DEEP LEARNING assignment 2

Linear Classifer

苗鈺  
 *Institute of Data Science*  
 *National Cheng Kung University*Tainan, Taiwan  
a94035@gmail.com

# Introduction

本次作業須建立三種版本的lenet5，分別為keras靜態圖版本的lenet5、keras動態圖版本的lenet5、pytorch版本的lenet5，並將此三個版本的lenet5與上一次作業手刻的lenet5進行各項效能上的比較。

# Data

本次作業使用的資料為圖像資料，共有63325張影像，所有影像被分為50個類別，之後又被切割為三個子資料集用於訓練模型及驗證模型效能，三個子資料分別為train dataset、test dataset、validation dataset，分別有63325、450、450張影像。

|  |  |  |
| --- | --- | --- |
| 資料集名稱 | 總影像數 | 包含類別數 |
| Train | 63325 | 50 |
| Test | 450 | 50 |
| Validation | 450 | 50 |

# layers definition

## convolution Layer(捲積層)

捲積層是捲積神經網路的主要特色，假設有一個3\*3的捲積層，計算方法如下:

其中f(c,h,w)代表f在此位置下的數值，將所有數值加總後輸出，算法範例如圖一所示。

Local gradient:

一張含有 文字, 電子產品 的圖片

自動產生的描述

圖一:圖解捲積運算

## activate function(激活函數)

本次報告中使用到的激活函數共有三種:

### ReLU

### Function:

Local gradient:

### Sigmoid

### Function:

Local gradient:

### Sigmoid\_m

### Function:

Local gradient:

## Maxpooling(最大池化層)

假設最大池化層的大小為n\*n，則計算時會選擇這n\*n的範圍的最大值作為往下傳遞的值，而計算backward時，gradient則是被選擇的那一個神經元會繼承上游的gradient，其餘神經元的gradient為0。

最大池化層計算方式可參考圖二。

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

|  |  |
| --- | --- |
| 6 | 8 |
| 3 | 4 |

圖二:圖解最大池化層

## Fully connected layer(全連接層)

全連接層跟基礎的神經網路一樣，其gradient為該層所有神經元權重之和。

## Loss function

### Soft max

為了優化計算，避免數值溢位，分子分母同除，並改良為下式:

### Cross entropy

k代表預測目標的總類別數，x代表神經網路中forward最後傳遞的數值向量，y(k)則代表真實答案的數值。

# model

本次實驗使用了四種模型，每個模型輸出的維度大小及該層神經網路的重要參數將以表格的形式進行呈獻

## keras(靜態圖版本)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | chennel | size | stride | Activate function |
| image | 3 | 32\*32 |  |  |
| flatten | 1 | 3072 |  |  |
| Fully conected1 | 1 | 50 |  | ReLU |
| softmax | 1 | 1 |  |  |

## keras(動態圖版本)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | chennel | size | stride | Activate function |
| image | 3 | 32\*32 |  |  |
| flatten | 1 | 3072 |  |  |
| Fully conected1 | 1 | 192 |  | ReLU |
| Fully conected2 | 1 | 50 |  |  |
| softmax | 1 | 1 |  |  |

## pytorch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | chennel | size | stride | Activate function |
| image | 3 | 32\*32 |  |  |
| Convolution1(5\*5) | 6 | 28\*28 | 1 |  |
| Maxpooling1(2\*2) | 6 | 14\*14 | 2 |  |
| Batch normal1 | 6 | 14\*14 |  | ReLU |
| Convolution2(5\*5) | 16 | 10\*10 | 1 |  |
| Maxpooling2(2\*2) | 16 | 5\*5 | 2 |  |
| Batch normal2 | 16 | 5\*5 |  | ReLU |
| flatten | 1 | 400 |  |  |
| Fully conected1 | 1 | 84 |  | ReLU |
| Fully conected2 | 1 | 50 |  |  |
| softmax | 1 | 1 |  |  |

## 手刻網路

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| layer | chennel | size | stride | Activate function |
| image | 3 | 32\*32 |  |  |
| Convolution1(3\*3) | 6 | 30\*30 | 1 |  |
| Maxpooling1(2\*2) | 6 | 15\*15 | 2 |  |
| Batch normal1 | 6 | 15\*15 |  | Sigmoid\_m |
| Convolution2(3\*3) | 16 | 13\*13 | 1 |  |
| Maxpooling2(2\*2) | 16 | 7\*7 | 2 |  |
| Batch normal2 | 16 | 7\*7 |  | Sigmoid\_m |
| Convolution3(3\*3) | 120 | 5\*5 | 1 |  |
| flatten | 1 | 3000 |  |  |
| Fully conected1 | 1 | 84 |  | Sigmoid\_m |
| Fully conected2 | 1 | 50 |  |  |
| softmax | 1 | 1 |  |  |

# Experiment

所有的實驗都是設置epoch=10, batch size=100，餵入資料前先把所有影像都打亂一次，但之後每個epoch同一階段的batch使用的皆為同一批影像，繪製訓練階段時是每個epoch的每個batch訓練結束後都會輸出一次loss與accuracy以觀察模型訓練狀況，四種模型訓練時的準確率及損失函數曲線請參考附錄A~D。

# discution

## 綜合比較

綜合比四個model的模型表現。

可以發現表現最好的模型為簡單的單層的神經網路，我認為會產生此結果的原因是，所有的模型都還未收斂(參考附錄A~D)，所以造成較複雜的還無法有一個穩定的結果之前就停止訓練，進而影響準確率；增加epoch應該會讓更複雜的模型有更好的表現。

|  |  |  |
| --- | --- | --- |
| Model | Top-1 | |
| data | test | val |
| 1-Layer-perceptron | 0.073 | 0.093 |
| 2-Layers-perceptron | 0.06 | 0.078 |
| Lenet5 | 0.042 | 0.04 |
| Lenet5\_m | 0.029 | 0.033 |

## 1-Layer-perceptron v.s 2-Layers-perceptron

除了前段提到的問題之外，可以發現單層神經網路的訓練時間明顯小於多層的神經網路。

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Top-1 | | Training time |
| data | test | val |
| 1-Layer-perceptron | 0.073 | 0.093 | 6min |
| 2-Layers-perceptron | 0.06 | 0.078 | 19min |

## Lanet5 v.s Lanet5\_m

改進的lenet5主要有三點不同:

### 激活函數從ReLU更改為Sigmoid\_m

### 所有捲積層大小從5\*5更改為3\*3

### 新增一層捲積層

從結果來看表現比原始的lenet5略差一點，除了上述提到的原因之外，把捲積層從5\*5改為3\*3可能也會讓模型更難以捕捉到圖像的特徵，進而降低準確率。

|  |  |  |
| --- | --- | --- |
| Model | Top-1 | |
| data | test | val |
| Lenet5 | 0.042 | 0.04 |
| Lenet5\_m | 0.029 | 0.033 |

程式碼: [github link](https://github.com/miowyu/deep_learning)

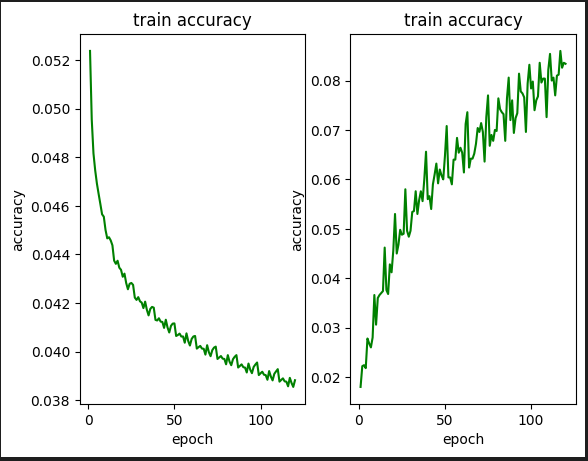
# reference

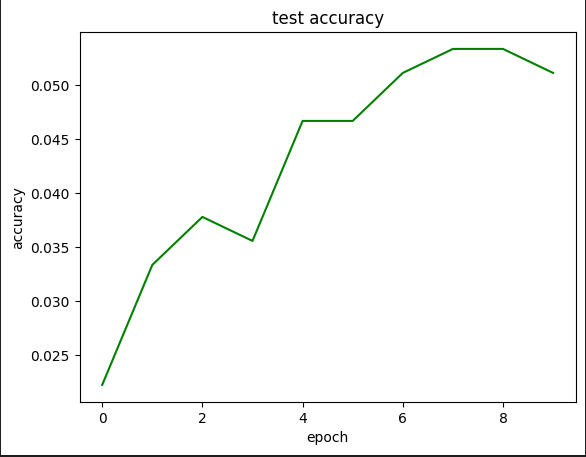
<http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

<https://github.com/mengxinZHANG/numpy-realizes-CNN>

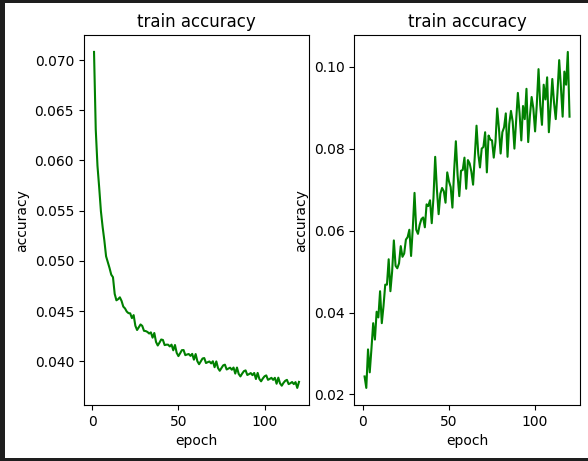
# 附錄

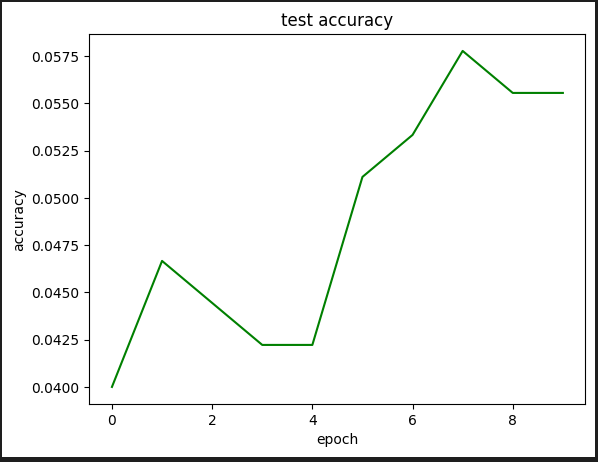
## 1-Layer-perceptron





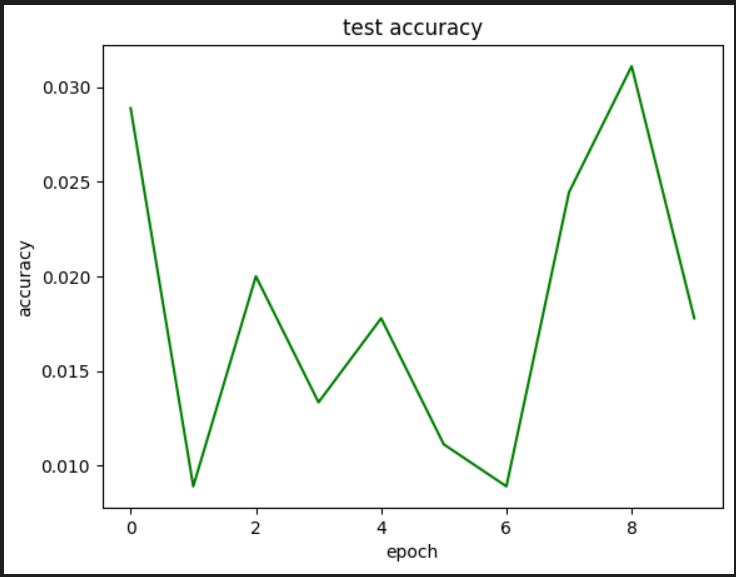
## 2-Layer-perceptron





## Lenet5

一張含有 圖表 的圖片

自動產生的描述

## Lenet5\_m

一張含有 圖表 的圖片

自動產生的描述