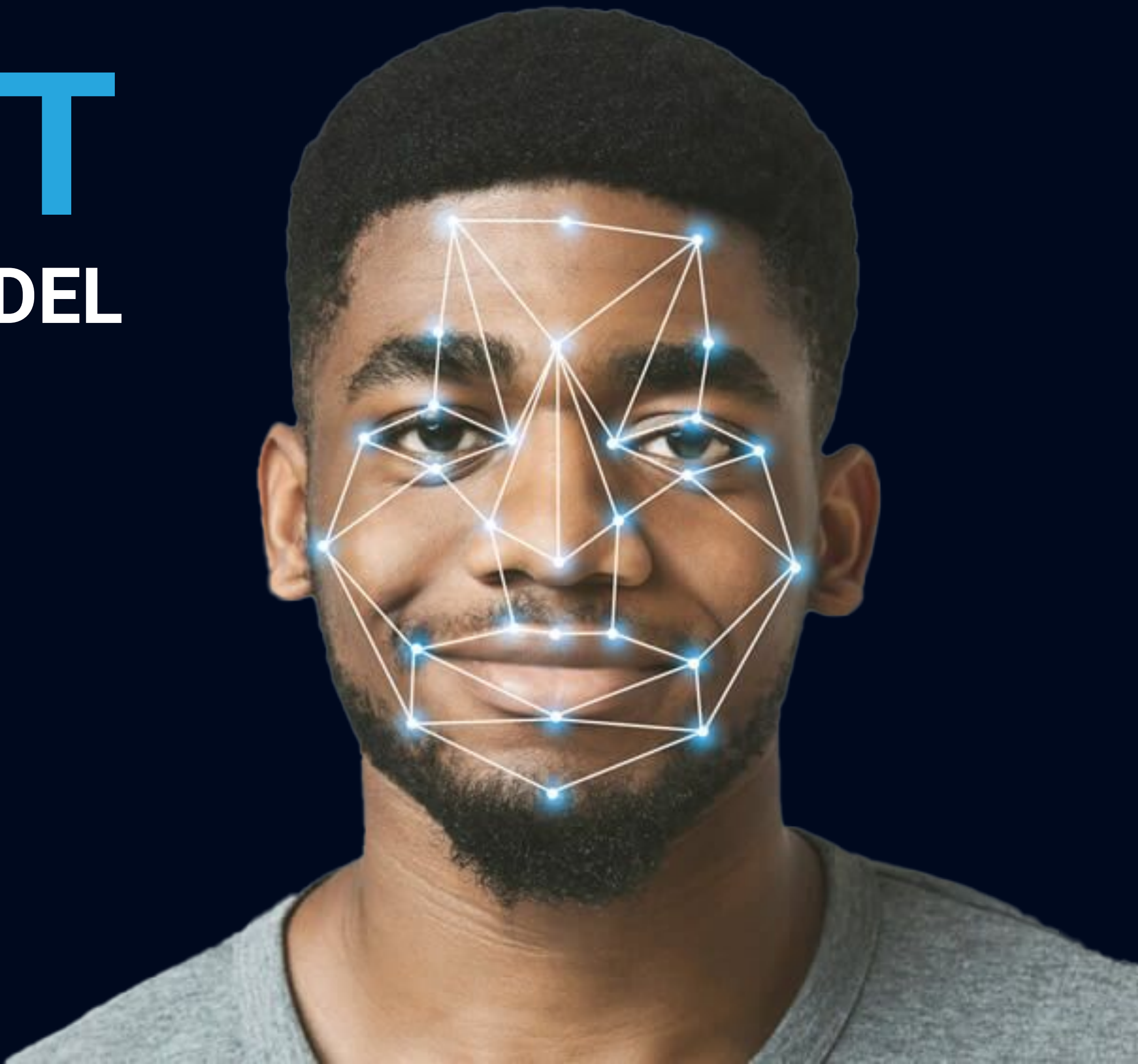
LET'S FACE IT

EMOTIONAL RECOGNITION MODEL



Georgiy Sekretaryuk, Rafael Arbex-Murut, Yeshwanth Somu

PROJECT OVERVIEW

THE DATA

- This Kaggle Facial Recognition Dataset contains grayscale images with different facial expressions.
    - surprise, anger, happiness, sad, neutral, fear
- Training Set: 28,079 (80%) // Testing set: 7,178 (20%)
- Attributes: 2304 (48 x 48 pixels)

How do we predict emotions?

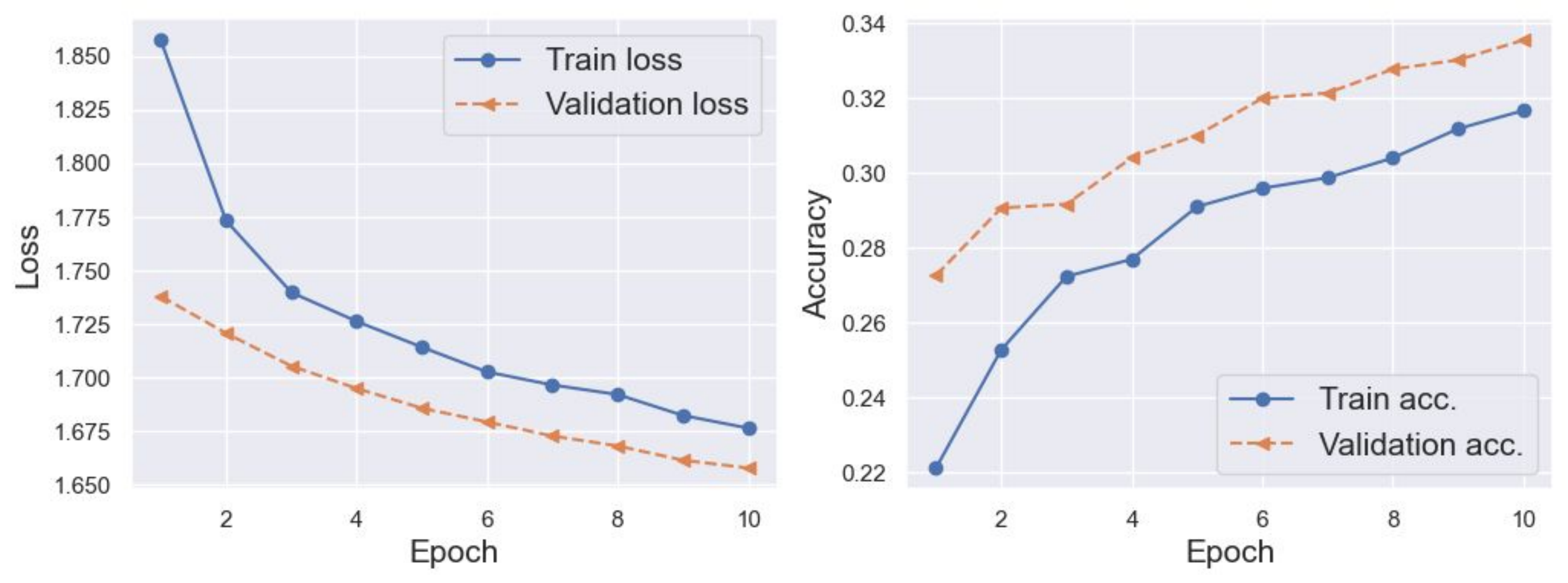
OUR PROBLEM

Our problem was - how do we build a good enough model to predict emotions (that may vary in perception)?

We started with an attempt to build an FNN model first and assess performance. We decided that if performance was low, we would build a Convolutional Neural Network Model (CNN).

INITIAL BASELINE MODEL - FNN

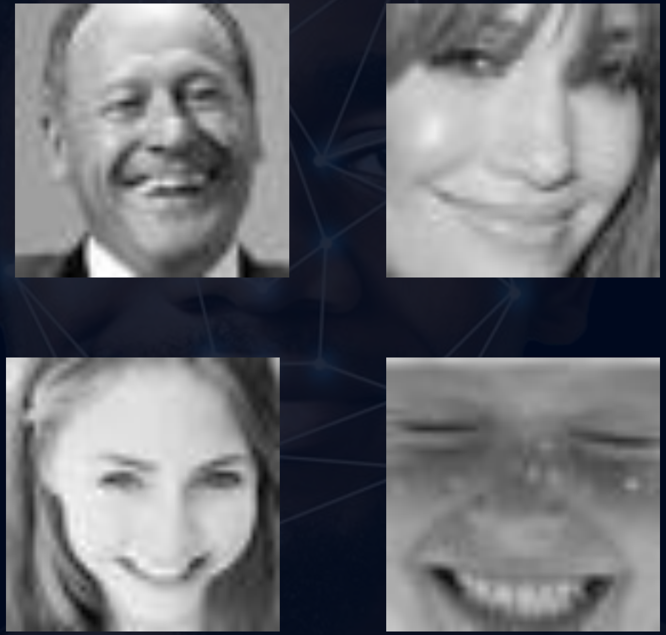
- The model performed better than random guessing would



DIFFERENT STRATEGY - CNN MODEL

- We decided to use a CNN Model instead. As we learned in the course, CNN models account for inconsistencies in images. While our images were of the same size, the faces were certainly not standardized like the MNIST dataset.
- Different zooms, face sizes, etc.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_1 (Conv2D)             (None, 48, 48, 32)        832

 pool_1 (MaxPooling2D)       (None, 24, 24, 32)        0

 conv_2 (Conv2D)             (None, 24, 24, 64)        51264

 pool_2 (MaxPooling2D)       (None, 12, 12, 64)        0

 flatten_1 (Flatten)         (None, 9216)              0

 fc_1 (Dense)                (None, 1024)              9438208

 dropout_1 (Dropout)         (None, 1024)              0

 fc_2 (Dense)                (None, 6)                 6150

=================================================================
Total params: 9,496,454
Trainable params: 9,496,454
Non-trainable params: 0
_________________________________________________________________
```

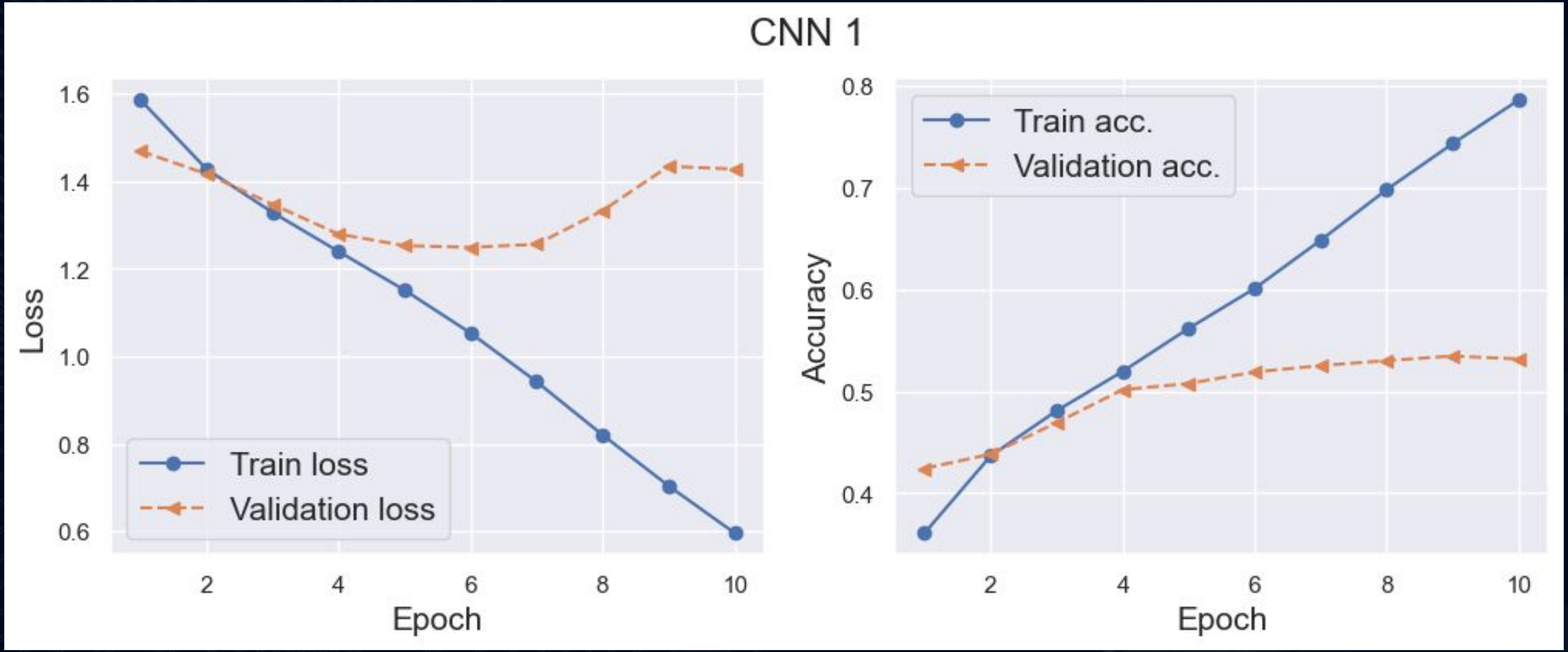


DIFFERENT STRATEGY - CNN MODEL

...RESULTS IMPROVED

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	1.5860	0.3612	1.4623	0.4208
2	1.4091	0.4413	1.3460	0.4694
3	1.2945	0.4984	1.2824	0.4974
4	1.2018	0.5389	1.2534	0.5131
5	1.1008	0.5828	1.2163	0.5353
6	0.9949	0.6255	1.2260	0.5373
7	0.8701	0.6776	1.2473	0.5397
8	0.7450	0.7249	1.3136	0.5455
9	0.6271	0.7702	1.3630	0.5541
10	0.5250	0.8119	1.4674	0.5538

DIFFERENT STRATEGY - CNN MODEL



DIFFERENT STRATEGY - CNN MODEL

True Label \ Predicted Label	Happy	Suprise	Angry	Neutral	Fear	Sad
Happy	1338	41	47	162	76	110
Suprise	46	605	22	58	73	27
Angry	108	28	353	185	108	176
Neutral	131	26	55	716	95	210
Fear	88	90	96	168	374	208
Sad	145	29	110	282	153	528

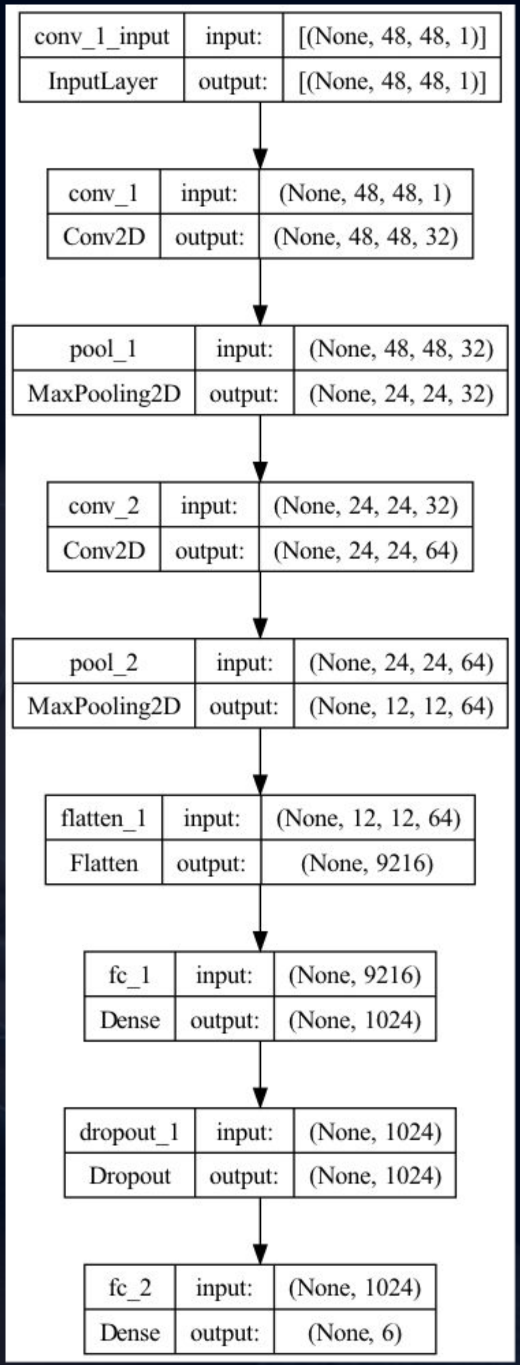
	precision	recall	f1-score	support
0	0.721	0.754	0.737	1774
1	0.739	0.728	0.733	831
2	0.517	0.368	0.430	958
3	0.456	0.581	0.511	1233
4	0.425	0.365	0.393	1024
5	0.419	0.423	0.421	1247
accuracy			0.554	7067
macro avg	0.546	0.537	0.538	7067
weighted avg	0.553	0.554	0.550	7067

2ND MODEL - TWEAKING MODEL 1

- We decided to adjust Model 1 and try:
  - a different optimizer (SGD)
  - Adjusting the kernel size to 4x4 instead of 5x5 pixels.

Layer (type)	Output Shape	Param #
conv_1 (Conv2D)	(None, 48, 48, 32)	544
pool_1 (MaxPooling2D)	(None, 24, 24, 32)	0
conv_2 (Conv2D)	(None, 24, 24, 64)	32832
pool_2 (MaxPooling2D)	(None, 12, 12, 64)	0
flatten_1 (Flatten)	(None, 9216)	0
fc_1 (Dense)	(None, 1024)	9438208
dropout_1 (Dropout)	(None, 1024)	0
fc_2 (Dense)	(None, 6)	6150

Total params: 9477734 (36.15 MB)
Trainable params: 9477734 (36.15 MB)
Non-trainable params: 0 (0.00 Byte)



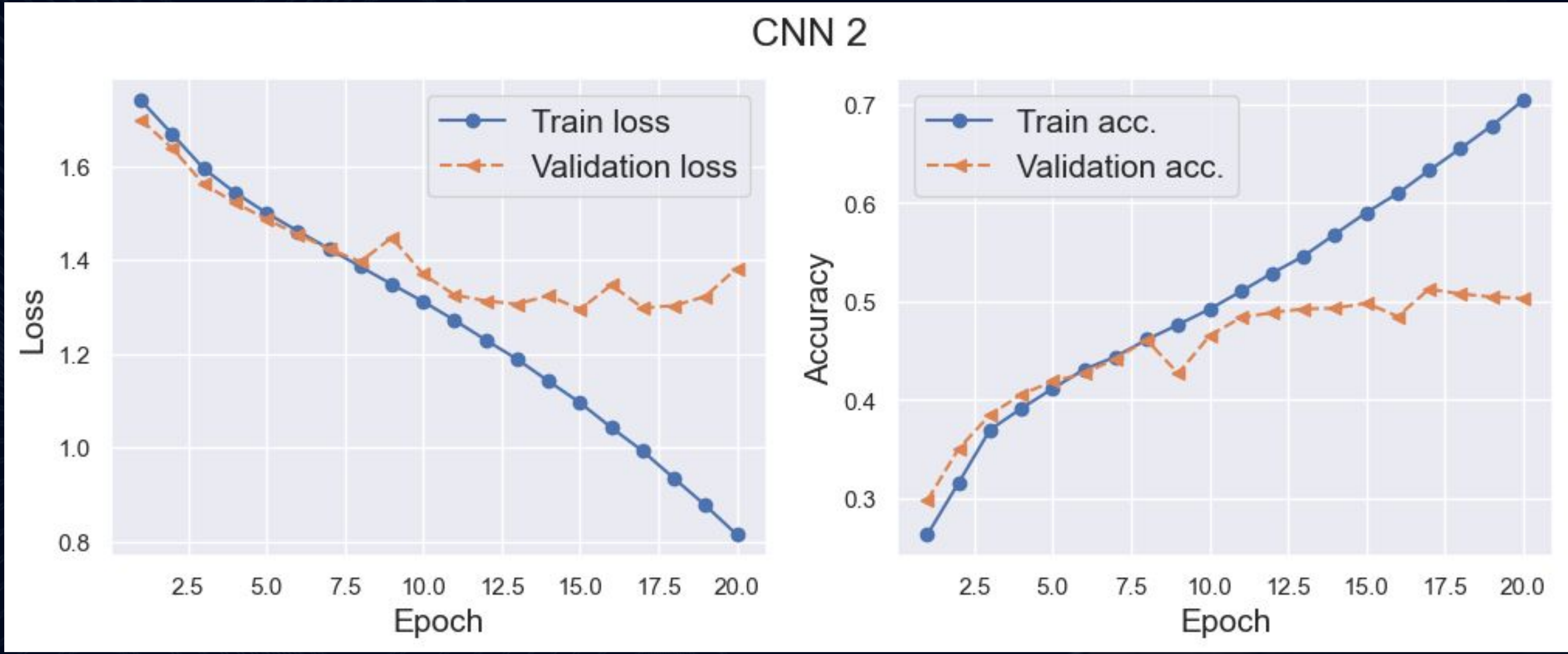
AND THE RESULTS FOR MODEL 2...

DIDN'T IMPROVE.

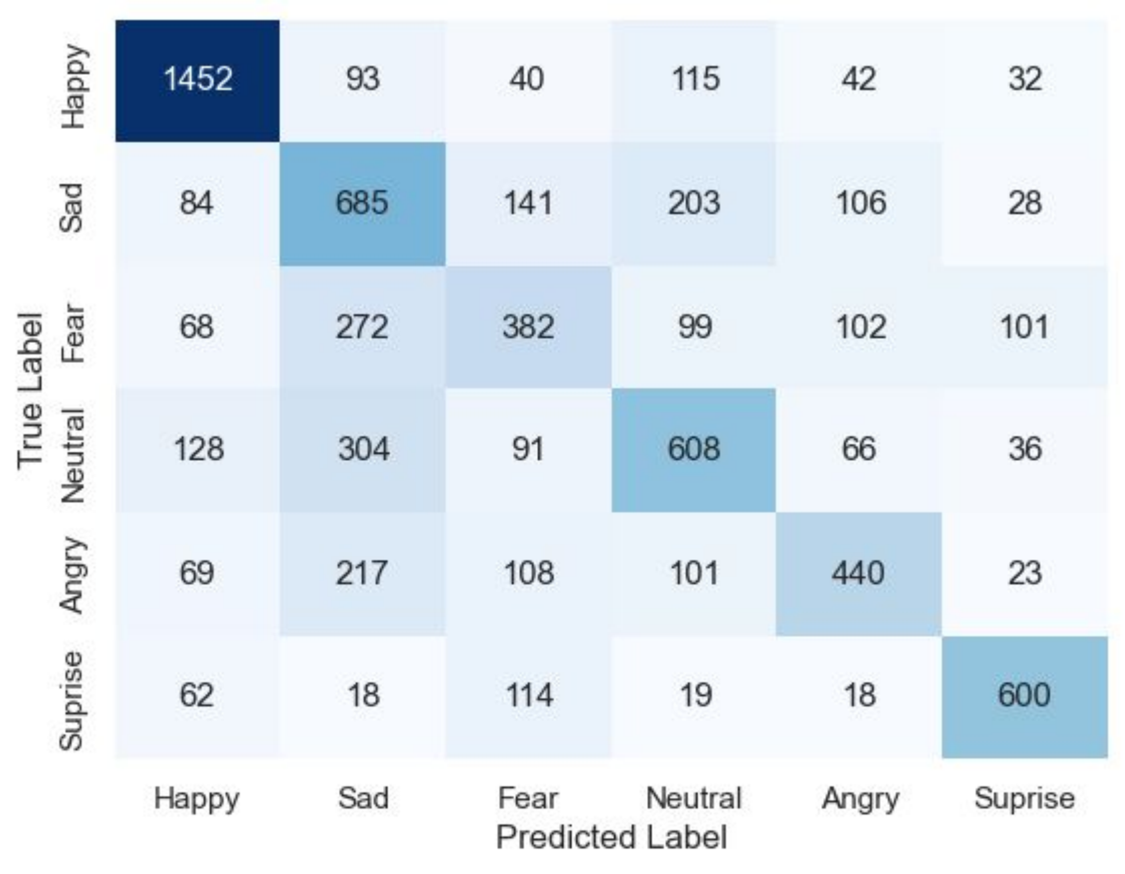
Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	1.5584	0.3712	1.4057	0.4439
2	1.3557	0.4685	1.3015	0.4900
3	1.2409	0.5204	1.2580	0.5110
4	1.1337	0.5691	1.2330	0.5299
5	1.0233	0.6151	1.2335	0.5370
6	0.8933	0.6690	1.2322	0.5489
7	0.7547	0.7249	1.2941	0.5427
8	0.6264	0.7709	1.3548	0.5544
9	0.5131	0.8149	1.4346	0.5536
10	0.4264	0.8465	1.5872	0.5531

2ND MODEL PERFORMANCE

VALIDATION LOSS AND ACCURACY SUFFERED



2ND MODEL - WORSE THAN MODEL 1

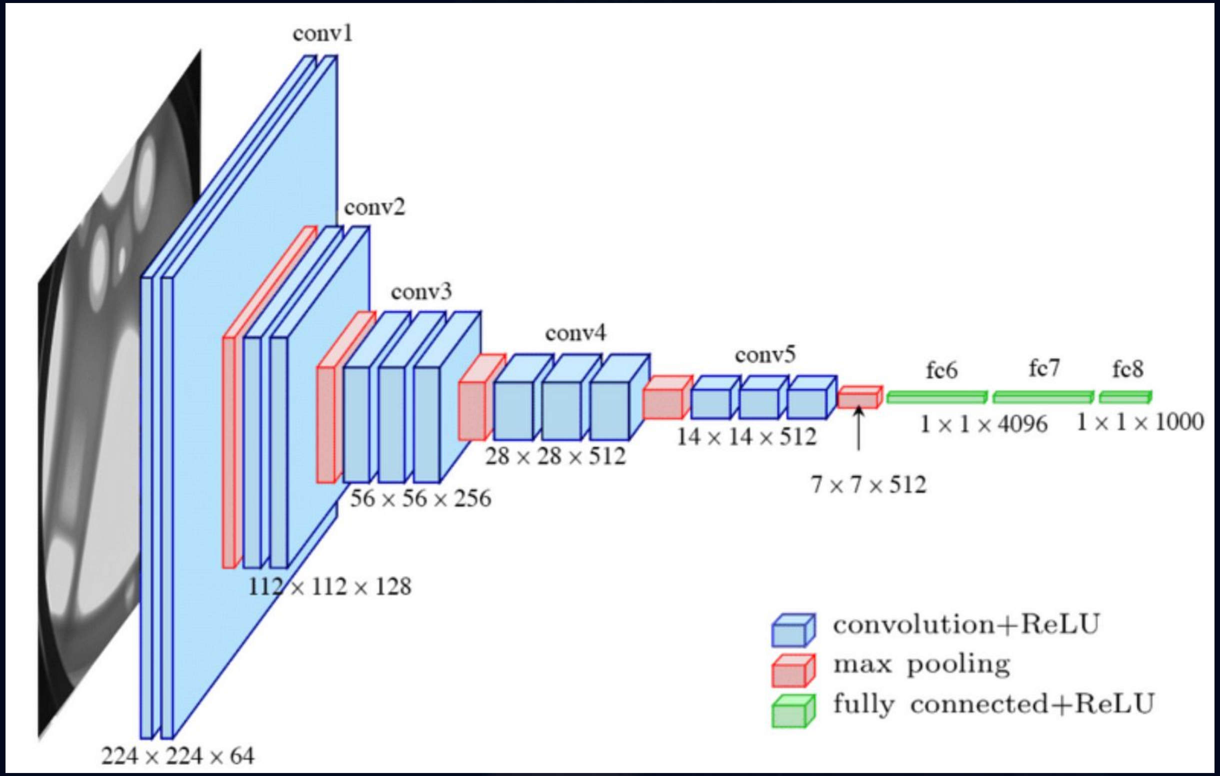


	precision	recall	f1-score	support
0	0.531	0.702	0.605	1774
1	0.600	0.005	0.010	1247
2	0.346	0.225	0.273	1024
3	0.345	0.397	0.370	1233
4	0.368	0.281	0.319	958
5	0.340	0.777	0.473	831
accuracy			0.408	7067
macro avg	0.422	0.398	0.341	7067
weighted avg	0.439	0.408	0.356	7067

LET'S TRY A DIFFERENT ARCHITECTURE…

ENTER VGG 16

- VGG 16 is a CNN architecture used to win the Imagenet competition in 2014.
- Considered to be a high performing vision model.
- Focuses on convolutional layers of 3x3 filters with a stride of 1, always uses the same padding, and a maxpool layer of 2x2 filter with a stride of 2.
- Has 16 layers (hence the name).
- Approximately 138 million parameters.



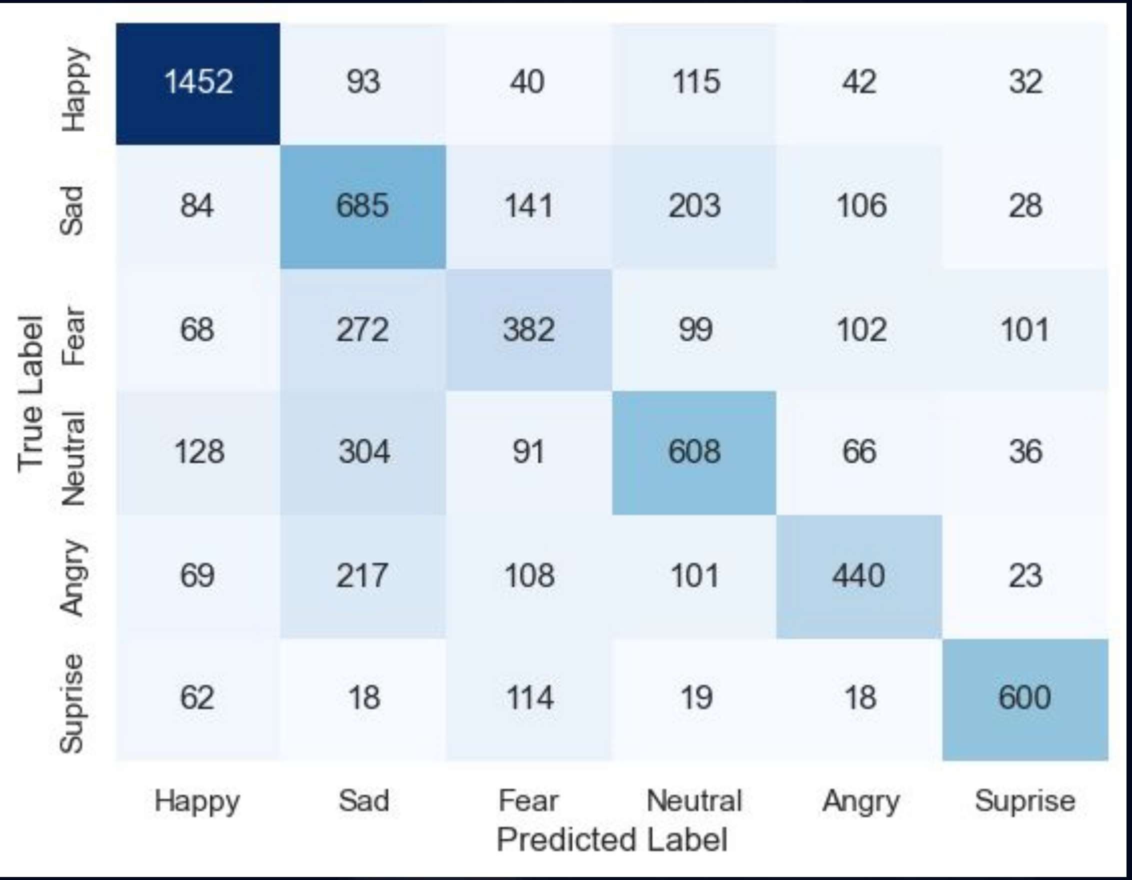
VGG 16 TRAINING

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	1.6487	0.3093	1.4372	0.431
2	1.3526	0.4551	1.2756	0.4868
3	1.1843	0.5332	1.1727	0.5373
4	1.0539	0.5909	1.1011	0.5714
5	0.9257	0.6452	1.0988	0.586
6	0.7773	0.7076	1.1524	0.593
7	0.6197	0.7707	1.1552	0.6029
8	0.4638	0.8328	1.3542	0.5981
9	0.3201	0.8871	1.5211	0.5821
10	0.231	0.9204	1.6232	0.586

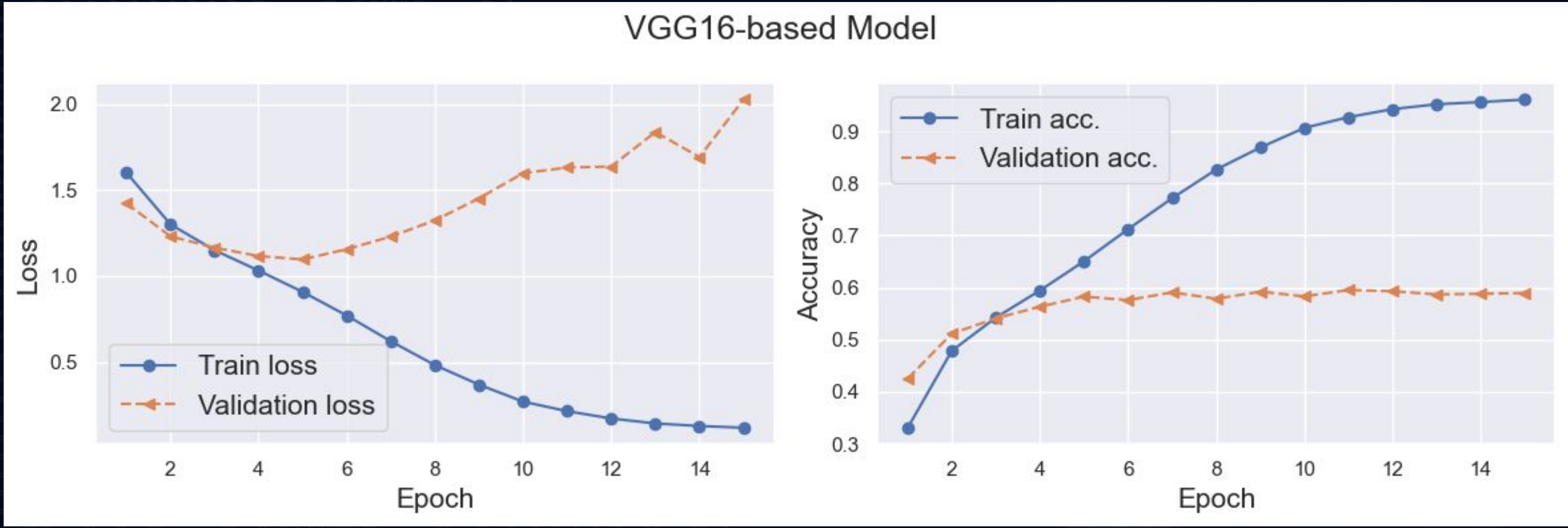
PERFORMANCE WITH VGG 16

RESULTS IMPROVED, SLIGHTLY

	precision	recall	f1-score	support
0	0.779	0.818	0.798	1774
1	0.431	0.549	0.483	1247
2	0.436	0.373	0.402	1024
3	0.531	0.493	0.511	1233
4	0.568	0.459	0.508	958
5	0.732	0.722	0.727	831
accuracy			0.590	7067
macro avg	0.580	0.569	0.572	7067
weighted avg	0.591	0.590	0.587	7067

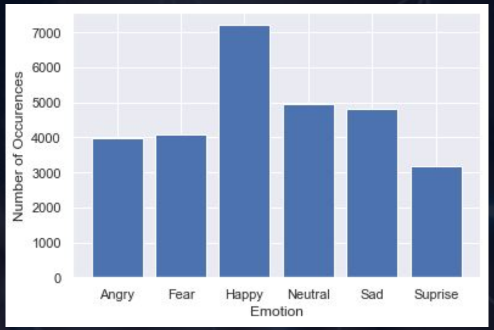


PERFORMANCE WITH VGG 16



LEARNING

- Model architecture learnings
  - Model 2 did not perform better with SGD / smaller Kernels
  - VGG 16 model performed better
- Lot of happy faces in training data
  - 25% of the training data was happy faces, leading to unequal classes
  - Overfitting on the “Happy” faces in some cases



- Location of faces in images matters
  - Non-centered faces were predicted poorly

- Emotion labeling can be subjective
  - Many faces can be interpreted as different emotions such as Neutral OR Angry


If we had more time

- Explore other model architectures
  - Potentially more parameters, more kernel sizes, or ensemble models
- Work on larger data sets with more equally weighted classes
- Have more emotions and define better method for initial labeling
    - Especially for expressions that can be interpreted as multiple emotions
    - Additional emotions: Doubt, Pain, Disgust
- Connect to camera and turn into an app
  - Explored creating a Flask app that uses your camera to predict your emotion in real-time, but ran out of time

THANK YOU!

HAPPY TO ANSWER ANY ADDITIONAL QUESTIONS