

## 108-2 Deep Learning Homework 1 - 0853412 吳宛儒

### 1. Deep Neural Network for Classification

- i. You should decide the following hyperparameters: number of hidden layers, number of hidden units, learning rate, number of iterations and mini-batch size. You have to show your (a) learning curve, (b) training error rate and (c) test error rate in the report. You should design the network architecture by yourself.

以下針對各種超參數比較與設定的過程作敘述：

*number of hidden layers* = 2, 3

在選擇隱藏層的數量的時候，除了比較不同層數的效果以外，也把其中 hidden units 的數量納入參考。由以下結果可以推測，2 層 hidden layers 的準確度在 training 和 testing 會稍微比 3 層 hidden layers 表現得還好，而在 Loss 的部分則是在 3 層的較多 hidden nodes 的情況下有最低的 Loss。考量到 weights 在初始化時的隨機性，以及以下幾種測試結果，推測在 2 層的時候使用[512,256]會有較好的效果，其 Testing acc 雖不是測試中最高，但差距也相對微小，因此後面的實驗決定使用 2 層 hidden layers。

Number of hidden layers	Training acc rate	Testing acc rate	Loss
2 [64,32]	0.9908	0.9018	0.001522982912986206
3 [64,32,16]	0.9936	0.8985	0.0011779387895317244
2 [512,256]	0.9999	0.8720	0.0001107692271499326
3 [512,256,128]	0.9999	0.8510	0.00015172037504256403

*number of hidden units* = [512,256], [256,128], [128,64], [64,32], [32,16]

在 hidden units 的選擇上，嘗試了以下幾種組合，而 hidden layers 的數量經過前面的實驗，決定固定在 2 個 layers。另外，各層 units 的數量偏好選取 2 的倍數，期望效果會較好。

Number of hidden units	Training acc rate	Testing acc rate	Loss
[512,256]	0.9999	0.8720	0.0001107692271499326
[256,128]	0.9998	0.8899	0.00024855158543373324
[128,64]	0.9977	0.9075	0.0006880426410740182
[64,32]	0.9921	0.8989	0.0014515771669385946
[32,16]	0.9753	0.8978	0.0031453040882808802

經過上述實驗各種不同的 hidden units 數量組合，發現[512,256]平均表現較好，因此後面也會繼續使用兩層 layers、[512,256]的組合。雖然參數量較大會使得訓練較為緩慢，但目前的組合數量都還在可接受的範圍內。

*learning rate* = 0.01, 0.001, 0.099

在設定的時候初始以 0.01 作為 learning rate，在其他參數固定的情況下，調整 learning rate 發現 0.099 目前有最好的效果。

Learning rate	Training acc rate	Testing acc rate	Loss
0.01	0.9914	0.8378	0.0019462965487882546
0.001	0.7988	0.7213	0.02477862782140357
0.099	0.9999	0.8720	0.0001107692271499326

實驗結果可以看出，設定成 0.099 會有比較好的效果，而在 0.01 和 0.001 的結果上就有非常明顯的差異，最後微調至 0.099 得到相對較好的模型效果。

*number of iterations* = 120, 1000

由上面幾次的實驗選定超參數以後，模型效果可以穩定在 120epochs 內達到不錯的效果(上述實驗皆在 120 epochs 下完成)。另外，當 epochs 數增多以後，考量在此處設定了簡易的提早結束方法，類似 early stopping 之概念，規則訂為：

- 1) 當 training 的 accuracy 超過十次沒有提升，提早結束 training。
- 2) 當 training 的 loss 超過十次沒有下降，提早結束 training。

最後為了避免混淆，僅先使用第一種規則，在訓練中會出現例如：

Early Stopping because the accuracy didn't improve...

等情況，中斷訓練。

*mini-batch size* = 8,16,32

設定不同的 batch size，一樣選取 2 的倍數，期望效果會較好。

Size of mini-batch	Training acc rate	Testing acc rate	Loss
8	0.999	0.8923	0.00048224379848218715
16	0.9998	0.8796	0.0003805761410311823
32	0.9999	0.8720	0.0001107692271499326

考量到選用後面 64,128 的大小時，一次訓練會花上太多時間，因此只有比較到 32。可以看到當 batch size 為 32 的時候效果相對好，即使 testing accuracy 有下降的趨勢，整體來說在 batch size 為 32 的時候

Loss 還是會最佳的。

除了上述設定以外，也加入一小段比較：儲存最佳準確度時的 weights，若此次 epoch 之準確度沒有比最佳準確度更高的話，將它的 weights 改成最佳準確度時的 weights，以提升模型效果。

最終設定：

number of hidden layers = 2

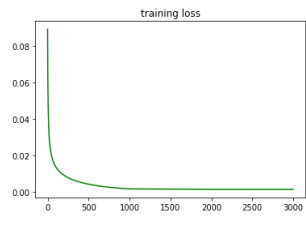
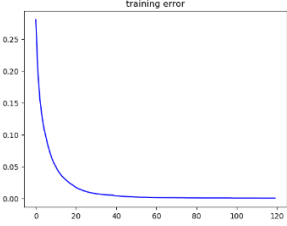
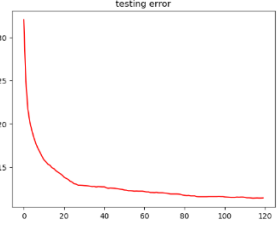
number of hidden units = [512,128]

learning rate = 0.099

number of iterations = 120

mini-batch size = 32

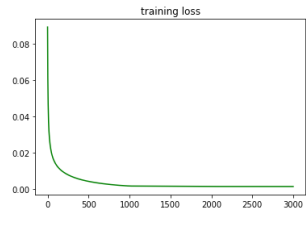
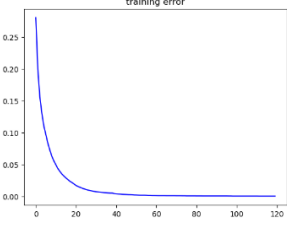
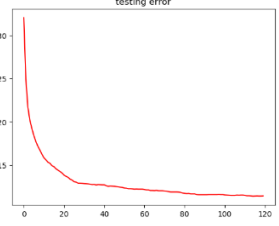
最終結果：

Learning Curve	Training error rate	Testing error rate
		
0.0002345907169628365	0.9998333333333334	0.8859223300970874

- ii. Please perform **zero and random initializations** for the model **weights** and compare the corresponding **error rates**. Are there any difference between two initializations? Please discuss in the report.

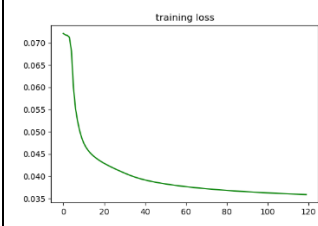
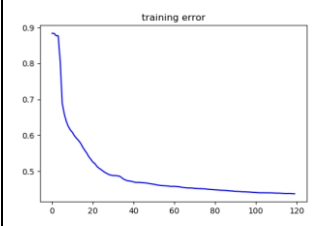
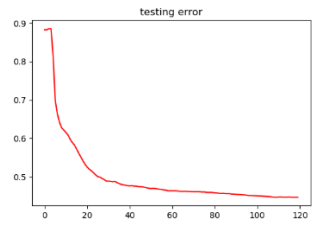
Initialize w **randomly**:

(w: random; b: zeros)

Learning Curve	Training error rate	Testing error rate
		
0.0002345907169628365	0.9998333333333334	0.8859223300970874

Initialize w with **zeros**:

(w: zeros; b: zeros)

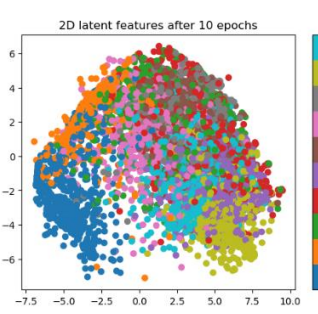
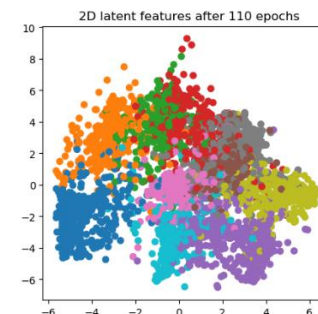
Learning Curve	Training error rate	Testing error rate
		
0.031424696895915316	0.5624166666666667	0.5532246879334258

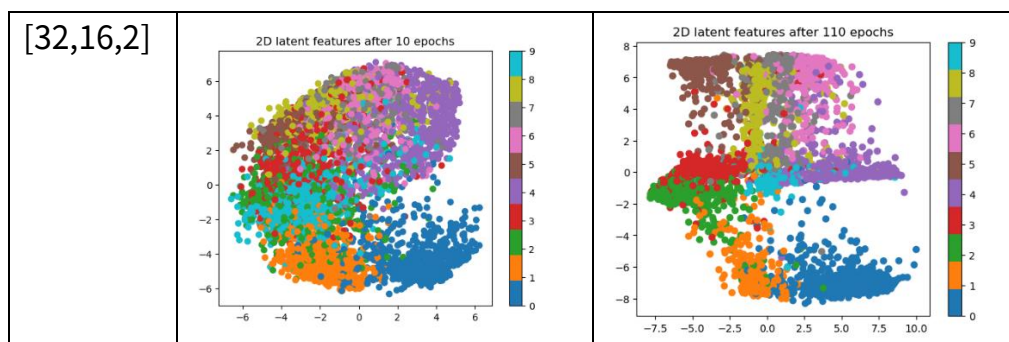
可以看出來當 weights 初始化成 0 的時候，效果降低了很多，可見 random 帶有的隨機性，可以讓模型有比較好也較快的收斂效果。另外也將 bias 試試看不同的初始化效果，由於 weights 隨機初始化會有比較好的效果，因此在 bias 的初始化比較上，僅會使用隨機初始化的 weights，而不再使用初始化為 0 的 weights。最後發現 bias 初始化為 0 效果會比 random 初始化好很多。

- iii. Design your network architecture with the layer of 2 nodes before the output layer. (1) **Plot** the distributions of latent features at different training stages. For example, you may show the results when running at 20th and 80th learning epochs. (2) Please **discuss** the **evolution** of latent features at different training stage.

原本使用在 output layer 前加上 2 nodes 的隱藏層，但是發現這樣 training 效果會大幅下降，因此除此之外也用了 PCA 以及 tSNE 降維的方式，取 output layer 前的隱藏層之 latent features 作為觀察。另外，在討論區有同學向助教詢問確認以後，是以 test data 在其中幾次 epoch 作為比較與紀錄。(程式每 10 個 epochs 會記錄一次 latent features)

在最後一層前加上 2 nodes 的隱藏層：

Hidden units	10 <sup>th</sup> epoch	110 <sup>th</sup> epoch
[16,2]		

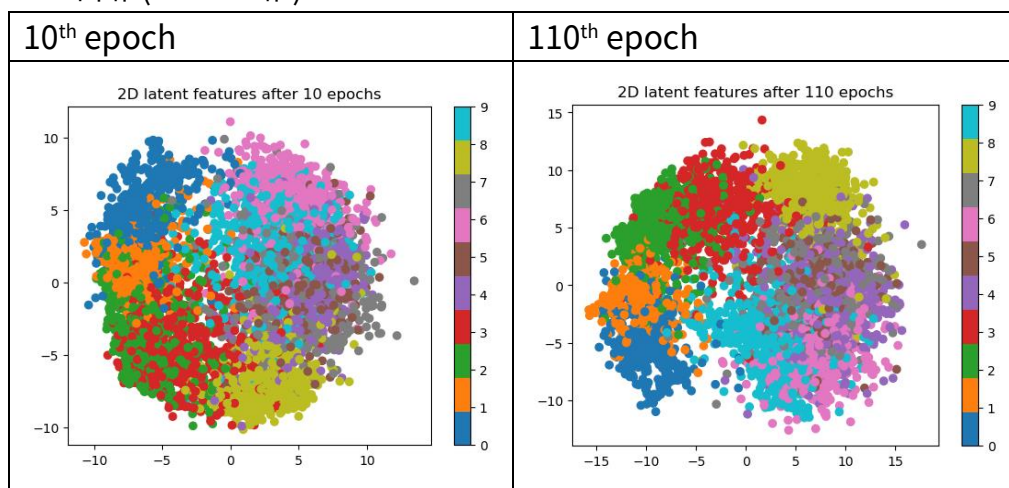


Hidden nodes 數量為[16,2]或[32,16,2]，可以看到在[32,16,2]這樣相對於[16,2]較多層的網路中，經過 110 個 epochs 以後，latent features 有更分開的效果。

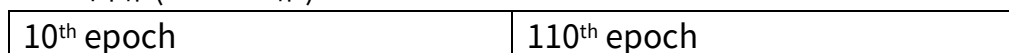
降維方法：

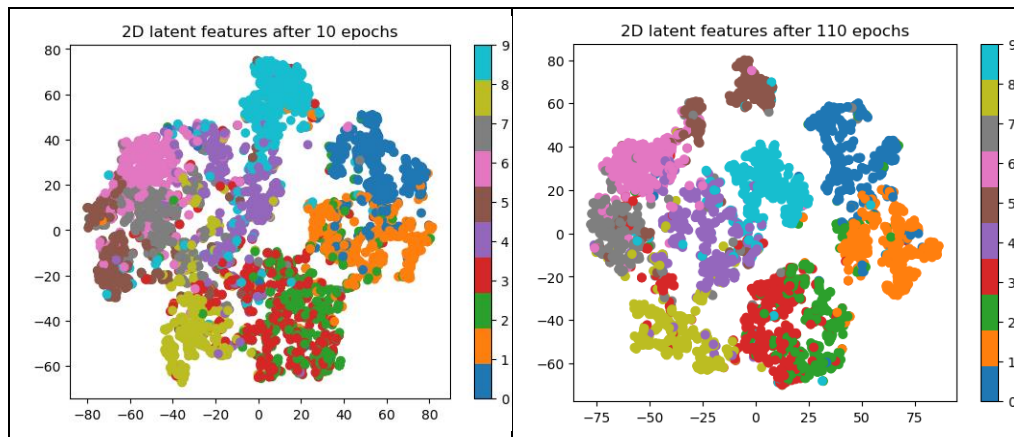
在實作過程中，發現較高維度的 hidden nodes 在降維上除了時間較久 (tSNE 較有感覺)，在二維圖中呈現也比較沒辦法明顯呈現 evolution 的過程，有可能是因為 256 維壓至 2 維差距過大使得沒辦法有效投影。因此在這邊選擇使用[20,15]作為 hidden nodes 之數量，發現這樣投影在二維會比較清楚，所花時間也會比較短。實作時以[20,15]作為 hidden nodes 之數量，其他設定皆以上述最終設定為主。

PCA 降維(15 至 2 維)



tSNE 降維(15 至 2 維)





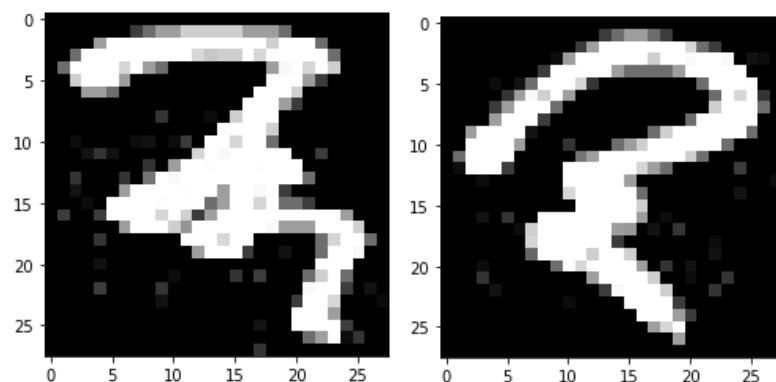
由 PCA 和 tSNE 的結果可以看出來，PCA 在經過 110 個 epochs 以後，雖然點沒有很外散，但是可以看到原本在中間的淺藍色群或其他在一開始混雜的群有像四處分散的趨勢，像一個圓。在 tSNE 降維的結果可以明顯看出來在第 10epoch 中尚未分清楚的類別，在 110epoch 已經明顯分開，且同群的點有集中的趨勢。

iv. Please list your **confusion matrix** and discuss about your results.

```
array([[1332, 0, 0, 0, 0, 0, 0, 0, 0, 0], array([[620, 25, 13, 0, 1, 0, 2, 2, 0, 1],
[ 0, 1473, 0, 0, 0, 0, 0, 0, 0, 0], [11, 631, 13, 5, 0, 0, 0, 0, 0, 1],
[ 1, 0, 1287, 0, 0, 0, 0, 0, 0, 0], [10, 23, 458, 83, 2, 0, 0, 1, 3, 4],
[ 0, 0, 0, 1216, 0, 0, 0, 0, 0, 0], [ 4, 4, 102, 436, 6, 8, 7, 11, 16, 6],
[ 0, 0, 0, 0, 1316, 0, 0, 0, 0, 0], [ 2, 0, 2, 3, 558, 2, 14, 4, 28, 38],
[ 0, 0, 0, 0, 0, 943, 0, 0, 0, 0], [ 0, 1, 0, 7, 6, 378, 4, 5, 5, 1],
[ 0, 0, 0, 0, 0, 0, 1116, 0, 0, 0], [ 1, 1, 0, 1, 5, 14, 451, 22, 4, 3],
[ 0, 0, 0, 0, 0, 0, 0, 853, 0, 0], [ 1, 0, 4, 9, 9, 15, 20, 376, 8, 7],
[ 0, 0, 0, 0, 0, 0, 0, 0, 1070, 0], [ 0, 0, 1, 11, 18, 9, 1, 8, 522, 2],
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1393]]) [ 4, 7, 3, 6, 46, 7, 4, 3, 3, 595]])
```

左表代表總共分錯 1 個 training samples。右表代表總共分錯 120 個 testing samples。除了斜對角正確答案以外，最大的數字，也就是最常被分錯的為分錯 102 次的真實為 3、預測為 2 的類別。

以下是將真實 label 為 3、預測類別為 2 的 samples 印出來：





## 2. Convolutional Neural Network for Image Recognition

- i. Please describe in details how to **preprocess images** because of the different resolution images and various bounding boxes region in Medical Masks dataset and explain why. You have to submit your preprocessing code.

### 第一種方法：

因為不同圖片上戴口罩的部分不一樣，因此透過 pillow 中 `crop((left, bottom, right, top))` 的方法，將圖片裁剪成需要的樣子。而對於裁減以後圖片大小不一的問題，則是在網路最初始的地方使用 PyTorch 中的 `AdaptiveAvgPool2d` 將圖片統一變成  $32 \times 32$  的大小。

### 第二種方法：

因為第一種方法是在網路前面才做修改，因此 batch size 只能為 1，否則會有 input 不同維度的問題。因此常是另種方法，在 PyTorch `ImageFolder` 中的 `transform` 參數設定裡，加入 `transform.Resize((32,32))`，將 image 大小轉成統一的  $32 \times 32$  大小，如此便可以在後面的 dataloader 設定非 1 的 batchsize，可訓練較快。

- ii. Please implement a CNN for image recognition by using Medical Masks dataset. You need to design the network architecture, **describe your network architecture and analyze the effect of different settings including stride size and filter size**. Plot the learning curve and the accuracy rate of training and test data.

### Network Architecture

`AdaptiveAvgPool2d(32,32)` → 第一種 resize 方法才會用到

`Conv2d=2` 層

`MaxPool2d=2` 層，各別接在 convolution 層後， $2 \times 2$

Epochs=10

Learning rate=0.001

以下會針對 stride size, filter size, 以及 optimizer 做比較。

### Stride size=1,2

Stride size	Training acc rate	Testing acc rate	Loss
1	0.950 [0.93 0.97 0.27]	0.934 [0.93 0.99 0.27]	0.158
2	0.957 [0.95 0.98 0.44]	0.820 [0.40 1.00 0.18]	0.128

嘗試了兩種 stride size，發現兩者效果在 training accuracy 跟 loss 上並沒有太大差別，但在 testing 上很 stride=1 效果比 stride=2 好很多，因此後面的實驗還是先以 stride=1 為主。

*Filter size=3\*3, 5\*5*

Filter size	Training acc rate	Testing acc rate	Loss
3*3	0.950 [0.93 0.97 0.27]	0.934 [0.93 0.99 0.27]	0.158
5*5	0.940 [0.90 0.97 0.27]	0.934 [0.97 0.97 0.36]	0.186

在 filter size 大小比較上，比較了 3\*3 和 5\*5 兩種，發現平均來看 filter size=3\*3 會有比較好的表現。

*Optimizer=SGD, Adam*

Optimizer	Training acc rate	Testing acc rate	Loss
SGD	0.952 [0.93 0.98 0.29]	0.926 [0.91 0.97 0.41]	0.145
Adam	0.950 [0.93 0.97 0.27]	0.934 [0.93 0.99 0.27]	0.158

嘗試了兩種不同的 optimizer，發現 SGD 和 Adam 兩者差別沒有很大。

最終設定：

Optimizer=SGD

Learning rate=0.001

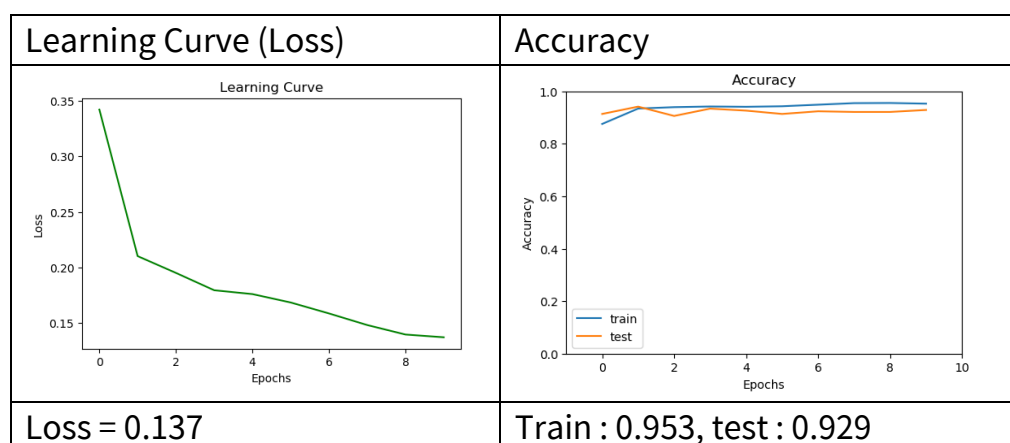
Stride size=1

Filter size=3\*3

Epochs=10

Batch size=16

兩層 convolution，搭配兩層 max pooling

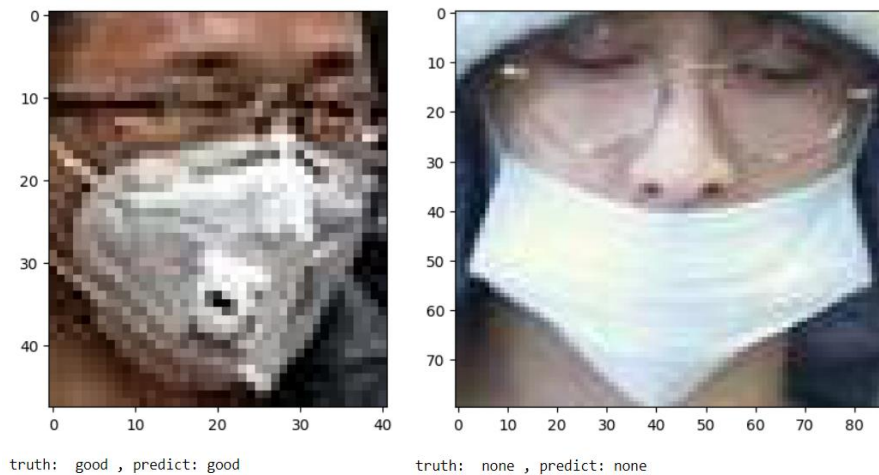




- iii. Show some examples of classification result, *list your accuracy of each classes for both training and test data*, and answer the following questions:

### Some examples of classification result

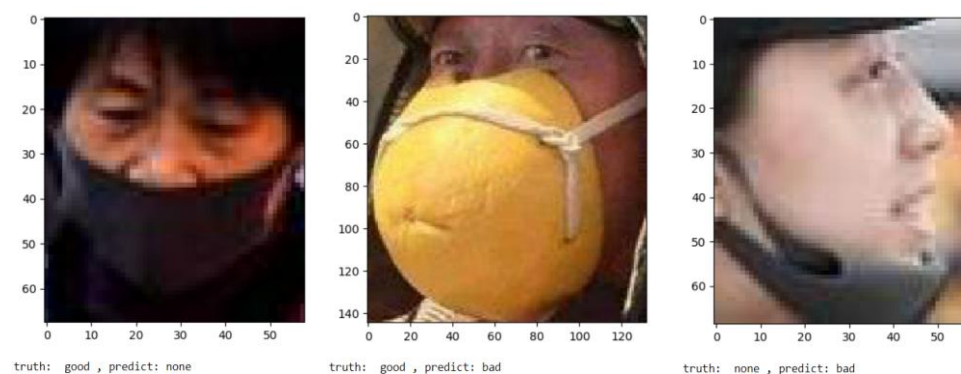
分類成功的案例：



Training 中分類失敗的案例：



Testing 中分類失敗的案例：



在失敗的案例中，可以看出大多屬於較難分辨的例子：例如口罩其實有遮住口鼻，但鼻子輪廓也很明顯，或是口罩顏色並非藍綠白色，會讓模型以為並沒有戴口罩。另外也有解析度非常差(原圖很小)的例子，也會讓分類難度提高許多。另外，也試著畫出像是作業說明檔案上的圖片，如下：



以 testing data 為資料，分別印出每張圖中 true 以及 predict 的 label 為何。框線的顏色以 predict 結果來區分。為了有比較效果，這些圖片是經過兩次 epoch 訓練以後的模型結果，因此效果比較沒那麼好。也可以看到在初期的時候，許多未戴口罩的會被分類為有戴的，許多不是戴一般顏色口罩的則會被誤認為沒有戴。另外，在這部分畫圖的時候，發現會有位置偏移的情況，因此在實際繪圖的時候，top 的值減去 60，讓圖片的框(rectangle)會在相對較正確的位置上。

## Accuracy of each classes

Class	Train Accuracy	Test Accuracy
0 bad	100.0%	93.3%
1 good	97.4%	98.2%
2 none	38.5%	27.3%

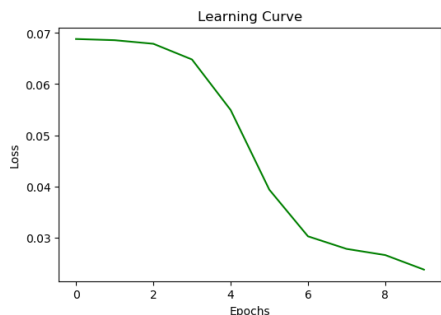
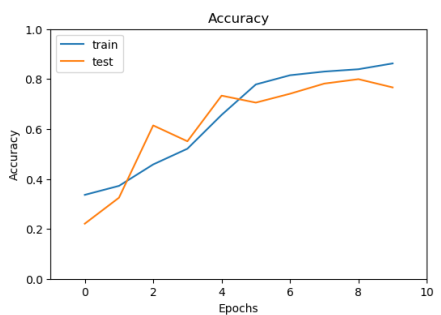
(1) Which class has the worst classification result and why?

None 分得最差，因為從資料數量就可以看出來，None 標籤的資料數量最少，因此模型在 None 特徵學習上可能不夠多，也學不好。

(2) How to solve this problem? (explain and do some experiment to compare the result)

可以增加 None 的樣本(over-sampling)或是減少另外兩類的樣本(under-sampling)，為了達成這樣的目的，使用了 PyTorch 中的 WeightedRandomSampler 來改變 model sample data 時的權重。做法為：首先取得每個 class 的權重，有三個類別，而其權重以各 class 資料量之倒數得到，因此類別資料量越大，其 sample 之權重會越低。Trainset 之 sample weights 為[0.0017301, 0.00035137, 0.00961538] Testset 之 sample weights 為[0.01123596, 0.00353357, 0.04545455] 接著，將設定好的 sampler 做為 dataloader 之參數即可。以下是設定完以後重新訓練 10 個 epochs 的訓練結果：

整體：

Learning Curve (Loss)	Accuracy
	
Loss = 0.024	Train : 0.863, test : 0.766

各類別：

Class	Train Accuracy	Test Accuracy
0 bad	100.0%→94.3%	93.3%→94.6%
1 good	97.4%→85.2%	98.2%→78.2%
2 none	<b>38.5%→79.5%</b>	<b>27.3%→63.0%</b>

從訓練結果可以看出，純粹改變 sample 的權重，網路架構不變的話，有可能會使得整體的效果(不論是 loss 或是準確度上)會表現較差，但這都可以再透過調整網路架構與參數來提升整體效果。另外也看到了我們原本想要解決的「因 imbalanced dataset 而使得資料量少的類別訓練效果差」的問題，在增加 sampling 的權重以後效果有所提升，而且效果蠻顯著的，在表格上以粗體表示。

### (3) Do some discussion about your results.

在口罩影像分類的任務上，可以從兩個角度來看模型訓練：首先是訓練結果的分析，如上述，其實可以很直覺地瞭解到，當口罩並非常見的顏色或是形狀時，模型較難以分辨類別，容易誤判為沒有戴口罩，而若是有正確地戴口罩但是鼻子輪廓相對明顯者，也會容易被分類到 none 的類別，而且加上解析度與其他訓練參數考量，模型在分類任務上難度也會相對提升。另一個角度是 imbalanced dataset，從類別數量來看，可以很明顯看出各類別的資料量非常不平均，雖然 good 和 bad 也有將近六倍數量之差距，但基本兩層 Conv2D 仍可以輕易地將它們訓練至 90% 以上的準確度，不過 none 這個類別不然，none 的類別資料量非常少且不容易判別(相較於有戴與沒戴，有戴與沒戴較容易分辨)，若直接將資料數量原封不動進行訓練，none 的類別的準確度是非常低的，因此在這裡使用了帶有權重的 sampler，將各類別被 sample 到的機率/權重和其各自的數量呈反比，使得各類別數量能夠較為平均，而實驗結果也顯示，這樣的手法讓 none 這個類別的準確度大大地提升。