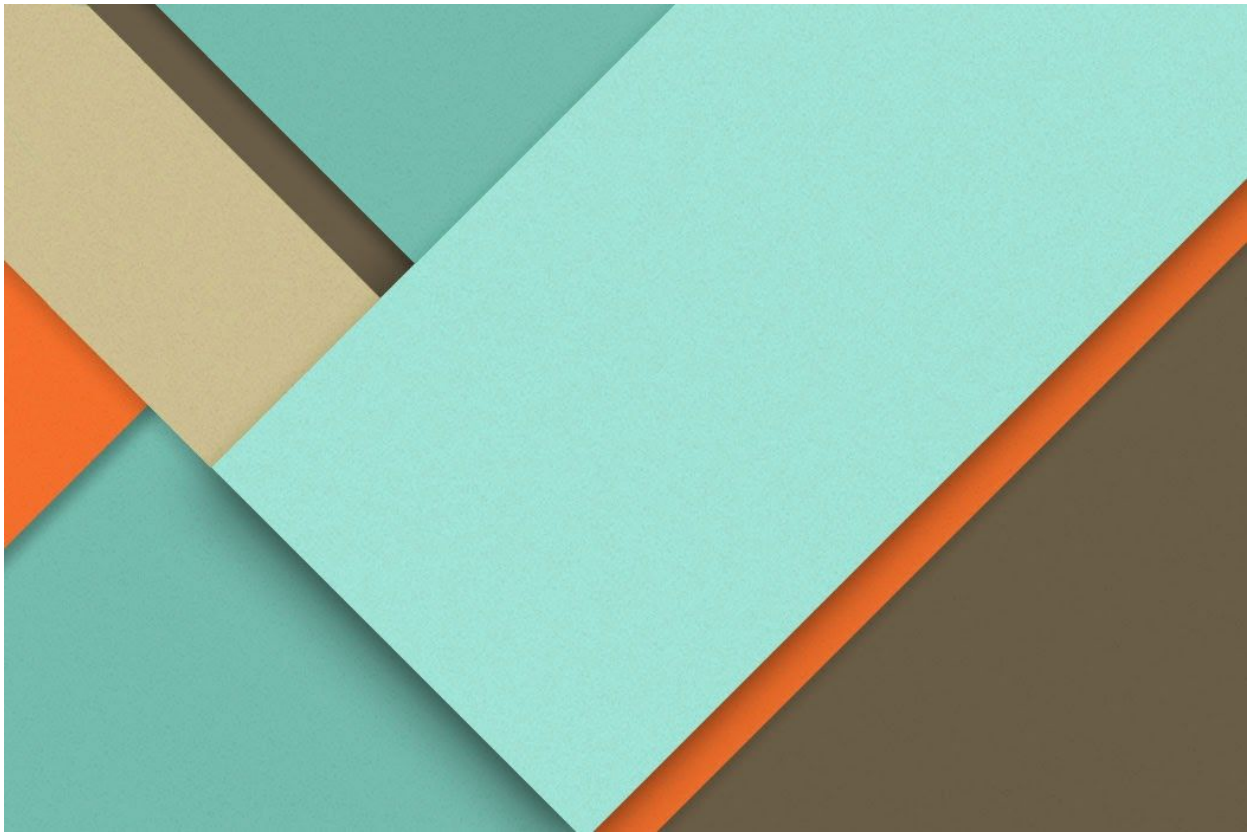


Facial Expression Recognition

Toka Alaa Elgindy

Nada Salama Mohammed

Youmna Gamal

Overview

The task is to categorize each image based on the emotion shown in the facial expression into one of seven categories:

- (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral)

Data Description

1. The data consists of 48x48 pixel grayscale images of faces
2. fer2013.csv contains three columns, "emotion" ,"pixels" and ”usage”.
3. The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image.
4. The “usage” column is a string from the following: “Training” , “PublicTest” and ”PrivateTest”
5. The training set consists of 28,709 examples.
6. The public test set used for the leaderboard consists of 3,589 examples.
7. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples

Emotion labels in the dataset:

0: -4593 images- *Angry*
1: -547 images- *Disgust*
2: -5121 images- *Fear*
3: -8989 images- *Happy*
4: -6077 images- *Sad*
5: -4002 images- *Surprise*
6: -6198 images- *Neutral*

Challenges

1- The images have a low resolution.

2- The faces are not in the same position.

3- Some images have text written on them.

4- Some people hide part of their faces with their hands.

Model

First attempt

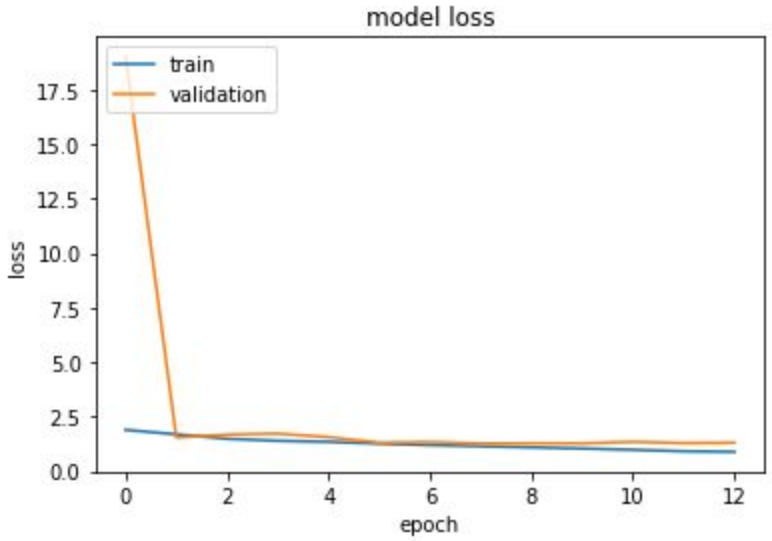
- Apply transfer learning
- use “imagenet“ weights -> convert images from grayscale to RGB
- Build 3 different models with 3 different feature extraction models (VGG16, InceptionV3, Resnet)
- Choose the model with the best results and try to improve it
- Minimum input size for VGG16 and Resnet is 32*32*3 -> our dataset is fine
- Minimum input size for InceptionV3 is 75 * 75 *3 -> Resize images to larger scale
- VGG Model

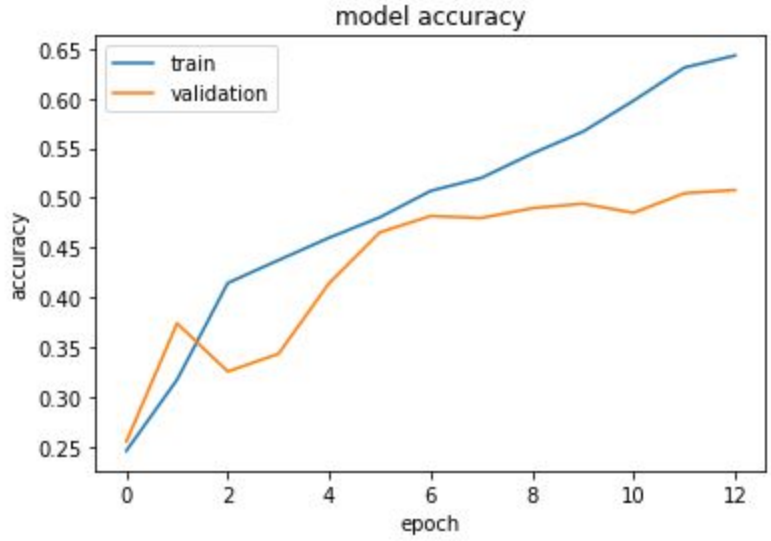
```
[ ] model.history= model.fit(np.array(X_train), np.array(y_train),
                             batch_size=128,
                             epochs=3,
                             validation_data=(np.array(X_val), np.array(y_val)))
    model.save('VGG_first_attempt.h5')
```

```
Epoch 1/3
228/228 [==============================] - 1164s 5s/step - loss: 1.6356 - accuracy: 0.3591 - val_loss: 1.5711 - val_accuracy: 0.3802
Epoch 2/3
228/228 [==============================] - 1159s 5s/step - loss: 1.5350 - accuracy: 0.4071 - val_loss: 1.5414 - val_accuracy: 0.3960
Epoch 3/3
228/228 [==============================] - 1141s 5s/step - loss: 1.4959 - accuracy: 0.4257 - val_loss: 1.5237 - val_accuracy: 0.4111
```

Inception model

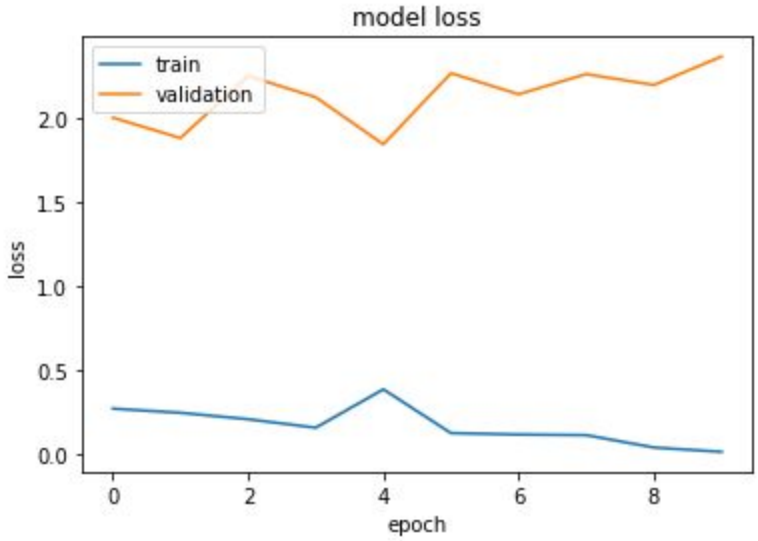
```
base_model = InceptionV3(weights = 'imagenet', include_top = False, input_shape=(W, H, 3))
  # add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
  # let's add a fully-connected layer
x = Dense(1024, activation='relu')(x)
  # and a logistic layer
predictions = Dense(7, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
for layer in model.layers:
  layer.trainable = True
mycallbacks = [                                          # define the callbacks
      keras.callbacks.ReduceLROnPlateau(                       # callback to reduce learning rate
          monitor='val_loss',
          factor = 0.1,
          patience=3,
          verbose=1,
          min_lr=1e-5
      ),
      keras.callbacks.EarlyStopping(                          # callback to stop training if no improvement in validation_loss.
        monitor='val_loss',
        verbose=1,
        patience= 5,
        restore_best_weights=True
        ),
      keras.callbacks.ModelCheckpoint(                      # callback to save the best model if no improvement in validation_AUC.
        monitor='val_accuracy',
        filepath= "../content/drive/My Drive/Facial/checkpoint",
        save_best_only=True,
        mode= 'max',
        verbose=1
        )
    ]
model.compile(optimizer   = Adam(lr = 1e-3, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08, decay = 0.0),loss='categorical_crossentropy', metrics=['accuracy'])
```

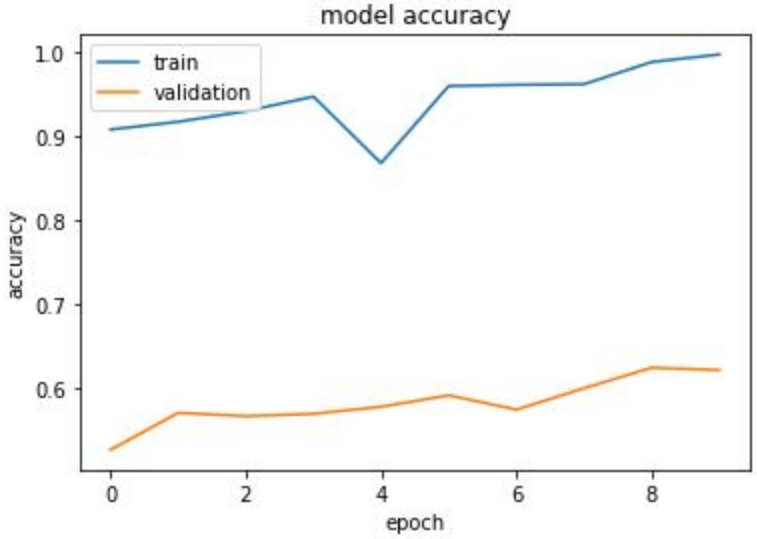




loss: 0.8842 - accuracy: 0.6431 - val_loss: 1.3022 - val_accuracy: 0.5078

Resnet Model





loss: 0.0387 - accuracy: 0.9877 - val_loss: 2.2020 - val_accuracy: 0.6248

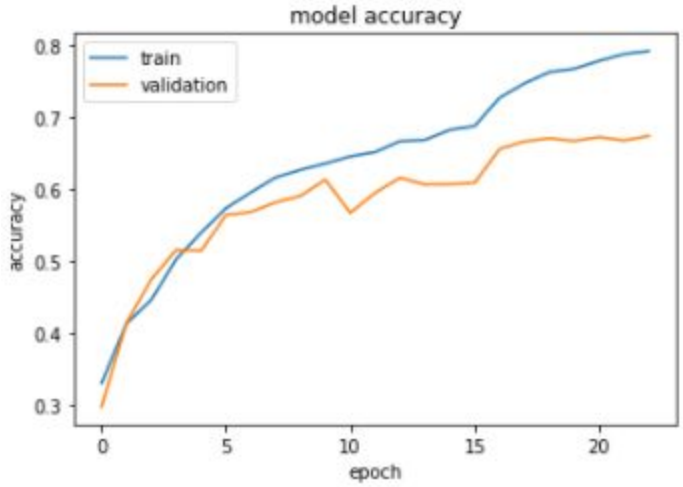
Results

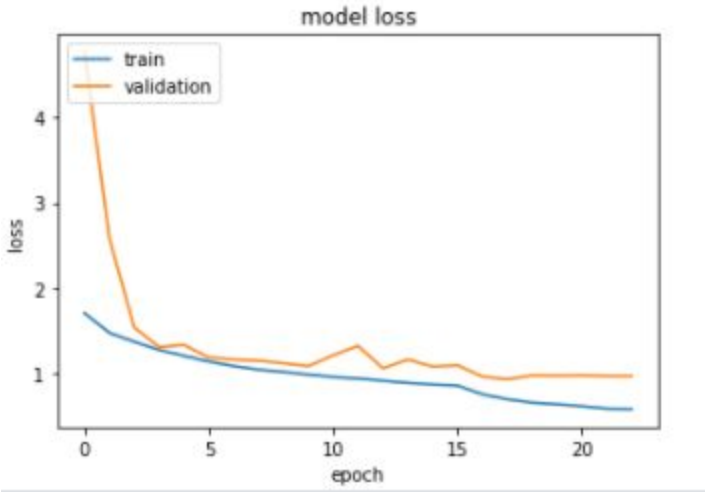
Model	loss	accuracy	val_loss	val_accuracy
Inception	0.8842	0.6431	1.3022	0.5078
Resnet	0.0387	0.9877	2.2020	0.6248
VGG	1.1726	0.5658	1.4930	0.4375

Second attempt

```
N = 35887
W = 75
H = 75
  # this could also be the output a different Keras model or layer

base_model = InceptionV3(weights = 'imagenet', include_top = False, input_shape=(W, H, 3))
  # add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
  # let's add a fully-connected layer
x = tf.keras.layers.Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
  # and a logistic layer
predictions = Dense(7, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

for layer in model.layers:
  layer.trainable = True
mycallbacks = [                                        # define the callbacks
      keras.callbacks.ReduceLROnPlateau(                     # callback to reduce learning rate
          monitor='val_loss',
          factor = 0.1,
          patience=3,
          verbose=1,
          min_lr=1e-8
      ),
      keras.callbacks.EarlyStopping(                         # callback to stop training if no improvement in validation_loss.
        monitor='val_loss',
        verbose=1,
        patience= 5,
        restore_best_weights=True
        ),
      keras.callbacks.ModelCheckpoint(                    # callback to save the best model if no improvement in validation_AUC.
        monitor='val_accuracy',
        filepath= "/content/drive/My Drive/Facial/checkpoint/Inception_Dropout",
        save_best_only=True,
        mode= 'max',
        verbose=1
        )
    ]
model.compile(optimizer   = Adam(lr = 1e-3, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08, decay = 0.0),loss='categorical_crossentropy', metrics=['accuracy'])

```

loss: 0.5864 - accuracy: 0.7921 - val_loss: 0.9731 - val_accuracy: 0.6743






```
weightsFile = '/content/drive/My Drive/Facial/Inception_Dropout.h5'
model.load_weights(weightsFile)
evaluate_results = model.evaluate(dataLoader.get_test_generator(),
                                                  workers= 6)
```

```
29/29 [==============================] - 3s 105ms/step - loss: 0.9269 - accuracy: 0.6626
```

Conclusion:

Training Accuracy:

Validation Accuracy: 67%

Test Accuracy: 66%