

1. (1%) 請說明這次使用的model架構，包含各層維度及連接方式。

以Convolution - Batch Normalization - ReLU - MaxPooling - Dropout為一個單位，共疊了4次，攤平後再接2層全連接層(同樣用ReLU)，再接輸出。使用Adam進行梯度下降，learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.99$ 。Loss function使用Cross entropy。

Convolution的filter數是呈金字塔形，由少變多再變少(32→64→32)。考慮到前幾層會先篩出簡單的幾何圖形，而後幾層的像素數不多，故給予較少filter。中間層不但filter較多，也用了較大的filter (5x5)，為了讓主要的特徵在中間層篩出來。最後加入batch normalization讓梯度下降的更快，以及dropout減少overfitting。

以下以Keras的輸出做修改

Layer (param)	Output Shape	Param #
[Input]		
Input	(None, 48, 48, 1)	
[Conv 0]		
Conv2D (32, (3, 3))	(None, 48, 48, 32)	320
Batch Normalization	(None, 48, 48, 32)	128
Activation (ReLU)	(None, 48, 48, 32)	0
MaxPooling2D (2, 2)	(None, 24, 24, 32)	0
Dropout (0.25)	(None, 24, 24, 32)	0
[Conv 1]		
Conv2D (64, (5, 5))	(None, 24, 24, 64)	51264
Batch Normalization	(None, 24, 24, 64)	256
Activation (ReLU)	(None, 24, 24, 64)	0
MaxPooling2D (2, 2)	(None, 12, 12, 64)	0
Dropout (0.25)	(None, 12, 12, 64)	0
[Conv 2]		
Conv2D (64, (3, 3))	(None, 12, 12, 64)	36928
Batch Normalization	(None, 12, 12, 64)	256

Activation (ReLU)	(None, 12, 12, 64)	0
MaxPooling2D (2, 2)	(None, 6, 6, 64)	0
Dropout (0.25)	(None, 6, 6, 64)	0

[Conv 3]

Conv2D (32, (3, 3))	(None, 6, 6, 32)	18464
Batch Normalization	(None, 6, 6, 32)	128
Activation (ReLU)	(None, 6, 6, 32)	0
MaxPooling2D (2, 2)	(None, 3, 3, 32)	0
Dropout (0.25)	(None, 3, 3, 32)	0

[Flatten]

Flatten	(None, 288)	0
---------	-------------	---

[Dense 0]

Dense (256)	(None, 256)	73984
Batch Normalization	(None, 256)	1024
Activation (ReLU)	(None, 256)	0
Dropout (0.25)	(None, 256)	0

[Dense 1]

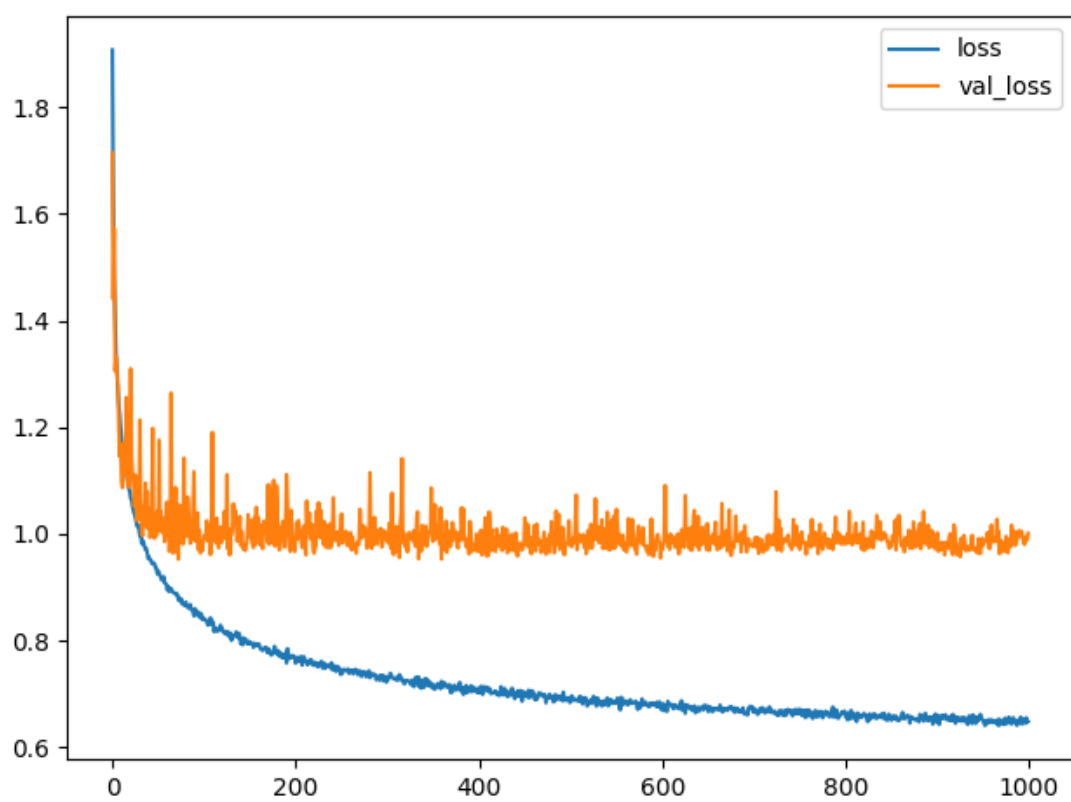
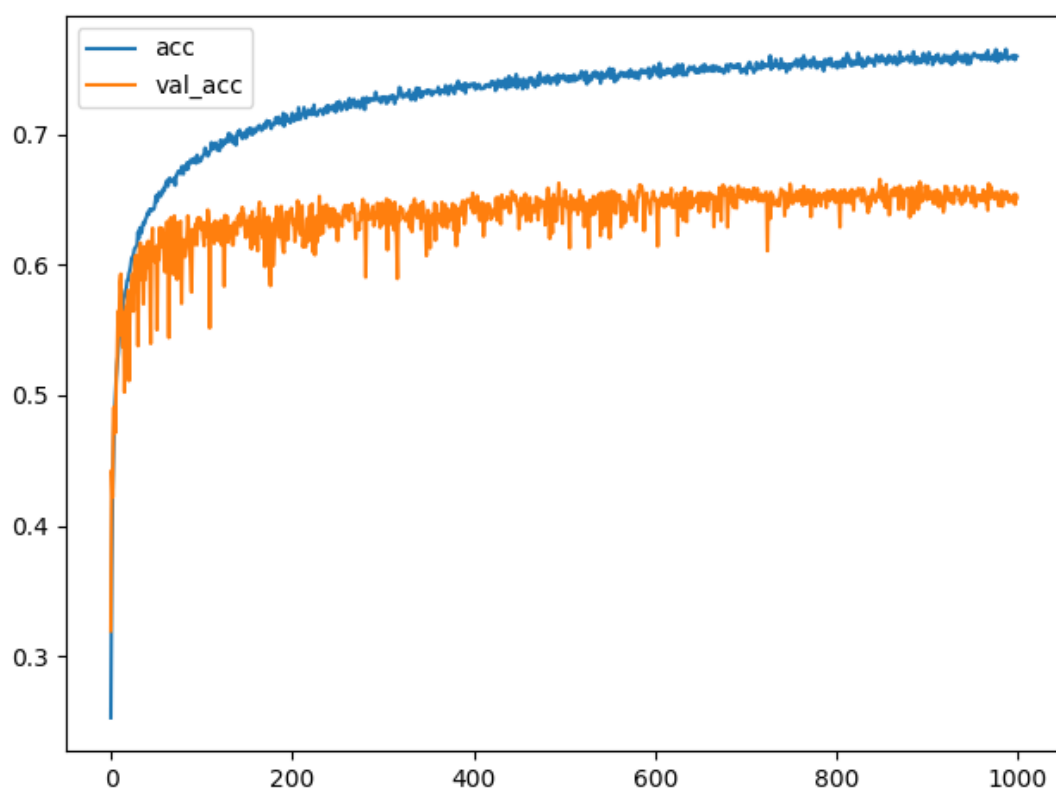
Dense (512)	(None, 512)	131584
Batch Normalization	(None, 512)	2048
Activation (ReLU)	(None, 512)	0
Dropout (0.25)	(None, 512)	0

[Output]

Dense (7)	(None, 7)	3591
Activation (softmax)	(None, 7)	0

Total params: 319,975
Trainable params: 318,055
Non-trainable params: 1,920

2. (1%) 請附上model的training/validation history (loss and accuracy)。

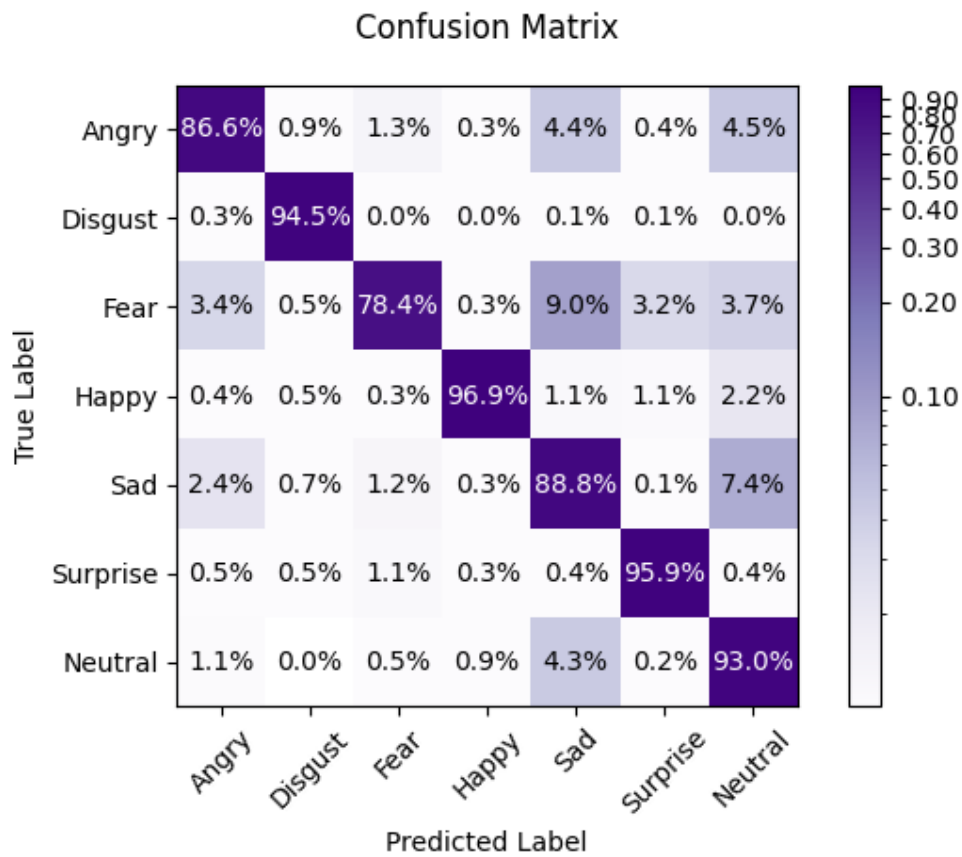


Final: loss = 0.6485, acc = 0.7597, val_loss = 1.0013, val_acc = 0.6510

3. (1%) 畫出confusion matrix分析哪些類別的圖片容易使model搞混，並簡單說明。

(ref: https://en.wikipedia.org/wiki/Confusion_matrix)

(注：圖中使用對數化的色表)

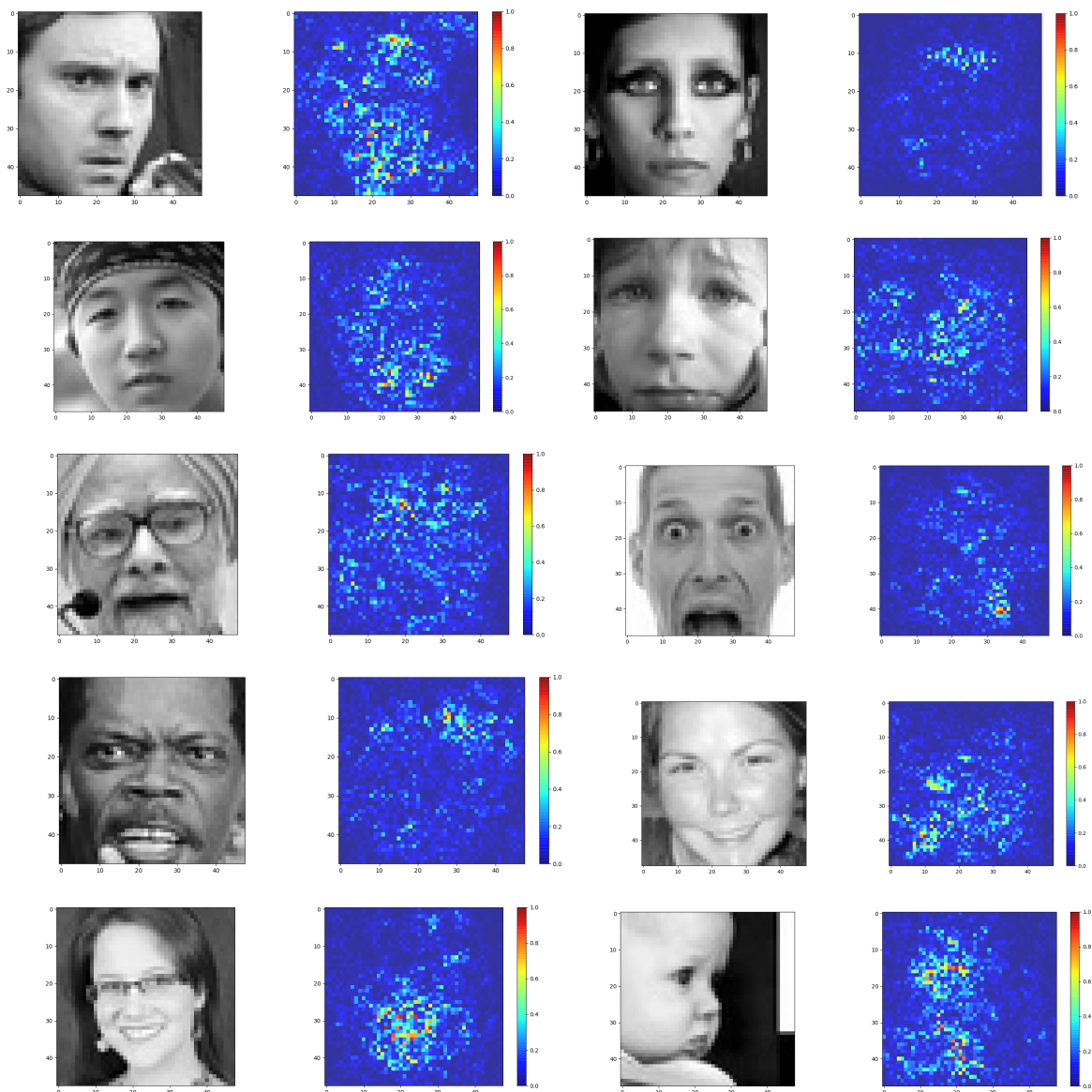


可以看到fear和angry容易被誤判，而且最常被判斷成sad。fear的variance又屬最高，可能的原因有，恐懼的表情在不同人身上差異很大，或者恐懼時本來肌肉移動的幅度就不大。

[關於第四及第五題]

可以使用簡單的 3-layer CNN model [64, 128, 512] 進行實作。

4. (1%) 畫出CNN model的saliency map，並簡單討論其現象。

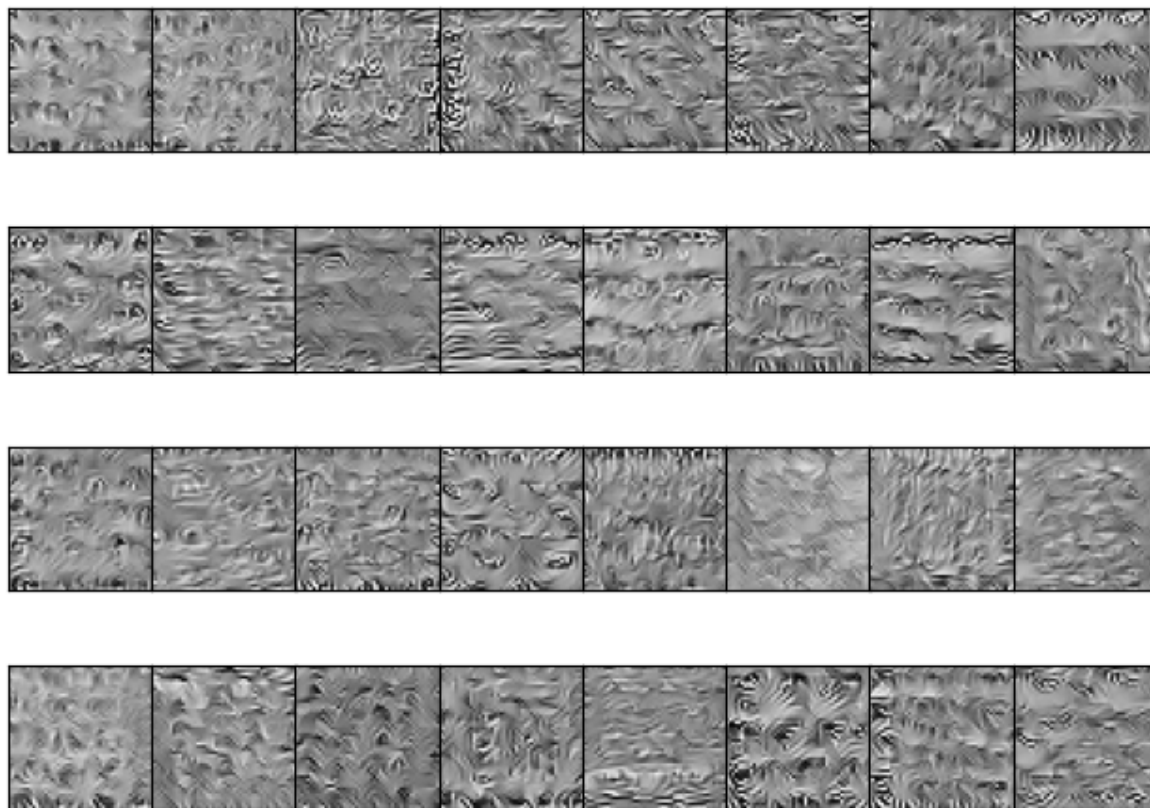


以上是以第1題提及的4層模型實作

定義圖1-10為以由左至右，由上而下的順序。可以發現影響顯著的pixel都集中在臉部，但不一定集中在與表情有關的部位。較理想的例子比如圖9，顯著點集中在開口笑的嘴；圖5集中在眉毛附近，算是表情嚴肅的特徵。而有些卻集中在不太正確的部位，如圖2和圖7在額頭，可能純粹是因為它比較亮。要解決這種問題，可能嘗試對圖片做預處理，還有更大的原因是模型訓練得不夠好，不然應該要看得更清楚的臉部輪廓。

5. (1%) 畫出最後一層的filters最容易被哪些feature activate。

第4個Conv層32個filter分別的output的feature map (activation map) 如下：



可以發現當中有很多一塊塊可能是五官的東西分布在不同位置。

6. (3%)Refer to math problem

<https://hackmd.io/@ASZWRvp7SjOEdYLqF3JYdg/HJMbtPOdD>

6-1. Convolution (1%)

As we mentioned in class, image size may change after convolution layers. Consider a batch of image data with shape (B, W, H, input_channels), how will the shape change after the convolution layer?

Conv2D (input_channels, output_channels, kernel_size = (k₁, k₂), stride = (s₁, s₂), padding = (p₁, p₂))

To simplify the answer: the padding tuple means that we pad p₁ pixels on both left and right side, and p₂ pixels for top and bottom

Solution.

$$\left(B, \left\lfloor \frac{W + 2p_1 - k_1}{s_1} \right\rfloor + 1, \left\lfloor \frac{H + 2p_2 - k_2}{s_2} \right\rfloor + 1, output_channels \right)$$

6-2. Batch Normalization (1%)

Besides Dropout, we usually use Batch Normalization in training nowadays [ref]. The trick is popular within the deep networks due to its convenience while training. It preserves the distribution within hidden layers and avoids gradient vanish.

The algorithm can be written as below:

Input : values of x over a mini – batch : B = {x_{1..m}};

Output : y_i = BN_{γ,β}(x_i)

Parameters to be learned : γ, β

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad //mini - batch\ mean$$

$$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad //mini - batch\ variance$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad //normalize$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i) \quad //scale\ and\ shift$$

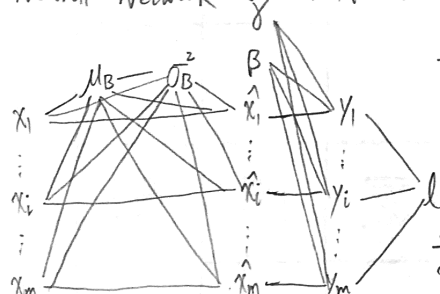
How to update γ and β from the optimization process of loss?

Just try to derive $\frac{\partial l}{\partial \hat{x}_i}, \frac{\partial l}{\partial \sigma_B^2}, \frac{\partial l}{\partial \mu_B}, \frac{\partial l}{\partial x_i}, \frac{\partial l}{\partial \gamma}, \frac{\partial l}{\partial \beta}$

Solution.

$$\begin{cases}
 \mu_B = \frac{1}{m} \sum_{i=1}^m x_i \\
 \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \\
 \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \\
 y_i = \gamma \hat{x}_i + \beta \\
 l(y_1, \dots, y_m)
 \end{cases}$$

Neural Network of $l(\gamma, \beta, x_1, \dots, x_m)$:



$$\begin{aligned}
 \frac{\partial l}{\partial \hat{x}_i} &= \frac{\partial l}{\partial y_i} \cdot \frac{\partial y_i}{\partial \hat{x}_i} = \gamma \frac{\partial l}{\partial y_i} \\
 \frac{\partial l}{\partial \sigma_B^2} &= \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial \sigma_B^2} = \frac{-1}{2} (\sigma_B^2 + \epsilon)^{-3/2} \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} (x_i - \mu_B) \\
 \frac{\partial l}{\partial \mu_B} &= \frac{\partial l}{\partial \sigma_B^2} \frac{\partial \sigma_B^2}{\partial \mu_B} + \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial \mu_B} \\
 &= \frac{\partial l}{\partial \sigma_B^2} \frac{-2}{m} \sum_{i=1}^m (x_i - \mu_B) + \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \\
 &\quad \underbrace{\sum_{i=1}^m (x_i - \mu_B)}_{=0} \\
 &= \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}} \sum_{i=1}^m \frac{\partial l}{\partial \hat{x}_i} \\
 \frac{\partial l}{\partial x_i} &= \frac{\partial l}{\partial \mu_B} \frac{\partial \mu_B}{\partial x_i} + \frac{\partial l}{\partial \sigma_B^2} \frac{\partial \sigma_B^2}{\partial x_i} + \frac{\partial l}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial x_i} \\
 &= \frac{\partial l}{\partial \mu_B} \frac{1}{m} + \frac{\partial l}{\partial \sigma_B^2} \frac{2}{m} (x_i - \mu_B) + \frac{\partial l}{\partial \hat{x}_i} \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} \\
 \frac{\partial l}{\partial \gamma} &= \sum_{i=1}^m \frac{\partial l}{\partial y_i} \frac{\partial y_i}{\partial \gamma} = \sum_{i=1}^m \frac{\partial l}{\partial y_i} \hat{x}_i \\
 \frac{\partial l}{\partial \beta} &= \sum_{i=1}^m \frac{\partial l}{\partial y_i} \frac{\partial y_i}{\partial \beta} = \sum_{i=1}^m \frac{\partial l}{\partial y_i}
 \end{aligned}$$

再根據選擇的 l 去求 $\frac{\partial l}{\partial y_i}$ 代入，求得 $\partial l / \partial \gamma$ 和 $\partial l / \partial \beta$ 之後，令 $\gamma_{t+1} \leftarrow \gamma_t - \eta \frac{\partial l}{\partial \gamma}$ ，

$\beta_{t+1} \leftarrow \beta_t - \eta \frac{\partial l}{\partial \beta}$ ，其中 η 為 learning rate。或者可以套其它梯度下降的算法。

6-3. Softmax and Cross Entropy (1%)

In classification problem, we use softmax as activation function and cross entropy as loss function.

$$\text{softmax}(z_t) = \frac{e^{z_t}}{\sum_i e^{z_i}}$$

$$\text{cross_entropy} = L(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i$$

$$\text{cross_entropy} = L_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$\hat{y}_t = \text{softmax}(z_t)$$

$$\text{Derive that } \frac{\partial L_t}{\partial z_t} = \hat{y}_t - y_t$$

Solution.

Clarify: I could not get the expected result for $\frac{\partial L_t}{\partial z_t}$ in the problem statement, so I calculate both

$$\frac{\partial L_t}{\partial z_t} \text{ and } \frac{\partial L}{\partial z_t}$$

$$\text{Recall: } \begin{cases} L(y, \hat{y}) &= \sum_i -y_i \log \hat{y}_i \\ \hat{y}_i &= \frac{e^{z_i}}{e^{z_1} + \dots + e^{z_n}} \\ \sum_i y_i &= 1 \end{cases}$$

$$\text{We want: } \frac{\partial L}{\partial z_t} = \sum_i \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_t}$$

where:

$$\frac{\partial L}{\partial \hat{y}_i} = \frac{-y_i}{\hat{y}_i} \quad \text{for all } i$$

$$\frac{\partial \hat{y}_i}{\partial z_t} = \frac{-e^{z_i} e^{z_t}}{(e^{z_1} + \dots + e^{z_n})^2} = -\hat{y}_i \hat{y}_t \quad \text{for } i \neq t$$

$$\frac{\partial \hat{y}_t}{\partial z_t} = \hat{y}_t - \hat{y}_t^2$$

$$\begin{aligned} \text{So: } \frac{\partial L}{\partial z_t} &= \sum_{i \neq t} \frac{-y_i}{\hat{y}_i} (-\hat{y}_i \hat{y}_t) + \frac{-y_t}{\hat{y}_t} (\hat{y}_t - \hat{y}_t^2) = \hat{y}_t \left(\sum_{i \neq t} y_i + y_t \right) - y_t \\ &= \hat{y}_t - y_t \end{aligned}$$

If we solve for $\frac{\partial L_t}{\partial z_t}$

$$\begin{aligned} \frac{\partial L_t}{\partial z_t} &= \frac{\partial}{\partial z_t} (-y_t \log \hat{y}_t) = \frac{\partial}{\partial z_t} \left[-y_t \log \left(\frac{e^{z_t}}{e^{z_1} + \dots + e^{z_n}} \right) \right] \\ &= \frac{\partial}{\partial z_t} (-y_t) [z_t - \log(e^{z_1} + \dots + e^{z_n})] = (-y_t) \left[1 - \frac{e^{z_t}}{e^{z_1} + \dots + e^{z_n}} \right] \\ &= -y_t (1 - \hat{y}_t) \end{aligned}$$