Homework 3 Report

電機三 B05901009 高瑋聰

1.(1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 48, 48, 1)	0
conv2d_1 (Conv2D)	(None, 48, 48, 64)	640
batch_normalization_1 (Batch	(None, 48, 48, 64)	256
p_re_lu_1 (PReLU)	(None, 48, 48, 64)	147456
LibreOffice Writer	(None, 48, 48, 64)	36928
batch_normalization_2 (Batch	(None, 48, 48, 64)	256
p_re_lu_2 (PReLU)	(None, 48, 48, 64)	147456
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856
batch_normalization_3 (Batch	(None, 24, 24, 128)	512
p_re_lu_3 (PReLU)	(None, 24, 24, 128)	73728
conv2d_4 (Conv2D)	(None, 24, 24, 128)	147584
batch_normalization_4 (Batch	(None, 24, 24, 128)	512
p_re_lu_4 (PReLU)	(None, 24, 24, 128)	73728
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
dropout_2 (Dropout)	(None, 12, 12, 128)	0
conv2d_5 (Conv2D)	(None, 12, 12, 128)	147584
batch_normalization_5 (Batch	(None, 12, 12, 128)	512
p_re_lu_5 (PReLU)	(None, 12, 12, 128)	18432
conv2d_6 (Conv2D)	(None, 12, 12, 128)	147584
batch_normalization_6 (Batch	(None, 12, 12, 128)	512
p_re_lu_6 (PReLU)	(None, 12, 12, 128)	18432
dropout_3 (Dropout)	(None, 12, 12, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 512)	2359808
batch_normalization_7 (Batch	(None, 512)	2048
p_re_lu_7 (PReLU)	(None, 512)	512
dropout_4 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
batch_normalization_8 (Batch	(None, 256)	1024
p_re_lu_8 (PReLU)	(None, 256)	256
dropout_5 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 7)	1799
activation_1 (Activation)	(None, 7)	0

Total params: 3,532,743
Trainable params: 3,529,927
Non-trainable params: 2,816

我的 CNN 模型架構如下，結構上大致參考 VGG 使用多個 3X3 Conv 後 再經過一次 Pool ，最後再經過兩層 Dense ，並且有加入 Dropout 以防止 overfitting ，且使用 Batch Normalization 來穩定訓練過程。

Activation 方面使用了 PReLU ，但是其結果與使用 ReLU 的模型差不多。

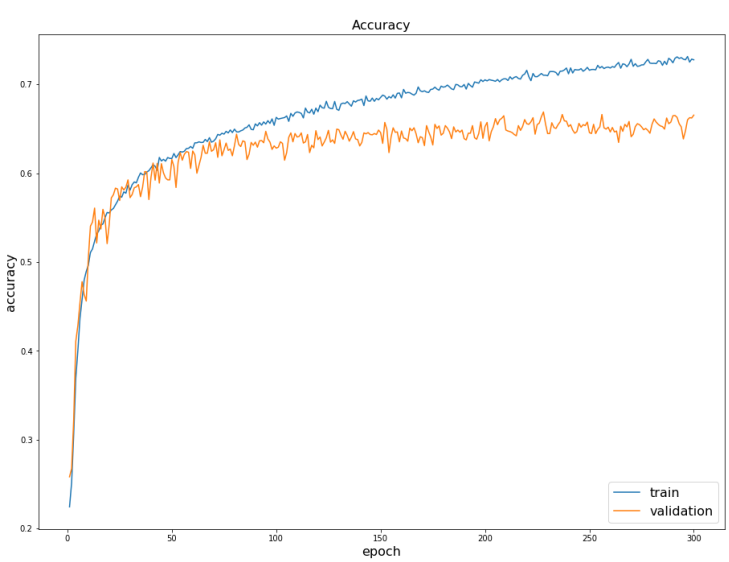
在資料處理方面，除了將圖片除以 255 做 normalize 外，也有使用 ImageGenerator 做 data augmentation ，以避免 overfitting 。

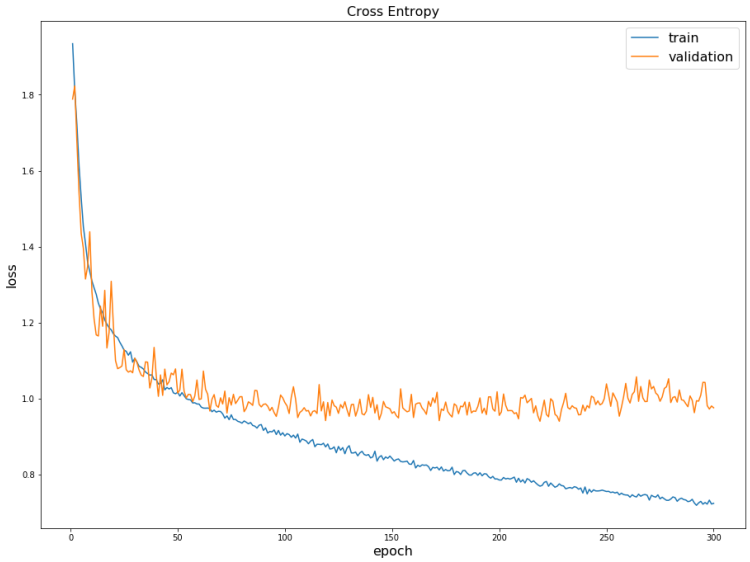
準確率方面，單一個 model 在 kaggle 上 public 約 0.668 ，經過 ensemble 後 public 約能達到 0.694 的準確率。

訓練過程請見下頁圖

從下頁的訓練過程圖形可看出，即使 training 的 loss 與 accuracy 有不斷變好，在 validation set 上的結果也不見得會跟著變好，這也是這次作業要採取多個防止 overfitting 技巧的原因。

訓練過程：（左圖維準確率，右圖為 loss 大小，圖中藍線為 training，橘線為 validation）





2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model，其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 48, 48, 1)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 600)	1383000
batch_normalization_1 (Batch	(None, 600)	2400
p_re_lu_1 (PReLU)	(None, 600)	600
dropout_1 (Dropout)	(None, 600)	0
dense_2 (Dense)	(None, 600)	360600
batch_normalization_2 (Batch	(None, 600)	2400
p_re_lu_2 (PReLU)	(None, 600)	600
dropout_2 (Dropout)	(None, 600)	0
dense_3 (Dense)	(None, 600)	360600
batch_normalization_3 (Batch	(None, 600)	2400
p_re_lu_3 (PReLU)	(None, 600)	600
dropout_3 (Dropout)	(None, 600)	0
dense_4 (Dense)	(None, 600)	360600
batch_normalization_4 (Batch	(None, 600)	2400
p_re_lu_4 (PReLU)	(None, 600)	600
dropout_4 (Dropout)	(None, 600)	0
dense_5 (Dense)	(None, 600)	360600
batch_normalization_5 (Batch	(None, 600)	2400
p_re_lu_5 (PReLU)	(None, 600)	600
dropout_5 (Dropout)	(None, 600)	0
dense_6 (Dense)	(None, 600)	360600
batch_normalization_6 (Batch	(None, 600)	2400
p_re_lu_6 (PReLU)	(None, 600)	600
dropout_6 (Dropout)	(None, 600)	0
dense_7 (Dense)	(None, 537)	322737
batch_normalization_7 (Batch	(None, 537)	2148
p_re_lu_7 (PReLU)	(None, 537)	537
dropout_7 (Dropout)	(None, 537)	0
dense_8 (Dense)	(None, 7)	3766
softmax_1 (Softmax)	(None, 7)	0

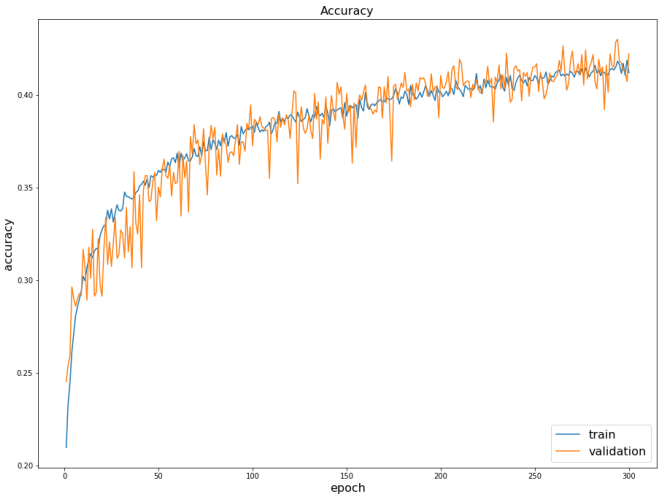
Total params: 3,533,188
Trainable params: 3,524,914
Non-trainable params: 8,274

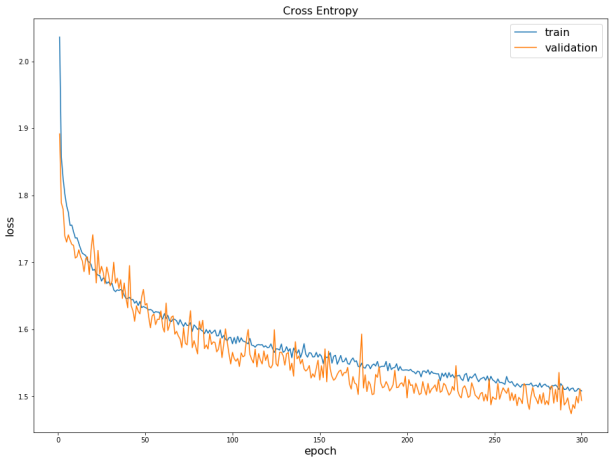
根據上面 CNN 的總參數量，我架了一個七層的 DNN ，並且與上面的 CNN 相同，有使用 BatchNormalization 與 Dropout ，資料的預處理也與上面的模型相同。

準確率方面，可以看到即便與 CNN 一樣訓練了 300 個 epoch ， DNN 的準確率卻只有 0.42 左右，可見在同樣的參數量下， CNN 在表現上有相當大的提昇，由此也可見 CNN 特殊的結構的確有其道理 (能夠利用 convolution 來提取空間上像素與像素間彼此相關的特性) 。不過相對的， CNN 訓練一個 epoch 約需要 DNN 4.5 倍的時間，應是使用的基本數學運算不同導致。即便如此， CNN 花時間訓練仍是值得的。

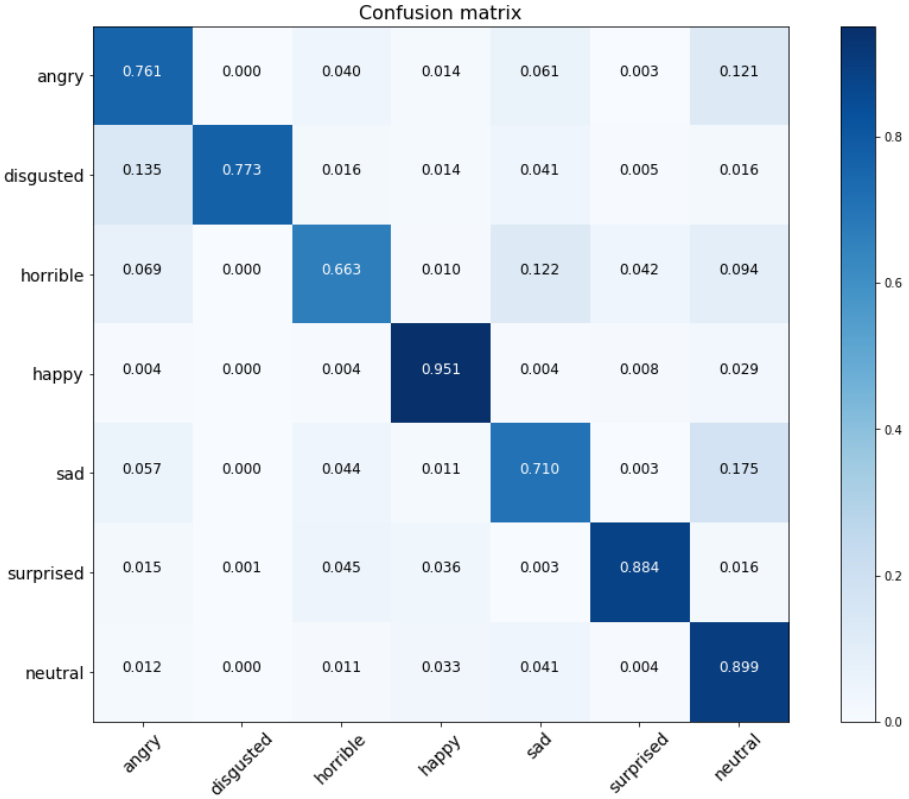
訓練過程可見下頁圖，與 CNN 相同，隨著 training acc 與 loss 變好， validation 的表現會出現震盪，但是因為 DNN 在 training 上就無法 fit 得很好，因此不像 CNN 的 train 與 validation 差異那麼大。

DNN 訓練過程





3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？ 並說明你觀察到了什麼？ [繪出 confusion matrix 分析]



在此七個 class 中，horrible(恐懼)與 sad(難過)這兩個類別分類正確的準確率較低，其中恐懼最容易被分錯成難過，而難過最容易被分成中立。我認為恐懼被分成難過算是蠻合理的結果，例如從助教作業說明投影片的那兩張圖，恐懼的那張圖也蠻像難過的表情。至於難過被分成中立，或許是有些圖片的表情不夠明顯，因此 model 才會將它歸類為中立類別。

4. 5.題請見下頁圖片

ML HW3 B0590009 高瑋聰

4. (a) Layer A : $2\times2\times5\times6+6 = 126$ ※

Layer A output : $(3, 3, 6)$

Layer B : $3\times2\times6\times4+4 = 100$ ※

(b) Layer A : 乘法 : $(2\times2\times5\times9)\times6 = 1080$

加法 : $(2\times2\times5-1)\times9\times6 = 1026$

Layer B : 乘法 : $(2\times2\times6\times1)\times4 = 96$

加法 : $(2\times2\times6-1)\times1\times4 = 92$ ※

(c) notation : C_i : i-th layer, # of filters

K_{iH} : i-th layer, height of kernel

K_{iW} : i-th layer, width of kernel

P_{iH} : i-th layer, # of padding of H (each side) ($H_i \to H_i + 2P_{iH}$)

P_{iW} : i-th layer, # of padding of W (each side) ($W_i \to W_i + 2P_{iW}$)

S_{iH} : i-th layer, # of stride of H

S_{iW} : i-th layer, # of stride of W

Input shape : (H_0, W_0, C_0)

Output shape of i-th layer : $H_i = \frac{H_{i-1} + 2\times P_{iH} - K_{iH}}{S_{iH}} + 1$ $\quad C_i = C_i$

$W_i = \frac{W_{i-1} + 2\times P_{iW} - K_{iW}}{S_{iW}} + 1$

Time Complexity : $\sum_{i=1}^{l} (2C_{i-1} \times K_{iH} \times K_{iW} - 1)\, H_i \times W_i \times C_i$

if $S_{iW} = S_{iH} = S_i$, $P_{iH} = P_{iW} = P_i$, $K_{iH} \times K_{iW} = K_i$, $H_0 = W_0 = n$ $\Rightarrow H_i = W_i = N_i$

$N_i = \frac{N_{i-1} + 2P_i - K_i}{S_i} + 1$

$\Rightarrow \sum_{i=1}^{l} (2C_{i-1} \times K_i^2 - 1) \times C_i \times \left(\frac{N_{i-1} + 2P_i - K_i}{S_i} + 1\right)^2$

$= \sum_{i=1}^{l} O\left((2C_{i-1} \times K_i^2 - 1) \times C_i \times \left(\frac{N_{i-1} + 2P_i - K_i}{S_i} + 1\right)^2\right)$ ※

5.(a) $X = \left[\begin{bmatrix}1\\2\\3\end{bmatrix}_{X_1}, \begin{bmatrix}4\\8\\5\end{bmatrix}_{X_2}, \begin{bmatrix}3\\12\\9\end{bmatrix}, \begin{bmatrix}1\\8\\5\end{bmatrix}, \begin{bmatrix}5\\14\\2\end{bmatrix}, \begin{bmatrix}2\\4\\1\end{bmatrix}, \begin{bmatrix}9\\8\\4\end{bmatrix}, \begin{bmatrix}3\\8\\1\end{bmatrix}, \begin{bmatrix}11\\5\\6\end{bmatrix}_{X_9}, \begin{bmatrix}10\\11\\7\end{bmatrix}_{X_{10}} \right]$

$\mu = \begin{bmatrix}5.4\\8\\4.8\end{bmatrix} \Rightarrow S = \frac{1}{10}\sum_{i=1}^{10}(X_i-\mu)(X_i-\mu)^T = \frac{1}{10}(X-\mu)(X-\mu)^T$

$= \begin{bmatrix}12.04 & .95 & 3.28\\ .95 & 12.2 & 2.9\\ 3.28 & 2.9 & 8.16\end{bmatrix}$

$\Rightarrow S = U\Lambda U^T$, $\det\left(\begin{bmatrix}12.04-\lambda & .95 & 3.28\\ .95 & 12.2-\lambda & 2.9\\ 3.28 & 2.9 & 8.16-\lambda\end{bmatrix}\right) = 0$

$= (12.04-\lambda)[(12.2-\lambda)(8.16-\lambda) - 2.9^2] - .95[.95(8.16-\lambda) - 2.9\times 3.28] + 3.28[.95\times 2.9 - 3.28(12.2-\lambda)]$

$= -\lambda^3 + 32.4\lambda^2 + -336.2764\lambda + 1097.34968 + .25\lambda + 2.716 + -126.49648 + 10.7584\lambda$

$= -\lambda^3 + 32.4\lambda^2 - 325.268\lambda + 973.5492 = 0$

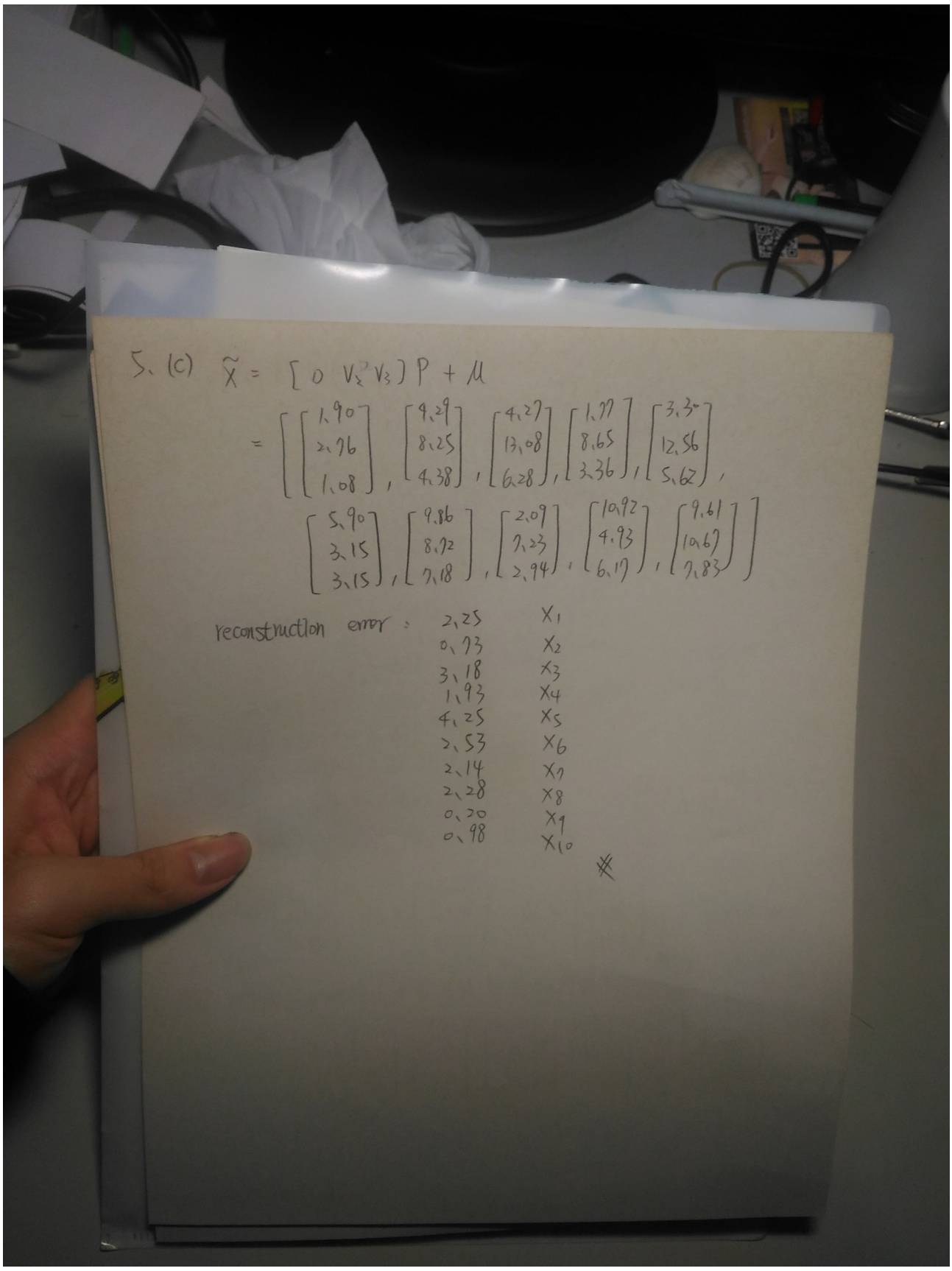
$\lambda_1 = 5.47203\cdots$
$\lambda_2 = 11.43052\cdots$
$\lambda_3 = 15.29744\cdots$

$\Rightarrow V_1 \approx \begin{bmatrix}-.399\\-.338\\.852\end{bmatrix}$ $V_2 \approx \begin{bmatrix}.678\\-.734\\.027\end{bmatrix}$ $V_3 \approx \begin{bmatrix}.617\\.589\\.523\end{bmatrix}$ $\Rightarrow$ principle axes ※

(b) principle component (平移 μ 後和 $V_{1,2,3}$ 內積):

$\Rightarrow P = \begin{bmatrix}V_1^T\\V_2^T\\V_3^T\end{bmatrix}[X-\mu] = \left[\begin{bmatrix}2.25\\1.37\\-7.19\end{bmatrix}_{X_1}, \begin{bmatrix}.73\\-.94\\-.76\end{bmatrix}_{X_2}, \begin{bmatrix}3.19\\-4.45\\3.09\end{bmatrix}, \begin{bmatrix}1.93\\-2.98\\-2.61\end{bmatrix}, \begin{bmatrix}-4.25\\-4.75\\1.82\end{bmatrix}, \right.$

$\left. \begin{bmatrix}-2.53\\3.91\\-3.35\end{bmatrix}, \begin{bmatrix}2.14\\2.56\\4.41\end{bmatrix}, \begin{bmatrix}-2.28\\-1.73\\-3.47\end{bmatrix}, \begin{bmatrix}-.20\\6.93\\2.31\end{bmatrix}_{X_9}, \begin{bmatrix}-.98\\.98\\5.75\end{bmatrix}_{X_{10}} \right]$ ※


5. (c) $\tilde{X} = [v_1\ v_2\ v_3]P + M$
$= \left[\begin{bmatrix}1.90\\2.76\\1.08\end{bmatrix}, \begin{bmatrix}9.29\\8.25\\4.38\end{bmatrix}, \begin{bmatrix}4.27\\13.08\\6.28\end{bmatrix}, \begin{bmatrix}1.77\\8.65\\3.36\end{bmatrix}, \begin{bmatrix}3.35\\12.56\\5.62\end{bmatrix}, \begin{bmatrix}5.90\\3.15\\3.15\end{bmatrix}, \begin{bmatrix}9.86\\8.72\\7.18\end{bmatrix}, \begin{bmatrix}2.09\\7.23\\2.94\end{bmatrix}, \begin{bmatrix}10.92\\4.93\\6.17\end{bmatrix}, \begin{bmatrix}9.61\\10.67\\7.83\end{bmatrix}\right]$
reconstruction error :
2.25 x_1
0.73 x_2
3.18 x_3
1.93 x_4
4.25 x_5
2.53 x_6
2.14 x_7
2.28 x_8
0.20 x_9
0.98 x_{10}
※