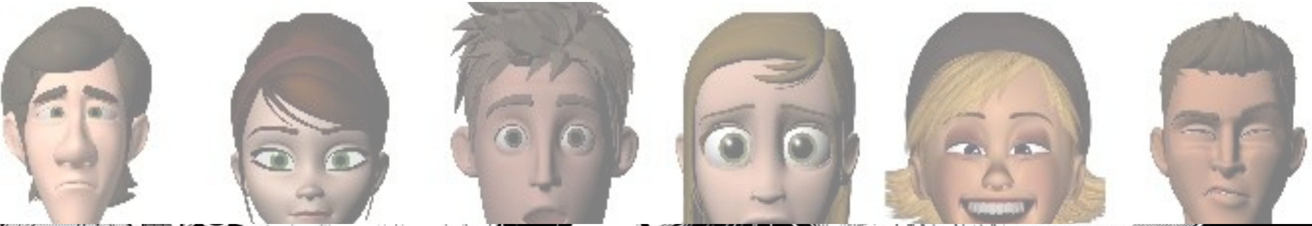
Emotion Classification of Animated Characters

Transfer Learning Performance Comparison

Katelyn Wang
Tracy Zhang



Executive Summary

Goal

Emotion Classification of Cartoon Characters of different style (anime vs. 3D cartoon)

Solution

- Transfer learning and fine-tuning
- Compare the accuracy of different pretrained models & baseline CNNs

Value

- Few studies explored this subject before => improve image search result quality
- Understand how neural networks differentiate emotions of fictional figures, whose characteristics vary dramatically between artists

Technical Challenges

- Animated faces have different characteristics from real human faces
- Limited availability of existing labeled data

Our Approach

1. Data augmentation
2. Benchmark: Vanilla CNN
3. Fine-tuning pretrained models
    a. Early stopping
    b. Batch Normalization
    c. Dropout layers

Implementation Details: Models and Data sets

Models

GoogleNet
ResNet50, pretrained on ImageNet
Vanilla CNNs

Data Sets

Facial Expression Research Group 2D Database (FERG-DB)

55767 annotated face images of 6 characters

Manga Facial Expressions Data Set (462 images)

- Pleased (38)
- Angry (54)
- Crying (56)
- Sad (57)
- Embarrassed (67)
- Happy (87)
- Shock/Surprised (103)

Shortage of the Dataset

Manga Facial Expressions Data Set (462 images)

Vague labeling

The labeling of the dataset is subjective

- Pleased (38)
- Angry (54)
- Crying (56)
- Sad (57)
- Embarrassed (67)
- Happy (87)
- Shock/Surprised (103)

Facial Expression Research Group 2D Database (FERG-DB)

Limited Categories ⟶

```
{'anger': 0,
 'disgust': 1,
 'fear': 2,
 'joy': 3,
 'neutral': 4,
 'sadness': 5,
 'surprise': 6}
```

Limited Characters ⟶ could be “overfitting”

Similar in training and validation

Experimental Evaluation - Analysis

Subjective labeling in training/validation data

P:pleased C:happy

P:happy C:pleased

P:shock C:sad

P:shock C:happy

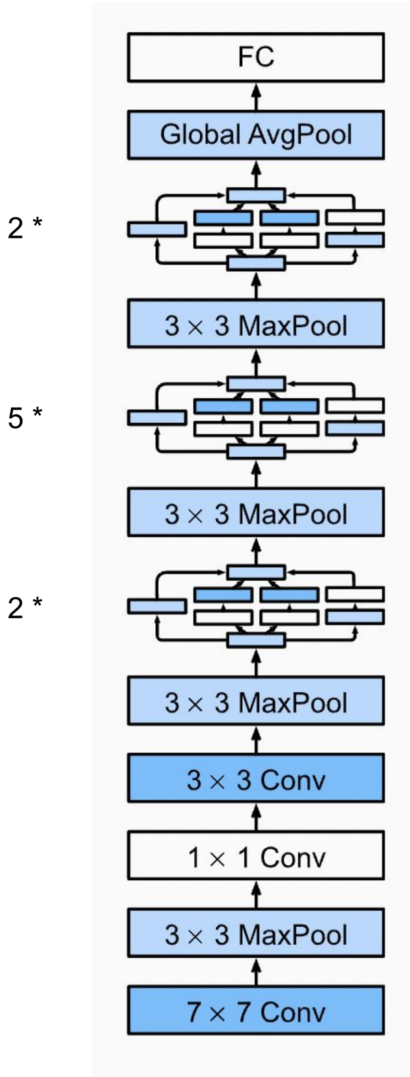
P:shock C:pleased

P:pleased C:shock

Experimental Evaluation - Result

model	dataset	model details	avg val accuracy	max val accuracy	pred accuracy
CNN from scratch	manga	baseline	0.3214	0.3835	
CNN from scratch	manga	baseline + leakyRELU	0.3155	0.411	
CNN from scratch	manga	baseline + leakyRELU + Batch Norm	0.2269	0.2877	
CNN from scratch	ferg	baseline	0.6815	0.7184	
ResNet50	manga	1 Dense layer + sigmoid	0.343	0.4167	0.31
ResNet50	manga	2 Dense 64 layer + leakyRELU + softmax	0.364	0.3796	0.47
ResNet50	ferg	baseline	0.6241	0.5406	
GoogleNet	manga	baseline	0.33	0.47	
GoogleNet	manga	baseline + Batch Norm + 2 Dense 64 Layer	0.35	0.47	
GoogleNet	manga	baseline+early stop	0.27	0.37	
GoogleNet	manga	L2 Regularization	0.3	0.56	
GoogleNet	manga	one more Inception on block 3 and 5	0.27	0.4	
GoogleNet	ferg	baseline	0.59	0.9	
GoogleNet	ferg	baseline+ Batch Norm + 2 Dense 64 Layer		0.459	

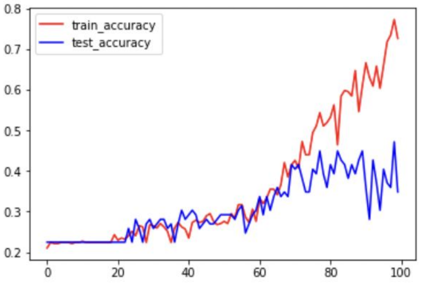
GoogleNet

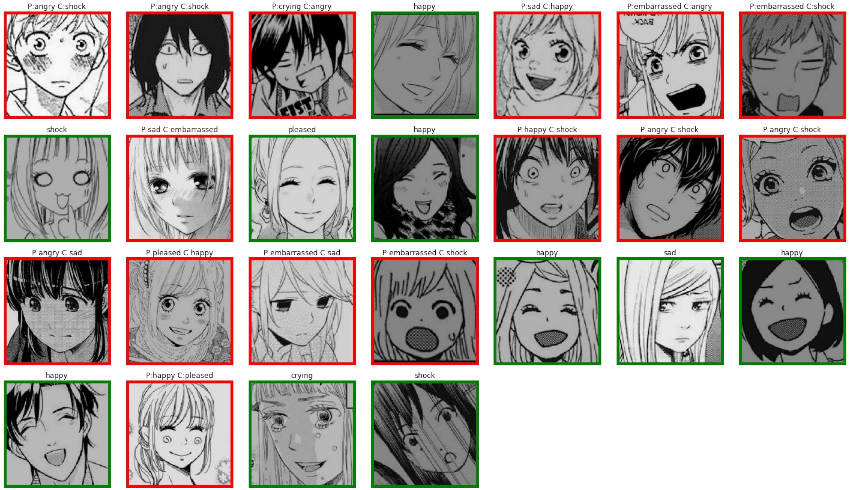



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_57 (Conv2D)          (None, 112, 112, 64)      9472

 max_pooling2d_13 (MaxPoolin  (None, 56, 56, 64)       0
 g2D)

 conv2d_58 (Conv2D)          (None, 56, 56, 64)        36928

 conv2d_59 (Conv2D)          (None, 56, 56, 192)       110784

 max_pooling2d_14 (MaxPoolin  (None, 28, 28, 192)      0
 g2D)

 inception_9 (Inception)     (None, 28, 28, 256)       163696

 inception_10 (Inception)    (None, 28, 28, 480)       388736

 max_pooling2d_17 (MaxPoolin  (None, 14, 14, 480)      0
 g2D)

 inception_11 (Inception)    (None, 14, 14, 512)       376176

 inception_12 (Inception)    (None, 14, 14, 512)       449160

 inception_13 (Inception)    (None, 14, 14, 512)       510104

 inception_14 (Inception)    (None, 14, 14, 528)       605376

 inception_15 (Inception)    (None, 14, 14, 832)       868352

 max_pooling2d_23 (MaxPoolin  (None, 7, 7, 832)        0
 g2D)

 inception_16 (Inception)    (None, 7, 7, 832)         1043456

 inception_17 (Inception)    (None, 7, 7, 1024)        1444080

 global_average_pooling2d_1   (None, 1024)             0
 (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 1024)              0

 flatten_1 (Flatten)         (None, 1024)              0

 dense_3 (Dense)             (None, 64)                65600

 dense_4 (Dense)             (None, 7)                 455

=================================================================
Total params: 6,072,375
Trainable params: 6,072,375
Non-trainable params: 0
_________________________________________________________________
```

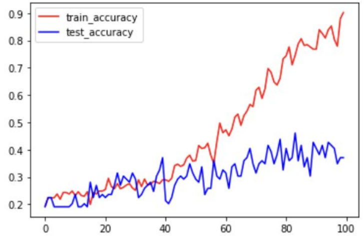
Comparison(Manga)

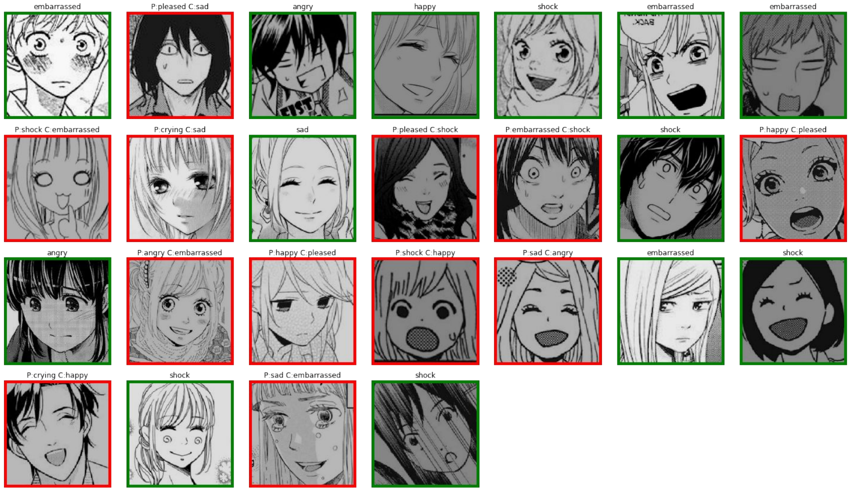
Baseline



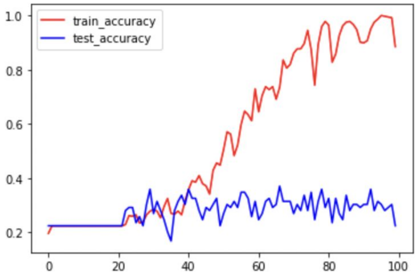


baseline + Batch Norm + 2 Dense 64 Layer

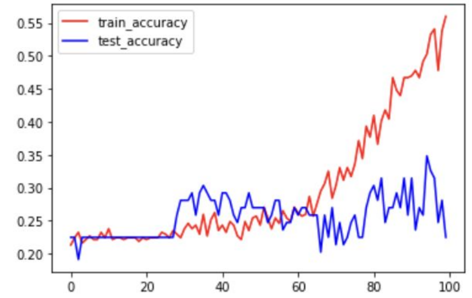


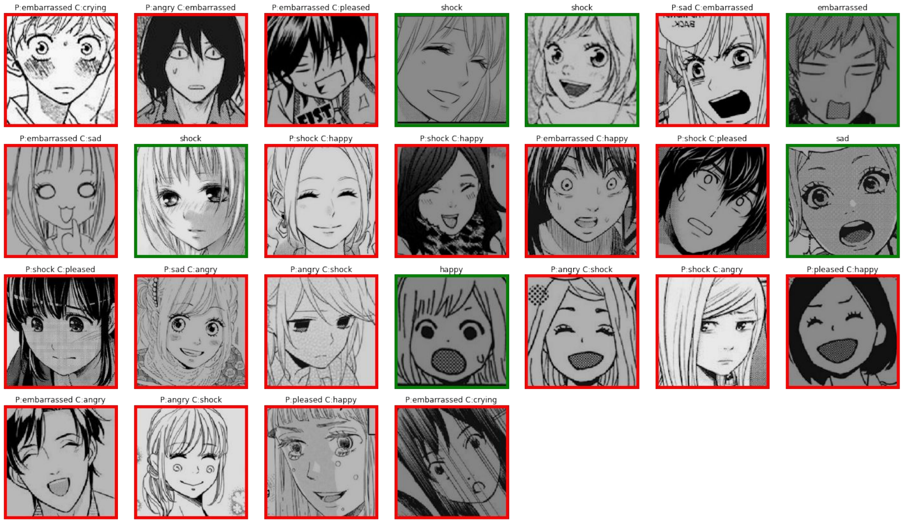


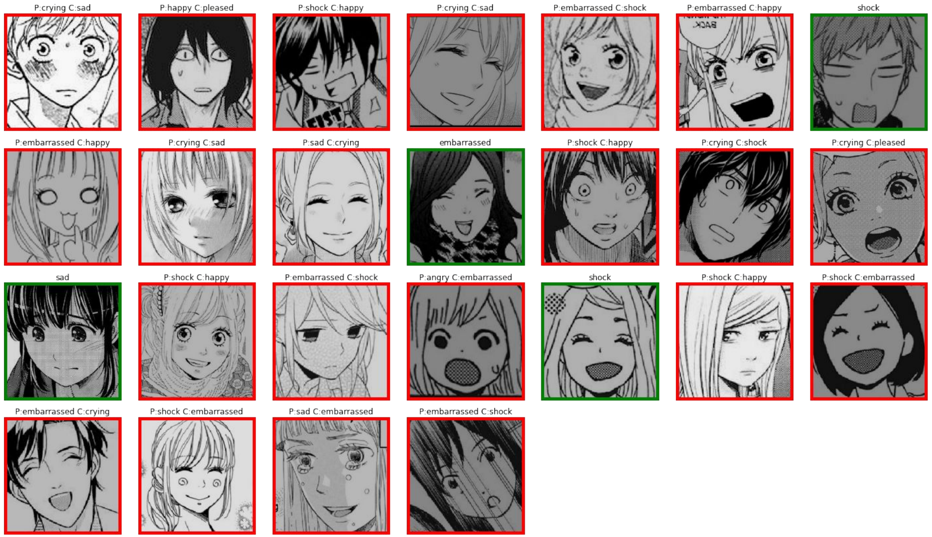
baseline+early stop



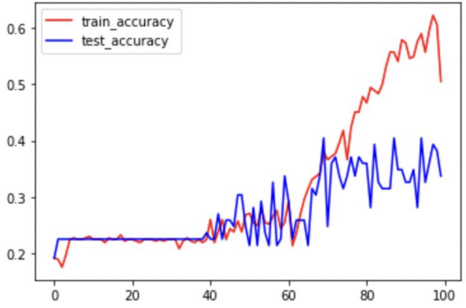
L2 Regularization

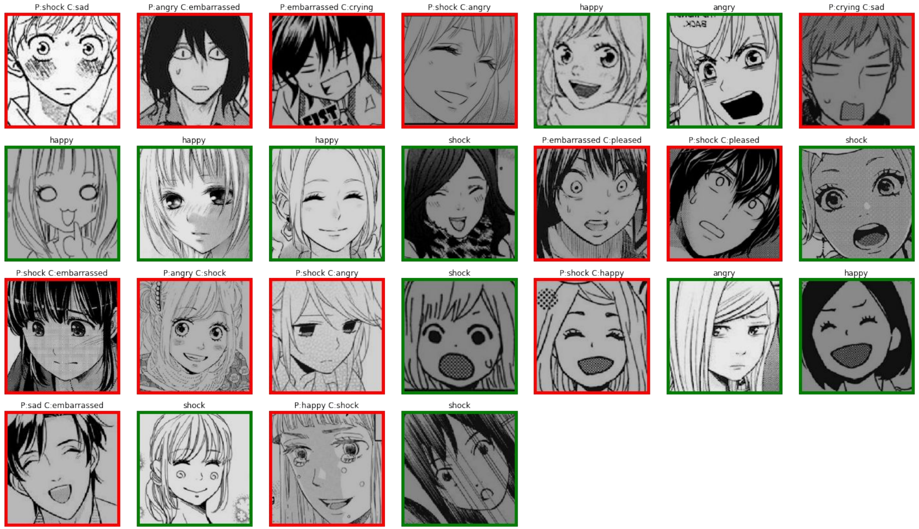






one more Inception on block 3 and 5

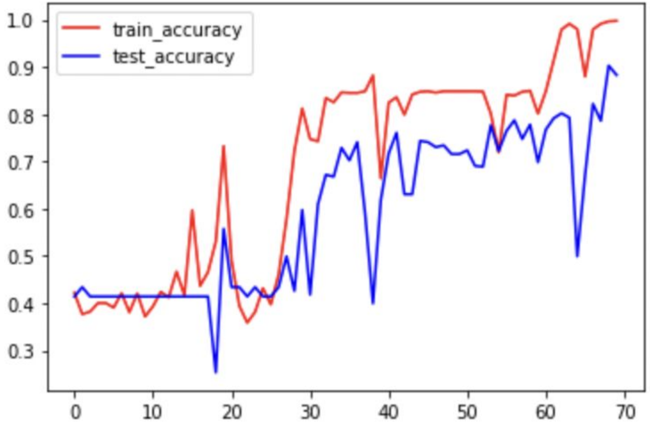




conv2d_448 (Conv2D)	(None, 112, 112, 64)	9472
max_pooling2d_104 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_449 (Conv2D)	(None, 56, 56, 64)	36928
conv2d_450 (Conv2D)	(None, 56, 56, 192)	110784
max_pooling2d_105 (MaxPooling2D)	(None, 28, 28, 192)	0
inception_67 (Inception)	(None, 28, 28, 256)	163696
inception_68 (Inception)	(None, 28, 28, 256)	177008
inception_69 (Inception)	(None, 28, 28, 480)	388736
max_pooling2d_109 (MaxPooling2D)	(None, 14, 14, 480)	0
inception_70 (Inception)	(None, 14, 14, 512)	376176
inception_71 (Inception)	(None, 14, 14, 512)	449160
inception_72 (Inception)	(None, 14, 14, 512)	510104
inception_73 (Inception)	(None, 14, 14, 528)	605376
inception_74 (Inception)	(None, 14, 14, 832)	868352
max_pooling2d_115 (MaxPooling2D)	(None, 7, 7, 832)	0
inception_75 (Inception)	(None, 7, 7, 832)	1043456
inception_76 (Inception)	(None, 7, 7, 832)	1043456
inception_77 (Inception)	(None, 7, 7, 1024)	1444080
global_average_pooling2d_7 (GlobalAveragePooling2D)	(None, 1024)	0
dropout_5 (Dropout)	(None, 1024)	0
flatten_7 (Flatten)	(None, 1024)	0
dense_21 (Dense)	(None, 64)	65600
dense_22 (Dense)	(None, 64)	4160
dense_23 (Dense)	(None, 7)	455

```
=================================================================
Total params: 7,296,999
Trainable params: 7,296,999
Non-trainable params: 0
```

FERG



- [it took me 5 hours to train GoogleNet with]

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_57 (Conv2D)          (None, 112, 112, 64)      9472

 max_pooling2d_13 (MaxPoolin  (None, 56, 56, 64)       0
 g2D)

 conv2d_58 (Conv2D)          (None, 56, 56, 64)        36928

 conv2d_59 (Conv2D)          (None, 56, 56, 192)       110784

 max_pooling2d_14 (MaxPoolin  (None, 28, 28, 192)      0
 g2D)

 inception_9 (Inception)     (None, 28, 28, 256)       163696

 inception_10 (Inception)    (None, 28, 28, 480)       388736

 max_pooling2d_17 (MaxPoolin  (None, 14, 14, 480)      0
 g2D)

 inception_11 (Inception)    (None, 14, 14, 512)       376176

 inception_12 (Inception)    (None, 14, 14, 512)       449160

 inception_13 (Inception)    (None, 14, 14, 512)       510104

 inception_14 (Inception)    (None, 14, 14, 528)       605376

 inception_15 (Inception)    (None, 14, 14, 832)       868352

 max_pooling2d_23 (MaxPoolin  (None, 7, 7, 832)        0
 g2D)

 inception_16 (Inception)    (None, 7, 7, 832)         1043456

 inception_17 (Inception)    (None, 7, 7, 1024)        1444080

 global_average_pooling2d_1   (None, 1024)             0
 (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 1024)              0

 flatten_1 (Flatten)         (None, 1024)              0

 dense_3 (Dense)             (None, 64)                65600

 dense_4 (Dense)             (None, 7)                 455

=================================================================
Total params: 6,072,375
Trainable params: 6,072,375
Non-trainable params: 0
_________________________________________________________________
```

Conclusion

- When the dataset is small
  - Changing the structure of the model is able to increase the accuracy and control overfitting, with mild effect on runtime
- When the dataset is large:
  - GoogleNet is faster the baseline

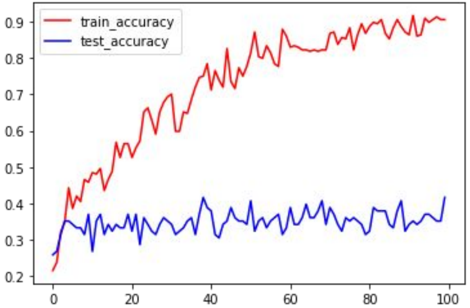
- Use `’val_categorical_accuracy’` to evaluate accuracy
- Overall top 3 performance:
  - L2 Regularization
  - baseline + Batch Norm + 2 Dense 64 Layer
  - baseline

```
{'angry': 0,
 'crying': 1,
 'embarrassed': 2,
 'happy': 3,
 'pleased': 4,
 'sad': 5,
 'shock': 6}
```

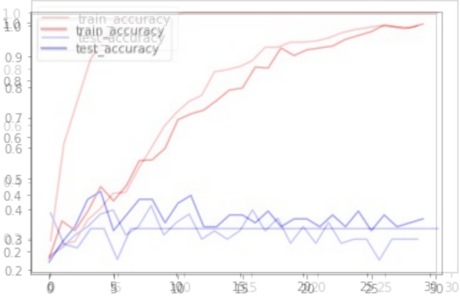
```
1/1 [==============================] - 1s 511ms/step
[6, 6, 2, 1, 5, 5, 0, 2, 6, 6, 5, 5, 0, 3, 2, 2, 4, 6, 1, 4, 2, 0, 6, 6, 3, 5, 6, 2, 5, 6, 5, 3]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.40      1.00      0.57         2
           2       1.00      0.83      0.91         6
           3       0.67      0.67      0.67         3
           4       0.67      1.00      0.80         2
           5       1.00      0.57      0.73         7
           6       0.89      0.89      0.89         9

    accuracy                           0.81        32
   macro avg       0.80      0.85      0.79        32
weighted avg       0.88      0.81      0.82        32
```

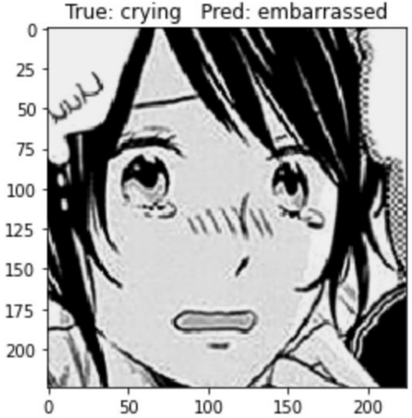
Manga

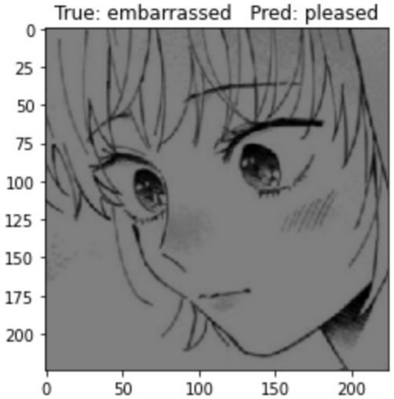
ResNet - RELU + sigmoid, single dense layer



Vanilla CNN



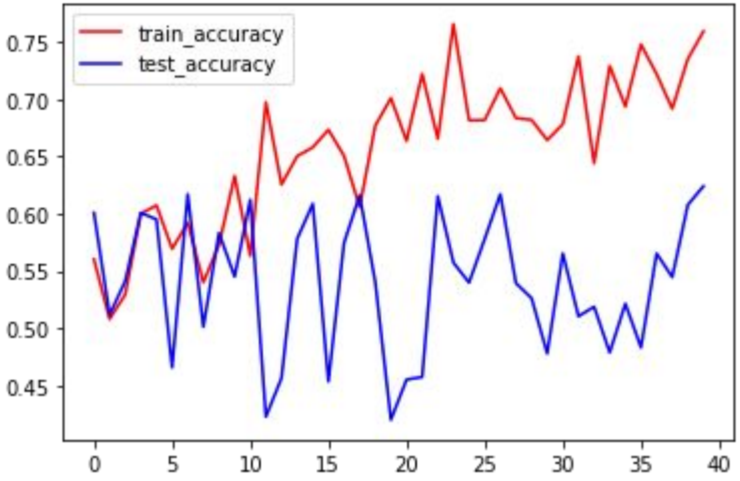




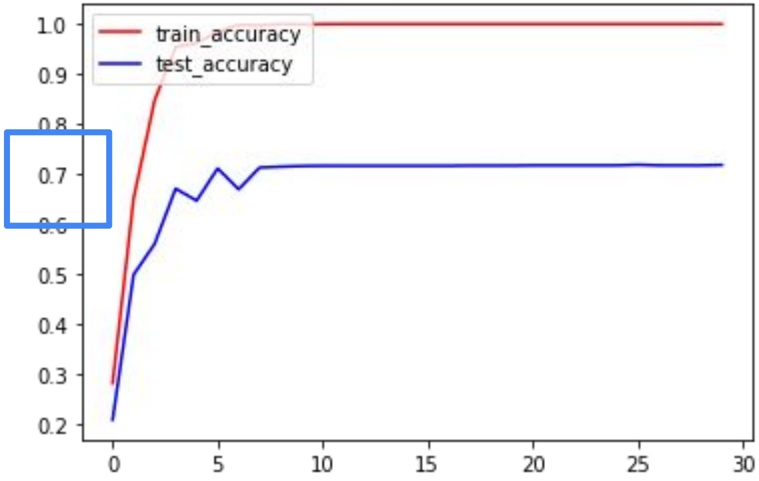
	precision	recall	f1-score	support
1	0.00	0.00	0.00	2
2	0.20	0.20	0.20	5
'happy' 3	0.50	0.44	0.47	9
4	0.20	0.50	0.29	2
'sad' 5	0.33	0.50	0.40	4
'shock' 6	0.50	0.40	0.44	10
accuracy			0.38	32
macro avg	0.29	0.34	0.30	32
weighted avg	0.38	0.38	0.37	32

FERG

ResNet



Vanilla CNN

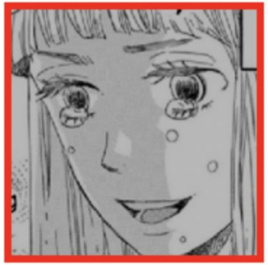


Fun Fact

Few shades below eyes → sad/crying

P:crying C:sad

P:sad C:embarrassed



Wide open eyes → angry

P:angry C:embarrassed

P:angry C:shock

Reference

http://grail.cs.washington.edu/projects/deepexpr/ferg-2d-db.html

https://ai.plainenglish.io/googlenet-inceptionv1-with-tensorflow-9e7f3a161e87

https://openaccess.thecvf.com/content_cvpr_2017/papers/Wu_A_Compact_DNN_CVPR_2017_paper.pdf

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Link to github