**AI training HW4**

STuser19 賴昱凱

1. **Please explain the pros and cons of “regular convolution” and “depth-wise separable convolution” ?**

( M: input channel, N: output channel, 𝐷k : kernel size, 𝐷f : feature map size )

**Regular convolution:**

對input做一般的convolution計算得到output，其計算量為

因為convolution計算方式就是將kernel大小的部分一對一相乘再相加得到output的一個像素，當 kernel像素總數，且input有M個channel時，每計算一個output像素的計算量就為。若我們希望output的大小為，有N個output channel所需總計算量就為。

**Pros:**

一、各大小區域皆被完整掃描，可以適應不同大小特徵類型的圖片。

二、有更多參數且channel不獨立，可以更有效地辨識複雜的特徵。

三、因為模型可以捕捉各種細節特徵，因此應用廣泛且表現佳。

**Cons:**

一、計算量大，需要更多的算力、記憶體。

二、訓練及計算效率低。

三、對電力耗能要求高。

**depth-wise separable convolution:**

對input做depth-wise separable convolution，其計算量為

depth-wise separable convolution分成Depth-wise以及pointwise convolution兩部分，Depth-wise為各input channel獨立計算出自己的output channel，計算量為。Pointwise則是利用M個1x1的kernel對depth-wise輸出的M個channel做convolution，因此有N個output channel就有個計算量。兩部分計算量共。

**Pros:**

一、計算量及參數少，可以更快的訓練及計算。

二、層數、activation function多，可能使其在某些情況表現更佳。

三、記憶體、算力等訓練成本要求減少。

**Cons:**

一­、由於參數減少且各channel獨立，對於捕捉特徵的能力減弱。

二、要求細節的特徵上無法完全替代regular convolution。

三、小模型使用此方法反而會因無法正確捕捉特稱而表現不佳。

1. **Please report the parameters of AlexNet by manual calculations. Show the actual “FLOPS / parameters” reported by code. Attached with Screenshot**

**CustomAlexNet structure:**

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

**Total number of parameters:**

**Screenshot:**

****

**一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

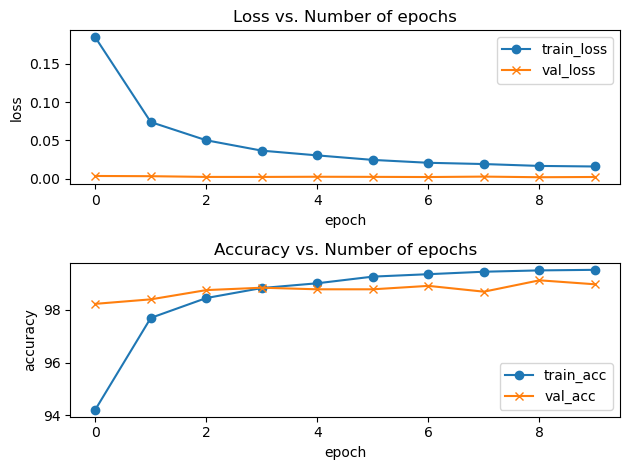
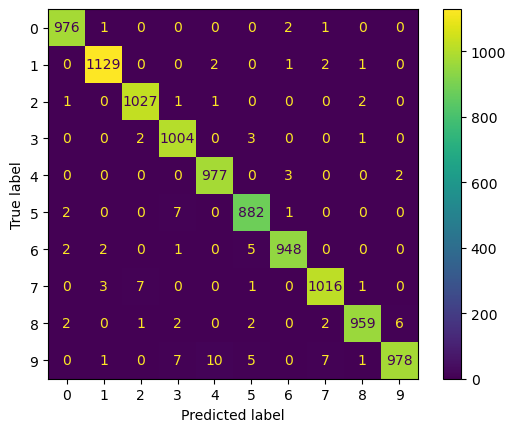
自動產生的描述**

**3. With hw4.py, train a CNN-based model without pre-trained weights.**

首先因為作業要求without pre-trained weights，因此我更改原先模型ResNet18參數pretrained=False並將param.requires\_grad = False拿掉，就跑出accuracy = 98.96%，如下圖:

一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

自動產生的描述



我推測是因為MNIST的手寫數字input圖檔只有，且皆為灰階圖片(channel = 1)，結構簡單且辨識結果可能性只有10種，因此隨意的CNN模型就可以有很好的表現。

由於ResNet18只有一個fully connection layer，因此我參考了2011年Ciresan等人在ICDAR上發表的CNN-based model (其在圖像辨識尤其是MNIST手寫數字辨識上有極優的表現)，並將其改成ResNet的形式拿來與原模型比較，更改後結構為一層convolution layer、3個residual blocks及3層fully connection layer，共10層神經網路，並在第一層convolution layer後及fully connection layer前使用Maxpooling讓計算量減少以簡短訓練時間(約30分鐘內降至約7分鐘)，結果如下:

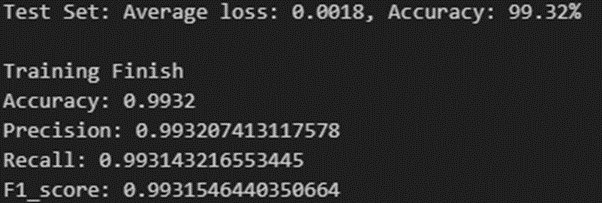
CiresanNet之結構及accuracy: 99.32%

一張含有 文字, 螢幕擷取畫面, 正方形, 圖表 的圖片

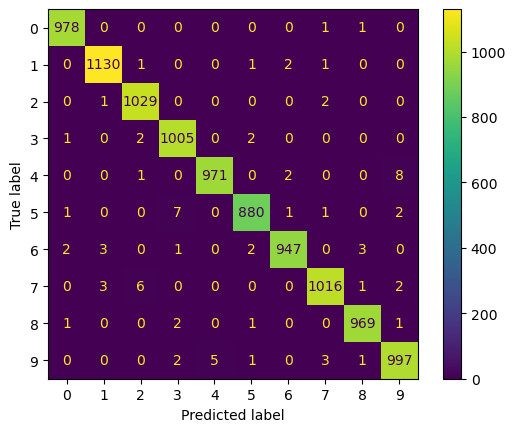
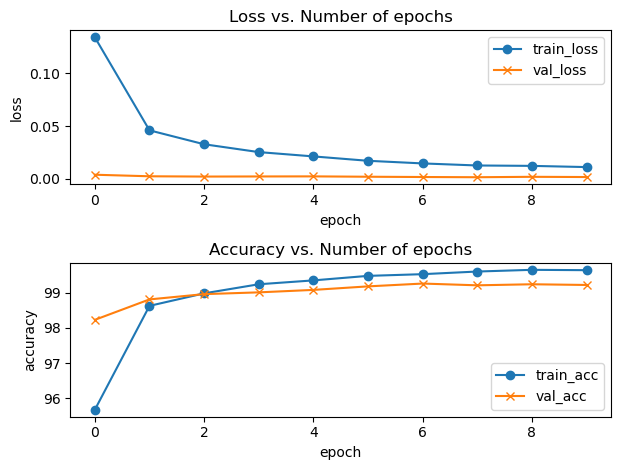
自動產生的描述一張含有 文字, 行, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

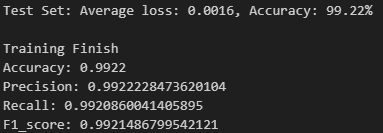
自動產生的描述



更改後CiresanNet之結構及accuracy: 99.22%

一張含有 文字, 螢幕擷取畫面, 陳列, 軟體 的圖片

自動產生的描述



將CiresanNet改成Residual blocks表現稍差於convolution + Maxpooling，但與同為Residual blocks架構的ResNet18比較起來表現明顯更佳，應該是fully connection layer由1層增為3層的表現彌補了Residual blocks的減少，以及channel數量變多的原因。因此我將fully connection layer以2、4層分別訓練得到以下結果: (原結構為3層，accuracy: 99.22%)

2層fully connection layers: 99.19%

一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

自動產生的描述

**一張含有 螢幕擷取畫面, 文字, 正方形, 圖表 的圖片

自動產生的描述一張含有 文字, 行, 圖表, 螢幕擷取畫面 的圖片

自動產生的描述**

4層fully connection layers: 99.42%

**一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 正方形, 圖表 的圖片

自動產生的描述一張含有 文字, 行, 圖表, 螢幕擷取畫面 的圖片

自動產生的描述**

結果表示在4層以下fully connection layer的accuracy有正相關，但我將其增加至5層後便出現98.98%的accuracy，應該是參數過多導致overfitting，因此2~4層是比較恰當的層數。

提到overfitting，我就想到是否減少convolution layer層數對於簡單的圖形辨識可以有更佳的表現?因此我將CiresanNet減少至4層convolution layers及3層fully connection layers，得到結果表現與原先的10層網路差不多:

一張含有 文字, 字型, 螢幕擷取畫面, 代數 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 正方形, 圖表 的圖片

自動產生的描述一張含有 文字, 行, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

於是我參考許多研究都有使用的elastic distortion將輸入資料做前處理，會隨機扭曲圖片及平滑邊緣，如下圖:

一張含有 螢幕擷取畫面, 黑與白, 圖形, 設計 的圖片

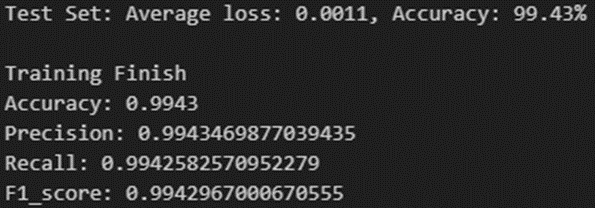
自動產生的描述一張含有 符號, 螢幕擷取畫面, 正方形, 像素 的圖片

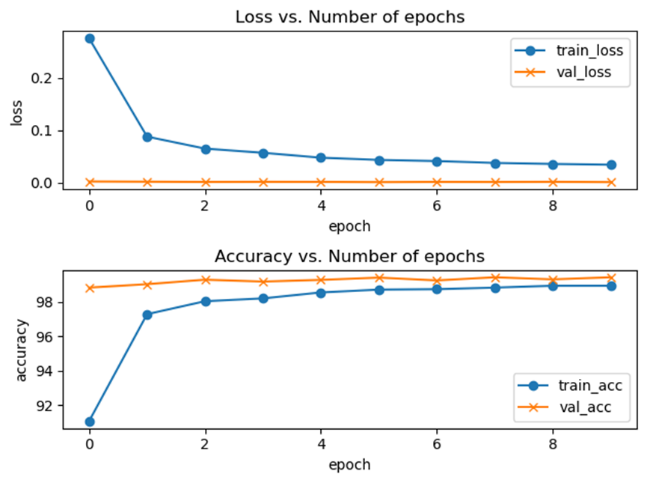
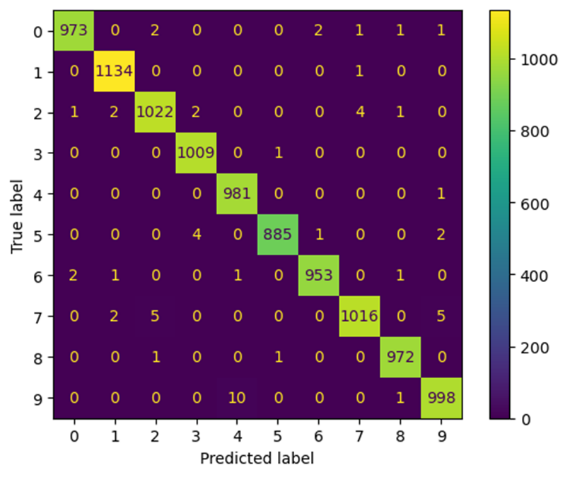
自動產生的描述一張含有 螢幕擷取畫面, 正方形, 設計 的圖片

自動產生的描述一張含有 螢幕擷取畫面, 符號, 圖形, 設計 的圖片

自動產生的描述

並使用剛才使用的7層CNN模型，結果如下:





**Summary:**

1. Fully connection layer的層數適合2~4層，相較於1層可有效提升accuracy。
2. Residual blocks雖然可以避免梯度消失問題，但在此任務中效果不佳，我推測是因為任務過於簡單，本就不易遇到梯度消失，用Residual blocks反而造成loss更難抵達minimum。
3. 簡單的任務若使用複雜的模型會很容易導致overfitting，層數少的模型反而表現更佳。
4. Maxpooling可以大幅減少訓練的計算及時間成本，且不會對於表現有太大影響。
5. Elastic distortion對於MNIST handwriting numbers recognition有正向影響。

**Reference:**

<https://yann.lecun.com/exdb/mnist/>

<https://blog.csdn.net/weixin_43112053/article/details/127374541>

<https://chih-sheng-huang821.medium.com/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-mobilenet-depthwise-separable-convolution-f1ed016b3467>

<https://blog.csdn.net/u011622208/article/details/112566676>