MLPR - AI3011

Quantifying Classroom Learning

ARSHIA SANGWAN & RAHATH MALLADI

Problem Statement

We aim to quantify classroom learning using a facial expression based attention model

Why: *To support students' learning methodologies and to give real-time assistance to the educator.*

Potential Applications: *Educator assistance, Personalized attention tracking*

Literature Survey

Literature Survey

- Attention can be categorized as follows based on the student engagement:
  Focused, Sustained, Selective, Alternating and Divided

- With 55% accuracy, our facial emotions are reflective of our attention.[1]

- Applied SVM, kNN and Decision Tree to gain an understanding of the accuracy of the classifier. Decision tree proved to be the most accurate. [2]

- Models: CNN, RNN, YOLO v3 deep-learning based algorithm, VGG16

- Models with different accuracy (YOLO-88%).

[1]Recognizing Students' Attention in a Virtual Class through Facial Expressions using Machine Vision
[2]ASSESSMENT OF LEARNERS' ATTENTION TO E-LEARNING BY MONITORING FACIAL EXPRESSIONS FOR COMPUTER NETWORK COURSES

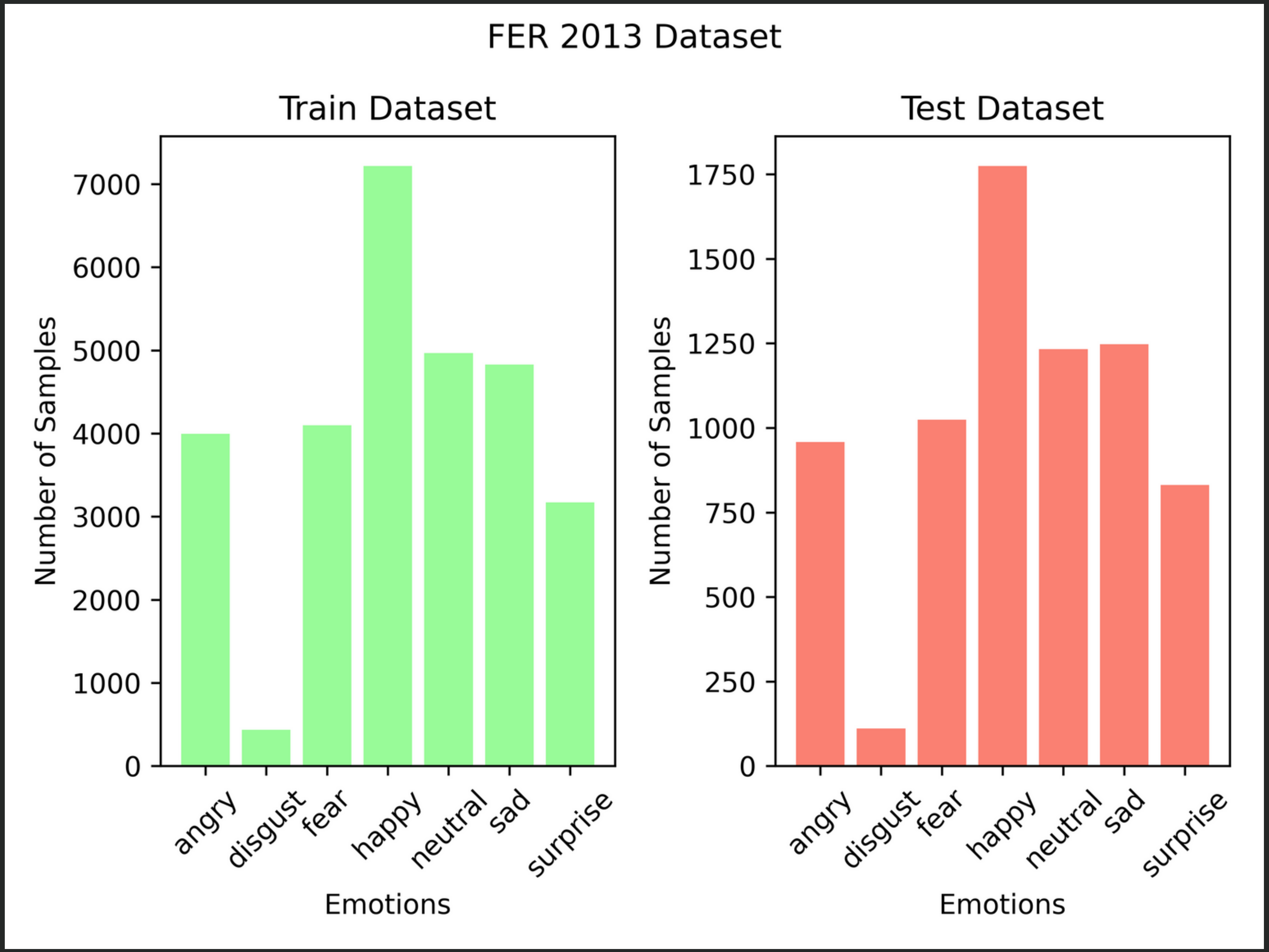
Datasets & Feature Preprocessing

Data Collection

1. Mixture of Datasets: FER_2013, UIBVFED
2. Testing on data collected at Plaksha: A total of 3600 images
3. Annotated on the basis of student feedback obtained via forms.
4. FER dataset : very large dataset (Total: 35,887; testing: 7178)
5. UIBVFED dataset : 635 , Includes both male and female, multiple facial patterns for a single expression too

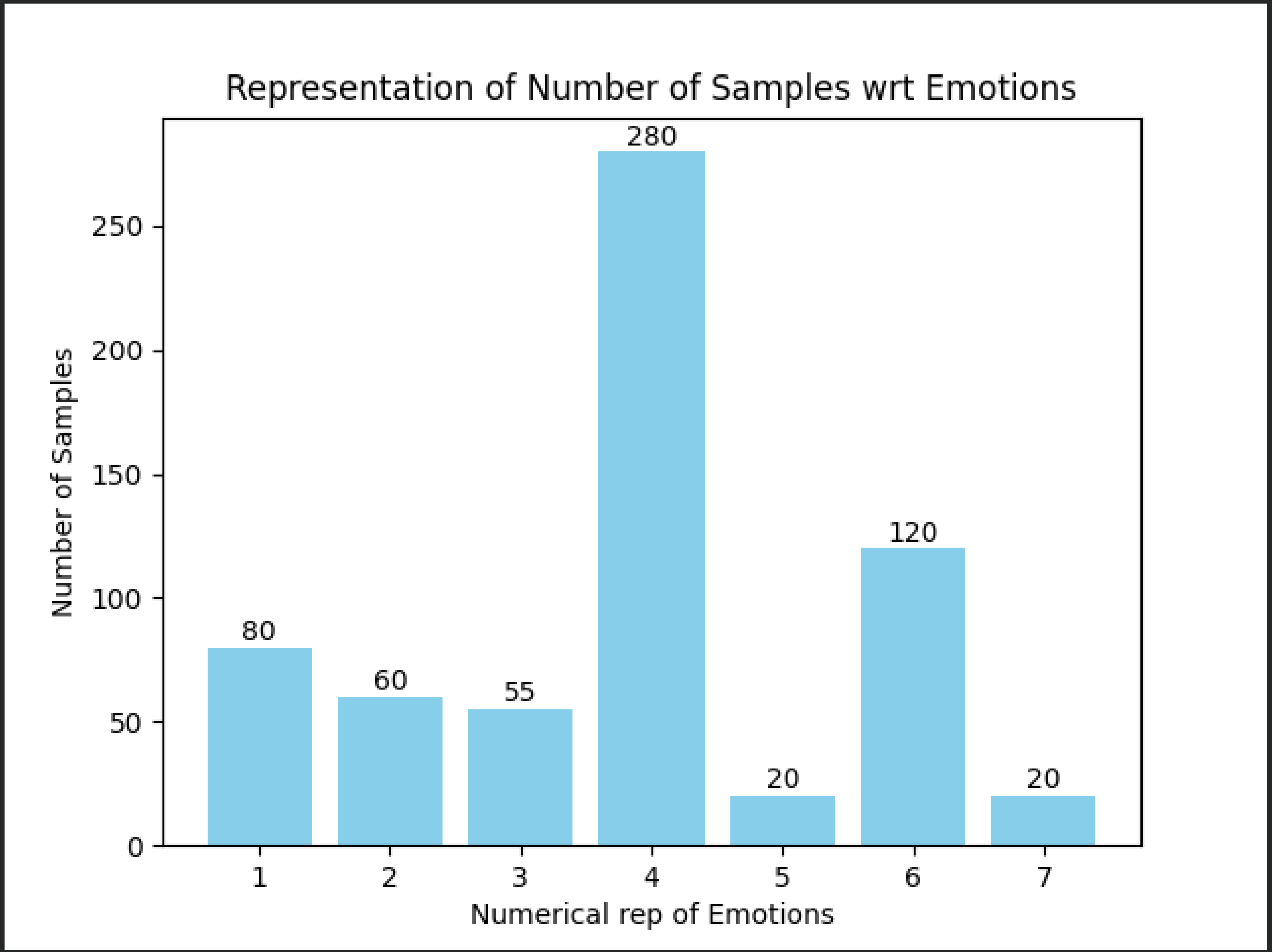
- Used Haar-Cascade to obtain bounding boxes
- Tried YOLOv5 but didn’t work due to generalized object detection
- OpenFace: Core Focus on FAUs
- 686 Features (34 FAUs, 68 Landmarks, etc.)
- Reduced to 50 Features by PCA
- Irregular Sample Sizes
- SMOTE to Equalize the Chance of training

FER 2013

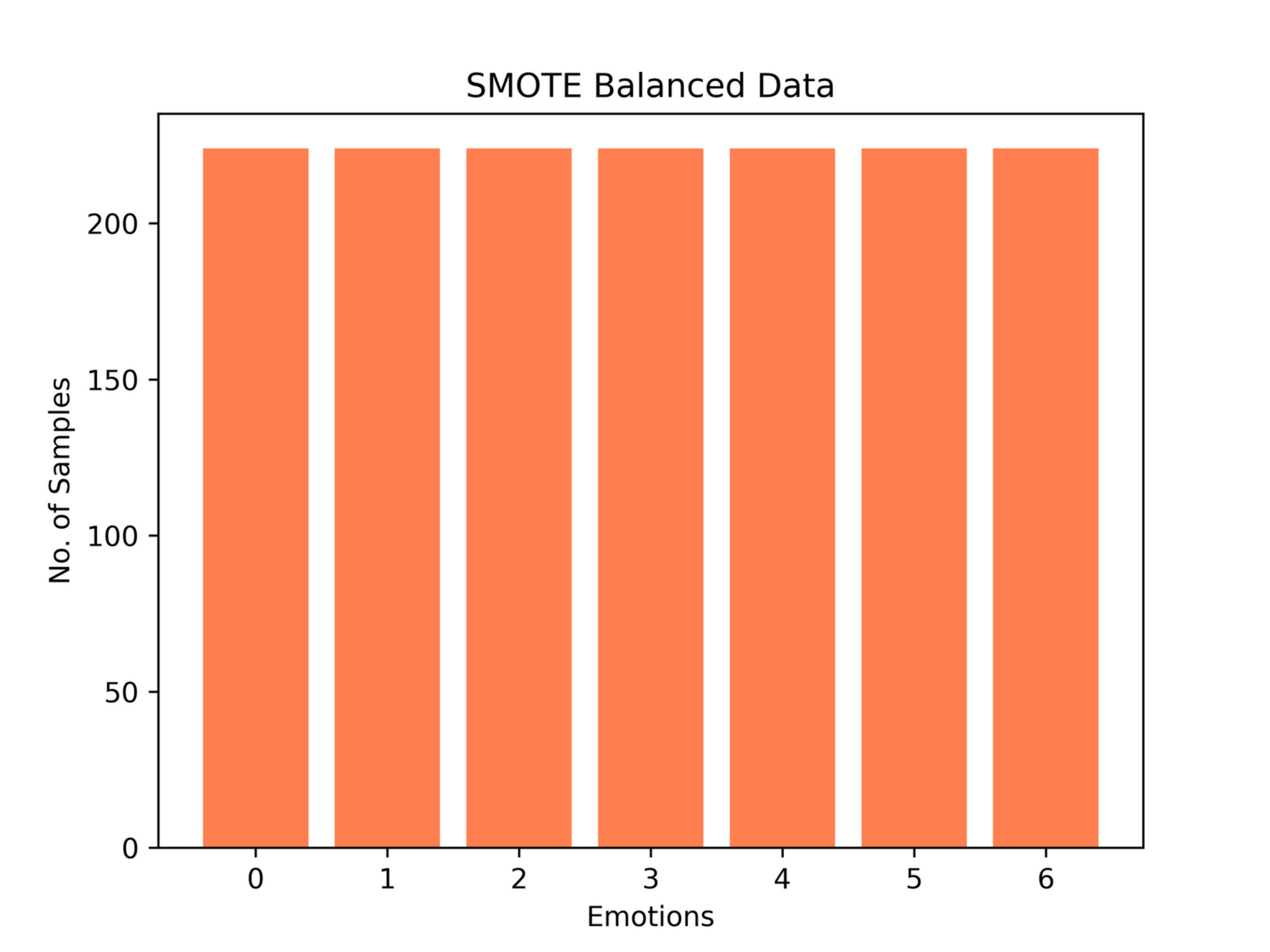


FER 2013 Dataset
Train Dataset
Test Dataset
Number of Samples
7000
6000
5000
4000
3000
2000
1000
0
1750
1500
1250
1000
750
500
250
0
angry
disgust
fear
happy
neutral
sad
surprise
Emotions

UIBVFED

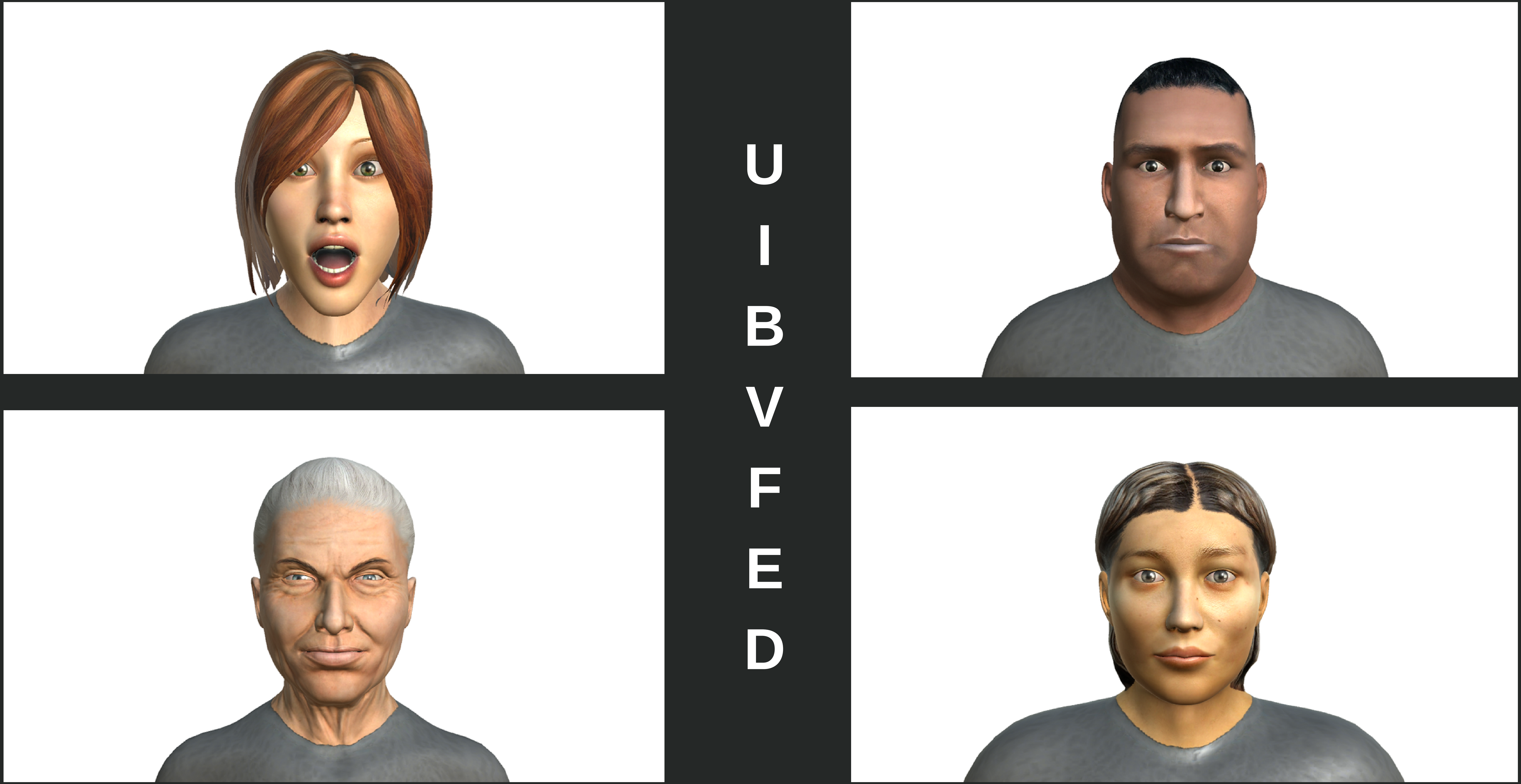

Representation of Number of Samples wrt Emotions
Number of Samples
0
50
100
150
200
250
80
60
55
280
20
120
20
1
2
3
4
5
6
7
Numerical rep of Emotions

after SMOTE



SMOTE Balanced Data
No. of Samples
0
50
100
150
200
0
1
2
3
4
5
6
Emotions

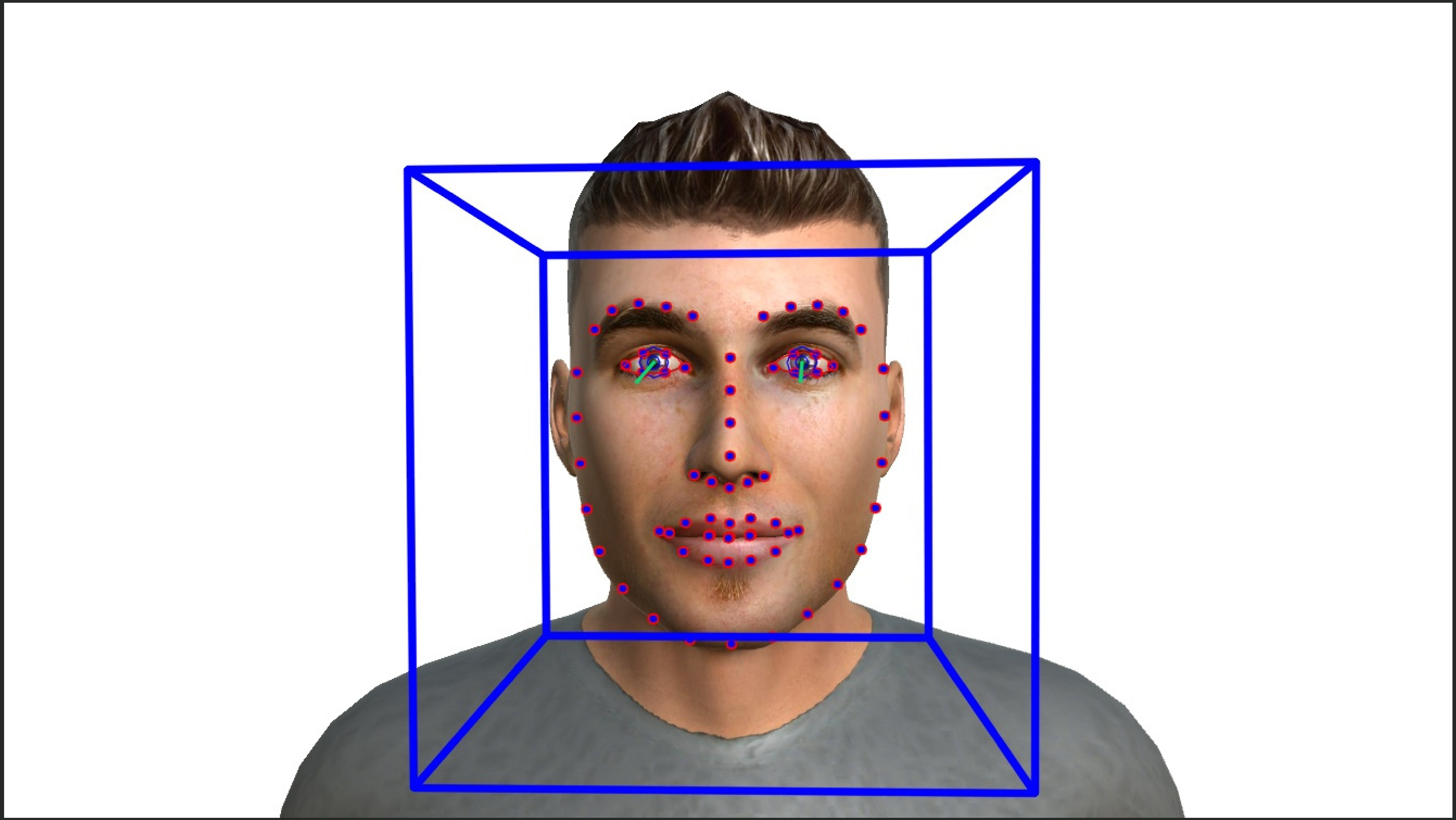
7 Emotions in FER 2013


UIBVFED


Functions





ML Methodology

```python
# Trying different algorithms:
# 1. Random Forest
model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```python
model.fit(X_train, y_train)
```

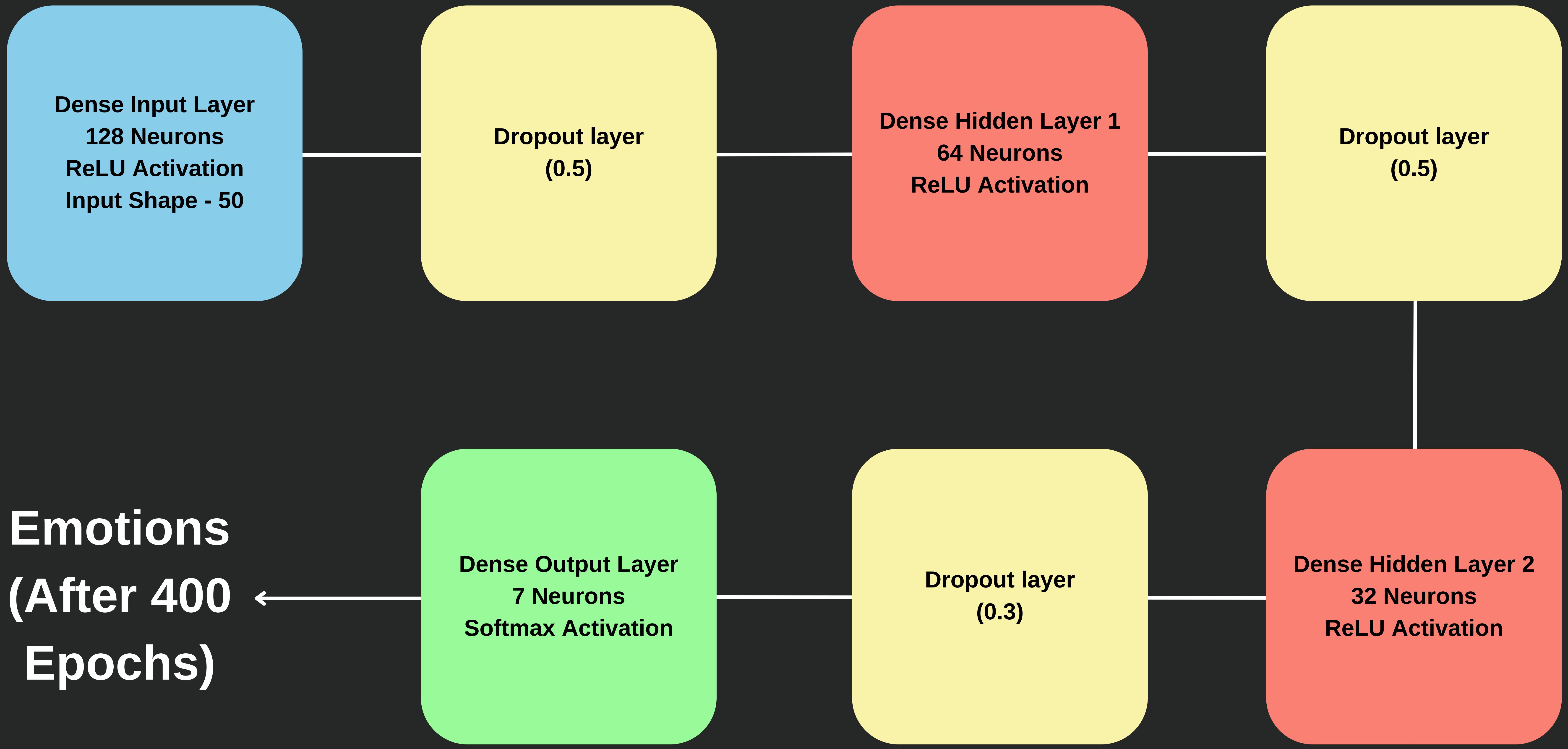
```python
# Hyperparameter tuning: Grid search
# Just an Attempt
param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [None, 10, 20]}
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

```python
# Why not a Voting classifier?
model1 = RandomForestClassifier(n_estimators=100, random_state=42)
model2 = GradientBoostingClassifier(n_estimators=100, random_state=42)
model3 = SVC(kernel='rbf', random_state=42)
voting_classifier = VotingClassifier(estimators=[('rf', model1), ('gb', model2), ('svc', model3)], voting='hard')
```

```python
# The CNN Way!
model = Sequential([
    Conv2D(64, (3, 3), activation='relu', input_shape=(img_size, img_size, 1), kernel_regularizer=l2(0.01)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', kernel_regularizer=l2(0.01)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu', kernel_regularizer=l2(0.01)),
    Dense(len(emotions), activation='softmax', kernel_regularizer=l2(0.01))
])
```

```python
basemodel2 = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(256, (3,3), activation='relu', input_shape=(48,48,1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.BatchNormalization(),

  tf.keras.layers.Conv2D(128, (3,3), activation='relu', input_shape=(48,48,1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.BatchNormalization(),

  tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(48,48,1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.BatchNormalization(),

  tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(48,48,1)),
  tf.keras.layers.MaxPooling2D(2, 2),
  tf.keras.layers.BatchNormalization(),

  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dense(7, activation='softmax')
])
```

```python
# Useing VGG16 model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(7, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```


Dense Input Layer
128 Neurons
ReLU Activation
Input Shape - 50
Dropout layer
(0.5)
Dense Hidden Layer 1
64 Neurons
ReLU Activation
Dropout layer
(0.5)
Dense Hidden Layer 2
32 Neurons
ReLU Activation
Dropout layer
(0.3)
Dense Output Layer
7 Neurons
Softmax Activation
Emotions
(After 400
Epochs)

Emotion to Attention

WELL ATTENTIVE

Greater Probabilities @ Neutral, Sad, Anger

MODERATELY ATTENTIVE

Fairly High Probabilities @
Neutral + any other emotion

WEAKLY ATTENTIVE

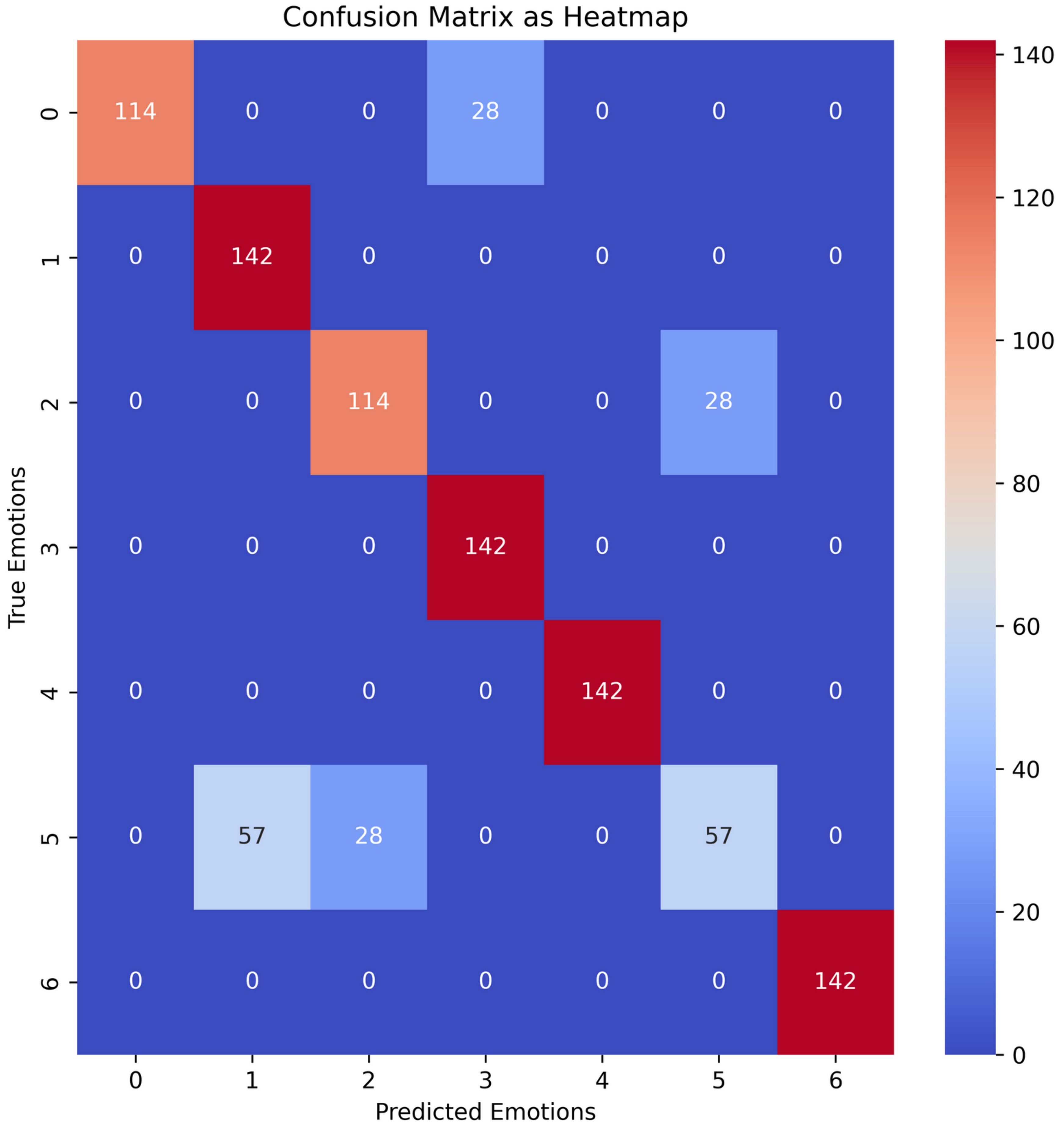
Greater Probabilities @ Surprised, Joy, Disgust

Performance Metrics

Accuracy - 73% (FER)
Accuracy - [87%,~99%] (UBIVFED)

```
20/20 [==============================] - 0s 2ms/step - loss: 0.0500 - accuracy: 0.9825 - val_loss: 0.0023 - val_accuracy: 1.0000
49/49 [==============================] - 0s 921us/step - loss: 0.0025 - accuracy: 0.9994
Accuracy: 99.94%
```

```
Class probabilities: [0.10630276 0.11434971 0.00970086 0.52772164 0.01606483 0.19533798
 0.03052231]
Predicted class: 3
Actual label: 0
Class probabilities: [1.0000000e+00 2.9221480e-15 2.8405174e-17 5.2368251e-16 3.1249684e-21
 1.0975439e-11 4.1803223e-17]
Predicted class: 0
Actual label: 0
Class probabilities: [2.60232156e-03 4.38167775e-12 5.21764904e-02 1.01924074e-04
 9.18397273e-16 8.29737663e-01 1.15381554e-01]
Predicted class: 5
Actual label: 5
Class probabilities: [9.2835299e-13 5.7325055e-15 9.9998271e-01 7.0815424e-13 3.1083932e-18
 1.1905426e-05 5.3469066e-06]
Predicted class: 2
Actual label: 0
```


Confusion Matrix as Heatmap
True Emotions
Predicted Emotions
0 1 2 3 4 5 6
114 0 0 28 0 0 0
0 142 0 0 0 0 0
0 0 114 0 0 28 0
0 0 0 142 0 0 0
0 0 0 0 142 0 0
0 57 28 0 0 57 0
0 0 0 0 0 0 142
0 20 40 60 80 100 120 140

So, Let’s be Attentive & Not be caught by the Model :)

ANY QUESTIONS? WE’RE OPEN TO ANSWER

Thank You!