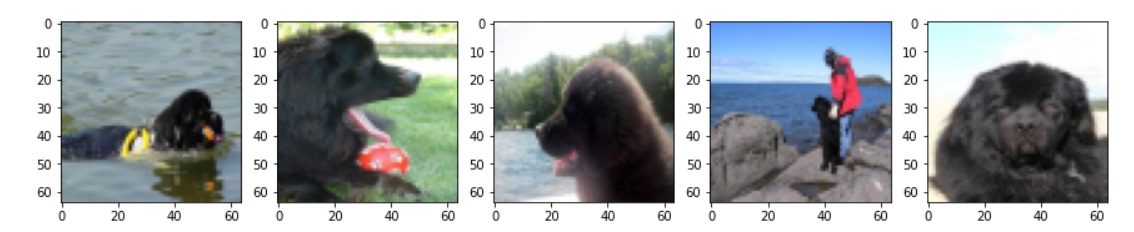
在Tensorflow架構中前7層採用論文中使用的tanh啟動函數，在Pytorch架構中前7層採用Relu為啟動函數，輸出層論文採用的是Guassian Connection，在實踐的代碼中我用的是現在採用的是softmax 函數



圖二：模型架構總結展示圖

**實驗設置：**

資料集共分為三個部分訓練集、驗證集、測試集，訓練集包含63325 張照片，驗證集包含450張照片, 測試集包含450張照片，大小不一，共50個類別，在資料前處理的部分第一步將所有照片的(H, W, channel)皆改為(64, 64, 3)作為模型輸入，圖三為調整圖片大小後的視覺化示例圖。第二步是做規一化，在tensorflow中直接分別對訓練集、驗證集、測試集各自做規一化，在Pytorch中則是進行針對每個mini batch的規一化，規一化即減掉該圖片子集平均再除以其標準差，並在訓練時使用批次訓練，參數將分別在用Pytorch和Tensorflow架構的兩模型結果展示中介紹。



圖三：調整圖片大小為(64, 64, 3)後的視覺化示例圖

設備：Python (version 3.8.5) on win32 (GTX1650 CUDA Version: 10.2)

**Tensorflow LeNet-5模型結果展示：**

參數設置：

Batch size = 32

Epochs = 20

Optimizer = SGD

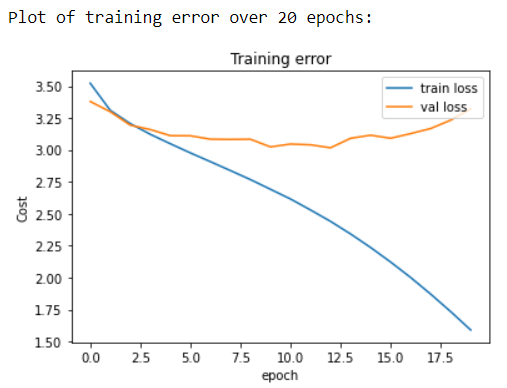
Loss = Categorical Crossentropy

Input：(32, 64, 64, 3)

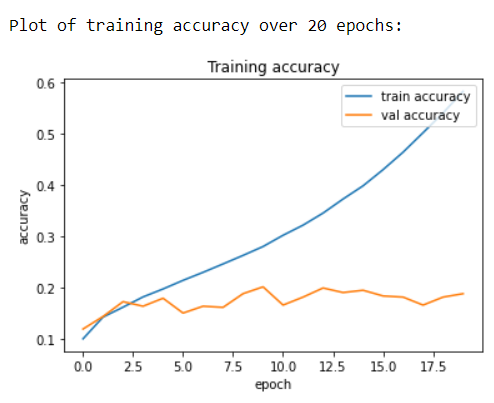
Tensorflow LeNet-5模型結果：

|  |  |  |  |
| --- | --- | --- | --- |
|  | train | vaild | test |
| accuracy | 0.5976 | 0.1889 | 0.0577778 |
| loss | 1.5632 | 3.3219 | 4.7033 |

圖四為Tensorflow LeNet-5模型訓練集和測試集的損失變化展示圖，圖五則為Tensorflow LeNet-5模型訓練集和測試集的準確率變化展示圖。



圖四：Tensorflow LeNet-5模型訓練集和測試集的損失變化展示圖



圖五：Tensorflow LeNet-5模型訓練集和測試集的準確率變化展示圖

**Pytorch LeNet-5模型結果展示：**

參數設置：

Batch size = 16

Epochs = 10

Learning rate = 0.001

Input：(16, 64, 64, 3)

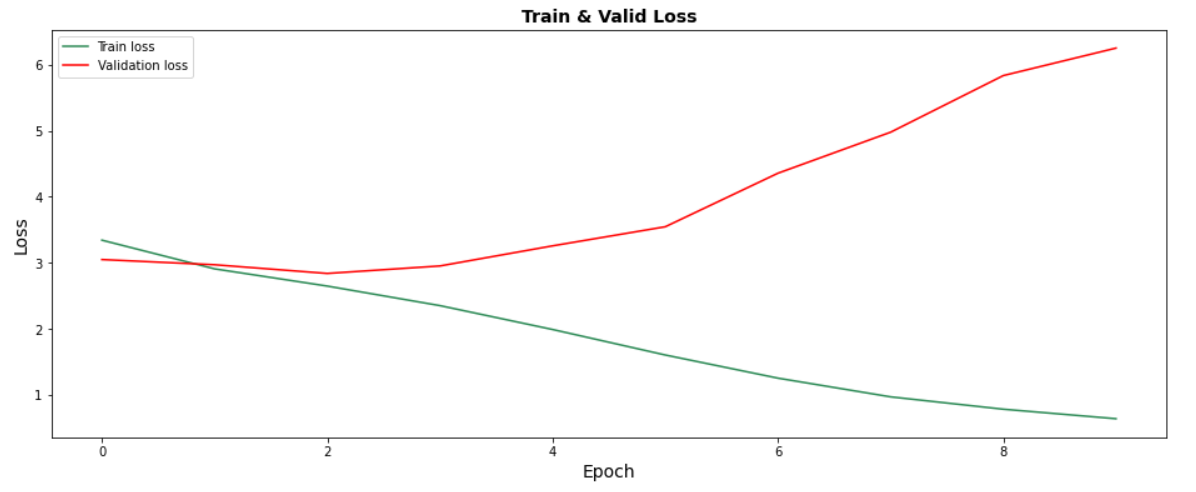
Optimizer = Adam

Loss = CrossEntropyLoss

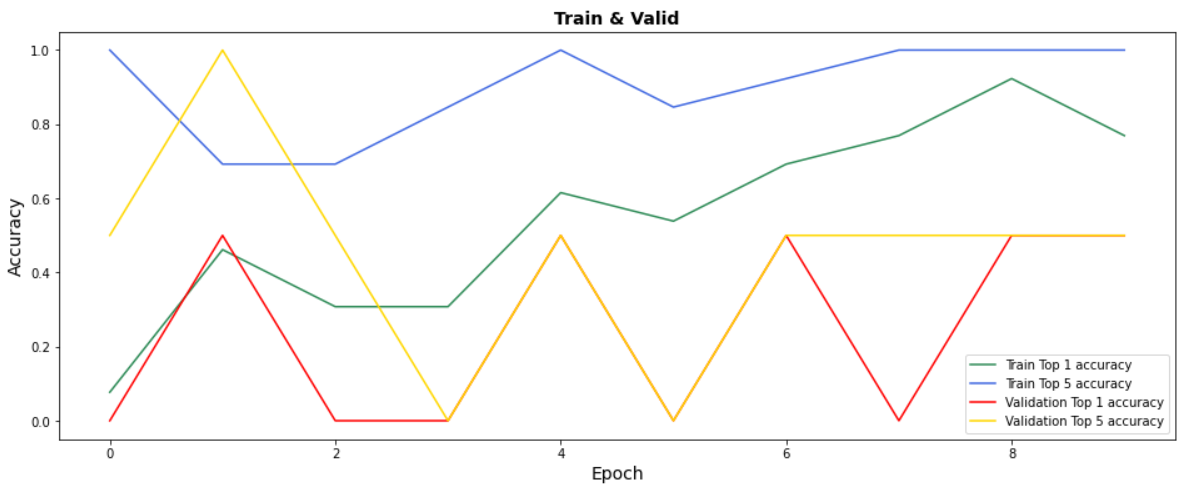
Pytorch LeNet-5模型結果：

|  |  |  |  |
| --- | --- | --- | --- |
|  | train | vaild | test |
| accuracy top1 | 0.76923 | 0.5 | 0.2 |
| loss | 0.63474 | 6.25482 | 5.996452 |

圖六為Pytorch LeNet-5模型訓練集和測試集的損失變化展示圖，圖七則為Pytorch LeNet-5模型訓練集和測試集的準確率top1與top5變化展示圖。



圖四：Pytorch LeNet-5模型訓練集和測試集的損失變化展示圖



圖七；Pytorch LeNet-5模型訓練集和測試集的準確率top1與top5變化展示圖

**總結：**

在架構Tensorflow與Pytorch LeNet-5模型和實驗設置上共有兩處不同，一是在Tensorflow架構中使用的是tanh做為啟動函數，在Pytorch架構中則採用Relu做為啟動函數，二是規一化的部分，在tensorflow中直接分別對訓練集、驗證集、測試集各自做規一化，在Pytorch中則是在訓練時針對每個mini batch做規一化，實驗發現總體而言Pytorch的實驗結果更好一些，可能原因以兩個，一是推測使用mini batch做規一化能有更好的泛化能力，二是Relu做為現在模型常用的啟動函數，推測其能更好的幫助模型迭代。

**參考資料：**

Tensorflow

<https://hackmd.io/@bouteille/S1WvJyqmI#LeNet-5-Summary-and-Implementation>

Pytorch

<https://jennaweng0621.pixnet.net/blog/post/403588244-%5Bpytorch%5D-lenet%E7%B6%B2%E8%B7%AF%E7%B5%90%E6%A7%8B-%2B-mnist%E6%95%B8%E5%AD%97%E8%A8%93%E7%B7%B4>

<https://medium.com/ching-i/%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1-cnn-%E7%B6%93%E5%85%B8%E6%A8%A1%E5%9E%8B-lenet-alexnet-vgg-nin-with-pytorch-code-84462d6cf60c>

<https://zhuanlan.zhihu.com/p/29716516>