CNN for TFBS predictions and model interpretability

Author	資工四甲 U0924028 符絜棻
Tags	CNN TFBS
程式碼來源	https://github.com/PacktPublishing/Deep-Learning-for-Genomics-/tree/main/Chapter10

主題描述

在調控基因表達和進化的過程中，如DNA複製及RNA轉錄，轉錄因子(TF)與TF結合位點(TFBS)的結合有至關重要的作用 ⇒ 精確類比基因的特異性及尋找TFBS有助於探索細胞表達的機制。

> TFBS預測問題中，所有基因序列會根據DNA序列中是否能找到TFBS分為兩類，由標籤0或1表示
>
> $$\begin{cases} 0 & 陰性 - 沒有與TF的結合位點 \\ 1 & 陽性 - 有TFBS \end{cases}$$

使用資料

sequences_mod.txt → 共2000筆資料，每一筆DNA序列的長度均為50

	sequence
0	CCGAGGGCTATGGTTTGGAAGTTAGAACCCTGGGGCTTCTCGCGGA...
1	GAGTTTATATGGCGCGAGCCTAGTGGTTTTTGTACTTGTTTGTCGC...
2	GATCAGTAGGGAAACAAACAGAGGGCCCAGCCACATCTAGCAGGTA...
3	GTCCAAAAGAGGAAGTTCACCTTGACCGCAGAGGTACCACCAGAGC...
4	GGAAAGAGGAAGTTAACTAGAACCTGCATAACTGGCCTGGGAGATA...

	sequence
1995	GTCGCGCGGGTGCGGAGGATGAGTCGCAGACGCATTTATGTCGCCC...
1996	GTTCGCAGCGTATTGAGTAATGTTTGACTAAAGAGGAAGTTATATT...
1997	ACTCGCTGTCCACGTCTATTCCTAGGGGTTTTATTTCGCAAGGTGA...
1998	TGCAAAGGGGAAAGAGGAAGTTCTTTACCGCGGAGTTATTCATAAT...
1999	AATGTAAAGAGGAAGTTGCACTGCTGGCCCGGGCCTATATCGAGAG...

labels.txt → sequences_mod.txt中每筆DNA序列中是否存在TFBS

```
[0 0 0 ... 0 1 1]
```

Distribution of Positive and Negative samples

```
In [7]: print(pd.DataFrame(Y).value_counts())
```

```
0    1013
1     987
dtype: int64
```

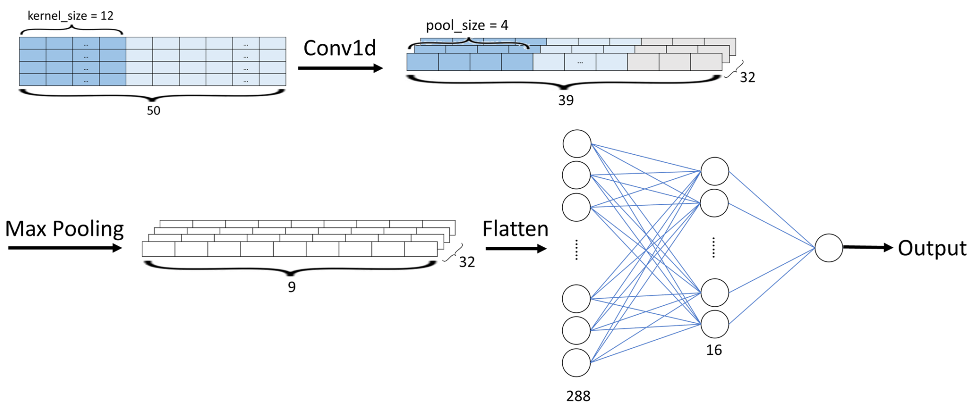
CNN Model

原始模型

In [10]:
```
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=12, input_shape=(50, 4)))
model.add(MaxPooling1D(pool_size=4))
model.add(Flatten())
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

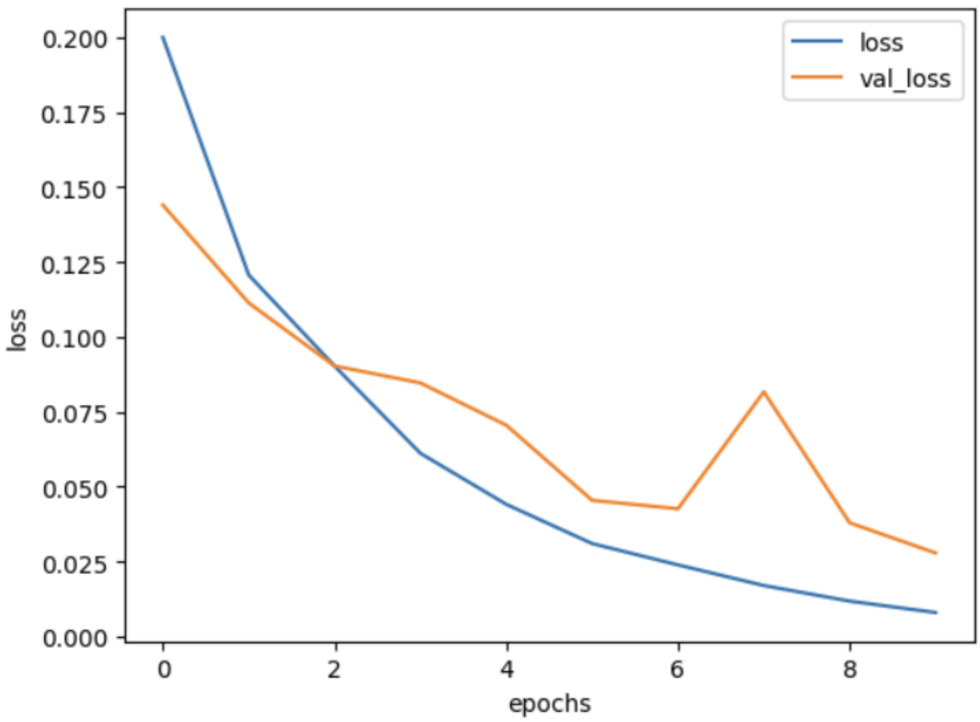
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 39, 32)            1568

 max_pooling1d (MaxPooling1D  (None, 9, 32)            0
 )

 flatten (Flatten)           (None, 288)               0

 dense (Dense)               (None, 16)                4624

 dense_1 (Dense)             (None, 1)                 17

=================================================================
Total params: 6,209
Trainable params: 6,209
Non-trainable params: 0
```

由於原始輸出層僅有一個神經元，模型的預測值是一個介於 0 到 1 之間的實數，用於表示是否存在 TFBS 的可能性。



訓練結果：

```
Epoch 1/10
135/135 [==============================] - 1s 2ms/step - loss: 0.2129 - val_loss: 0.1548
Epoch 2/10
135/135 [==============================] - 0s 1ms/step - loss: 0.1240 - val_loss: 0.1124
Epoch 3/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0896 - val_loss: 0.0909
Epoch 4/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0589 - val_loss: 0.0696
Epoch 5/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0398 - val_loss: 0.0570
Epoch 6/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0264 - val_loss: 0.0451
Epoch 7/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0177 - val_loss: 0.0630
Epoch 8/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0149 - val_loss: 0.0390
Epoch 9/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0104 - val_loss: 0.0341
Epoch 10/10
135/135 [==============================] - 0s 1ms/step - loss: 0.0069 - val_loss: 0.0339
```



雖然訓練集及驗證集的損失隨著epoch下降，但驗證集的損失呈現波動狀態，而非穩定下降 ⇒ 猜測模型可能有overfitting的情況發生。

修改模型

1. 使用padding進行填充，提升邊緣資訊的可參考程度
2. 修改pool_size，使所有特徵值都能進行max pooling
3. 輸出層更改為兩個神經元，使預測結果與實際標籤更相符

```
#標籤做one hot encoding
labels = pd.read_csv('labels.txt')
Labels = np.array(labels).reshape(-1)
print(Labels[:5])
iecd = iec.fit_transform(Labels[:])
iecd = np.array(iecd).reshape(-1, 1)
ohed = ohe.fit_transform(iecd)
Y = np.stack(ohed.toarray())
```

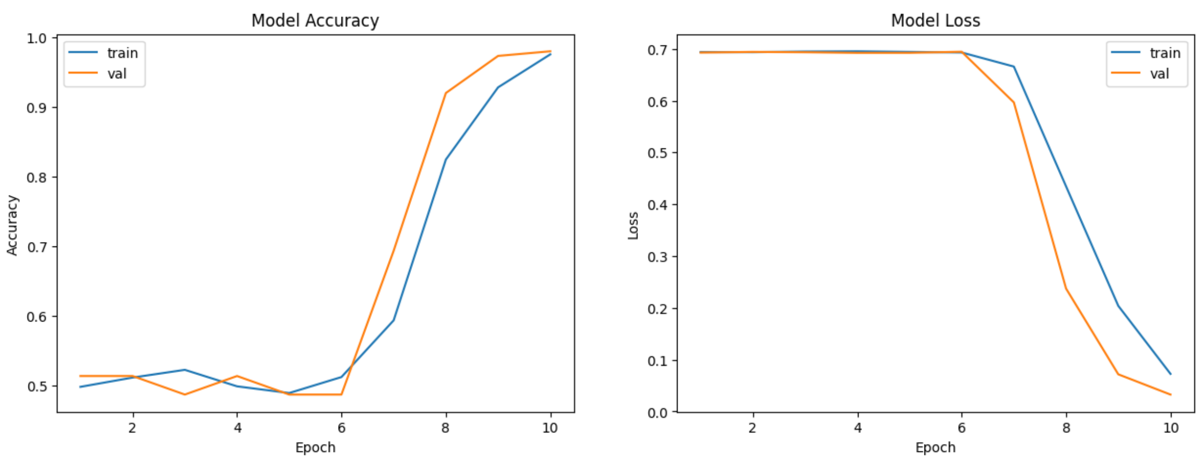
4. 增加模型的深度

5. 設定dropout ratio，避免overfitting

```
dropout = 0.2
model = Sequential()
model.add(Conv1D(filters=32, kernel_size=12, input_shape=(50, 4), padding='same'))
model.add(MaxPooling1D(pool_size=5))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(128, activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(64, activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(32, activation='relu'))
model.add(Dropout(dropout))
model.add(Dense(16, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

訓練結果：

```
Epoch 1/10
135/135 [==============================] - 2s 7ms/step - loss: 0.6938 - accuracy: 0.4978 - val_loss: 0.6927 - val_accuracy: 0.5133
Epoch 2/10
135/135 [==============================] 1s 6ms/step - loss: 0.6935 - accuracy: 0.5111 - val_loss: 0.6940 - val_accuracy: 0.5133
Epoch 3/10
135/135 [==============================] - 1s 5ms/step - loss: 0.6948 - accuracy: 0.5222 - val_loss: 0.6934 - val_accuracy: 0.4867
Epoch 4/10
135/135 [==============================] - 1s 5ms/step - loss: 0.6952 - accuracy: 0.4985 - val_loss: 0.6922 - val_accuracy: 0.5133
Epoch 5/10
135/135 [==============================] - 1s 5ms/step - loss: 0.6940 - accuracy: 0.4889 - val_loss: 0.6922 - val_accuracy: 0.4867
Epoch 6/10
135/135 [==============================] - 1s 6ms/step - loss: 0.6929 - accuracy: 0.5119 - val_loss: 0.6943 - val_accuracy: 0.4867
Epoch 7/10
135/135 [==============================] - 1s 6ms/step - loss: 0.6657 - accuracy: 0.5933 - val_loss: 0.5966 - val_accuracy: 0.6933
Epoch 8/10
135/135 [==============================] - 1s 8ms/step - loss: 0.4345 - accuracy: 0.8244 - val_loss: 0.2372 - val_accuracy: 0.9200
Epoch 9/10
135/135 [==============================] - 1s 8ms/step - loss: 0.2039 - accuracy: 0.9281 - val_loss: 0.0717 - val_accuracy: 0.9733
Epoch 10/10
135/135 [==============================] - 1s 8ms/step - loss: 0.0726 - accuracy: 0.9756 - val_loss: 0.0326 - val_accuracy: 0.9800
```



修改後的模型訓練結果相較於原始模型，雖然損失較多，但降低了訓練資料過度擬和的影響。

Evaluating the Model

- **roc_curve** → 在不同閾值設定下，分類模型的真陽性率(True Positive Rate, TPR)和假陽性率(False Positive Rate, FPR)之間的關係。
- **AUC** → 考慮了不同閾值下的真陽性率及偽陽性率，並將其合成一個數值，通常取值範圍在0.5~1之間。當AUC愈大，模型的預測能力越強。
- **AUPRC** (精確率-召回率曲線) → 關注模型的精確率及召回率之間的權衡。

使用測試集資料評估模型的預測能力

原始

AUC 0.9971483042022461
AUPRC 0.9969029074117326

修改後

AUC 0.9903474903474904
AUPRC 0.9796747967479674

雖然修改後模型的評估指標分數低於原始模型，但也顯示了修改後模型的預測能力不會差到很多。