# 以 NSL 進行臉部表情辨識

學生：109034518 蕭詠仁 指導教授：邱銘傳教授
國立清華大學 工業工程與工程管理學系

## 一、 摘要:

現今人臉辨識已經被普遍的使用，也大量的應用在日常生活中，例如：疲勞駕駛的辨識、防盜鎖的應用、證件比對及確認身分等等，目前人類情緒的辨識應用也越來越多，分類方法也不盡相同，本專題以 Neural Structure Learning 為架構利用對抗性學習(Adversarial Learning)進行臉部表情分類，透過臉部表情辨識幫助其他應用來分辨人類情緒。

關鍵字：NSL、臉部表情分類、對抗性學習

## 二、 緒論：

1. 問題定義 5W1H

   What: 利用人臉部表情進行分辨是否為快樂、悲傷、中性等 7 種表情。
   Where: 人臉表情辨識可以運用在醫學研究、影像遊戲體驗及影片體驗評分等等。
   When: 需要判別人類情緒時。
   Who: 醫學研究人員、遊戲設計人員與影片創作者等等。
   Why: 透過臉部表情來識別使用者的體驗感受，針對辨識結果來進行所設計的環境或實驗是否要進行改善或調整。
   How: 利用 NSL 進行對抗性學習。

2. 研究動機與目的

   藉由上述問題定義中 5W1H 可以發現在未來人類情緒辨識，除了上述應用外可能會越來越多，在過去人類情緒分類大多數研究利用 Neural Network、SVM、AdaBoost 等等[6]，這些方法為現在機器學習的主要架構，因此本次專題主要動機為透過較新的神經網路架構進行對抗式學習來訓練人類表情分類，並期望達到較佳的分類結果。

# 三、 研究方法：

1. NSL

神經結構學習(Neural Structure Learning)是一種新的深度學習架構，透過特徵輸入內容(給予類神經網路的特徵)和結構化信號(圖片或文字之間特徵關連性)來訓練類神經網路，這個結構可以是以圖形表示的顯示結構，也可以是由對抗式擾動產生的隱式結構。

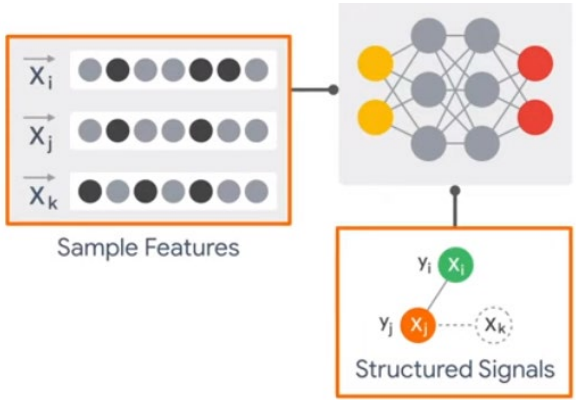


圖 3.1 NSL 結構透過結合結構化信號與特徵訓練類神經網路[7]

結構化信號通常用於表示樣本之間的關係或相似性(樣本可能已加上標籤或未加上標籤)，因此在訓練類神經網路時可運用這些信號，較能夠掌握已加上或未加上標籤的資料，進而提高模型準確率，NSL 也能解決當已加上標籤的資料較少時的情況。如果當訓練模型的樣本是透過增加對抗式擾動所產生，模型將能有效對抗惡意攻擊(惡意攻擊的目標是讓模型產生不正確的預測或分類)。

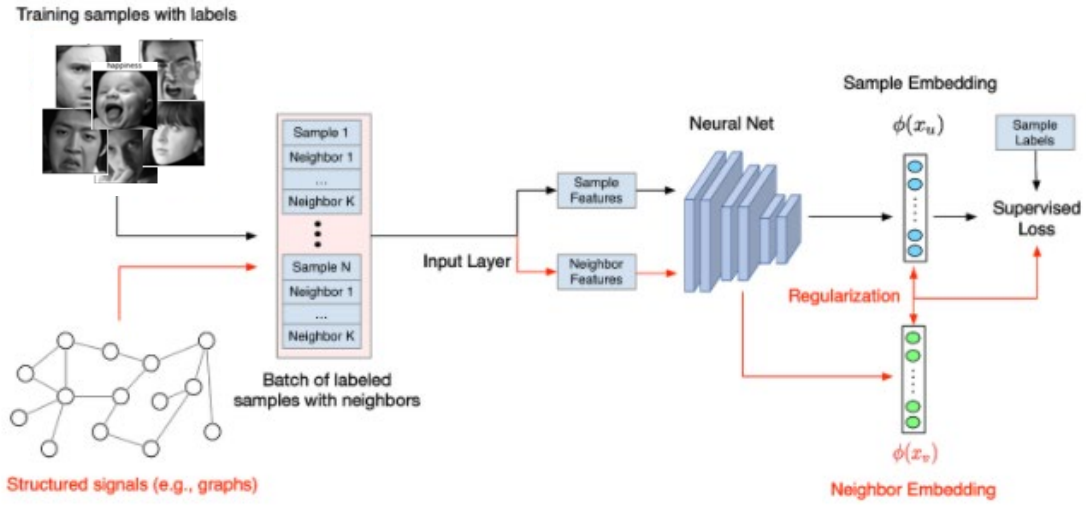


圖 3.2 NSL 的架構圖[8]

在 NSL 的結構化信號可一般化為類神經圖形學習(Neural Graph Learning)與對抗式學習(Adversarial Learning)，不管是明確的定義為圖片或是隱式學習作為對抗式樣本，都是利用正規化的方式來訓練神經網路，透過最小化 supervised loss 來迫使模型學習準確的預測，並同時透過最小化 neighbor loss 維持 input 及 structure 之間的相關性(如下圖三)，而 supervised loss 也可以應用在任意的神經網路結構，例如前饋 NN(Feed-forward NNs)、捲積 NN(Convolutional NNs)與遞迴 NN(Recurrent NNs)，本專題將這項技術結合到 CNN 中。

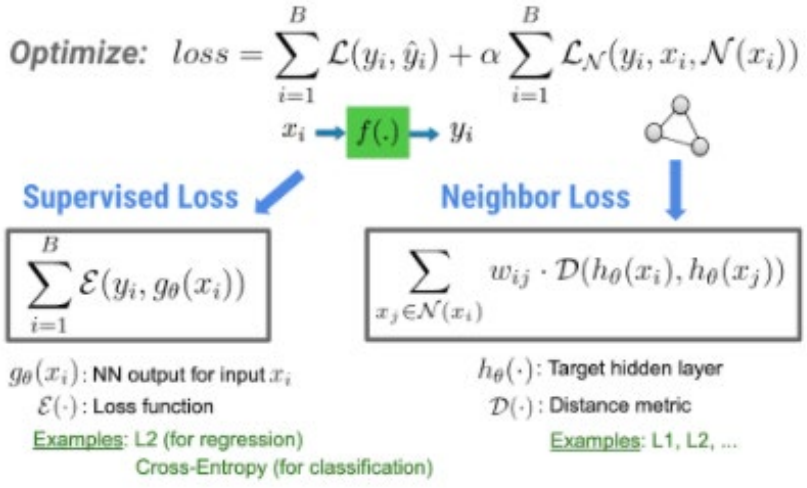


圖 3.3 NSL 最佳化的 loss 公式[8]

2. Adversarial Learning

對抗學習(Adversarial Learning)屬於 NSL 一般化中的隱式學習，訓練模型中除了原本的訓練數據外，還使用一組具有對抗性干擾數據(對抗性樣本)的模型進行訓練。對抗性樣本主要目的在故意誤導模型做出錯誤的預測或分類，這些誤導可能是利用人眼無法辨識出來的雜訊，這些雜訊可能會導致訓練模型判斷錯誤，該模型在進行預測時將學會抵抗對抗性擾動。在訓練過程中對抗性樣本也會持續與正常樣本保持關聯性，意旨在告訴訓練模型記住這些錯誤的分類及如何判別正確圖片其中的相似性，透過對抗性學習可以讓學習模型的預測或分類結果更加的穩定。

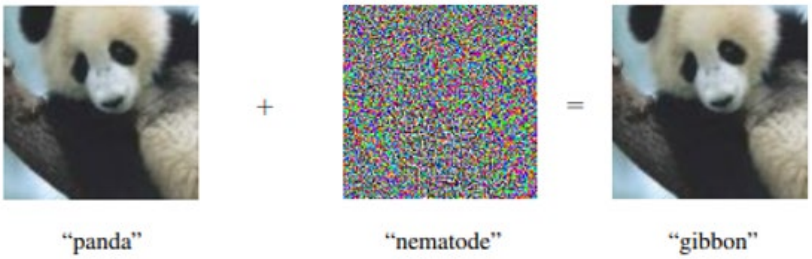


圖 3.4 對抗式學習誤導模型[9]

# 四、 個案研究：

1. 資料介紹

本次專題所使用的 dataset 是現有的資料，其中這個資料集包含 7 種情緒的圖片，分別為 anger(0)、disgust(1)、fear(2)、happiness(3)、sadness(4)、surprise(5)、neutral(6)，資料利用像素的方式存在 csv 檔。7 種情緒共有 35887 張，以下透過程式碼將資料可視化。

表 4.1 情緒張數表

| 情緒 | 張數 |
|---|---|
| Happiness | 8989 |
| Neutral | 6198 |
| Sadness | 6077 |
| Fear | 5121 |
| Anger | 4953 |
| Surprise | 4002 |
| Disgust | 547 |

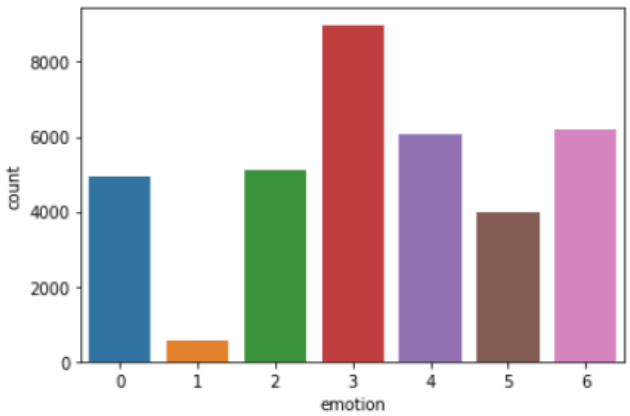


圖 4.1 情緒張數條狀圖

圖 4.2 可視化情緒圖片

2. 資料處理及建模

Step1:

將資料轉換成符合 model 的形式，並重新定義標籤。

```
df = df[df.emotion.isin(INTERESTED_LABELS)]
df.shape
```

```
(35887, 3)
```

```
img_array = df.pixels.apply(lambda x: np.array(x.split(' ')).reshape(48, 48, 1).astype('float32'))
img_array = np.stack(img_array, axis=0)
```

```
le = LabelEncoder()
img_labels = le.fit_transform(df.emotion)
img_labels = np_utils.to_categorical(img_labels)
img_labels.shape
```

```
(35887, 7)
```

```
le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print(le_name_mapping)
```

```
{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6}
```

圖 4.3 轉換圖片大小及重新給予訓練標籤

Step2:

把資料切成訓練及測試，這裡將測試及訓練比例分為 9:1。

```
X_train, X_valid, y_train, y_valid = train_test_split(img_array, img_labels, shuffle=True, stratify=img_labels, test_size=0.1, random_state=42)
X_train.shape, X_valid.shape, y_train.shape, y_valid.shape
```

```
((32298, 48, 48, 1), (3589, 48, 48, 1), (32298, 7), (3589, 7))
```

圖 4.4 設定訓練集與測試集

Step3:

建立 CNN 模型，建立架構為兩層 convolution layer 加上一層 pooling layer 及一層 drop out layer 重複三次，共六層 convolution layer 三層 pooling layer 及三層 drop out layer，之後建立 fully-connected layer，其中包含一層 flatten、兩層 dense 及一層 drop out。模型 model 如下圖 4.5，使用參數如下表 4.2。

接著建立 Adversarial Learning，參數上 multiplier 表示訓練中對抗式學習 loss 的權重相對於標籤的 loss，透過參考範例的試誤測出來較好的範圍大約在 0.1 到 0.4 之間，這邊選擇設定成 0.12，而 step_size 表示給予對抗式學習影響的大小，訓練較佳的範圍大約在 0.01 到 0.1 之間，這裡選擇設定成 0.01 是因為所建立好的模型先前並未接觸過這些圖片，如果第一次就給予很大的影響比例訓練結果可能不會很好，再將建立好的 CNN model 與 Adversarial Learning 結合，形成 Neural Structure Learning，如下圖 4.6。

表 4.2 CNN 各層參數設定

| Layer | Hyperparameter |
|---|---|
| Input layer | 48,48,1 |
| Convolution layer 1(2) | 64 (5,5) |
| Pooling layer 1(Max pooling) | 2,2 |
| Dropout 1 | 0.4 |
| Convolution layer 2(2) | 128 (3,3) |
| Pooling layer 2(Max pooling) | 2,2 |
| Dropout 2 | 0.4 |
| Convolution layer 3(2) | 256 (3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 3 | 0.5 |
| Fully connected layer | |
| Dense layer 1 | 128 |
| Dropout | 0.6 |
| Dense layer 2 | 7 |

```
Model: "DCNN"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
image (Conv2D)               (None, 48, 48, 64)        1664
_________________________________________________________________
batchnorm_1 (BatchNormalizat (None, 48, 48, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 64)        102464
_________________________________________________________________
batchnorm_2 (BatchNormalizat (None, 48, 48, 64)        256
_________________________________________________________________
maxpool2d_1 (MaxPooling2D)   (None, 24, 24, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
_________________________________________________________________
batchnorm_3 (BatchNormalizat (None, 24, 24, 128)       512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 128)       147584
_________________________________________________________________
batchnorm_4 (BatchNormalizat (None, 24, 24, 128)       512
_________________________________________________________________
maxpool2d_2 (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       295168
_________________________________________________________________
batchnorm_5 (BatchNormalizat (None, 12, 12, 256)       1024
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 256)       590080
_________________________________________________________________
batchnorm_6 (BatchNormalizat (None, 12, 12, 256)       1024
_________________________________________________________________
maxpool2d_3 (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1179776
_________________________________________________________________
batchnorm_7 (BatchNormalizat (None, 128)               512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
out_layer (Dense)            (None, 7)                 903
=================================================================
```

圖 4.5 Model summery

```
adv_config = nsl.configs.make_adv_reg_config(multiplier=0.3, adv_step_size=0.12)
adv_model = nsl.keras.AdversarialRegularization(base_model, adv_config=adv_config)
```

圖 4.6 對抗式學習參數設定與整合 Model

Step4.

接著進行設定訓練的 batch_size、epochs 與 model 所使用的 optimizer、loss，設定訓練參數程式如下圖 4.7，而本研究所使用的 Neural Structure Learning 的 loss 是結合 crossentropy 與對抗式學習的 loss 當作整個 Model 的 loss。

```
batch_size = 50
epochs = 30
```

```
adv_model.compile(
        optimizer=optimizers.Adam(0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy']
)
```

圖 4.7 設定訓練參數與函數

## 五、 研究結果：

1. 初步訓練結果

主要結果跑出分類的準確率(Categorical Accuracy)、模型的 loss、對抗式學習的 loss 與分類的 Crossentropy 如下圖 5.1，可以看出一開始的訓練結果沒有很好，訓練的分類準確率只有 0.766 而測試的分類準確率只有 0.6882，從 Model 的 loss 來看分類的 Crossentropy 較高，可能是模型出現 under fitting 的情況。

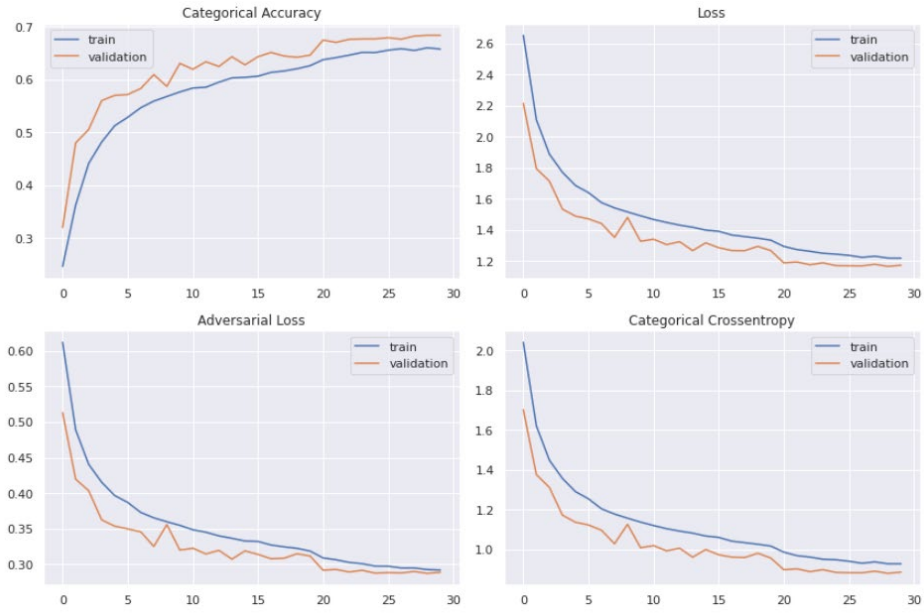


圖 5.1 初步訓練結果

2. 改善結果

先透過基本的增加訓練次數來觀察 Model 的學習是否改善，從原本訓練 30 次增加到 100 次，並觀察結果如下圖 5.2，可以看出增加訓練次數只有訓練集準確些微的提升及 loss 些微降低，且出現有 over fitting 的情況，訓練分類準確率為 0.8035，而測試的訓練準確率為 0.6980，改變幅度不大。

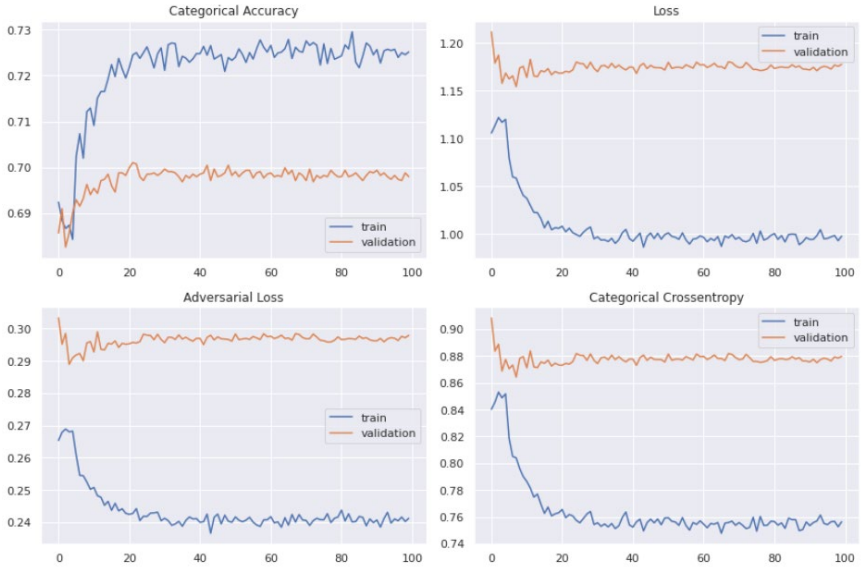


圖 5.2 增加訓練次數結果

由於改變訓練次數效果不大，所以試著增加 CNN 的 Convolution layer、Pooling layer、Dropout layer。共增加五層 Convolution layer、兩層 Pooling layer 及 Dropout layer，改變參數結果如下表 5.1，訓練結果如下圖 5.3，可以看出訓練結果的準確率雖然跟原本差異不大，但是從整體的 loss 及對抗學習的 loss 與分類的 Crossentropy 來看都相較第一次改善及原本的結果來的更低以及更收斂，但是在測試樣本的分類準確率及 loss 與原本差異不大。

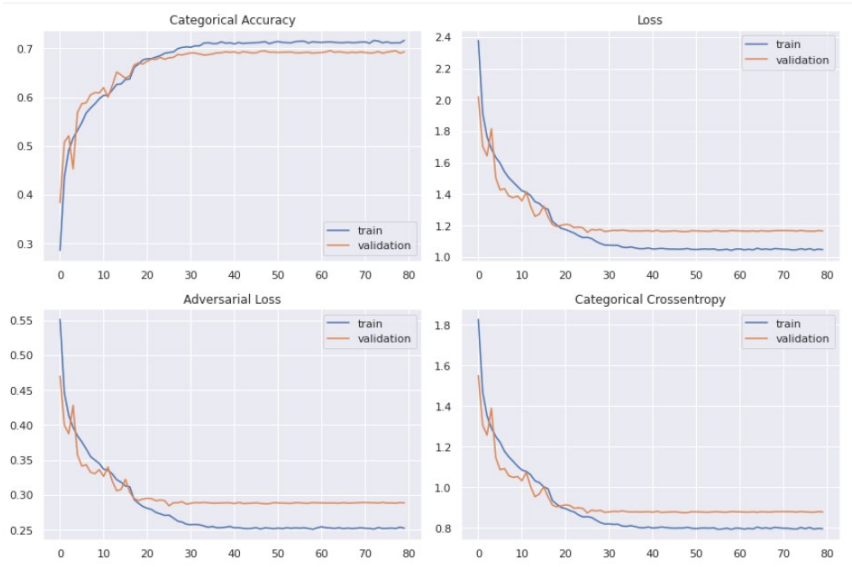


圖 5.3 改變神經網路深度結果

表 5.1 CNN 調整後各層參數設定

| Layer | Hyperparameter |
|---|---|
| Input layer | 48,48,1 |
| Convolution layer 1(2) | 64 (5,5) |
| Pooling layer 1(Max pooling) | 2,2 |
| Dropout 1 | 0.4 |
| Convolution layer 2(2) | 128 (3,3) |
| Pooling layer 2(Max pooling) | 2,2 |
| Dropout 2 | 0.4 |
| Convolution layer 3(2) | 256 (3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 3 | 0.5 |
| Convolution layer 4(2) | 512(3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 4 | 0.5 |
| Convolution layer 5(2) | 512 (3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 5 | 0.5 |
| Fully connected layer | |
| Dense layer 1 | 128 |
| Dropout | 0.6 |
| Dense layer 2 | 7 |

由於訓練結果已經趨近收斂，所以再切一個測試集出來看看 model 的訓練結果如何，其中再將各個卷積層的 filter size 改小為 3*3，每層輸出圖片也相對減少，改善結果如下表 5.2。總共張數為 35887 張，訓練集約為 80% (28709 張)，驗證集約 10% (3589 張)，測試集約 10% (3589 張)，其結果如下圖 5.4、5.5 及 5.6，由結果可以看出 Model 又出現 Overfitting 的情況，但是測試結果準確率達到了 70%。

```
[79]  adv_model.evaluate(
            {IMAGE_INPUT_NAME:  X_test,  LABEL_INPUT_NAME:  Y_test}
      )

      113/113 [==============================] - 5s 40ms/step - loss: 1.27
      [1.2703722715377808,
       0.9443823099136353,
       0.7049317359924316,
       0.3259899616241455]
```

圖 5.4 測試集辨識結果

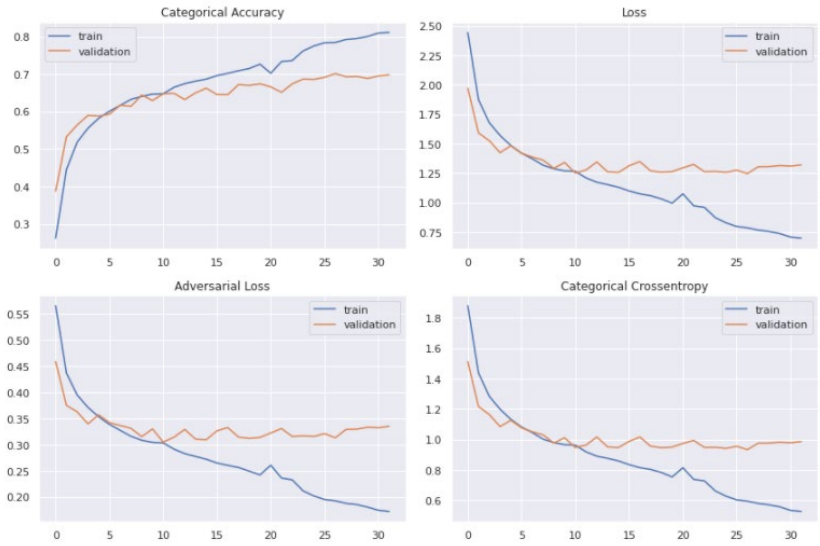


圖 5.5 最後訓練結果

```
total wrong validation predictions: 1059
accuracy: 70.493%
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.62 | 0.68 | 0.64 | 491 |
| 1 | 0.79 | 0.69 | 0.74 | 55 |
| 2 | 0.56 | 0.48 | 0.52 | 528 |
| 3 | 0.91 | 0.89 | 0.90 | 879 |
| 4 | 0.61 | 0.53 | 0.57 | 594 |
| 5 | 0.79 | 0.80 | 0.80 | 416 |
| 6 | 0.63 | 0.76 | 0.69 | 626 |
| accuracy | | | 0.70 | 3589 |
| macro avg | 0.70 | 0.69 | 0.69 | 3589 |
| weighted avg | 0.71 | 0.70 | 0.70 | 3589 |

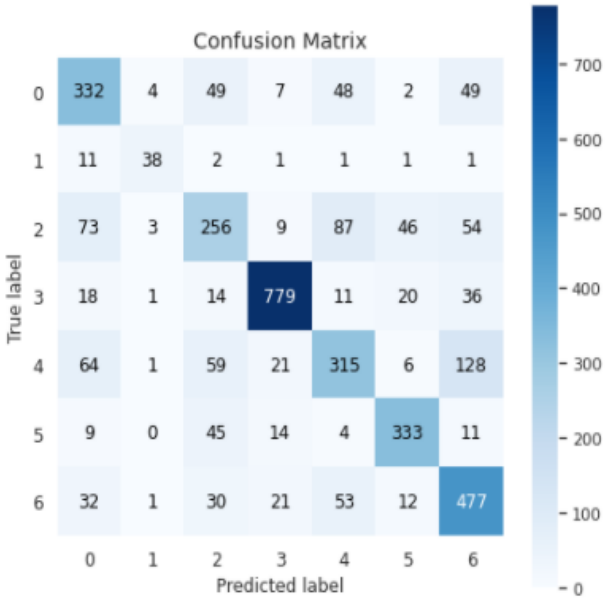


圖 5.6 最後訓練結果(混淆舉證)

表 5.2 CNN 調整後各層參數設定

| Layer | Hyperparameter |
|---|---|
| Input layer | 48,48,1 |
| Convolution layer 1(2) | 32 (5,5) |
| Pooling layer 1(Max pooling) | 2,2 |
| Dropout 1 | 0.5 |
| Convolution layer 2(3) | 64 (3,3) |
| Pooling layer 2(Max pooling) | 2,2 |
| Dropout 2 | 0.5 |
| Convolution layer 3(4) | 64 (3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 3 | 0.5 |
| Convolution layer 4(3) | 128(3,3) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 4 | 0.5 |
| Convolution layer 5(3) | 512 (2,2) |
| Pooling layer 3(Max pooling) | 2,2 |
| Dropout 5 | 0.5 |
| Fully connected layer | |
| Dense layer 1 | 128 |
| Dropout | 0.5 |
| Dense layer 2 | 7 |

因為增加神經網路深度與調整參數的改變效果不大，可能是因為七種情緒中有些情緒的照片量不夠，所以試著訓練分類較少的情緒類別，透過觀察張數較多的情緒來進行分類，設定維持不變，所分類的情緒類別為 happiness(3)、sadness(4)與 neutral(6)，其結果如下圖 5.7。透過減少分類，可以看出模型的學習成果還不錯，訓練及測試準確率分別達到 0.8551 及 0.8295，而 loss 也在 0.5 左右或以下，結果也較沒有 overfitting 的情況。

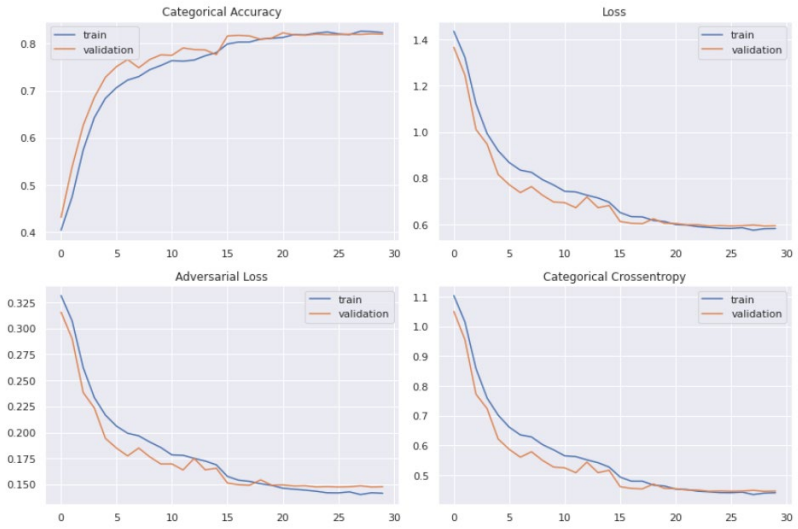


圖 5.7 減少分類訓練結果

3. 與其他研究比較結果

由於所選的資料集的複雜程度較高，所以所訓練結果沒有很高的準確率，透過回顧其他使用相同資料集的研究進行比較，比較結果如下表 5.3，由結果可以看出，所使用的 CNN Model 結合 NSL 的對抗式學習表現其實不差，而且在資料前處理方面，如資料增強上沒有使用特殊方法，只使用基本的旋轉、放大及縮小等，也並未使用額外的資料進行訓練，所以對比其他研究下，表現結果還不錯，但是還有待後續更深入的研究。

表 5.3 與個別不同研究比較

| Algorithm | Test accuracy |
|---|---|
| Zhang .J. [1] | 64.15% |
| Ionescu, R. T., Popescu, M., & Grozea, C. [2] | 67.484% |
| Gao, H., & Ma, B [3] | 65.20% |
| Raksarikorn, T., & Kangkachit, T [4] | 71.69% |
| Sheng, M. et al [5] | 73.28% |
| Proposed method | 70.187% |

# 六、 結論

1. 研究限制

因為模型架構及參數設定大多數參考網路上以訓練好的 CNN 模型，以 vgg16 當作模板來進行架構改變，但是神經網路的深度並沒有像 vgg16 一樣有 13 層 Convolution 及其他細節的參數設定，所以訓練結果可能沒辦法達到 9 成的辨識率。在資料方面，有些表情辨識可能連人都容易混淆，如下圖 6.1 可以看出一些中性和悲傷情緒的圖片，其實差異不大，所以如果要達到較好的訓練結果，可能要增加資料量。


圖 6.1 分類錯誤圖片

2. 未來方向

由於本專題只能在分類較少的情況下訓練出較好的辨識模型，所以在之後希望透過其他的方式，如預先訓練、修改參數或是增加資料量的方式來改進模型，使模型能夠完整的分辨人的 7 種情緒，增加模型的實用性。

# 七、 參考資料

[1] Zhang, J. (2020, October). Movies and Pop Songs Recommendation System by Emotion Detection through Facial Recognition. In *Journal of Physics: Conference Series* (Vol. 1650, No. 3, p. 032076). IOP Publishing

[2] Ionescu, R. T., Popescu, M., & Grozea, C. (2013, June). Local learning to improve bag of visual words model for facial expression recognition. In Workshop on challenges in representation learning, ICML.

[3] Gao, H., & Ma, B. A Robust Improved Network for Facial Expression Recognition.

[4] Raksarikorn, T., & Kangkachit, T. (2018, July). Facial Expression Classification using Deep Extreme Inception Networks. In 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 1-5). IEEE.

[5] Sheng, M., Zhang, L., Yan, L., Wang, C., Li, M., Xia, H., & Zhang, Y. (2020, August). Facial Expression Recognition Based on Sparse Autoencoder and Shallow Convolutional Neural Network. In 2020 15th International Conference on Computer Science & Education (ICCSE) (pp. 669-674). IEEE.
[6] http://nccur.lib.nccu.edu.tw/bitstream/140.119/37112/9/100309.pdf
[7] https://www.tensorflow.org/neural_structured_learning
[8] https://blog.tensorflow.org/2019/09/introducing-neural-structured-learning.html
[9] https://arxiv.org/pdf/1412.6572.pdf
[10] https://reurl.cc/3L3EpM
[11] https://arxiv.org/pdf/1412.6572.pdf
[12] https://chtseng.wordpress.com/2017/11/11/data-augmentation-%E8%B3%87%E6%96%99%E5%A2%9E%E5%BC%B7/
[13] https://keras.io/api/preprocessing/image/
[14] https://keras-cn.readthedocs.io/en/latest/preprocessing/image/