Personal Project_04_v10_test1_2conv-layer_run042_advanced contro_autorun

May 5, 2025

[1]:

```
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


ACC=0.1
try_num = 1

while (ACC<0.88 and try_num<10):
    # DOE factors:
    learning_rate = 0.005
    dropout_value = 0.5
    # n_conv_layers = 2
    n_units_last_layer = 1024
    n_filters_l1 = 8
    n_filters_l2 = 16

    # 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']
```

```
epochs = 29
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)

# Reduce LR every 10 epochs (Learning rate decay factor)
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 / 1
        return lr
    elif epoch < 30:
        if epoch % 5 == 0 and epoch > 0:
            return lr / 1
        return lr
    else:
        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),
            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],
                                 verbose=2)
    result_history = pd.DataFrame(model.history.history)
    ACC = result_history['val_accuracy'].iloc[-1]
    print(f"Current validation accuracy: {ACC}")
    model.save('trained_model_run42_advanced_control.h5')
    # 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']))

result_history.head(15)
```

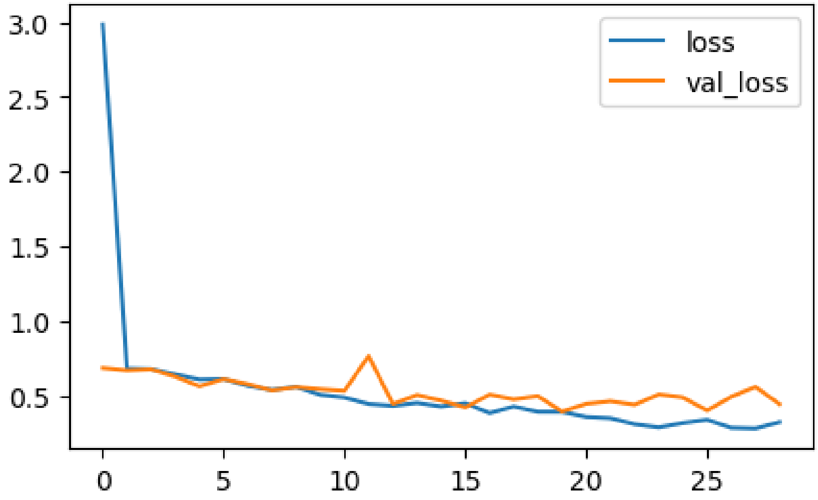
```
Found 943 files belonging to 2 classes.
Using 849 files for training.
Found 943 files belonging to 2 classes.
Using 94 files for validation.
Epoch 1/29
27/27 - 4s - 137ms/step - accuracy: 0.5241 - loss: 2.9836 - val_accuracy: 0.5851
- val_loss: 0.6829 - learning_rate: 0.0050
Epoch 2/29
27/27 - 2s - 70ms/step - accuracy: 0.6019 - loss: 0.6791 - val_accuracy: 0.5851
- val_loss: 0.6669 - learning_rate: 0.0050
Epoch 3/29
27/27 - 2s - 68ms/step - accuracy: 0.5901 - loss: 0.6753 - val_accuracy: 0.6702
- val_loss: 0.6742 - learning_rate: 0.0050
Epoch 4/29
27/27 - 2s - 69ms/step - accuracy: 0.6419 - loss: 0.6400 - val_accuracy: 0.6277
- val_loss: 0.6250 - learning_rate: 0.0050
Epoch 5/29
27/27 - 2s - 78ms/step - accuracy: 0.6678 - loss: 0.6058 - val_accuracy: 0.7340
- val_loss: 0.5599 - learning_rate: 0.0050
Epoch 6/29
27/27 - 2s - 72ms/step - accuracy: 0.6855 - loss: 0.6090 - val_accuracy: 0.7340
- val_loss: 0.6086 - learning_rate: 0.0050
Epoch 7/29
27/27 - 2s - 73ms/step - accuracy: 0.7102 - loss: 0.5634 - val_accuracy: 0.7766
- val_loss: 0.5745 - learning_rate: 0.0050
Epoch 8/29
27/27 - 2s - 76ms/step - accuracy: 0.7244 - loss: 0.5360 - val_accuracy: 0.7660
- val_loss: 0.5326 - learning_rate: 0.0050
Epoch 9/29
27/27 - 2s - 67ms/step - accuracy: 0.7256 - loss: 0.5567 - val_accuracy: 0.7766
- val_loss: 0.5552 - learning_rate: 0.0050
Epoch 10/29
27/27 - 2s - 70ms/step - accuracy: 0.7397 - loss: 0.5029 - val_accuracy: 0.7340
- val_loss: 0.5408 - learning_rate: 0.0050
Epoch 11/29
27/27 - 2s - 73ms/step - accuracy: 0.7703 - loss: 0.4850 - val_accuracy: 0.7553
- val_loss: 0.5296 - learning_rate: 0.0050
Epoch 12/29
27/27 - 2s - 84ms/step - accuracy: 0.7951 - loss: 0.4414 - val_accuracy: 0.6702
```

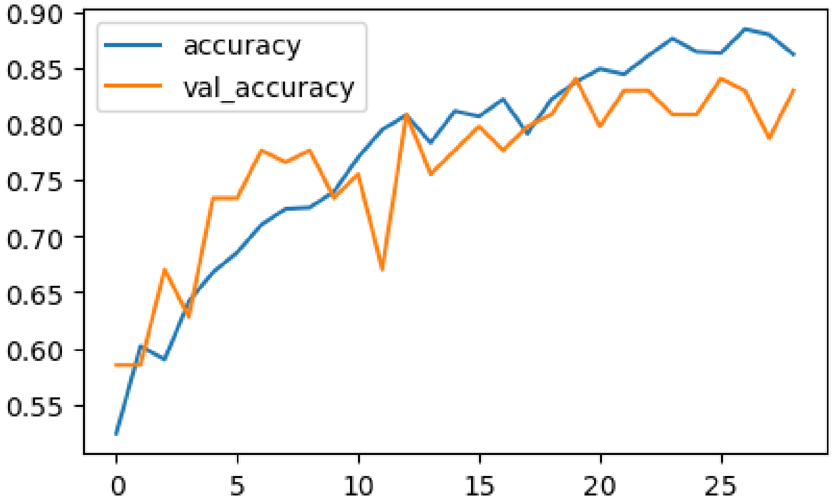
```
- val_loss: 0.7635 - learning_rate: 0.0050
Epoch 13/29
27/27 - 2s - 75ms/step - accuracy: 0.8080 - loss: 0.4288 - val_accuracy: 0.8085
- val_loss: 0.4418 - learning_rate: 0.0050
Epoch 14/29
27/27 - 2s - 72ms/step - accuracy: 0.7833 - loss: 0.4486 - val_accuracy: 0.7553
- val_loss: 0.5014 - learning_rate: 0.0050
Epoch 15/29
27/27 - 2s - 72ms/step - accuracy: 0.8115 - loss: 0.4241 - val_accuracy: 0.7766
- val_loss: 0.4649 - learning_rate: 0.0050
Epoch 16/29
27/27 - 2s - 68ms/step - accuracy: 0.8068 - loss: 0.4465 - val_accuracy: 0.7979
- val_loss: 0.4181 - learning_rate: 0.0050
Epoch 17/29
27/27 - 2s - 67ms/step - accuracy: 0.8221 - loss: 0.3806 - val_accuracy: 0.7766
- val_loss: 0.5044 - learning_rate: 0.0050
Epoch 18/29
27/27 - 2s - 68ms/step - accuracy: 0.7915 - loss: 0.4238 - val_accuracy: 0.7979
- val_loss: 0.4731 - learning_rate: 0.0050
Epoch 19/29
27/27 - 2s - 67ms/step - accuracy: 0.8221 - loss: 0.3890 - val_accuracy: 0.8085
- val_loss: 0.4946 - learning_rate: 0.0050
Epoch 20/29
27/27 - 2s - 67ms/step - accuracy: 0.8375 - loss: 0.3886 - val_accuracy: 0.8404
- val_loss: 0.3906 - learning_rate: 0.0050
Epoch 21/29
27/27 - 2s - 67ms/step - accuracy: 0.8492 - loss: 0.3548 - val_accuracy: 0.7979
- val_loss: 0.4425 - learning_rate: 0.0050
Epoch 22/29
27/27 - 2s - 67ms/step - accuracy: 0.8445 - loss: 0.3464 - val_accuracy: 0.8298
- val_loss: 0.4605 - learning_rate: 0.0050
Epoch 23/29
27/27 - 2s - 67ms/step - accuracy: 0.8610 - loss: 0.3066 - val_accuracy: 0.8298
- val_loss: 0.4365 - learning_rate: 0.0050
Epoch 24/29
27/27 - 2s - 68ms/step - accuracy: 0.8763 - loss: 0.2850 - val_accuracy: 0.8085
- val_loss: 0.5054 - learning_rate: 0.0050
Epoch 25/29
27/27 - 2s - 67ms/step - accuracy: 0.8645 - loss: 0.3144 - val_accuracy: 0.8085
- val_loss: 0.4857 - learning_rate: 0.0050
Epoch 26/29
27/27 - 2s - 67ms/step - accuracy: 0.8634 - loss: 0.3359 - val_accuracy: 0.8404
- val_loss: 0.3977 - learning_rate: 0.0050
Epoch 27/29
27/27 - 2s - 68ms/step - accuracy: 0.8846 - loss: 0.2822 - val_accuracy: 0.8298
- val_loss: 0.4897 - learning_rate: 0.0050
Epoch 28/29
27/27 - 2s - 67ms/step - accuracy: 0.8799 - loss: 0.2776 - val_accuracy: 0.7872
```

```
- val_loss: 0.5563 - learning_rate: 0.0050
Epoch 29/29
27/27 - 2s - 67ms/step - accuracy: 0.8622 - loss: 0.3202 - val_accuracy: 0.8298
- val_loss: 0.4403 - learning_rate: 0.0050
```

```
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')`.
```

```
Current validation accuracy: 0.8297872543334961
Reseting all weights…
Current number of trials: 1
```





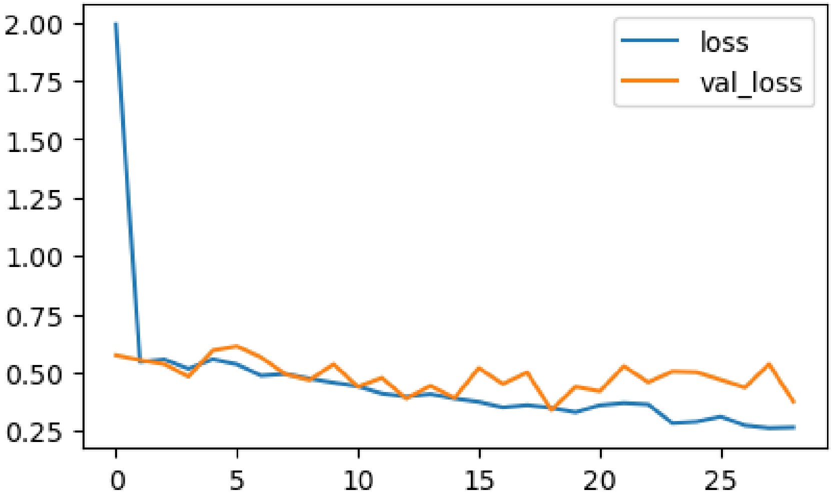
```
['loss', 'compile_metrics']
3/3             0s 17ms/step -
accuracy: 0.8172 - loss: 0.4643
[0.4402659833431244, 0.8297872543334961]
3/3             0s 49ms/step
Classification Report:
              precision    recall  f1-score   support

      Female       0.79      0.83      0.81        41
        Male       0.86      0.83      0.85        53

    accuracy                           0.83        94
   macro avg       0.83      0.83      0.83        94
weighted avg       0.83      0.83      0.83        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.
Epoch 1/29
27/27 - 4s - 139ms/step - accuracy: 0.6478 - loss: 1.9918 - val_accuracy: 0.7234
- val_loss: 0.5742 - learning_rate: 0.0050
Epoch 2/29
27/27 - 2s - 68ms/step - accuracy: 0.7102 - loss: 0.5456 - val_accuracy: 0.7872
- val_loss: 0.5530 - learning_rate: 0.0050
Epoch 3/29
27/27 - 2s - 67ms/step - accuracy: 0.7044 - loss: 0.5561 - val_accuracy: 0.7447
- val_loss: 0.5362 - learning_rate: 0.0050
Epoch 4/29
27/27 - 2s - 67ms/step - accuracy: 0.7562 - loss: 0.5157 - val_accuracy: 0.7553
- val_loss: 0.4819 - learning_rate: 0.0050
Epoch 5/29
27/27 - 2s - 67ms/step - accuracy: 0.7550 - loss: 0.5572 - val_accuracy: 0.7234
- val_loss: 0.5941 - learning_rate: 0.0050
Epoch 6/29
27/27 - 2s - 67ms/step - accuracy: 0.7385 - loss: 0.5349 - val_accuracy: 0.7979
- val_loss: 0.6125 - learning_rate: 0.0050
Epoch 7/29
27/27 - 2s - 66ms/step - accuracy: 0.7621 - loss: 0.4866 - val_accuracy: 0.7766
- val_loss: 0.5653 - learning_rate: 0.0050
Epoch 8/29
27/27 - 2s - 67ms/step - accuracy: 0.7715 - loss: 0.4939 - val_accuracy: 0.8085
- val_loss: 0.4927 - learning_rate: 0.0050
Epoch 9/29
27/27 - 2s - 67ms/step - accuracy: 0.7880 - loss: 0.4740 - val_accuracy: 0.7979
- val_loss: 0.4676 - learning_rate: 0.0050
```

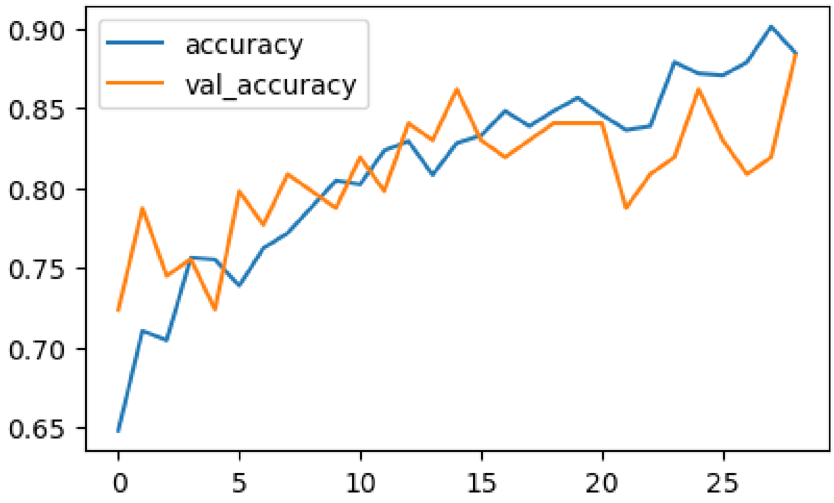
```
Epoch 10/29
27/27 - 2s - 68ms/step - accuracy: 0.8045 - loss: 0.4564 - val_accuracy: 0.7872
- val_loss: 0.5353 - learning_rate: 0.0050
Epoch 11/29
27/27 - 2s - 67ms/step - accuracy: 0.8021 - loss: 0.4411 - val_accuracy: 0.8191
- val_loss: 0.4392 - learning_rate: 0.0050
Epoch 12/29
27/27 - 2s - 67ms/step - accuracy: 0.8233 - loss: 0.4097 - val_accuracy: 0.7979
- val_loss: 0.4771 - learning_rate: 0.0050
Epoch 13/29
27/27 - 2s - 67ms/step - accuracy: 0.8292 - loss: 0.3962 - val_accuracy: 0.8404
- val_loss: 0.3880 - learning_rate: 0.0050
Epoch 14/29
27/27 - 2s - 67ms/step - accuracy: 0.8080 - loss: 0.4072 - val_accuracy: 0.8298
- val_loss: 0.4432 - learning_rate: 0.0050
Epoch 15/29
27/27 - 2s - 67ms/step - accuracy: 0.8280 - loss: 0.3881 - val_accuracy: 0.8617
- val_loss: 0.3907 - learning_rate: 0.0050
Epoch 16/29
27/27 - 2s - 67ms/step - accuracy: 0.8327 - loss: 0.3737 - val_accuracy: 0.8298
- val_loss: 0.5187 - learning_rate: 0.0050
Epoch 17/29
27/27 - 2s - 67ms/step - accuracy: 0.8481 - loss: 0.3500 - val_accuracy: 0.8191
- val_loss: 0.4510 - learning_rate: 0.0050
Epoch 18/29
27/27 - 2s - 67ms/step - accuracy: 0.8386 - loss: 0.3600 - val_accuracy: 0.8298
- val_loss: 0.5003 - learning_rate: 0.0050
Epoch 19/29
27/27 - 2s - 67ms/step - accuracy: 0.8481 - loss: 0.3481 - val_accuracy: 0.8404
- val_loss: 0.3387 - learning_rate: 0.0050
Epoch 20/29
27/27 - 2s - 67ms/step - accuracy: 0.8563 - loss: 0.3295 - val_accuracy: 0.8404
- val_loss: 0.4390 - learning_rate: 0.0050
Epoch 21/29
27/27 - 2s - 67ms/step - accuracy: 0.8457 - loss: 0.3594 - val_accuracy: 0.8404
- val_loss: 0.4201 - learning_rate: 0.0050
Epoch 22/29
27/27 - 2s - 67ms/step - accuracy: 0.8363 - loss: 0.3688 - val_accuracy: 0.7872
- val_loss: 0.5267 - learning_rate: 0.0050
Epoch 23/29
27/27 - 2s - 67ms/step - accuracy: 0.8386 - loss: 0.3630 - val_accuracy: 0.8085
- val_loss: 0.4585 - learning_rate: 0.0050
Epoch 24/29
27/27 - 2s - 67ms/step - accuracy: 0.8787 - loss: 0.2830 - val_accuracy: 0.8191
- val_loss: 0.5042 - learning_rate: 0.0050
Epoch 25/29
27/27 - 2s - 67ms/step - accuracy: 0.8716 - loss: 0.2890 - val_accuracy: 0.8617
- val_loss: 0.5012 - learning_rate: 0.0050
```

```
Epoch 26/29
27/27 - 2s - 67ms/step - accuracy: 0.8704 - loss: 0.3109 - val_accuracy: 0.8298
- val_loss: 0.4689 - learning_rate: 0.0050
Epoch 27/29
27/27 - 2s - 67ms/step - accuracy: 0.8787 - loss: 0.2732 - val_accuracy: 0.8085
- val_loss: 0.4356 - learning_rate: 0.0050
Epoch 28/29
27/27 - 2s - 68ms/step - accuracy: 0.9011 - loss: 0.2619 - val_accuracy: 0.8191
- val_loss: 0.5356 - learning_rate: 0.0050
Epoch 29/29
27/27 - 2s - 67ms/step - accuracy: 0.8846 - loss: 0.2646 - val_accuracy: 0.8830
- val_loss: 0.3757 - learning_rate: 0.0050
```

```
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')`.
```

```
Current validation accuracy: 0.8829787373542786
Reseting all weights…
Current number of trials: 2
```





```
['loss', 'compile_metrics']
3/3              0s 17ms/step -
accuracy: 0.8829 - loss: 0.3935
[0.37573546171188354, 0.8829787373542786]
3/3              0s 45ms/step
Classification Report:
              precision    recall  f1-score   support

      Female       0.83      0.93      0.87        41
        Male       0.94      0.85      0.89        53

    accuracy                           0.88        94
   macro avg       0.88      0.89      0.88        94
weighted avg       0.89      0.88      0.88        94
```

```
[1]:     accuracy      loss  val_accuracy  val_loss  learning_rate
     0   0.647821  1.991831      0.723404  0.574178          0.005
     1   0.710247  0.545627      0.787234  0.552998          0.005
     2   0.704358  0.556103      0.744681  0.536161          0.005
     3   0.756184  0.515687      0.755319  0.481906          0.005
     4   0.755006  0.557162      0.723404  0.594113          0.005
     5   0.738516  0.534944      0.797872  0.612482          0.005
     6   0.762073  0.486636      0.776596  0.565275          0.005
     7   0.771496  0.493853      0.808511  0.492743          0.005
     8   0.787986  0.473971      0.797872  0.467577          0.005
     9   0.804476  0.456402      0.787234  0.535287          0.005
     10  0.802120  0.441130      0.819149  0.439167          0.005
     11  0.823322  0.409732      0.797872  0.477067          0.005
     12  0.829211  0.396230      0.840426  0.388031          0.005
```

```
13  0.808009  0.407230      0.829787  0.443219          0.005
14  0.828033  0.388145      0.861702  0.390725          0.005
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```