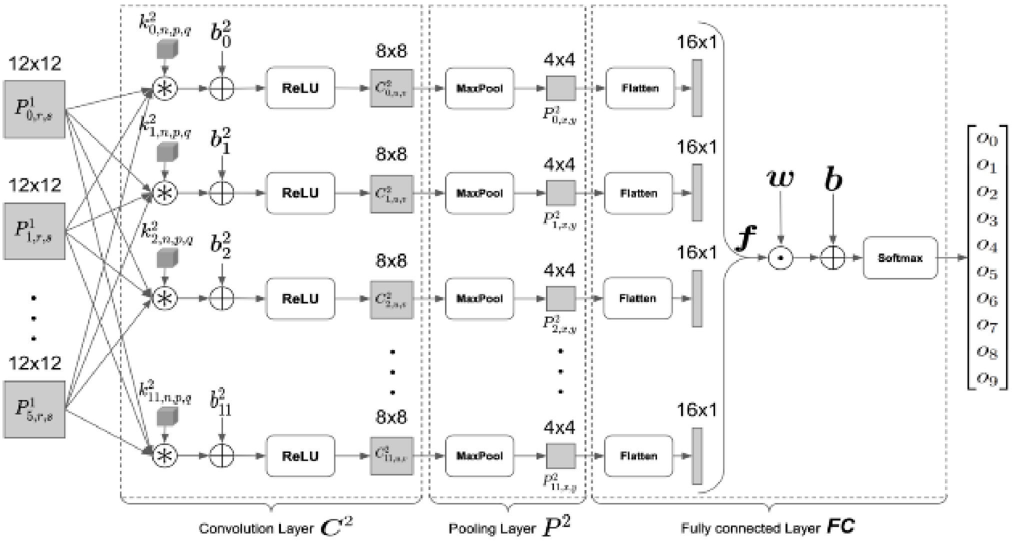
Neural Networks

HW3 Computer Project2 CNN	
授課老師	王振興
Due date 12/16	
學號	N26094883
姓名	杜冠勳
系所	電機所

derivative process	
handcraft	CNN
Model	RBF
CNN	參考論文: REF.A gentle explanation of Backpropagation in Convolutional Neural Network Son Nguyen February 27, 2020
	Gradient & BP 推導



首先先從 softmax 層往回推

$$\begin{aligned} L &= L(O_0, ..., O_9), \\ &= L(O_{label}) \\ &= -\ln(O_{label}),\ label \in \{0, ..., 9\} \end{aligned}$$

因為 是採用 softmax 故使用 crossentrophy 當作 loss 依據

$$\boldsymbol{f} = \text{flatten}(\boldsymbol{P^2})$$

$$S_i = \sum_{j=0}^{191} w_{ij} f_j + b_i,$$

$$O_i = \text{softmax}(S_i) = \frac{e^{S_i}}{\sum_{k=0}^{9} e^{S_k}},$$

$$i = 0, ..., 9$$

藉由上式第二行可推

$$\frac{\partial S_k}{\partial b_i} = \begin{cases} 1 \text{ if } k = i \\ 0 \text{ if } k \neq i \end{cases}$$

而因為

	$$\frac{\partial L}{\partial b_i} = \sum_{k=0}^{9} \frac{\partial L}{\partial S_k}\frac{\partial S_k}{\partial b_i},\ i = 0, ..., 9$$ 可得 $$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial S_i},\ i = 0, ..., 9$$ 利用 chain rule $$\frac{\partial L}{\partial S_i} = \frac{\partial L(O_{label})}{\partial O_{label}}\frac{\partial O_{label}}{\partial S_i}$$ $$\frac{\partial L(O_{label})}{\partial O_{label}} = \frac{\partial(-\ln(O_{label}))}{\partial O_{label}} = -\frac{1}{O_{label}}$$ 而因為 $$O_{label} = \frac{e^{S_{label}}}{\sum_{k=0}^{9} e^{S_k}}$$ 將 T 用代數帶換 $$T = \sum_{k=0}^{9} e^{S_k},$$ $$O_{label} = \frac{e^{S_{label}}}{T} = e^{S_{label}}T^{-1}$$ 若是 label hits 的 case 令 $$e^{S_{label}} = T - C_1,$$ C 為一常數 可得 $$\frac{\partial O_{label}}{\partial S_{label}} = \frac{\partial\big((T - C_1)T^{-1}\big)}{\partial S_{label}}$$

$$= C_1 T^{-2} \frac{\partial T}{\partial S_{label}}$$

$$\begin{aligned}
\frac{\partial O_{label}}{\partial S_{label}} &= C_1 T^{-2} e^{S_{label}} \\
&= (C_1 T^{-1})(e^{S_{label}} T^{-1}) \\
&= (T - e^{S_{label}}) T^{-1} (e^{S_{label}} T^{-1}) \\
&= (1 - e^{S_{label}} T^{-1})(e^{S_{label}} T^{-1}) \\
&= (1 - O_{label}) O_{label}
\end{aligned}$$

而當 label 不相同時 亦是此方式得到其 gradient

Softmax 階段 小結

$$\frac{\partial O_{label}}{\partial S_i} = \begin{cases} (1 - O_{label}) O_{label} & \text{if } i = label \\ -O_{label} O_i & \text{if } i \neq label \end{cases}, \ i = 0, \ldots, 9$$

$$\frac{\partial L}{\partial S_i} = \begin{cases} O_{label} - 1 & \text{if } i = label \\ O_i & \text{if } i \neq label \end{cases}, \ i = 0, \ldots, 9$$

$$\frac{\partial L}{\partial b_i} = \begin{cases} O_{label} - 1 & \text{if } i = label \\ O_i & \text{if } i \neq label \end{cases}, \ i = 0, \ldots, 9$$

接下來再往回推 要更新 FC layer 的 weight

$$\frac{\partial L}{\partial w_{ij}} = \sum_{k=0}^{9} \frac{\partial L}{\partial S_k} \frac{\partial S_k}{\partial w_{ij}}, \ i = 0, \ldots, 9, \ j = 0, \ldots, 191$$

$$\frac{\partial S_k}{\partial w_{ij}} = \frac{\partial(\sum_{t=0}^{191} w_{kt} f_t + b_k)}{\partial w_{ij}} = \begin{cases} f_j & \text{if } k = i \\ 0 & \text{if } k \neq i \end{cases}$$

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial S_i} f_j,\ i = 0, ..., 9,\ j = 0, ..., 191$$

回推到 maxpooling & convolution 層 要更新

該 bias

$$\frac{\partial L}{\partial b_m^2} = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{\partial L}{\partial S_{muv}^2} \frac{\partial S_{muv}^2}{\partial b_m^2}$$

$$\frac{\partial L}{\partial S_{muv}^2} = \frac{\partial L}{\partial C_{muv}^2} \frac{\partial C_{muv}^2}{\partial S_{muv}^2}$$

Maxpooling 前 為 convolution 3D output shape 為 (8 8 12)

故 bias 更新數量應為 12

若今天更新回去的路徑上 是 maxpooling 最大值的元素則

$$\frac{\partial L}{\partial C_{muv}^2} = \frac{\partial L}{\partial C_{mu_{max}v_{max}}^2} = \frac{\partial L}{\partial P_{mxy}^2}$$

令 fk=Pmxy

$$\frac{\partial L}{\partial C_{mu_{max}v_{max}}^2} = \frac{\partial L}{\partial P_{mxy}^2} = \frac{\partial L}{\partial f_k}$$

By chain rule 從 output 端口回推

$$\frac{\partial L}{\partial f_k} = \sum_{i=0}^{9} \frac{\partial L}{\partial S_i} \frac{\partial S_i}{\partial f_k}$$

$$\frac{\partial L}{\partial f_k} = \sum_{i=0}^{9} \frac{\partial L}{\partial S_i} w_{ik}$$

令 convolution 為 1D arrary 則

$$\mathbf{dLf} = \mathbf{w}^T \cdot \mathbf{dLS}$$

令 convolution 為 3D arrary 則

$$\mathbf{dLP2} = \mathbf{dLf}.\mathrm{reshape}(\boldsymbol{P}^2.\mathrm{shape})$$

若今天 要對 pooling 不是最大值的 nuron 更新 可以發現 其不需要更新 因為其梯度造成的影響甚小

<table>
<tr>
<td></td>
<td>

小結

$$\frac{\partial L}{\partial C_{muv}^2} = \begin{cases} \frac{\partial L}{\partial P_{mxy}^2},\ \big(x = \text{floor}(u/2),\ y = \text{floor}(v/2)\big) \text{ if } C_{muv}^2 \text{ is the maximum element} \\ \text{out of 4 elements.} \\ 0 \text{ otherwise} \end{cases}$$

</td>
</tr>
<tr>
<td>RBFN</td>
<td>

```
def call(self, inputs):
    diff = K.expand_dims(inputs) - self.mu
    l2 = K.sum(K.pow(diff, 2), axis=1)
    res = K.exp(-1 * self.gamma * l2)
    # res = 1/K.sqrt(1 + self.gamma * l2)
    return res
```

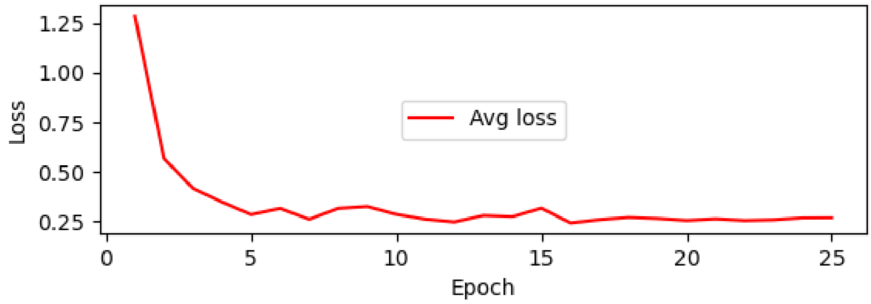
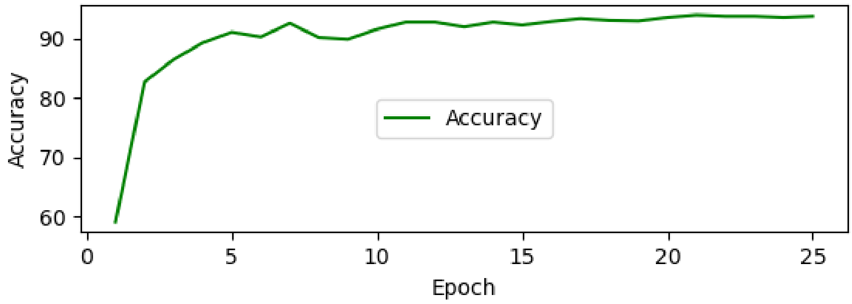
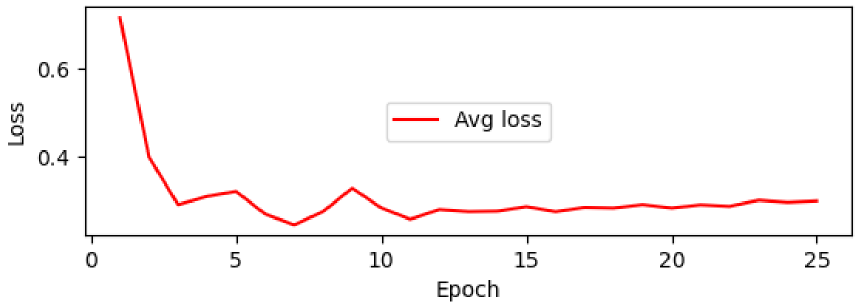
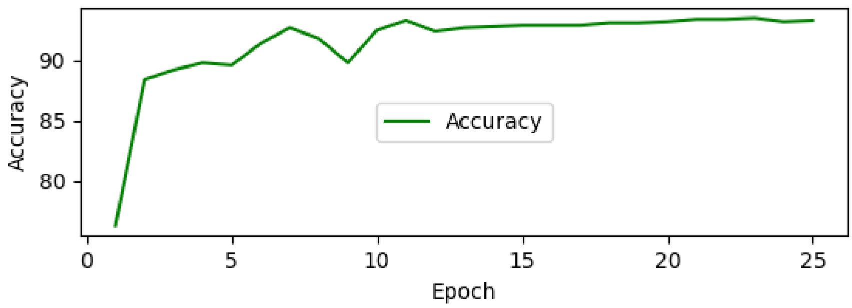
$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i\, \varphi(\|\mathbf{x} - \mathbf{x}_i\|),$$

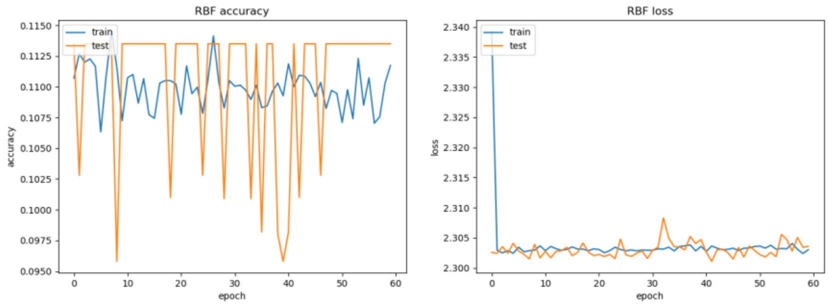
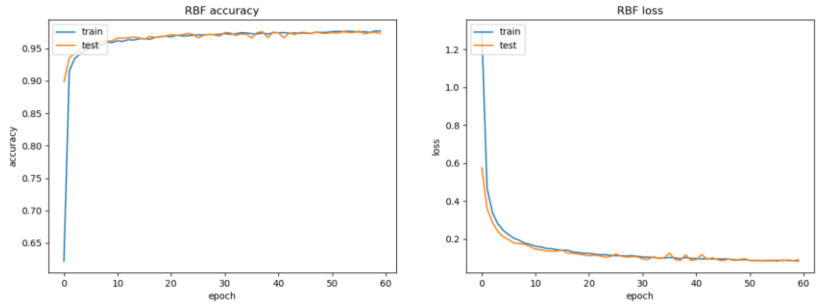
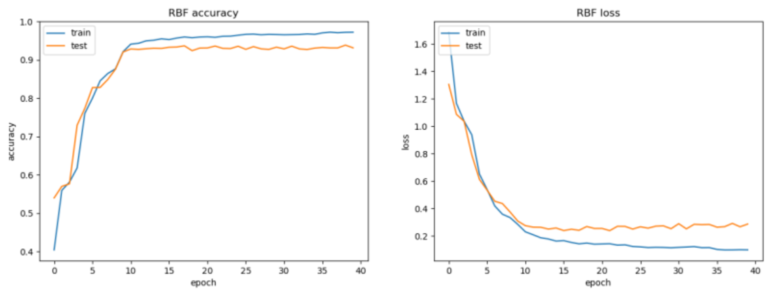
N 為 hidden layer 的 neuron 個數 Xi 代表的是中心點位置 w 為 權重 y 為 該 neuron 的輸出值

淨向基函數將 input 與一定點去做距離比較
距離的計算方式有許多種 本次作業採用了高斯函數逆多二次函數

$$\exp\left[-\beta\|\mathbf{x} - \mathbf{c}_i\|^2\right].$$

</td>
</tr>
</table>

Result		
Model_name	lr	PICTURE
CNN	0.05	 
CNN	0.01	 

<table>
<tr><td>CNN+RBF</td><td>0.05</td><td>
</td></tr>
<tr><td>CNN+RBF</td><td>0.005</td><td>
</td></tr>
<tr><td>RBF</td><td>0.05</td><td>
</td></tr>
</table>

RBF&& NN 比較與討論
● 從結果上來看純粹用 Dense + RBF 的網路 因為沒有經過特徵篩選 故會相較於 CNN based RBF 網路在相同的 learning rate 下較慢 提升 accuracy ● 選用 CNN 取代 dense 除了獲取到較好的特徵外 也減少了 weight 的數量 對硬體上設計也是較好的 ● 相同的 learning rate 下 CNN 相較於 CNN +RBF 可以接受較大的 learning rate 來找到更好的 performance ● 同樣的網路架構 learning rate 調過大可能導致無法收斂 CNN +RBF 需用較小的 learning rate tune 出 model 但也不排除是因為網路較為複雜的緣故。