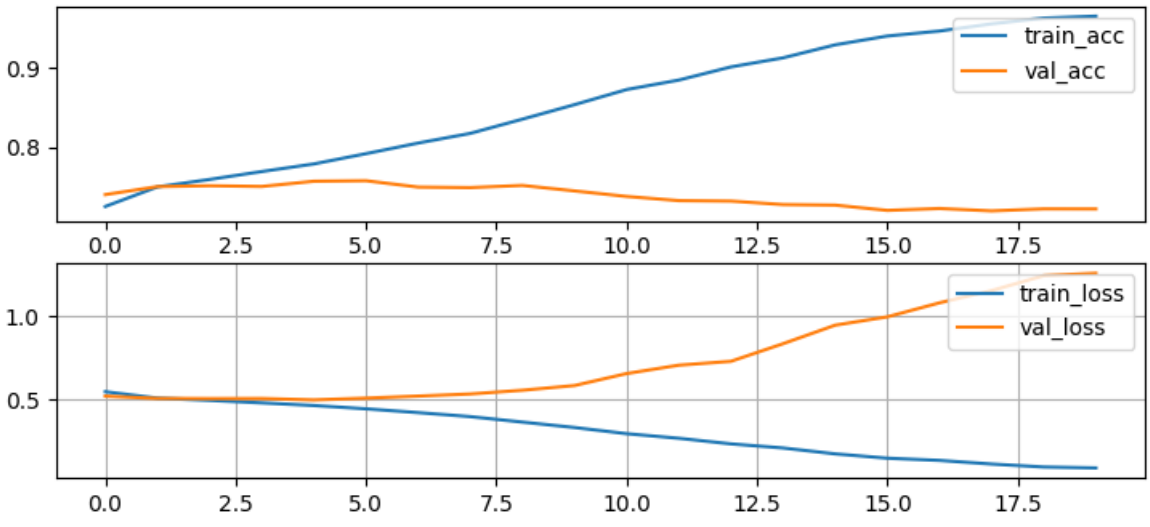
Machine Learning HW6 Report

學號：B05901170 系級：電機三 姓名：陳柏志

1. (1%) 請說明你實作之 RNN 模型架構及使用的 word embedding 方法，回報模型的正確率並繪出訓練曲線*
   a. 模型架構：

```
class my_RNN_Net(nn.Module):
    def __init__(self, weight):
        super(my_RNN_Net, self).__init__()

        self.input_size = 250
        self.dropout = 0.5
        self.num_hiddens = 200
        self.num_layers = 2
        self.bidirectional = True

        self.embedding = nn.Embedding.from_pretrained(weight)
        self.embedding.weight.requires_grad = False

        self.encoder = nn.LSTM(self.input_size, hidden_size=self.num_hiddens,
                               num_layers=self.num_layers, bidirectional=self.bidirectional,
                               dropout=self.dropout)
    def forward(self, inputs):
        embeddings = self.embedding(inputs)

        states, hidden_out = self.encoder(embeddings.permute([1, 0, 2]))
        encoding = torch.cat([states[0], states[-1]], dim=1)
        outputs = self.decoder(encoding)
        return outputs
```

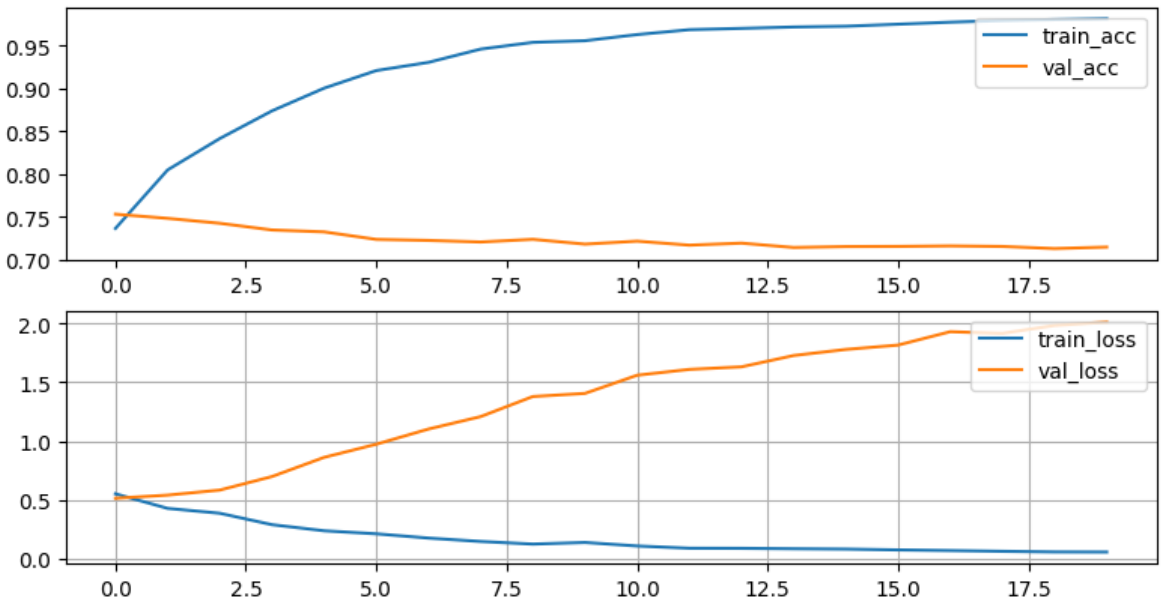
   b. word embedding 方法:
      將所有 test data 及 train data 中的辭彙編成辭典，再將所有句子依此辭典編碼成 id 的形式。再模型的 embedding layer 中會將每個字轉成 250 維的 vector。此一 vector 的產生使用了 gensim 套件。
   c. 正確率：
      i. public: 0.75600
      ii. private: 0.74840
   d. 訓練曲線：



2. (1%) 請實作 BOW+DNN 模型，敘述你的模型架構，回報模型的正確率並繪出訓練曲線*
    a. 模型架構：

```
class my_RNN_Net(nn.Module):
    def __init__(self, weight):
        super(my_RNN_Net, self).__init__()

        self.input_size = 250
        self.dropout = 0.5
        self.num_hiddens = 200
        self.num_layers = 2
        self.bidirectional = True

        self.embedding = nn.Embedding.from_pretrained(weight)
        self.embedding.weight.requires_grad = False

        self.encoder = nn.LSTM(self.input_size, hidden_size=self.num_hiddens,
                               num_layers=self.num_layers, bidirectional=self.bidirectional,
                               dropout=self.dropout)
    def forward(self, inputs):
        embeddings = self.embedding(inputs)

        states, hidden_out = self.encoder(embeddings.permute([1, 0, 2]))
        encoding = torch.cat([states[0], states[-1]], dim=1)
        outputs = self.decoder(encoding)
        return outputs
```

    b. 正確率：
        i. public: 0.75660
        ii. private: 0.75480
    c. 訓練曲線：



3. (1%) 請敘述你如何 improve performance（preprocess, embedding, 架構等），並解釋為何這些做法可以使模型進步。

   我調整不同參數後使用 3 個 model 進行 emsemble，進而提高模型表現。
   emsemble 可以讓不同模型的結果進行平均，因此能達到更好的效果。

4. (1%) 請比較不做斷詞 (e.g., 以字為單位) 與有做斷詞，兩種方法實作出來的效果差異，並解釋為何有此差別。

	Public Score	Private Score
RNN 不做斷詞	0.72850	0.72000
RNN 做斷詞	0.75600	0.74840

   可以看出不做斷詞的表現較差，可能是因為不同意思的詞中可能出先同樣的字，因在做詞之間的相關向量時可能得到沒什麼意義的向量，因而導致訓練結果不佳。

5. (1%) 請比較 RNN 與 BOW 兩種不同 model 對於 "在說別人白痴之前，先想想自己"與"在說別人之前先想想自己，白痴" 這兩句話的分數（model output），並討論造成差異的原因。

	"在說別人白痴之前，先想想自己"	"在說別人之前先想想自己，白痴"
RNN	[0.20301035 -0.20443627]	[0.2115915 -0.21291995]
BOW	[-0.02443599 0.0223346]	[-0.02443599 0.0223346]

   BOW 只看詞出現的次數，因此白痴不論出現在那都會被判斷成負面句子，且因為組成這兩句的詞完全一樣，因此他們的 BOW 也是一樣的，因此輸出的結果也是一樣的。
   而 RNN 則可能考慮辭彙出現的前後順序來判斷結果，因此輸出結果可能與 BOW 完全不同。而此處兩句話都輸出一樣的結果（非惡意）則可能是我的 model 沒訓練好。