Personal Project_04_v10_test1_4conv-layer_run78_very advanced control 4_autorun

May 7, 2025

[1]:

```python
from tensorflow.keras.callbacks import LearningRateScheduler
from sklearn.metrics import classification_report, confusion_matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.image as mpimg
import tensorflow as tf
import os


class EarlyStoppingCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        train_accuracy = logs.get('accuracy')
        val_accuracy = logs.get('val_accuracy')
        if train_accuracy >= desired_train_accuracy and val_accuracy >= desired_val_accuracy:
            self.model.stop_training = True
            print("Reached desired accuracy so cancelling training!")


# target accuracy values:
desired_train_accuracy = 0.89
desired_val_accuracy = 0.89

# maximum trial number:
trial_num = 50

# maximum possible epoch:
epochs = 20


TRAIN_ACC=0.1
VAL_ACC=0.1
try_num = 1
condition = True
```

```python
while (try_num<trial_num and condition==True):
    # DOE factors:
    learning_rate = 0.0005
    dropout_value = 0.2
    # n-conv_layers = 4
    n_units_last_layer = 4096
    n_filters_l1 = 32
    n_filters_l2 = 64

    # other factors:
    img_size = 130
    batch_size = 32
    validation_split = 0.1  # 10% for validation
    test_split = 0.00  # 0% for testing
    shuffle_buffer_size = 1000
    seed_num = 101
    desired_accuracy = 0.99  # it should be active if EarlyStoppingCallback is activated
    loss = 'binary_crossentropy'
    #optimizer = tf.keras.optimizers.RMSprop(learning_rate=learning_rate)
    optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
    metrics = ['accuracy']
    f_mode = 'nearest'  # fill_mode in image augmentation


    #DATA_DIR = "D:\\CS online courses\\Free DataSets\\Free Images\\Easier portrait images_GPU_03"
    DATA_DIR = "/Users/hossein/Downloads/Easier portrait images_GPU_03"


    # Subdirectories for each class
    data_dir_woman = os.path.join(DATA_DIR, 'woman')
    data_dir_man = os.path.join(DATA_DIR, 'man')

    image_size = (img_size, img_size)  # Resize images to this size

    # Load train dataset (excluding validation & test set):
    train_dataset = tf.keras.utils.image_dataset_from_directory(
        directory = DATA_DIR,
        image_size = image_size,
        batch_size = batch_size,
        label_mode='binary',
        validation_split = validation_split + test_split,  # Total split for val + test
        subset = "training",
        seed = seed_num
    )

    # Load validation dataset
    val_dataset = tf.keras.utils.image_dataset_from_directory(
        directory = DATA_DIR,
        image_size = image_size,
        batch_size = batch_size,
        label_mode='binary',
        validation_split = validation_split + test_split,
        subset = "validation",
        seed = seed_num
    )

    # Further manually split validation dataset to extract test dataset
    val_batches = tf.data.experimental.cardinality(val_dataset)
    # Compute test dataset size (number of batches)
    test_size = round(val_batches.numpy() * (test_split / (validation_split + test_split)))
    # Split validation dataset into validation and test subsets
    test_dataset = val_dataset.take(test_size)
    val_dataset = val_dataset.skip(test_size)

    # Optimize for performance
    AUTOTUNE = tf.data.AUTOTUNE
    training_dataset = train_dataset.cache().shuffle(shuffle_buffer_size).prefetch(buffer_size = AUTOTUNE)
    validation_dataset = val_dataset.cache().prefetch(buffer_size = AUTOTUNE)
    test_dataset = test_dataset.cache().prefetch(buffer_size = AUTOTUNE)

    # Get the first batch of images and labels
    for images, labels in training_dataset.take(1):
            example_batch_images = images
            example_batch_labels = labels

    max_pixel = np.max(example_batch_images)

    def scheduler(epoch, lr):
        if epoch < 10:
            if epoch % 5 == 0 and epoch > 0:
                return lr / 1
            return lr
        elif epoch < 15:
            if epoch % 5 == 0 and epoch > 0:
                return lr / 2
            return lr
        elif epoch < 30:
```

```
        if epoch % 5 == 0 and epoch > 0:
            return lr / 1
        return lr
    return lr
lr_callback = LearningRateScheduler(scheduler)

# augmentation_model
def augment_model():
    augmentation_model = tf.keras.Sequential([
        # Specify the input shape.
        tf.keras.Input(shape = (img_size, img_size, 3)),
        tf.keras.layers.RandomFlip("horizontal"),
        tf.keras.layers.RandomRotation(0.1, fill_mode = f_mode),
        #tf.keras.layers.RandomTranslation(0.1, 0.1, fill_mode = f_mode),
        #tf.keras.layers.RandomZoom(0.1, fill_mode=f_mode)
        ])
    return augmentation_model

def create_and_compile_model():
    augmentation_layers = augment_model()
    model = tf.keras.Sequential([
        # Note: the input shape is the desired size of the image: 150x150␣
↪with 3 bytes for color
        tf.keras.layers.InputLayer(shape = (img_size, img_size, 3)),
        augmentation_layers,
        tf.keras.layers.Rescaling(1./255),
        #####    CONV_LAYER_1:     #####
        tf.keras.layers.Conv2D(n_filters_l1, (4, 4), activation = 'linear'),
        tf.keras.layers.MaxPooling2D(2, 2),
        #####    CONV_LAYER_2:     #####
        tf.keras.layers.Conv2D(n_filters_l2, (3, 3), activation = 'relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        #####    CONV_LAYER_3:     #####
        tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        #####    CONV_LAYER_4:     #####
        tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dropout(dropout_value),
        #####    BEFORE_LAST_LAYER:     #####
        tf.keras.layers.Dense(n_units_last_layer, activation = 'relu'),
        # It will contain a value from 0-1 where 0 for the class 'female'␣
↪and 1 for the 'male'
        tf.keras.layers.Dense(1, activation = 'sigmoid')])
    model.compile(
        loss = loss,
                optimizer = optimizer,
                metrics = metrics
            )
            return model


    # Create the compiled but untrained model
    def reset_weights(model):
        for layer in model.layers:
            if hasattr(layer, 'kernel_initializer'):
                layer.kernel.assign(layer.kernel_initializer(layer.kernel.
↪shape))
            if hasattr(layer, 'bias_initializer'):
                layer.bias.assign(layer.bias_initializer(layer.bias.shape))

    model = create_and_compile_model()
    reset_weights(model)  # Reset all layer weights
    training_history = model.fit(training_dataset,
                                 epochs=epochs,
                                 validation_data=validation_dataset,
                                 callbacks=[lr_callback,␣
↪EarlyStoppingCallback()],
                                 verbose=1)
    result_history = pd.DataFrame(model.history.history)
    TRAIN_ACC = result_history['accuracy'].iloc[-1]
    print(f"Current training accuracy: {TRAIN_ACC}")
    VAL_ACC = result_history['val_accuracy'].iloc[-1]
    print(f"Current validation accuracy: {VAL_ACC}")
    # Restart script
    print("Reseting all weights...")
    print(f'Current number of trials: {try_num}')
    try_num += 1
    result_history[['loss', 'val_loss']].plot(figsize=(5, 3))
    result_history[['accuracy', 'val_accuracy']].plot(figsize=(5, 3))
    plt.show()
    print(model.metrics_names)
    print(model.evaluate(validation_dataset))
    y_true = np.concatenate([y.numpy() for _, y in validation_dataset])
    y_pred_prob = model.predict(validation_dataset)
    # Convert probabilities to class labels (0:Female or 1:Male)
    y_pred = (y_pred_prob > 0.5).astype(int).flatten()
    print("Classification Report:\n", classification_report(y_true, y_pred,␣
↪target_names=['Female', 'Male']))

    if (TRAIN_ACC>=desired_train_accuracy and VAL_ACC>=desired_val_accuracy):
        condition = False
        model.save('trained_model_run78_very_advanced_control.h5')
```

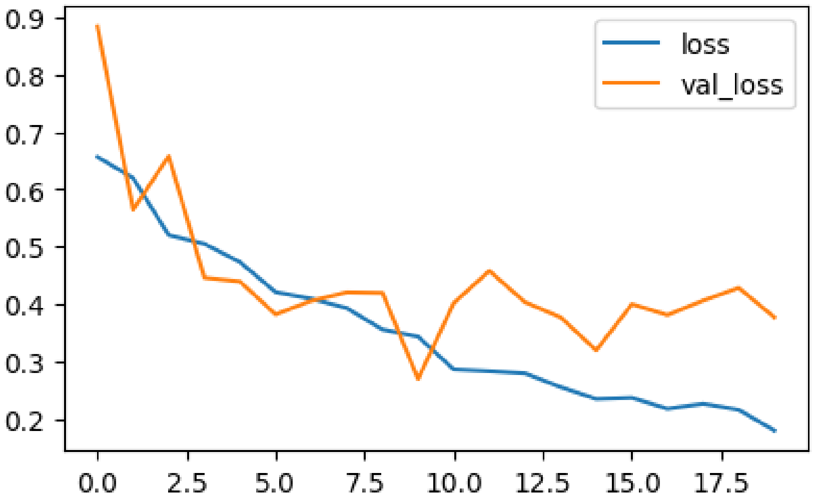
```python
result_history.head(15)
```

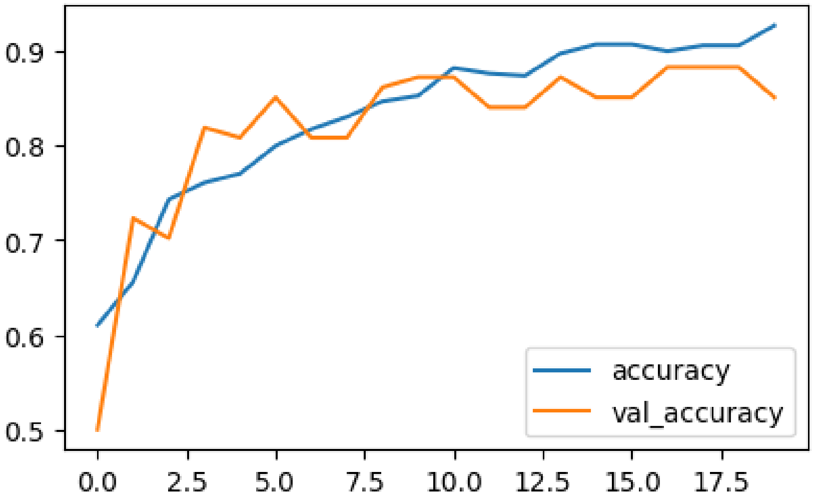
```
Found 943 files belonging to 2 classes.
Using 849 files for training.
Found 943 files belonging to 2 classes.
Using 94 files for validation.
Epoch 1/20
```

```
2025-05-07 12:22:49.495620: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
27/27            3s 98ms/step -
accuracy: 0.5822 - loss: 0.6885 - val_accuracy: 0.5000 - val_loss: 0.8843 -
learning_rate: 5.0000e-04
Epoch 2/20
27/27            3s 95ms/step -
accuracy: 0.6371 - loss: 0.6461 - val_accuracy: 0.7234 - val_loss: 0.5644 -
learning_rate: 5.0000e-04
Epoch 3/20
27/27            3s 98ms/step -
accuracy: 0.7372 - loss: 0.5271 - val_accuracy: 0.7021 - val_loss: 0.6582 -
learning_rate: 5.0000e-04
Epoch 4/20
27/27            3s 95ms/step -
accuracy: 0.7479 - loss: 0.5298 - val_accuracy: 0.8191 - val_loss: 0.4458 -
learning_rate: 5.0000e-04
Epoch 5/20
27/27            3s 97ms/step -
accuracy: 0.7642 - loss: 0.4580 - val_accuracy: 0.8085 - val_loss: 0.4392 -
learning_rate: 5.0000e-04
Epoch 6/20
27/27            3s 96ms/step -
accuracy: 0.7932 - loss: 0.4415 - val_accuracy: 0.8511 - val_loss: 0.3819 -
learning_rate: 5.0000e-04
Epoch 7/20
27/27            3s 97ms/step -
accuracy: 0.8219 - loss: 0.4030 - val_accuracy: 0.8085 - val_loss: 0.4056 -
learning_rate: 5.0000e-04
Epoch 8/20
27/27            3s 96ms/step -
accuracy: 0.8265 - loss: 0.4040 - val_accuracy: 0.8085 - val_loss: 0.4204 -
learning_rate: 5.0000e-04
Epoch 9/20
27/27            3s 96ms/step -
accuracy: 0.8598 - loss: 0.3465 - val_accuracy: 0.8617 - val_loss: 0.4194 -
learning_rate: 5.0000e-04
Epoch 10/20
```

```
27/27           3s 96ms/step -
accuracy: 0.8604 - loss: 0.3343 - val_accuracy: 0.8723 - val_loss: 0.2692 -
learning_rate: 5.0000e-04
Epoch 11/20
27/27           3s 97ms/step -
accuracy: 0.8968 - loss: 0.2748 - val_accuracy: 0.8723 - val_loss: 0.4017 -
learning_rate: 2.5000e-04
Epoch 12/20
27/27           3s 96ms/step -
accuracy: 0.8852 - loss: 0.2814 - val_accuracy: 0.8404 - val_loss: 0.4579 -
learning_rate: 2.5000e-04
Epoch 13/20
27/27           3s 96ms/step -
accuracy: 0.8547 - loss: 0.3136 - val_accuracy: 0.8404 - val_loss: 0.4032 -
learning_rate: 2.5000e-04
Epoch 14/20
27/27           3s 96ms/step -
accuracy: 0.9058 - loss: 0.2361 - val_accuracy: 0.8723 - val_loss: 0.3769 -
learning_rate: 2.5000e-04
Epoch 15/20
27/27           3s 97ms/step -
accuracy: 0.8999 - loss: 0.2315 - val_accuracy: 0.8511 - val_loss: 0.3192 -
learning_rate: 2.5000e-04
Epoch 16/20
27/27           3s 97ms/step -
accuracy: 0.9107 - loss: 0.2446 - val_accuracy: 0.8511 - val_loss: 0.3997 -
learning_rate: 2.5000e-04
Epoch 17/20
27/27           3s 98ms/step -
accuracy: 0.9136 - loss: 0.2122 - val_accuracy: 0.8830 - val_loss: 0.3810 -
learning_rate: 2.5000e-04
Epoch 18/20
27/27           3s 97ms/step -
accuracy: 0.9086 - loss: 0.2189 - val_accuracy: 0.8830 - val_loss: 0.4060 -
learning_rate: 2.5000e-04
Epoch 19/20
27/27           3s 97ms/step -
accuracy: 0.9036 - loss: 0.2123 - val_accuracy: 0.8830 - val_loss: 0.4281 -
learning_rate: 2.5000e-04
Epoch 20/20
27/27           3s 97ms/step -
accuracy: 0.9264 - loss: 0.1941 - val_accuracy: 0.8511 - val_loss: 0.3769 -
learning_rate: 2.5000e-04
Current training accuracy: 0.9269729256629944
Current validation accuracy: 0.8510638475418091
Reseting all weights…
Current number of trials: 1
```





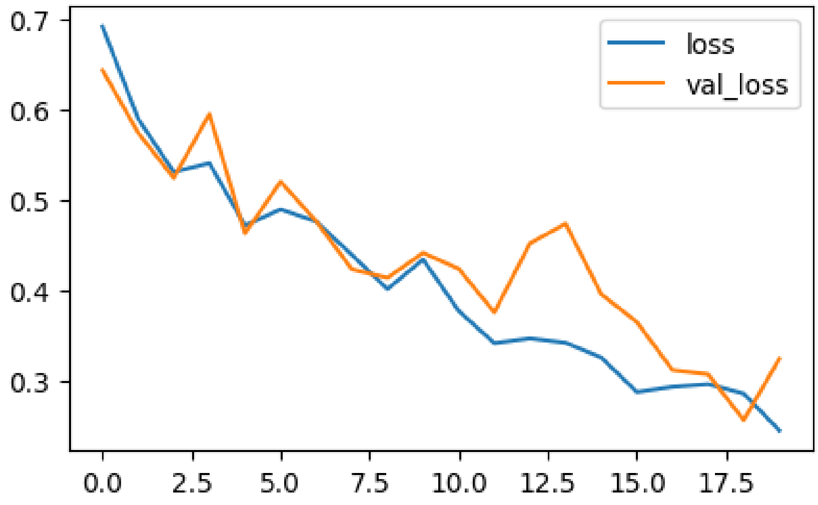
```
['loss', 'compile_metrics']
3/3              0s 25ms/step -
accuracy: 0.8318 - loss: 0.3831
[0.37694793939590454, 0.8510638475418091]
3/3              0s 36ms/step
Classification Report:
               precision    recall  f1-score   support

      Female       0.81      0.85      0.83        41
        Male       0.88      0.85      0.87        53

    accuracy                           0.85        94
```

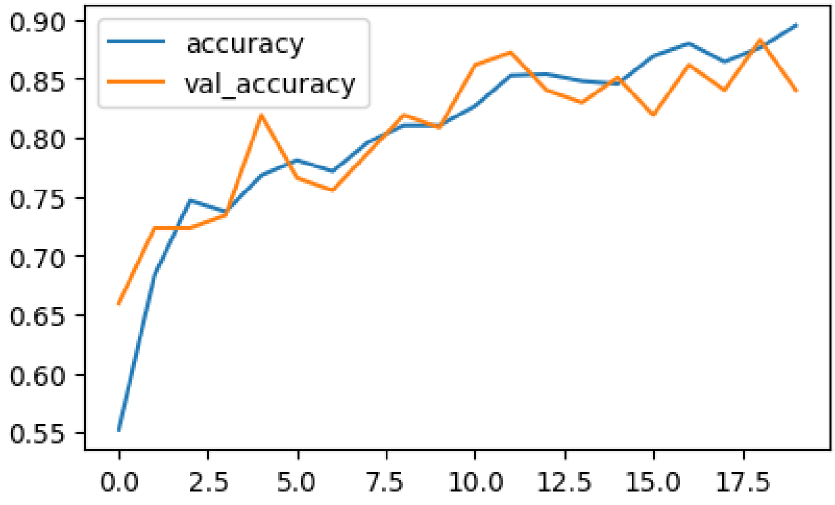
```
   macro avg       0.85      0.85      0.85        94
weighted avg       0.85      0.85      0.85        94

Found 943 files belonging to 2 classes.
Using 849 files for training.
Found 943 files belonging to 2 classes.
Using 94 files for validation.
```

```
2025-05-07 12:23:42.466022: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Epoch 1/20
27/27            3s 100ms/step -
accuracy: 0.5201 - loss: 0.7306 - val_accuracy: 0.6596 - val_loss: 0.6441 -
learning_rate: 5.0000e-04
Epoch 2/20
27/27            3s 96ms/step -
accuracy: 0.6903 - loss: 0.5974 - val_accuracy: 0.7234 - val_loss: 0.5745 -
learning_rate: 5.0000e-04
Epoch 3/20
27/27            3s 98ms/step -
accuracy: 0.7712 - loss: 0.5243 - val_accuracy: 0.7234 - val_loss: 0.5244 -
learning_rate: 5.0000e-04
Epoch 4/20
27/27            3s 96ms/step -
accuracy: 0.7126 - loss: 0.5493 - val_accuracy: 0.7340 - val_loss: 0.5952 -
learning_rate: 5.0000e-04
Epoch 5/20
27/27            3s 96ms/step -
accuracy: 0.7730 - loss: 0.4587 - val_accuracy: 0.8191 - val_loss: 0.4628 -
learning_rate: 5.0000e-04
Epoch 6/20
27/27            3s 96ms/step -
accuracy: 0.7853 - loss: 0.4603 - val_accuracy: 0.7660 - val_loss: 0.5203 -
learning_rate: 5.0000e-04
Epoch 7/20
27/27            3s 96ms/step -
accuracy: 0.7849 - loss: 0.4686 - val_accuracy: 0.7553 - val_loss: 0.4762 -
learning_rate: 5.0000e-04
Epoch 8/20
27/27            3s 96ms/step -
accuracy: 0.7938 - loss: 0.4376 - val_accuracy: 0.7872 - val_loss: 0.4229 -
learning_rate: 5.0000e-04
Epoch 9/20
27/27            3s 100ms/step -
accuracy: 0.8030 - loss: 0.4236 - val_accuracy: 0.8191 - val_loss: 0.4136 -
learning_rate: 5.0000e-04
Epoch 10/20
27/27            3s 96ms/step -
```

```
accuracy: 0.8022 - loss: 0.4394 - val_accuracy: 0.8085 - val_loss: 0.4412 -
learning_rate: 5.0000e-04
Epoch 11/20
27/27           3s 97ms/step -
accuracy: 0.8375 - loss: 0.3721 - val_accuracy: 0.8617 - val_loss: 0.4235 -
learning_rate: 2.5000e-04
Epoch 12/20
27/27           3s 96ms/step -
accuracy: 0.8414 - loss: 0.3441 - val_accuracy: 0.8723 - val_loss: 0.3752 -
learning_rate: 2.5000e-04
Epoch 13/20
27/27           3s 96ms/step -
accuracy: 0.8530 - loss: 0.3435 - val_accuracy: 0.8404 - val_loss: 0.4519 -
learning_rate: 2.5000e-04
Epoch 14/20
27/27           3s 96ms/step -
accuracy: 0.8327 - loss: 0.3620 - val_accuracy: 0.8298 - val_loss: 0.4735 -
learning_rate: 2.5000e-04
Epoch 15/20
27/27           3s 98ms/step -
accuracy: 0.8462 - loss: 0.3397 - val_accuracy: 0.8511 - val_loss: 0.3953 -
learning_rate: 2.5000e-04
Epoch 16/20
27/27           3s 97ms/step -
accuracy: 0.8662 - loss: 0.2780 - val_accuracy: 0.8191 - val_loss: 0.3644 -
learning_rate: 2.5000e-04
Epoch 17/20
27/27           3s 97ms/step -
accuracy: 0.8829 - loss: 0.3070 - val_accuracy: 0.8617 - val_loss: 0.3110 -
learning_rate: 2.5000e-04
Epoch 18/20
27/27           3s 97ms/step -
accuracy: 0.8687 - loss: 0.2847 - val_accuracy: 0.8404 - val_loss: 0.3064 -
learning_rate: 2.5000e-04
Epoch 19/20
27/27           3s 97ms/step -
accuracy: 0.8816 - loss: 0.2991 - val_accuracy: 0.8830 - val_loss: 0.2557 -
learning_rate: 2.5000e-04
Epoch 20/20
27/27           3s 97ms/step -
accuracy: 0.8928 - loss: 0.2598 - val_accuracy: 0.8404 - val_loss: 0.3236 -
learning_rate: 2.5000e-04
Current training accuracy: 0.8951708078384399
Current validation accuracy: 0.8404255509376526
Reseting all weights…
Current number of trials: 2
```





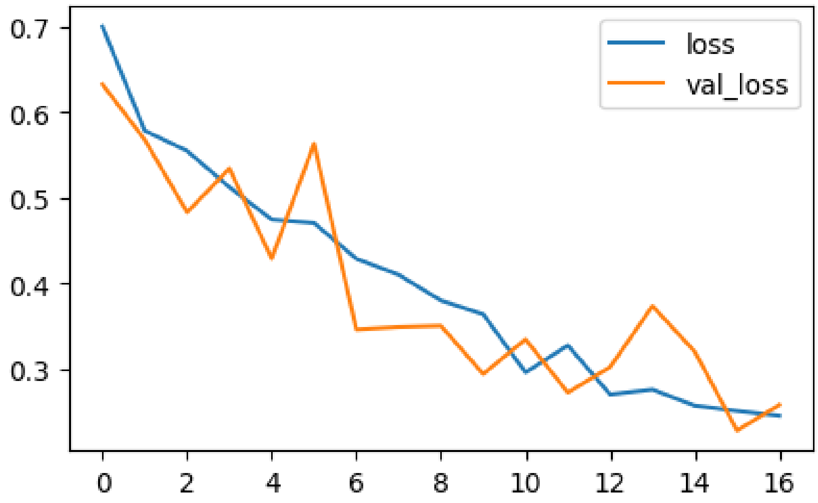
```
['loss', 'compile_metrics']
3/3              0s 24ms/step -
accuracy: 0.8343 - loss: 0.3530
[0.32360365986824036, 0.8404255509376526]
3/3              0s 37ms/step
Classification Report:
               precision    recall  f1-score   support

      Female       0.78      0.88      0.83        41
        Male       0.90      0.81      0.85        53

    accuracy                           0.84        94
```

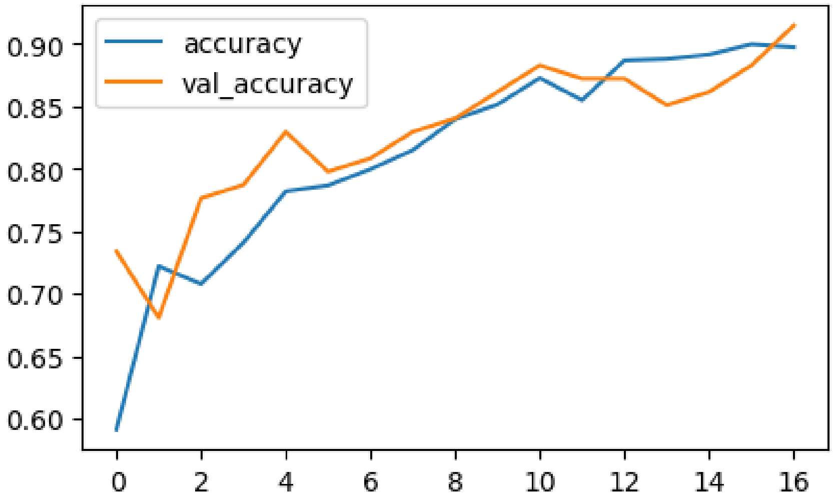
```
   macro avg       0.84      0.84      0.84        94
weighted avg       0.85      0.84      0.84        94

Found 943 files belonging to 2 classes.
Using 849 files for training.
Found 943 files belonging to 2 classes.
Using 94 files for validation.
```

```
2025-05-07 12:24:35.960210: I tensorflow/core/framework/local_rendezvous.cc:405]
Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
```

```
Epoch 1/20
27/27            3s 102ms/step -
accuracy: 0.5591 - loss: 0.7182 - val_accuracy: 0.7340 - val_loss: 0.6327 -
learning_rate: 5.0000e-04
Epoch 2/20
27/27            3s 97ms/step -
accuracy: 0.7219 - loss: 0.5909 - val_accuracy: 0.6809 - val_loss: 0.5680 -
learning_rate: 5.0000e-04
Epoch 3/20
27/27            3s 97ms/step -
accuracy: 0.6785 - loss: 0.5778 - val_accuracy: 0.7766 - val_loss: 0.4832 -
learning_rate: 5.0000e-04
Epoch 4/20
27/27            3s 97ms/step -
accuracy: 0.7385 - loss: 0.5046 - val_accuracy: 0.7872 - val_loss: 0.5345 -
learning_rate: 5.0000e-04
Epoch 5/20
27/27            3s 97ms/step -
accuracy: 0.7804 - loss: 0.4804 - val_accuracy: 0.8298 - val_loss: 0.4290 -
learning_rate: 5.0000e-04
Epoch 6/20
27/27            3s 98ms/step -
accuracy: 0.7826 - loss: 0.4872 - val_accuracy: 0.7979 - val_loss: 0.5635 -
learning_rate: 5.0000e-04
Epoch 7/20
27/27            3s 98ms/step -
accuracy: 0.7955 - loss: 0.4416 - val_accuracy: 0.8085 - val_loss: 0.3463 -
learning_rate: 5.0000e-04
Epoch 8/20
27/27            3s 97ms/step -
accuracy: 0.8053 - loss: 0.4130 - val_accuracy: 0.8298 - val_loss: 0.3492 -
learning_rate: 5.0000e-04
Epoch 9/20
27/27            3s 97ms/step -
accuracy: 0.8544 - loss: 0.3677 - val_accuracy: 0.8404 - val_loss: 0.3506 -
learning_rate: 5.0000e-04
Epoch 10/20
27/27            3s 97ms/step -
```

```
accuracy: 0.8611 - loss: 0.3603 - val_accuracy: 0.8617 - val_loss: 0.2943 -
learning_rate: 5.0000e-04
Epoch 11/20
27/27             3s 97ms/step -
accuracy: 0.8438 - loss: 0.3434 - val_accuracy: 0.8830 - val_loss: 0.3348 -
learning_rate: 2.5000e-04
Epoch 12/20
27/27             3s 98ms/step -
accuracy: 0.8594 - loss: 0.3544 - val_accuracy: 0.8723 - val_loss: 0.2727 -
learning_rate: 2.5000e-04
Epoch 13/20
27/27             3s 99ms/step -
accuracy: 0.8869 - loss: 0.2528 - val_accuracy: 0.8723 - val_loss: 0.3023 -
learning_rate: 2.5000e-04
Epoch 14/20
27/27             3s 98ms/step -
accuracy: 0.8975 - loss: 0.2524 - val_accuracy: 0.8511 - val_loss: 0.3737 -
learning_rate: 2.5000e-04
Epoch 15/20
27/27             3s 97ms/step -
accuracy: 0.8742 - loss: 0.2777 - val_accuracy: 0.8617 - val_loss: 0.3205 -
learning_rate: 2.5000e-04
Epoch 16/20
27/27             3s 100ms/step -
accuracy: 0.9109 - loss: 0.2373 - val_accuracy: 0.8830 - val_loss: 0.2283 -
learning_rate: 2.5000e-04
Epoch 17/20
27/27             0s 95ms/step -
accuracy: 0.9242 - loss: 0.2151Reached desired accuracy so cancelling training!
27/27             3s 98ms/step -
accuracy: 0.9233 - loss: 0.2161 - val_accuracy: 0.9149 - val_loss: 0.2582 -
learning_rate: 2.5000e-04
Current training accuracy: 0.8975265026092529
Current validation accuracy: 0.914893627166748
Reseting all weights…
Current number of trials: 3
```





```
['loss', 'compile_metrics']
3/3               0s 25ms/step -
accuracy: 0.8989 - loss: 0.2818
[0.25818532705307007, 0.914893627166748]
WARNING:tensorflow:5 out of the last 7 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x17f228fe0> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
```

```
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/3            0s
47ms/stepWARNING:tensorflow:6 out of the last 9 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x17f228fe0> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
3/3            0s 37ms/step
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

```
Classification Report:
              precision    recall  f1-score   support

      Female       0.85      0.98      0.91        41
        Male       0.98      0.87      0.92        53

    accuracy                           0.91        94
   macro avg       0.91      0.92      0.91        94
weighted avg       0.92      0.91      0.92        94
```

```
[1]:     accuracy      loss  val_accuracy  val_loss  learning_rate
     0   0.591284  0.700387      0.734043  0.632742        0.00050
     1   0.722026  0.578647      0.680851  0.568016        0.00050
     2   0.707892  0.555076      0.776596  0.483235        0.00050
     3   0.740872  0.512653      0.787234  0.534464        0.00050
     4   0.782097  0.474838      0.829787  0.429026        0.00050
     5   0.786808  0.470764      0.797872  0.563464        0.00050
     6   0.799764  0.429010      0.808511  0.346285        0.00050
     7   0.815077  0.410316      0.829787  0.349219        0.00050
     8   0.839812  0.380174      0.840426  0.350584        0.00050
     9   0.851590  0.364124      0.861702  0.294300        0.00050
     10  0.872792  0.296062      0.882979  0.334795        0.00025
     11  0.855124  0.327675      0.872340  0.272722        0.00025
     12  0.886926  0.270424      0.872340  0.302330        0.00025
     13  0.888104  0.276107      0.851064  0.373744        0.00025
     14  0.891637  0.257347      0.861702  0.320483        0.00025
```

[2]:

```
from tensorflow.keras.models import Model
from tensorflow.keras.utils import load_img, img_to_array

img_size = img_size
model = tf.keras.models.load_model("trained_model_run78_very_advanced_control.
 ↪h5")

# Load your personal image if you are interested to predict:
#your_image_path = "D:\\Hossein's desktop files in Microsoft Studio␣
 ↪Laptop\\Personal Photos\\Hossein_10.jpg"
your_image_path = "/Users/hossein/Downloads/Hossein.png"

img = load_img(your_image_path, target_size=(img_size, img_size))
final_img = img_to_array(img)
# Adding a batch dimension:
final_img = np.expand_dims(final_img, axis=0)
prediction = model.predict(final_img)
result = "Female" if prediction > 0.5 else "Male"
if result=="Female":
    confidence = (model.predict(final_img)[0][0])*100
else:
    confidence = (1-model.predict(final_img)[0][0])*100
print(f"Prediction result: {result} (confidence= {confidence:.2f} %)")

# Visualize CNN Layers
dummy_input = np.random.rand(1, img_size, img_size, 3)  # Create random input␣
 ↪with correct shape
model.predict(dummy_input)  # Call the model to establish input tensors
visualization_model = Model(inputs=model.inputs, outputs=[layer.output for␣
 ↪layer in model.layers])
successive_feature_maps = visualization_model.predict(final_img)
layer_names = [layer.name for layer in model.layers]

for layer_name, feature_map in zip(layer_names, successive_feature_maps):
    if len(feature_map.shape) == 4:  # Only visualize conv/maxpool layers
        n_features = feature_map.shape[-1]  # Number of filters
        size = feature_map.shape[1]  # Feature map size
        display_grid = np.zeros((size, size * n_features))

        for i in range(n_features):
            x = feature_map[0, :, :, i]
            x -= x.mean()
            x /= (x.std() + 1e-8)  # Normalize
            x *= 64
            x += 128
            x = np.clip(x, 0, 255).astype('uint8')  # Convert to image format
            display_grid[:, i * size: (i + 1) * size] = x
        scale = 20. / n_features
        plt.figure(figsize=(scale * n_features, scale))
        plt.title(layer_name)
        plt.grid(False)
        plt.imshow(display_grid, aspect='auto', cmap='cividis')
        plt.show()
```

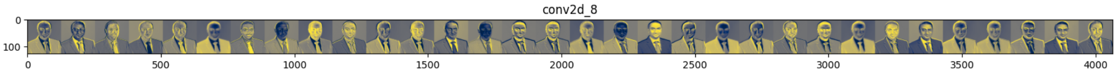
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be
built. `model.compile_metrics` will be empty until you train or evaluate the
model.
```

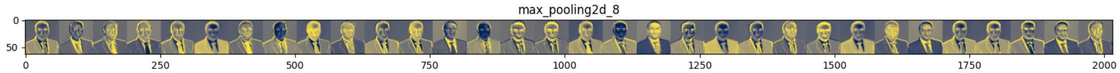
```
1/1               0s 38ms/step
1/1               0s 16ms/step
Prediction result: Male (confidence= 91.61 %)
1/1               0s 15ms/step
```

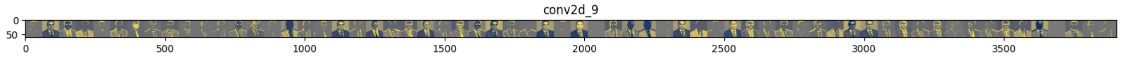
```
/opt/anaconda3/envs/mytfenv/lib/python3.12/site-
packages/keras/src/models/functional.py:237: UserWarning: The structure of
`inputs` doesn't match the expected structure.
Expected: ['input_layer_5']
Received: inputs=Tensor(shape=(1, 130, 130, 3))
  warnings.warn(msg)
```

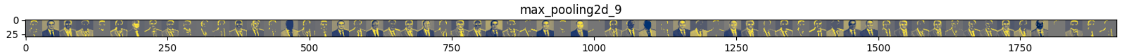
```
1/1               0s 150ms/step
```

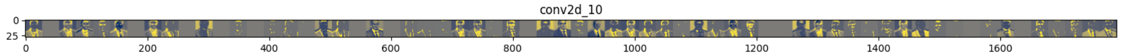
rescaling_2
0
20
40
60
80
100
120
0
50
100
150
200
250
300
350

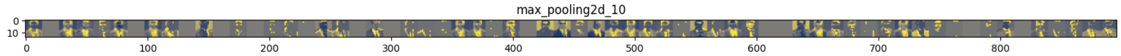

conv2d_8
0
100
0
500
1000
1500
2000
2500
3000
3500
4000

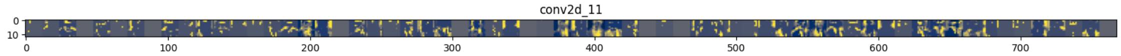

max_pooling2d_8
0
50
0
250
500
750
1000
1250
1500
1750
2000

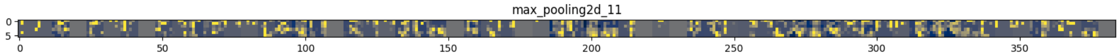

conv2d_9
0
50
0
500
1000
1500
2000
2500
3000
3500


max_pooling2d_9
0
25
0
250
500
750
1000
1250
1500
1750


conv2d_10
0
25
0
200
400
600
800
1000
1200
1400
1600


max_pooling2d_10
0
10
0
100
200
300
400
500
600
700
800


conv2d_11
0
10
0
100
200
300
400
500
600
700


max_pooling2d_11
0
5
0
50
100
150
200
250
300
350

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```