Applied Deep Learning (2022 Spring)
Homework 1

資管四 B07705015 劉鎮霆

Q1: Data processing

a. How do you tokenize the data

- Intent classification
  (1) 將 training set 與 validation set 中句子 [1] 的標點符號 [2] 替換為空白字元
  (2) 將句子依據空白字元切分成 token 的陣列
  (3) 將每個 token 轉換為小寫
- Slot Tagging
  (1) 由於資料集已提供 token 的陣列 [3]，因此僅將每個 token 轉換為小寫

b. The pre-trained embedding you used

(1) Intent classification 與 slot tagging 皆使用 GloVe pre-trained word vectors [4] 轉換成對應向量

(2) 所有未出現於 GloVe 的 token 皆被視為額外的 [UNK] token，並給予此 token 隨機生成的向量 (向量中的每個值位於 $[-1, 1)$ 的範圍中)

(3) 另外新增 [PAD] token 並以相同方式對應至隨機向量，此 token 將置於較短的句子之後，把所有句子增長至相同長度 [5]

Q2: Intent classification model

a. Your model

模型架構：

- 將 N 表示為輸入的資料筆數，L 表示為單句的 token 數

	Layer	Input	Output	Note
Word Embedding	GloVe	(N, L)	$(N, L, 300)$	-
Recurrent	GRU	$(N, L, 300)$	$(N, L, 256)$	# of layers: 1 bidirectional
Fully Connected	Dropout	$(N, 256)$	$(N, 256)$	0.5
	Linear	$(N, 256)$	$(N, 256)$	-
	ELU	$(N, 256)$	$(N, 256)$	$\alpha = 1$
	Dropout	$(N, 256)$	$(N, 256)$	0.5
	Linear	$(N, 256)$	$(N, 150)$	-

[1] 位於 intent classification 資料集的「text」欄位。

[2] 包含半形的「.」、「,」、「;」、「?」、「!」、「(」與「)」。

[3] 位於 slot tagging 資料集的「tokens」欄位。

[4] 使用 Common Crawl 版本，包含 840B tokens，並將其對應至 300 維度的向量。

[5] Intent classification 的最長長度為 64，slot tagging 則為最長句子的長度，超過此長度的句子去除後半多餘的 token，不足的則填入 [PAD] token。

以下描述輸入一個句子時，模型運算的步驟：

- 將 T 表示為句子增長後的 token 數，S 表示為句子實際的 token 數 (若句子超過設定的最長長度，則 $S = T$)
- 將 C 表示為 intent 類別總數
- 將 g(·,·) 表示為 fully connected layers (包含 Dropout、Linear 與 ELU)
- 將 $\overrightarrow{W}_{GRU}, \overleftarrow{W}_{GRU}, W_g$ 分別表示為 GRU 正反向及 fully connected layers 的參數
- 將 x_t 表示為第 t 個 token 的 embedding vector
- 將 $\vec{h}_t, \overleftarrow{h}_t$ 分別表示為 GRU 於第 t 個時間點輸出的正向與反向 hidden state，$\vec{h}_0, \overleftarrow{h}_{T+1}$ 分別為正向與反向的初始 hidden state
- 將 o 表示為模型的輸出值，o_c 為第 c 項類別的輸出值，預測的類別為 y^{pred}

$$\vec{h}_t = \mathrm{GRU}\left(\overrightarrow{W}_{GRU}, x_t, \vec{h}_{t-1}\right), \forall t \in \{1, \dots, T\},$$
$$\overleftarrow{h}_t = \mathrm{GRU}\left(\overleftarrow{W}_{GRU}, x_t, \overleftarrow{h}_{t+1}\right), \forall t \in \{1, \dots, T\},$$
$$\mathrm{o} = \mathrm{g}\left(W_g, \left[\vec{h}_S; \overleftarrow{h}_1\right]\right),$$
$$y^{pred} = \underset{c \in \{1,\dots,C\}}{\operatorname{argmax}} o_c.$$

b. <u>Performance of your model</u>

- Public score on kaggle：0.91644

c. <u>The loss function you used</u>

計算每筆資料的輸出預測機率分布與 intent 實際值的 cross entropy，

- 將 N 表示為輸入的資料筆數，C 表示為 intent 類別總數
- 將 o 表示為模型的輸出值，$o_{n,c}$ 為第 n 筆輸入、第 c 項類別的輸出值
- 將 y 表示為資料類別的實際值，y_n 為第 n 筆輸入的 intent 類別

模型的損失函數 $\ell(o, y)$ 為：

$$\ell(o, y) = -\frac{1}{N}\sum_{n=1}^{N} \log \frac{\exp\left(o_{n,y_n}\right)}{\sum_{c=1}^{C} \exp\left(o_{n,c}\right)}$$

d. <u>The optimization algorithm, learning rate and batch size</u>

- Optimizer：Adam

Learning rate	β_1	β_2
0.001	0.9	0.999

- Batch size：32

Q3: Slot tagging model

a. Your model

模型架構：

- 將 N 表示為輸入的資料筆數，L 表示為單句的 token 數

	Layer	Input	Output	Note
Word Embedding	GloVe	(N, L)	$(N, L, 300)$	-
Recurrent	GRU	$(N, L, 300)$	$(N, L, 256)$	# of layers: 1 bidirectional
Fully Connected	Dropout	$(NL, 256)$	$(NL, 256)$	0.5
	Linear	$(NL, 256)$	$(NL, 256)$	-
	ELU	$(NL, 256)$	$(NL, 256)$	$\alpha = 1$
	Dropout	$(NL, 256)$	$(NL, 256)$	0.5
	Linear	$(NL, 256)$	$(NL, 9)$	-

以下描述輸入一個句子時，模型運算的步驟：

- 將 T 表示為句子增長後的 token 數，S 表示為句子實際的 token 數 ，C 表示為 tag 類別總數
- 將 $\mathrm{g}(\cdot,\cdot)$ 表示為 fully connected layers (包含 Dropout、Linear 與 ELU)
- 將 $\overrightarrow{W}_{GRU}, \overleftarrow{W}_{GRU}, W_g$ 分別表示為 GRU 正反向及 fully connected layers 的參數
- 將 x_t 表示為第 t 個 token 的 embedding vector
- 將 $\vec{h}_t, \overleftarrow{h}_t$ 分別表示為 GRU 於第 t 個時間點輸出的正向與反向 hidden state，$\vec{h}_0, \overleftarrow{h}_{T+1}$ 分別為正向與反向的初始 hidden state
- 將 o_t 表示為模型第 t 個 token 的輸出值，$o_{t,c}$ 為該 token 第 c 項類別的輸出值，預測的類別為 y_t^{pred}

$$\vec{h}_t = \mathrm{GRU}\left(\overrightarrow{W}_{GRU}, x_t, \vec{h}_{t-1}\right), \forall t \in \{1, \dots, T\},$$
$$\overleftarrow{h}_t = \mathrm{GRU}\left(\overleftarrow{W}_{GRU}, x_t, \overleftarrow{h}_{t+1}\right), \forall t \in \{1, \dots, T\},$$
$$\mathrm{o}_t = \mathrm{g}\left(W_g, \left[\vec{h}_t; \overleftarrow{h}_t\right]\right), \forall t \in \{1, \dots, S\},$$
$$y_t^{pred} = \underset{c \in \{1,\dots,C\}}{\operatorname{argmax}} o_{t,c}, \forall t \in \{1, \dots, S\}.$$

b. Performance of your model

- Public score on kaggle：0.78766

c. The loss function you used

計算每筆 token 的輸出預測機率分布與 tag 實際值的 cross entropy，

- 將 N 表示為輸入的 token 總數[6]，C 表示為 tag 類別總數

[6] Token 總數為每個句子 token 數的總和，不包含增加句子長度的 [PAD] token。

- 將 o 表示為模型的輸出值，$o_{n,c}$ 為第 n 筆 token、第 c 項類別的輸出值
- 將 y 表示為資料類別的實際值，y_n 為第 n 筆 token 的 tag 類別

模型的損失函數 $\ell(o,y)$ 為：

$$\ell(o,y) = -\frac{1}{N}\sum_{n=1}^{N}\log\frac{\exp(o_{n,y_n})}{\sum_{c=1}^{C}\exp(o_{n,c})}$$

d. The optimization algorithm, learning rate and batch size

- Optimizer：Adam

Learning rate	β_1	β_2
0.001	0.9	0.999

- Batch size：32

Q4: Sequence tagging evaluation

a. Evaluation model in Q3 with seqeval, token accuracy, and join accuracy

使用 validation set 進行計算

- Seqeval report：

	precision	recall	f1-score	support
date	0.72	0.74	0.73	206
first_name	0.94	0.91	0.93	102
last_name	0.89	0.81	0.85	78
people	0.69	0.68	0.68	238
time	0.81	0.79	0.80	218
micro avg	0.78	0.76	0.77	842
macro avg	0.81	0.78	0.80	842
weighted avg	0.78	0.76	0.77	842

- Token accuracy：0.9618552923202515
- Join accuracy：0.7730000019073486

b. The differences between the evaluation method in seqeval, token accuracy, and join accuracy

- Seqeval 計算方式：IOB2 格式會從句子標註出數個 chunk，一個 chunk 會包含一或多個 token，其中 token 會被標記為：
  - 「O」代表不屬於任何一個 chunk
  - 「B-xxx」代表屬於 xxx 類型的 chunk，且是該 chunk 的第一個 token
  - 「I-xxx」代表屬於 xxx 類型的 chunk，且非該 chunk 的第一個 token

只有整個 chunk 的 token 都預測正確，才會被視為正確，針對不同類型的指標，可以計算出 precision、recall、F1 score 等：

$$precision_i = \frac{\#\ of\ chunks\ successfully\ predicted\ to\ be\ type\ i}{\#\ of\ chunks\ predicted\ to\ be\ type\ i}$$

$$recall_i = \frac{\#\ of\ chunks\ successfully\ predicted\ to\ be\ type\ i}{\#\ of\ chunks\ in\ type\ i}$$

$$F_{1,i} = \frac{2 \times precision_i \times recall_i}{precision_i + recall_i}$$

$$support_i = \#\ of\ chunks\ in\ type\ i$$

Micro average 考慮所有類型的 chunk，例如[7]：

$$precision_{micro} = \frac{\#\ of\ chunks\ successfully\ predicted}{\#\ of\ chunks\ predicted}$$

Macro average 將不同類型的指標平均，假設有 N 種類型的 chunk：

$$precision_{macro} = \frac{1}{N}\sum_{i=1}^{N} precision_i$$

Weighted average 將不同類型的指標依照 support 進行加權平均：

$$precision_{weighted} = \frac{1}{support}\sum_{i=1}^{N} support_i \times precision_i$$

其中，support 代表所有 chunk 的總數：

$$support = \sum_{i=1}^{N} support_i$$

- Token accuracy 計算方式：

$$\frac{\#\ of\ tokens\ successfully\ predicted}{\#\ of\ tokens}$$

- Join accuracy 計算方式：

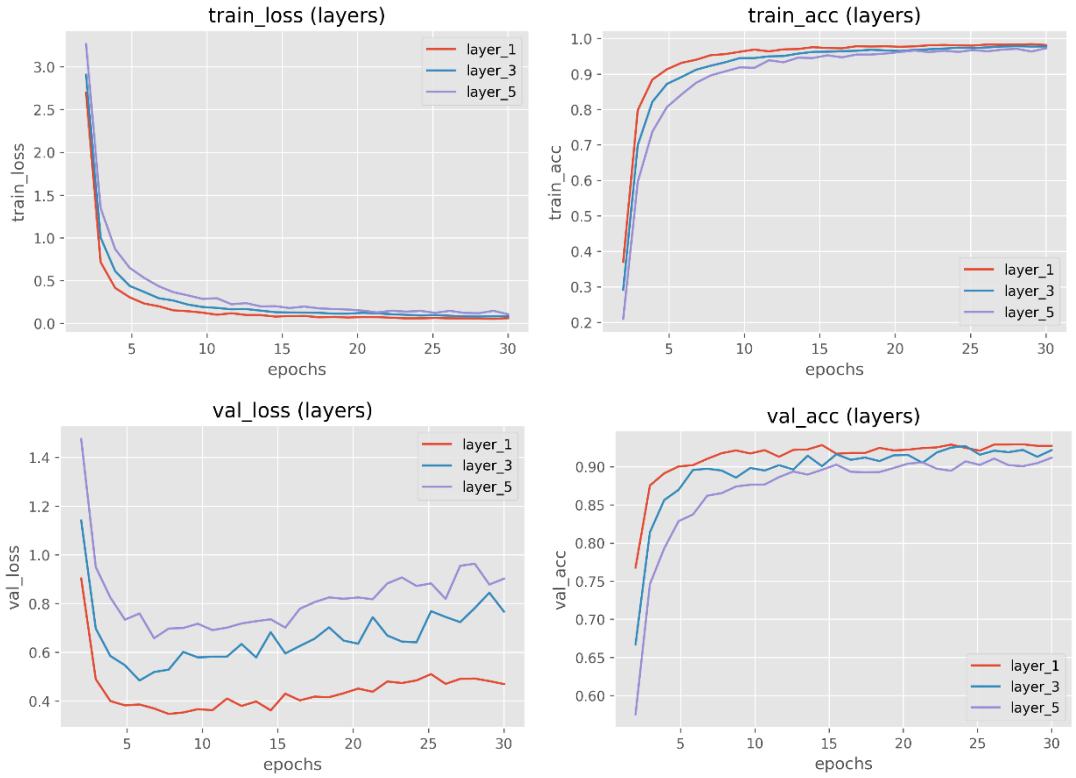
$$\frac{\#\ of\ sentences\ with\ all\ tokens\ successfully\ predicted}{\#\ of\ sentences}$$

[7] 由於其他指標的計算方法類似，因此只列出 precision。

Q5: Different configuration

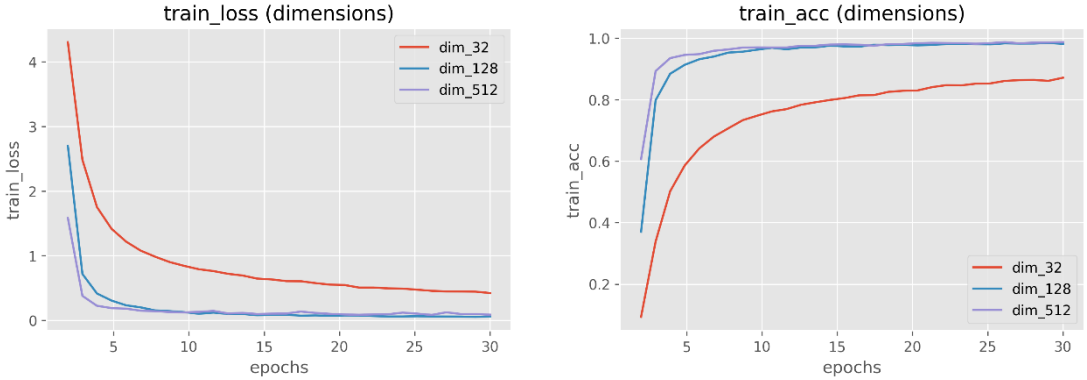
以 intent classification 進行測試，以下無特別說明的參數或設定皆與 Q2 相同：

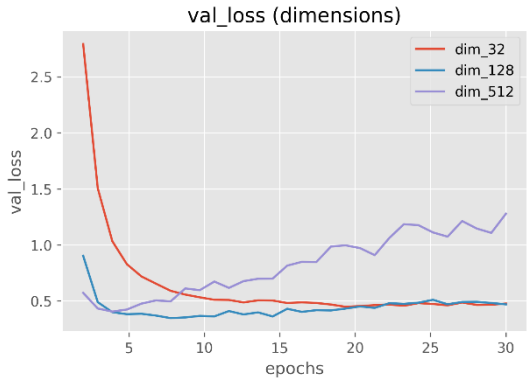
a. Number of layers (in GRU)

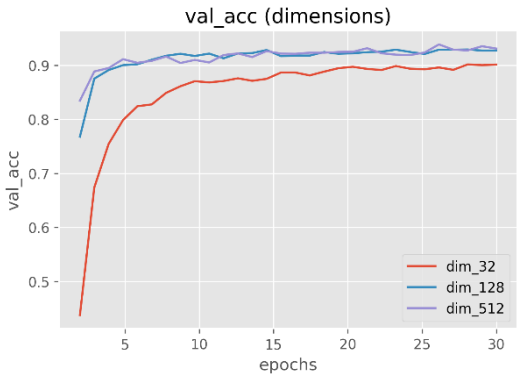


由上圖可以發現，對於不同層數的模型，training set 的 loss 與 accuracy 差異不大，可能是因為低層數的模型就足以表達句子和其類型的關聯；然而，對於 validation set 而言，也可以注意到當模型的層數由 1 增加至 5 時，loss 明顯地增加了，同時 accuracy 也有所下降，顯示高層數的模型可能出現 overfitting 的狀況。

b. Hidden dimension





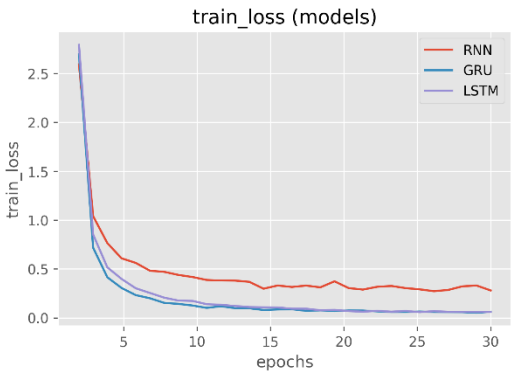


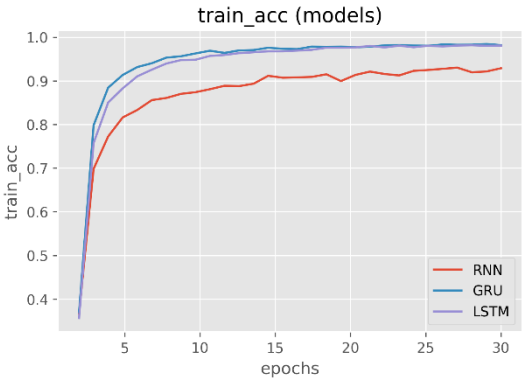
上圖顯示當 hidden dimension 設為 32 時，其 training loss、training accuracy 及 validation accuracy 的表現皆比其他維度更高的模型差，這可能顯示了低維度模型的 underfitting 問題； 可以看出當維度增加至 128 時模型的表現便有所改善，然而繼續將維度增加至 512 後，雖然模型對於 training set 的表現差異不大，但能夠觀察到 validation loss 隨著前幾個 epoch 下降後便持續增加，代表在高維度的模型下可能產生 overfitting 的問題。

c. GRU / LSTM / RNN

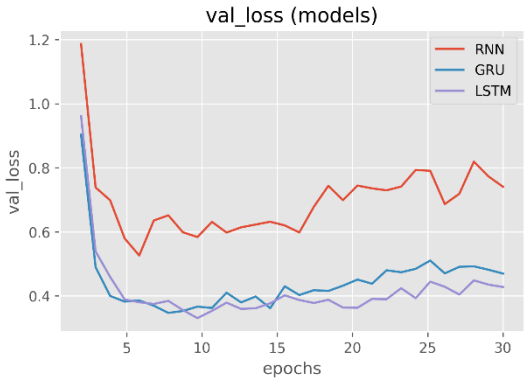
由於 GRU、LSTM 及 RNN 的模型架構不同，同樣層數和 hidden dimension 時會有不同的參數數量，因此實驗將固定模型層數，透過調整 hidden dimension 使三個模型的參數數量較為接近以進行比較，設定如下：

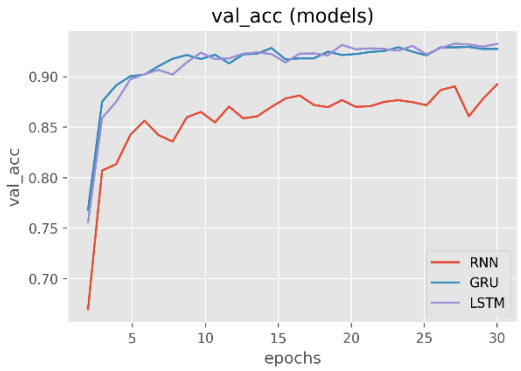
		RNN	GRU	LSTM
hidden dimension		204	128	108
參數數量 [8]	recurrent	206 K	330 K	354 K
	full connected	228 K	104 K	79.4 K
	total	434 K	434 K	433 K





[8] 由於三者使用的 embedding layer 相同，因此此處不討論 embedding layer 的參數數量。





從設定來看，可以注意到由於 LSTM 需要藉由 forget gate、input gate 與 output gate 紀錄長期資料，在同樣設定下有更多的參數，因此在實驗中的 hidden dimension 最少；GRU 相比 LSTM 省去了 cell state，所以在實驗中可以有較大的 hidden dimension；而 RNN 則因為架構最簡單，因此在實驗中的 hidden dimension 最大。

三個模型中，可以觀察到 RNN 在此設定下的表現較差，而 GRU 與 LSTM 較為接近，且 LSTM 的 validation loss 最低。

d. CNN-BiLSTM

模型架構參考 Sentiment Analysis of Movie Reviews Based on CNN-BLSTM 並進行調整，以 CNN output channel 為 128 為例，模型架構如下：

- 將 N 表示為輸入的資料筆數，L 表示為單句的 token 數，L' 表示為 max pool 的輸出長度

	Layer	Input	Output	Note
Word Embedding	GloVe	(N, L)	$(N, L, 300)$	-
Convolutional	Conv1d	$(N, 300, L)$	$(N, L, 128)$	kernel size: 1
	ELU	$(N, L, 128)$	$(N, L, 128)$	-
	MaxPool1d	$(N, L, 128)$	$(N, L', 128)$	kernel size: 4
Recurrent	LSTM	$(N, L', 128)$	$(N, L', 268)$	# of layers: 1 bidirectional
Fully Connected	Dropout	$(N, 268)$	$(N, 268)$	0.5
	Linear	$(N, 268)$	$(N, 268)$	-
	ELU	$(N, 268)$	$(N, 268)$	$\alpha = 1$
	Dropout	$(N, 268)$	$(N, 268)$	0.5
	Linear	$(N, 256)$	$(N, 150)$	-

以下描述輸入一個句子時，模型運算的步驟：

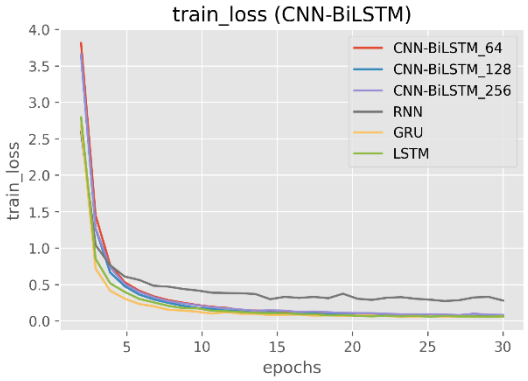
- 將 T 表示為句子增長後的 token 數，將 T' 表示為 max pooling 後的長度
- 將 C 表示為 intent 類別總數
- 將 g(·,·) 表示為 fully connected layers (包含 Dropout、Linear 與 ELU)

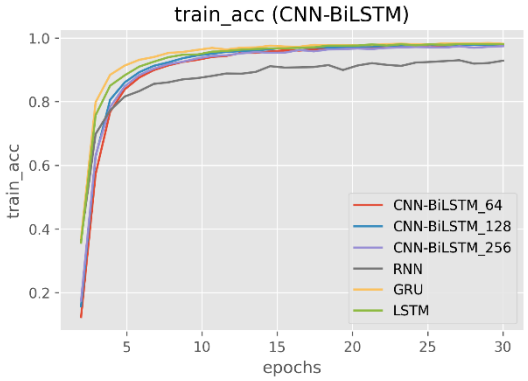
- 將 $W_{CNN}, \overrightarrow{W}_{LSTM}, \overleftarrow{W}_{LSTM}, W_g$ 分別表示為 CNN、 GRU 正反向及 fully connected layers 的參數
- 將 x_t 表示為第 t 個 token 的 embedding vector，s_t 表示為 CNN 輸出的第 t 個 output channel 的向量
- 將 $\vec{h}_t, \overleftarrow{h}_t, \vec{c}_t, \overleftarrow{c}_t$ 分別表示為 LSTM 於第 t 個時間點輸出的正向與反向 hidden state 及 cell state，$\vec{h}_0, \overleftarrow{h}_{T'+1}, \vec{c}_0, \overleftarrow{c}_{T'+1}$ 分別為正向與反向的初始 hidden state 及 cell state
- 將 o 表示為模型的輸出值，o_c 為第 c 項類別的輸出值，預測的類別為 y^{pred}

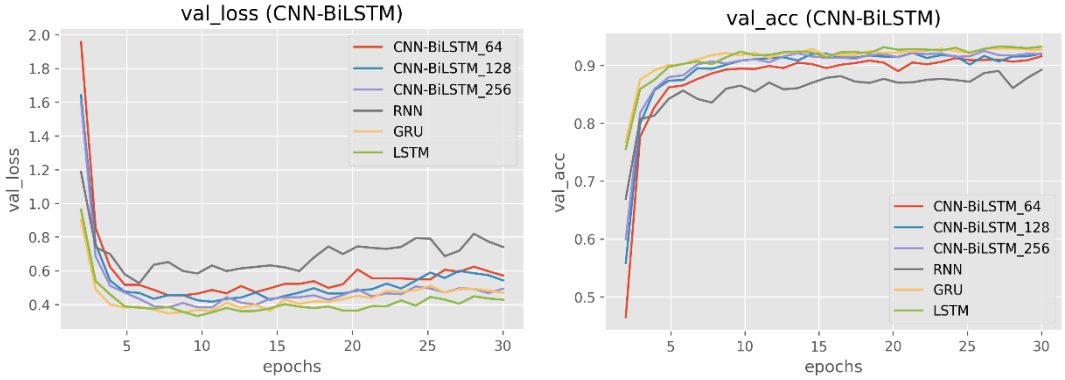
$$s = \mathrm{CNN}(W_{CNN}, x)$$
$$\vec{h}_t, \vec{c}_t = \mathrm{LSTM}\left(\overrightarrow{W}_{LSTM}, s_t, \vec{h}_{t-1}, \vec{c}_t\right), \forall t \in \{1, \ldots, T'\},$$
$$\overleftarrow{h}_t, \overleftarrow{c}_t = \mathrm{LSTM}\left(\overleftarrow{W}_{LSTM}, s_t, \overleftarrow{h}_{t+1}, \overleftarrow{c}_t\right), \forall t \in \{1, \ldots, T'\},$$
$$\mathrm{o} = \mathrm{g}\left(W_g, \left[\vec{h}_T; \overleftarrow{h}_1\right]\right),$$
$$y^{pred} = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}}\, o_c.$$

測試不同參數設定並與 c. 的三個模型進行比較：

		Bi-LSTM		
CNN output channels		64	128	256
LSTM hidden dimension		155	134	100
參數數量	convolutional	19.3 K	38.5 K	77.1 K
	recurrent	274 K	283 K	286 K
	full connected	143 K	112 K	70.4 K
	total	436 K	433 K	433 K







在 training set 的表現上，CNN-BiLSTM 與 LSTM、GRU 相似，而從 validation set 的表現可以看出 CNN 的 output channel 越大，模型的表現越好，且越接近一般的 LSTM，不過在以上設定中，CNN-BiLSTM 雖然比 RNN 的表現更好，但相對於 LSTM 與 GRU 則沒有更好的表現。