Machine Learning Project 3

Image Sentiment Classification

Team Member

- b03903089 資工三 林良翰

Questions

- The naming rule of `model_name` is the validation accuracy of last training epoch.
- Model of 1. 3. 4. 5. 6. is `0.643750_0.610833.h5`
- Model of 2. is `dnn_0.384187.h5`

1. 請說明你實作的 CNN model. 其模型架構, 訓練過程和準確率為何?

- Usage

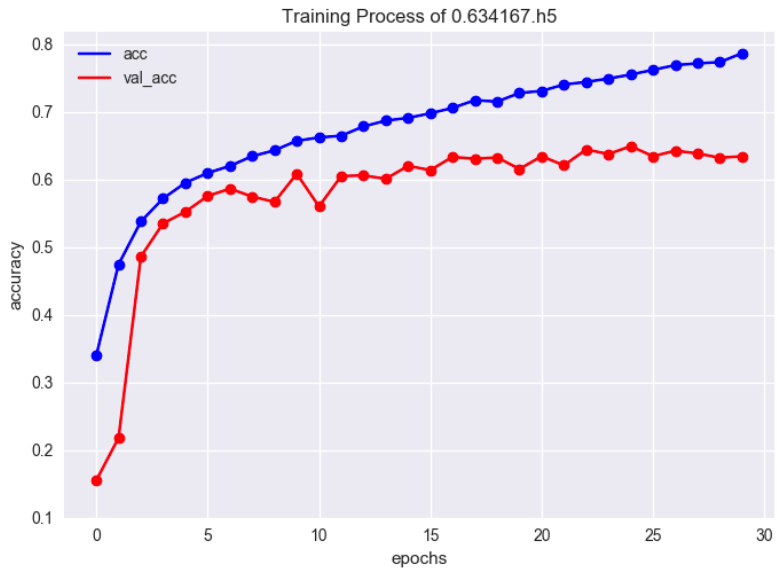
  ```
  $ python train.py train.csv
  $ python test.py test.csv result.csv [model_name].h5
  ```

- CNN Model Structure

Layer	Input	Output	Parameters
Input	2,304	(48, 48, 1)	0
Convolution 2D (32, (3, 3))	(48, 48, 1)	(48, 48, 32)	320
Batch Normalization	(48, 48, 32)	(48, 48, 32)	128
Convolution 2D (32, (3, 3))	(48, 48, 32)	(48, 48, 32)	9,248
Batch Normalization	(48, 48, 32)	(48, 48, 32)	128
Convolution 2D (32, (3, 3))	(48, 48, 32)	(48, 48, 32)	9,248
Batch Normalization	(48, 48, 32)	(48, 48, 32)	128
Max Pooling (2, 2)	(48, 48, 32)	(24, 24, 32)	0
Convolution 2D (64, (3, 3))	(24, 24, 32)	(24, 24, 64)	18,496
Batch Normalization	(24, 24, 64)	(24, 24, 64)	256
Convolution 2D (64, (3, 3))	(24, 24, 64)	(24, 24, 64)	36,928
Batch Normalization	(24, 24, 64)	(24, 24, 64)	256
Convolution 2D (64, (3, 3))	(24, 24, 64)	(24, 24, 64)	36,928
Batch Normalization	(24, 24, 64)	(24, 24, 64)	256
Max Pooling (2, 2)	(24, 24, 64)	(12, 12, 64)	0

Convolution 2D (128, (3, 3))	(12, 12, 64)	(12, 12, 128)	73,856
Batch Normalization	(12, 12, 128)	(12, 12, 128)	512
Convolution 2D (128, (3, 3))	(12, 12, 128)	(12, 12, 128)	147,584
Batch Normalization	(12, 12, 128)	(12, 12, 128)	512
Convolution 2D (128, (3, 3))	(12, 12, 128)	(12, 12, 128)	147,584
Batch Normalization	(12, 12, 128)	(12, 12, 128)	512
Max Pooling (2, 2)	(12, 12, 128)	(6, 6, 128)	0
Flatten	(6, 6, 128)	4,608	0
Dense (256)	4,608	256	(4,608+1)×256=1,179,904
Dense (128)	256	128	(256+1)×128=32,896
Dense (64)	128	64	(128+1)×64=8,256
Dense (7)	64	7	(64+1)×7=455
Output	7	7	0

- Trainable Parameters: 1,703,047
- Training Process
  - Epochs: 20
  - Batch Size: 128
  - Augment on Data: Randomly horizontal / vertical shift within ratio 0.1, randomly horizontal flip
  - Validation Data: First 2400 training data
  - Loss Function: Categorical Cross Entropy
  - Optimizer: Adam



- Accuracy
  - Validation Accuracy: 64.38%
  - Kaggle Public Accuracy: 65.06%

2. 承上題, 請用與上述 CNN 接近的參數量, 實做簡單的 DNN model. 其模型架構, 訓練過程和準確率為何? 試與上題結果做比較, 並說明你觀察到了什麼?

- Usage

  ```
  $ python dnn_train.py train.csv
  $ python dnn_test.py test.csv result.csv [model_name].h5
  ```

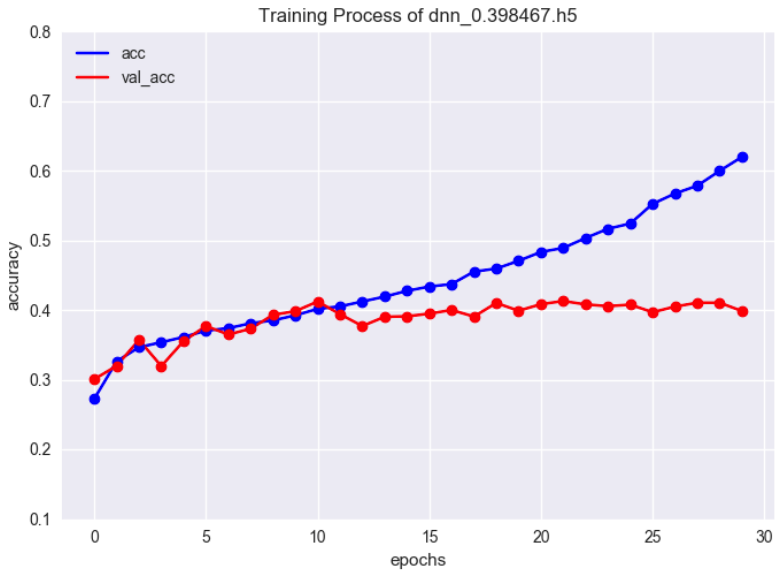
- DNN Model Structure

Layer	Input	Output	Parameters
Input	2,304	2,304	0
Dense (512)	2,304	512	(2,304+1)×512=1,180,160
Dense (512)	512	512	(512+1)×512=262,655
Dense (512)	512	512	(512+1)×512=262,656
Dense (256)	512	256	(512+1)×256=131,328
Dense (7)	256	7	(256+1)×7=1,799
Output	7	7	0

  - Trainable Paramters: 1,838,599

- Training Process

  - Epochs: 20
  - Batch Size: 128
  - Augment on Data: Randomly horizontal / vertical shift within ratio 0.1, randomly horizontal flip
  - Validation Data: 10% training data
  - Loss Function: Categorical Cross Entropy
  - Optimizer: Adam



- Accuracy

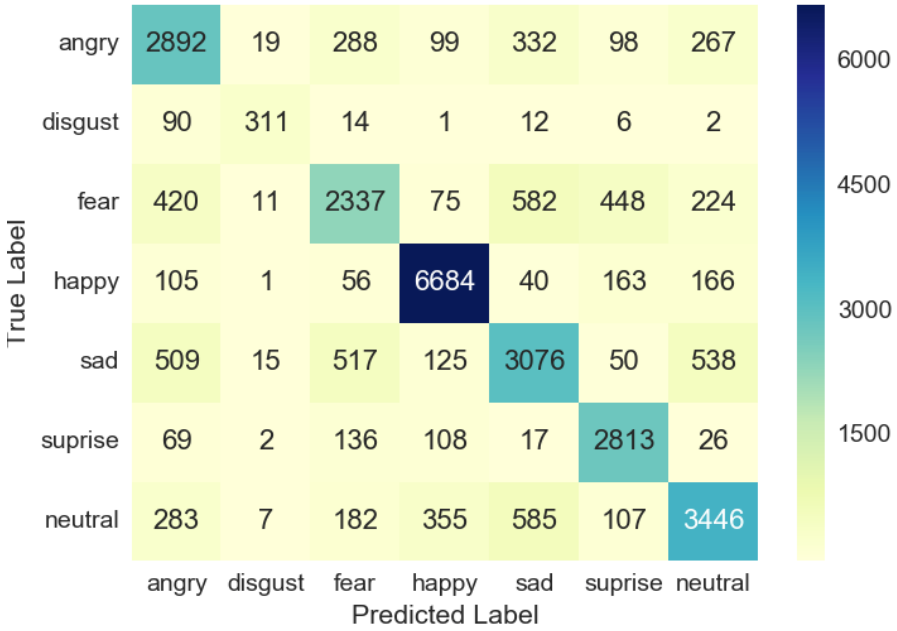
  - Validation Accuracy: 38.42%
  - Kaggle Public Accuracy: 38.01%

3. 觀察答錯的圖片中, 哪些 class 彼此間容易用混?

- Usage

  ```
  $ python plot_model.py trainc.csv [model_name].h5
  ```

- Confusion Matrix



- Accuracy

Expression	Recognition (Row)	Prediction (Column)
Angry生氣	72.390%	66.209%
Disgust 嘔心	71.330%	84.973%
Fear 害怕	57.042%	66.204%
Happy 開心	92.640%	89.754%
Sad 難過	63.685%	66.236%
Suprise 驚訝	88.710%	76.336%
Neutral 中立	69.406%	73.806%

- Conclusion
  - Happy 的準確率最高, Fear 的準確率最低
  - 容易混淆的 Class: (# Error ≥ 400)
    - Fear to Angry: 420
    - Sad to Angry: 509
    - Sad to Fear: 517
    - Fear to Sad: 582
    - Neutral to Sad: 585
    - Fear to Suprise: 448
    - Sad to Neutral: 538
  - Sad & Fear, Sad & Neutral 是最容易混淆的表情

4. 從1. 2.可以發現, 使用 CNN 的確有些好處, 試繪出其 saliency maps 觀察模型在做 classification 時, 是 focus 在圖片的哪些部份?

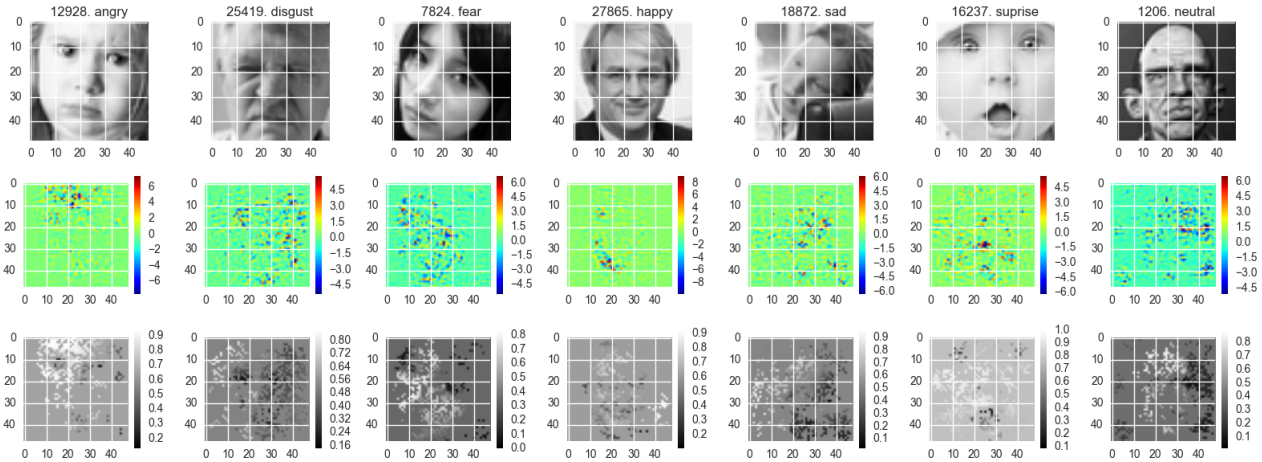
- Usage

  ```
  $ python plot_model.py train.csv [model_name].h5
  ```

- Method

  分別對圖片上每一個 pixel 的灰階值+1, 觀察最後預測出來的 Loss 和原本的差異並取平方

- Figure Selected: #12928, #25419, #7824, #27865, #18872, #16237, #1206



- Conclusion

  在五官的部分最容易影響 CNN 模型的判斷

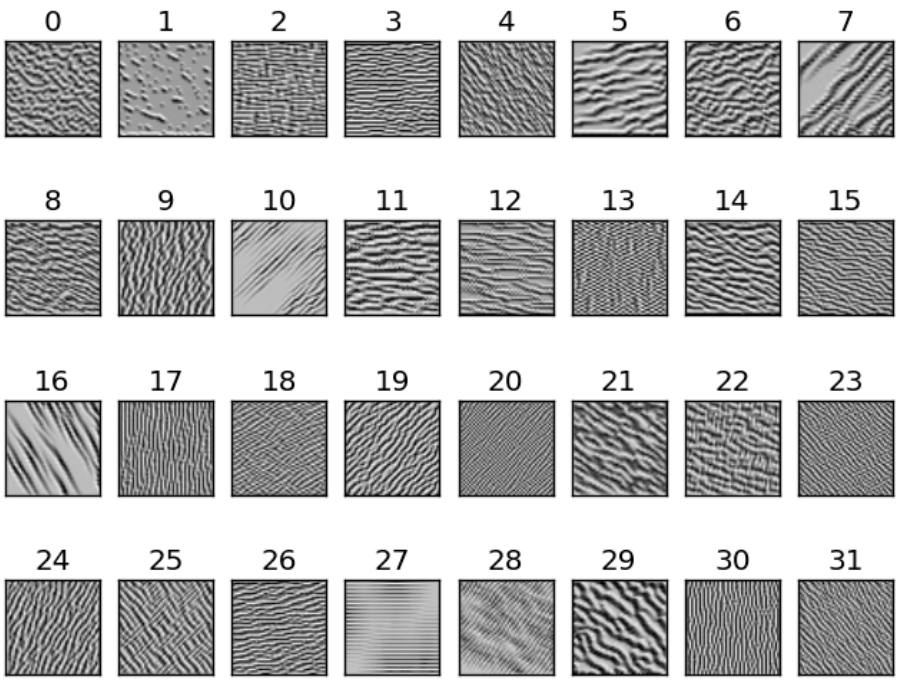
5. 承1. 2., 利用上課所提到的 gradient ascent 方法, 觀察特定層的filter最容易被哪種圖片 activate.

- Usage

  ```
  $ python gradient_ascent.py [model_name].h5
  ```

- Selected Layer: First layer, 3×3 convolution with 32 filters

- Result:



6. 從 training data 中移除部份 label, 實做 semi-supervised learning.

- Usage
  ```
  $ python semi_train.py train.csv test.csv [model_name].h5
  ```
- Training Process
  - Epochs: 20
  - Batch Size: 128
  - Augment on Data: Randomly horizontal / vertical shift within ratio 0.1, randomly horizontal flip
  - Original-Labeled Data Ratio: 0.5
  - Self-Labeled Training Data: Data with prediction expected value ≥ 0.5
  - Validation Data: First 2400 training data
  - Loss Function: Categorical Cross Entropy
  - Optimizer: Adam
- Accuracy
  - Validation Accuracy: 64.25%
  - Kaggle Public Accuracy: 66.54%

Program Usage

1. Convert data format

- Convert .csv to .npz
  ```
  $ python csv_to_npz.py train.csv test.csv
  ```
  Output
  ```
  data.npz
  ```

2. CNN

- Train a new CNN model
  ```
  $ python train.py train.csv
  ```
  Output
  ```
  [val_acc].h5
  ```
- Train an existed model with more epochs
  ```
  $ python retrain.py train.csv [existed_model_name].h5 [previous_epoch] [last_epoch]
  ```
  Output
  ```
  [val_acc]_[previous_model_name].h5
  ```
- Train a new CNN model with semi-supervised learning
  ```
  $ python semi_train.py train.csv test.csv [predictor_model_name].h5
  ```
  Output
  ```
  semi_[val_acc].h5
  ```

- Train an existed CNN model with semi-supervised learning
  `$ python semi_retrain.py train.csv test.csv [predictor_model_name].h5 [existed_model_name].h5 [previous epoch] [last_epoch]`
  Output
  `[val_acc]_[previous_model_name].h5`
- Test CNN model
  `$ python test.py test.csv result.csv [model_name].h5`

3. DNN

- Train a new DNN model
  `$ python dnn_train.py train.csv`
  Output
  `dnn_[val_acc].h5`
- Test DNN model
  `$ python dnn_test.py test.csv result.csv [model_name].h5`

4. Plot Model

- Plot model structure, confusion matrix, saliency map, training process
  `$ python plot_model.py train.csv [model_name].h5`
  Output
  `[model_name]_struct.png` , `[model_name]_cm.png` , `[model_name]_his.png` , `[model_name]_sm.png`
- Plot gradient ascent
  `$ python gradient_ascent.py [model_name].h5`
  Output
  `[model_name]_[layer_name].png`

Best Model

- Blending with 25 model: (weight & model name) `model_list.txt`

```
1 ./0.640000/25_0.658333.h5
2 ./0.640000/26_0.600417.h5
3 ./0.640000/27_0.632083.h5
4 ./0.640000/28_0.635833.h5
5 ./0.640000/29_0.640000.h5
1 ./0.638750/25_0.642500.h5
2 ./0.638750/26_0.659583.h5
3 ./0.638750/27_0.651667.h5
4 ./0.638750/28_0.638333.h5
5 ./0.638750/29_0.638750.h5
3 ./semi_0.645000/29_0.645000.h5
1 0.624583.h5
```

```
3 0.654167_semi_0.642500_0.643750_0.610833.h5
2 0.643750_0.610833.h5
3 semi_0.642500_0.643750_0.610833.h5
1 0.617083_0.603333.h5
3 semi_0.631667_0.617083_0.603333.h5
2 semi_0.626250_0.615417.h5
2 0.639583_0.594167.h5
2 0.803786.h5
2 semi_0.756897_0.803786.h5
2 0.785732.h5
2 semi_0.756975_0.785732.h5
1 0.612500.h5
2 0.632083.h5
3 semi_0.638750_0.632083.h5
```

- Usage

  ```
  $ python test_by_vote.py test.csv result.csv model_list.txt
  ```

- Accuracy
  - Kaggle Public Test: 71.245%