Personal Project_04_v10_test1_4conv-layer_run51_advanced control 1_autorun

May 6, 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<0.80 and try_num<10):
    # DOE factors:
    learning_rate = 0.005
    dropout_value = 0.2
    # n-conv_layers = 4
    n_units_last_layer = 4096
    n_filters_l1 = 8
    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']
```

```
    epochs = 17
    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)
```

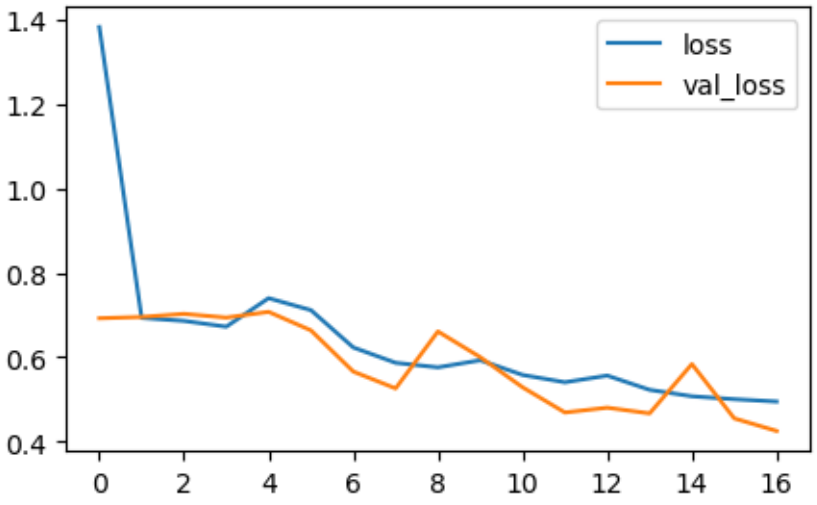
```
    # Get the first batch of images and labels
    for images, labels in training_dataset.take(1):
            example_batch_images = images
            example_batch_labels = labels
    max_pixel = np.max(example_batch_images)

    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:     #####
```

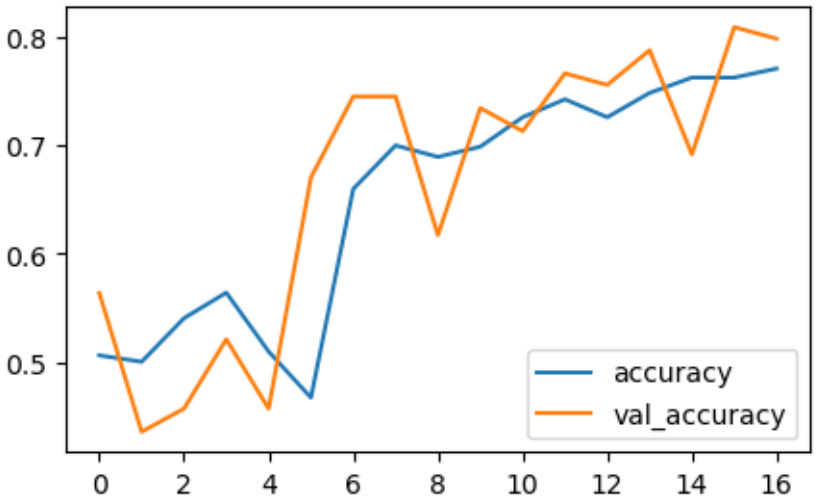
```
            tf.keras.layers.Conv2D(n_filters_l2, (3, 3), activation = 'relu'),
            tf.keras.layers.MaxPooling2D(2, 2),
            #####    CONV_LAYER_3:     #####
            tf.keras.layers.Conv2D(64, (3, 3), activation = 'relu'),
            tf.keras.layers.MaxPooling2D(2, 2),
            #####    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],
                                 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_run51_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 1/17
27/27 - 5s - 191ms/step - accuracy: 0.5065 - loss: 1.3823 - val_accuracy: 0.5638
- val_loss: 0.6923 - learning_rate: 0.0050
Epoch 2/17
27/27 - 2s - 83ms/step - accuracy: 0.5006 - loss: 0.6931 - val_accuracy: 0.4362
- val_loss: 0.6952 - learning_rate: 0.0050
Epoch 3/17
27/27 - 2s - 81ms/step - accuracy: 0.5406 - loss: 0.6859 - val_accuracy: 0.4574
- val_loss: 0.7026 - learning_rate: 0.0050
Epoch 4/17
27/27 - 2s - 84ms/step - accuracy: 0.5642 - loss: 0.6722 - val_accuracy: 0.5213
- val_loss: 0.6938 - learning_rate: 0.0050
Epoch 5/17
27/27 - 2s - 82ms/step - accuracy: 0.5100 - loss: 0.7398 - val_accuracy: 0.4574
- val_loss: 0.7074 - learning_rate: 0.0050
Epoch 6/17
27/27 - 2s - 81ms/step - accuracy: 0.4676 - loss: 0.7115 - val_accuracy: 0.6702
- val_loss: 0.6635 - learning_rate: 0.0050
Epoch 7/17
27/27 - 2s - 82ms/step - accuracy: 0.6596 - loss: 0.6231 - val_accuracy: 0.7447
- val_loss: 0.5659 - learning_rate: 0.0050
Epoch 8/17
27/27 - 2s - 82ms/step - accuracy: 0.6996 - loss: 0.5867 - val_accuracy: 0.7447
- val_loss: 0.5260 - learning_rate: 0.0050
Epoch 9/17
27/27 - 3s - 96ms/step - accuracy: 0.6890 - loss: 0.5759 - val_accuracy: 0.6170
- val_loss: 0.6609 - learning_rate: 0.0050
Epoch 10/17
27/27 - 2s - 88ms/step - accuracy: 0.6985 - loss: 0.5927 - val_accuracy: 0.7340
- val_loss: 0.6000 - learning_rate: 0.0050
Epoch 11/17
```

```
27/27 - 3s - 94ms/step - accuracy: 0.7256 - loss: 0.5579 - val_accuracy: 0.7128
- val_loss: 0.5287 - learning_rate: 0.0025
Epoch 12/17
27/27 - 2s - 82ms/step - accuracy: 0.7420 - loss: 0.5406 - val_accuracy: 0.7660
- val_loss: 0.4689 - learning_rate: 0.0025
Epoch 13/17
27/27 - 2s - 82ms/step - accuracy: 0.7256 - loss: 0.5564 - val_accuracy: 0.7553
- val_loss: 0.4803 - learning_rate: 0.0025
Epoch 14/17
27/27 - 2s - 83ms/step - accuracy: 0.7479 - loss: 0.5226 - val_accuracy: 0.7872
- val_loss: 0.4668 - learning_rate: 0.0025
Epoch 15/17
27/27 - 2s - 81ms/step - accuracy: 0.7621 - loss: 0.5073 - val_accuracy: 0.6915
- val_loss: 0.5837 - learning_rate: 0.0025
Epoch 16/17
27/27 - 2s - 83ms/step - accuracy: 0.7621 - loss: 0.5004 - val_accuracy: 0.8085
- val_loss: 0.4550 - learning_rate: 0.0025
Epoch 17/17
27/27 - 2s - 81ms/step - accuracy: 0.7703 - loss: 0.4949 - val_accuracy: 0.7979
- val_loss: 0.4252 - learning_rate: 0.0025

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: 0.7978723645210266
Reseting all weights…
Current number of trials: 1
```

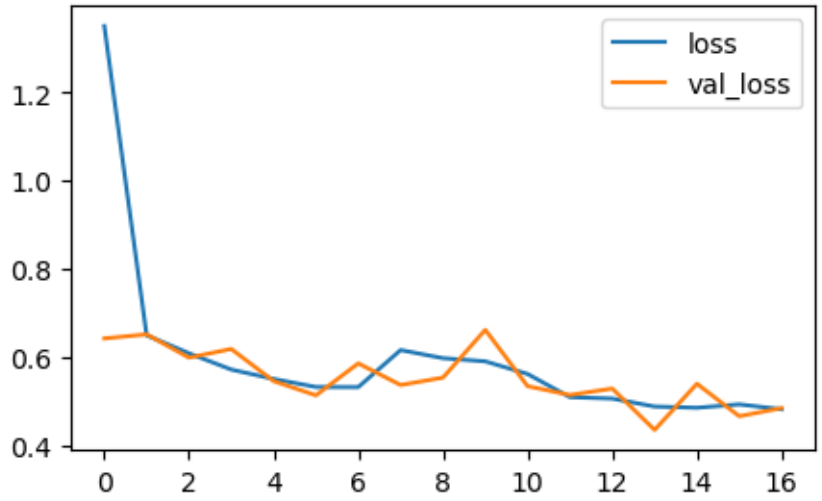


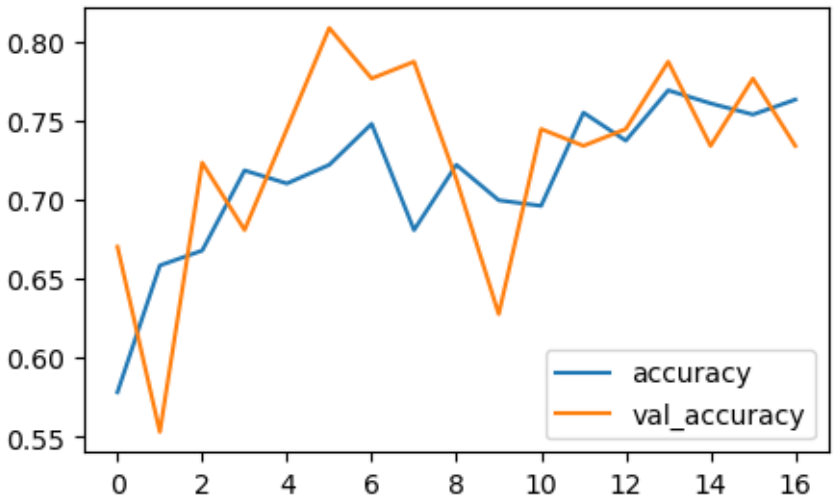



```
['loss', 'compile_metrics']
3/3             0s 33ms/step -
accuracy: 0.7661 - loss: 0.4402
[0.4252191483974457, 0.7978723645210266]
3/3             0s 82ms/step
Classification Report:
               precision    recall  f1-score   support

      Female       0.74      0.83      0.78        41
        Male       0.85      0.77      0.81        53

    accuracy                           0.80        94
   macro avg       0.80      0.80      0.80        94
weighted avg       0.80      0.80      0.80        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 1/17
27/27 - 5s - 191ms/step - accuracy: 0.5783 - loss: 1.3506 - val_accuracy: 0.6702
- val_loss: 0.6426 - learning_rate: 0.0050
Epoch 2/17
27/27 - 2s - 83ms/step - accuracy: 0.6584 - loss: 0.6498 - val_accuracy: 0.5532
- val_loss: 0.6519 - learning_rate: 0.0050
Epoch 3/17
27/27 - 2s - 83ms/step - accuracy: 0.6678 - loss: 0.6090 - val_accuracy: 0.7234
- val_loss: 0.5991 - learning_rate: 0.0050
Epoch 4/17
27/27 - 3s - 98ms/step - accuracy: 0.7185 - loss: 0.5721 - val_accuracy: 0.6809
```

```
- val_loss: 0.6188 - learning_rate: 0.0050
Epoch 5/17
27/27 - 2s - 82ms/step - accuracy: 0.7102 - loss: 0.5508 - val_accuracy: 0.7447
- val_loss: 0.5460 - learning_rate: 0.0050
Epoch 6/17
27/27 - 2s - 82ms/step - accuracy: 0.7220 - loss: 0.5327 - val_accuracy: 0.8085
- val_loss: 0.5137 - learning_rate: 0.0050
Epoch 7/17
27/27 - 2s - 82ms/step - accuracy: 0.7479 - loss: 0.5324 - val_accuracy: 0.7766
- val_loss: 0.5861 - learning_rate: 0.0050
Epoch 8/17
27/27 - 2s - 82ms/step - accuracy: 0.6808 - loss: 0.6163 - val_accuracy: 0.7872
- val_loss: 0.5371 - learning_rate: 0.0050
Epoch 9/17
27/27 - 2s - 83ms/step - accuracy: 0.7220 - loss: 0.5979 - val_accuracy: 0.7128
- val_loss: 0.5538 - learning_rate: 0.0050
Epoch 10/17
27/27 - 2s - 83ms/step - accuracy: 0.6996 - loss: 0.5908 - val_accuracy: 0.6277
- val_loss: 0.6622 - learning_rate: 0.0050
Epoch 11/17
27/27 - 2s - 82ms/step - accuracy: 0.6961 - loss: 0.5621 - val_accuracy: 0.7447
- val_loss: 0.5345 - learning_rate: 0.0025
Epoch 12/17
27/27 - 2s - 82ms/step - accuracy: 0.7550 - loss: 0.5095 - val_accuracy: 0.7340
- val_loss: 0.5145 - learning_rate: 0.0025
Epoch 13/17
27/27 - 2s - 84ms/step - accuracy: 0.7373 - loss: 0.5064 - val_accuracy: 0.7447
- val_loss: 0.5291 - learning_rate: 0.0025
Epoch 14/17
27/27 - 2s - 82ms/step - accuracy: 0.7691 - loss: 0.4883 - val_accuracy: 0.7872
- val_loss: 0.4353 - learning_rate: 0.0025
Epoch 15/17
27/27 - 2s - 82ms/step - accuracy: 0.7609 - loss: 0.4856 - val_accuracy: 0.7340
- val_loss: 0.5400 - learning_rate: 0.0025
Epoch 16/17
27/27 - 2s - 82ms/step - accuracy: 0.7538 - loss: 0.4933 - val_accuracy: 0.7766
- val_loss: 0.4664 - learning_rate: 0.0025
Epoch 17/17
27/27 - 2s - 83ms/step - accuracy: 0.7633 - loss: 0.4822 - val_accuracy: 0.7340
- val_loss: 0.4848 - learning_rate: 0.0025

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: 0.7340425252914429
Reseting all weights…
```

```
Current number of trials: 2
```

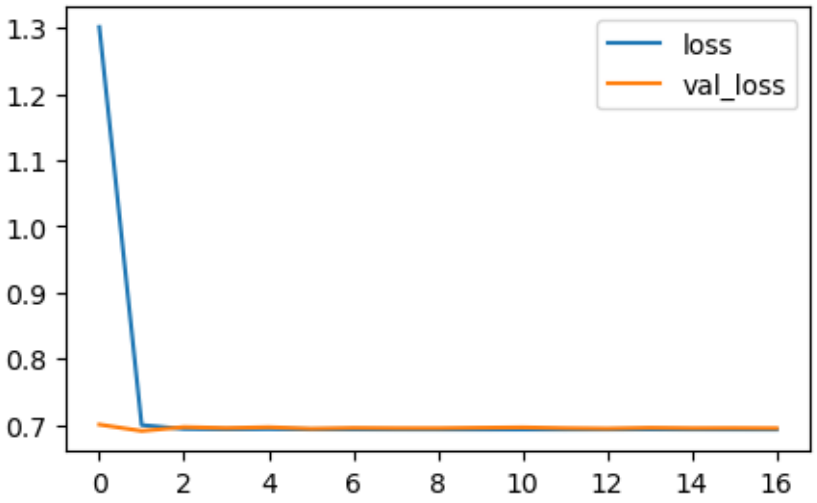


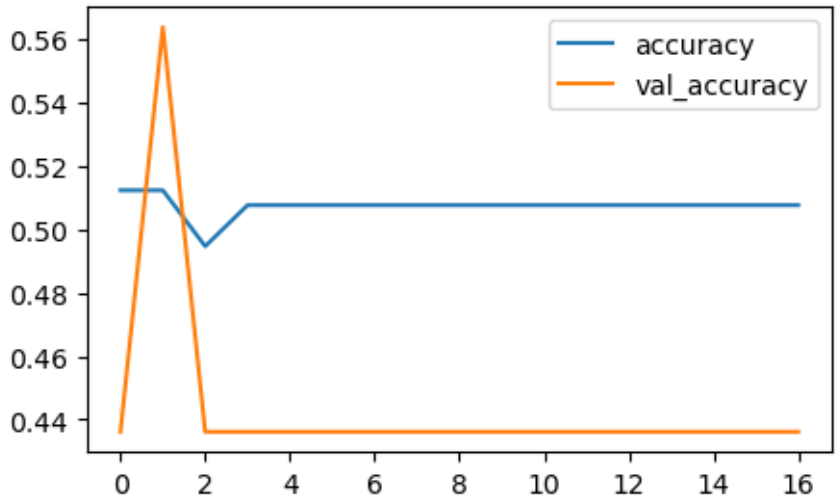



```
['loss', 'compile_metrics']
3/3             0s 23ms/step -
accuracy: 0.7225 - loss: 0.4928
[0.48484566807746887, 0.7340425252914429]
3/3             0s 98ms/step
Classification Report:
              precision    recall  f1-score   support

      Female       0.68      0.73      0.71        41
        Male       0.78      0.74      0.76        53
```

```
    accuracy                           0.73        94
   macro avg       0.73      0.73      0.73        94
weighted avg       0.74      0.73      0.73        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 1/17
27/27 - 5s - 183ms/step - accuracy: 0.5124 - loss: 1.3010 - val_accuracy: 0.4362
- val_loss: 0.7007 - learning_rate: 0.0050
Epoch 2/17
27/27 - 2s - 85ms/step - accuracy: 0.5124 - loss: 0.6999 - val_accuracy: 0.5638
- val_loss: 0.6910 - learning_rate: 0.0050
Epoch 3/17
27/27 - 2s - 82ms/step - accuracy: 0.4947 - loss: 0.6941 - val_accuracy: 0.4362
- val_loss: 0.6969 - learning_rate: 0.0050
Epoch 4/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6935 - val_accuracy: 0.4362
- val_loss: 0.6954 - learning_rate: 0.0050
Epoch 5/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6939 - val_accuracy: 0.4362
- val_loss: 0.6966 - learning_rate: 0.0050
Epoch 6/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6933 - val_accuracy: 0.4362
- val_loss: 0.6944 - learning_rate: 0.0050
Epoch 7/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6933 - val_accuracy: 0.4362
- val_loss: 0.6955 - learning_rate: 0.0050
Epoch 8/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6952 - learning_rate: 0.0050
Epoch 9/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6952 - learning_rate: 0.0050
Epoch 10/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6959 - learning_rate: 0.0050
Epoch 11/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6964 - learning_rate: 0.0025
Epoch 12/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6931 - val_accuracy: 0.4362
- val_loss: 0.6952 - learning_rate: 0.0025
Epoch 13/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6946 - learning_rate: 0.0025
```

```
Epoch 14/17
27/27 - 2s - 85ms/step - accuracy: 0.5077 - loss: 0.6934 - val_accuracy: 0.4362
- val_loss: 0.6958 - learning_rate: 0.0025
Epoch 15/17
27/27 - 2s - 83ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6952 - learning_rate: 0.0025
Epoch 16/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6932 - val_accuracy: 0.4362
- val_loss: 0.6953 - learning_rate: 0.0025
Epoch 17/17
27/27 - 2s - 82ms/step - accuracy: 0.5077 - loss: 0.6931 - val_accuracy: 0.4362
- val_loss: 0.6951 - learning_rate: 0.0025

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: 0.43617022037506104
Reseting all weights…
Current number of trials: 3
```



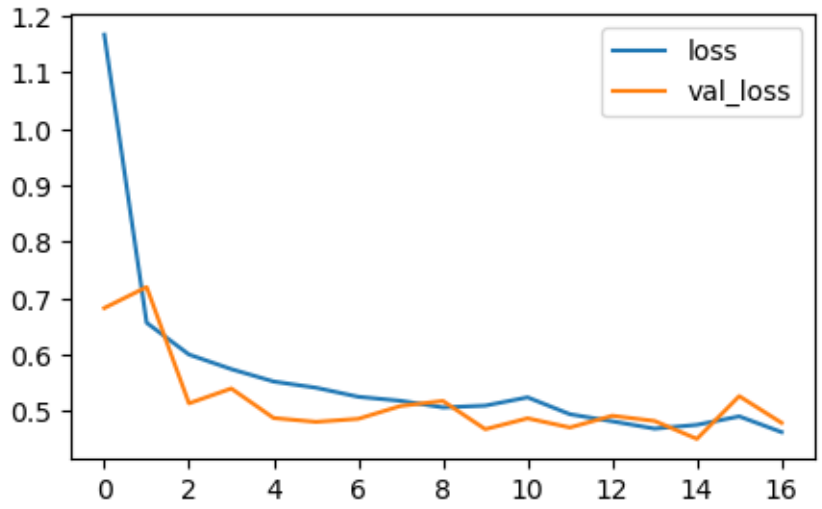


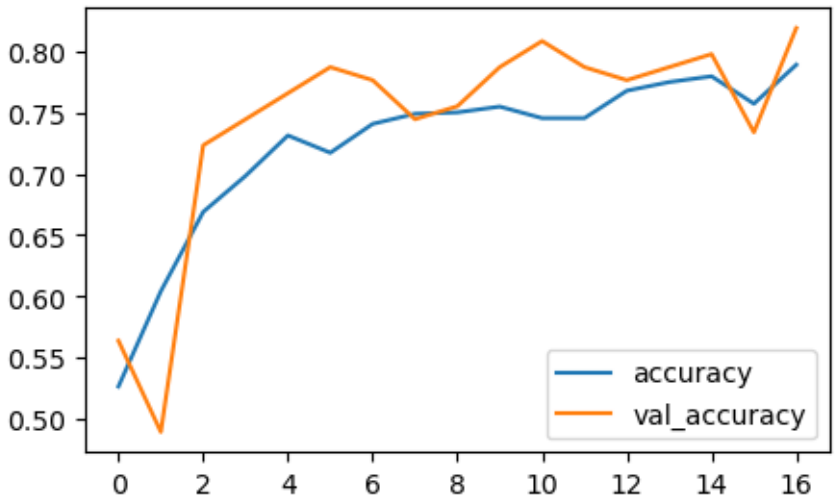

```
['loss', 'compile_metrics']
3/3             0s 33ms/step -
accuracy: 0.4603 - loss: 0.6944
[0.6951209306716919, 0.43617022037506104]
WARNING:tensorflow:5 out of the last 7 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000024FC4F90D60> triggered 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.

WARNING:tensorflow:5 out of the last 7 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000024FC4F90D60> 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
121ms/stepWARNING:tensorflow:6 out of the last 9 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000024FC4F90D60> 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.

WARNING:tensorflow:6 out of the last 9 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x0000024FC4F90D60> 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 86ms/step

C:\Users\MICROSOFT SURFACE\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Classification Report:
              precision    recall  f1-score   support

      Female       0.44      1.00      0.61        41
        Male       0.00      0.00      0.00        53

    accuracy                           0.44        94
   macro avg       0.22      0.50      0.30        94
weighted avg       0.19      0.44      0.26        94

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

C:\Users\MICROSOFT SURFACE\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\MICROSOFT SURFACE\anaconda3\Lib\site-
packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Found 943 files belonging to 2 classes.
```

```
Using 94 files for validation.
Epoch 1/17
27/27 - 5s - 174ms/step - accuracy: 0.5265 - loss: 1.1670 - val_accuracy: 0.5638
- val_loss: 0.6830 - learning_rate: 0.0050
Epoch 2/17
27/27 - 2s - 83ms/step - accuracy: 0.6042 - loss: 0.6569 - val_accuracy: 0.4894
- val_loss: 0.7201 - learning_rate: 0.0050
Epoch 3/17
27/27 - 2s - 83ms/step - accuracy: 0.6690 - loss: 0.6006 - val_accuracy: 0.7234
- val_loss: 0.5142 - learning_rate: 0.0050
Epoch 4/17
27/27 - 2s - 82ms/step - accuracy: 0.6985 - loss: 0.5748 - val_accuracy: 0.7447
- val_loss: 0.5405 - learning_rate: 0.0050
Epoch 5/17
27/27 - 2s - 82ms/step - accuracy: 0.7314 - loss: 0.5528 - val_accuracy: 0.7660
- val_loss: 0.4882 - learning_rate: 0.0050
Epoch 6/17
27/27 - 2s - 82ms/step - accuracy: 0.7173 - loss: 0.5418 - val_accuracy: 0.7872
- val_loss: 0.4814 - learning_rate: 0.0050
Epoch 7/17
27/27 - 2s - 82ms/step - accuracy: 0.7409 - loss: 0.5258 - val_accuracy: 0.7766
- val_loss: 0.4868 - learning_rate: 0.0050
Epoch 8/17
27/27 - 2s - 85ms/step - accuracy: 0.7491 - loss: 0.5187 - val_accuracy: 0.7447
- val_loss: 0.5096 - learning_rate: 0.0050
Epoch 9/17
27/27 - 2s - 82ms/step - accuracy: 0.7503 - loss: 0.5070 - val_accuracy: 0.7553
- val_loss: 0.5184 - learning_rate: 0.0050
Epoch 10/17
27/27 - 2s - 83ms/step - accuracy: 0.7550 - loss: 0.5102 - val_accuracy: 0.7872
- val_loss: 0.4684 - learning_rate: 0.0050
Epoch 11/17
27/27 - 2s - 83ms/step - accuracy: 0.7456 - loss: 0.5248 - val_accuracy: 0.8085
- val_loss: 0.4877 - learning_rate: 0.0025
Epoch 12/17
27/27 - 2s - 82ms/step - accuracy: 0.7456 - loss: 0.4949 - val_accuracy: 0.7872
- val_loss: 0.4714 - learning_rate: 0.0025
Epoch 13/17
27/27 - 2s - 82ms/step - accuracy: 0.7680 - loss: 0.4824 - val_accuracy: 0.7766
- val_loss: 0.4922 - learning_rate: 0.0025
Epoch 14/17
27/27 - 2s - 83ms/step - accuracy: 0.7750 - loss: 0.4694 - val_accuracy: 0.7872
- val_loss: 0.4830 - learning_rate: 0.0025
Epoch 15/17
27/27 - 3s - 101ms/step - accuracy: 0.7797 - loss: 0.4761 - val_accuracy: 0.7979
- val_loss: 0.4513 - learning_rate: 0.0025
Epoch 16/17
27/27 - 2s - 83ms/step - accuracy: 0.7574 - loss: 0.4911 - val_accuracy: 0.7340
```

```
- val_loss: 0.5269 - learning_rate: 0.0025
Epoch 17/17
27/27 - 2s - 83ms/step - accuracy: 0.7892 - loss: 0.4632 - val_accuracy: 0.8191
- val_loss: 0.4797 - learning_rate: 0.0025

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: 0.8191489577293396
Reseting all weights…
Current number of trials: 4
```





```
['loss', 'compile_metrics']
3/3             0s 23ms/step -
accuracy: 0.7963 - loss: 0.4961
[0.4797188639640808, 0.8191489577293396]
3/3             0s 77ms/step
Classification Report:
               precision    recall  f1-score   support

      Female       0.77      0.83      0.80        41
        Male       0.86      0.81      0.83        53

    accuracy                           0.82        94
   macro avg       0.82      0.82      0.82        94
weighted avg       0.82      0.82      0.82        94
```

[1]:

	accuracy	loss	val_accuracy	val_loss	learning_rate
0	0.526502	1.166981	0.563830	0.682958	0.0050
1	0.604240	0.656876	0.489362	0.720113	0.0050
2	0.669022	0.600641	0.723404	0.514176	0.0050
3	0.698469	0.574779	0.744681	0.540474	0.0050
4	0.731449	0.552824	0.765957	0.488200	0.0050
5	0.717314	0.541847	0.787234	0.481392	0.0050
6	0.740872	0.525833	0.776596	0.486792	0.0050
7	0.749117	0.518652	0.744681	0.509562	0.0050
8	0.750294	0.506974	0.755319	0.518418	0.0050
9	0.755006	0.510153	0.787234	0.468393	0.0050
10	0.745583	0.524811	0.808511	0.487692	0.0025
11	0.745583	0.494913	0.787234	0.471414	0.0025
12	0.767962	0.482384	0.776596	0.492203	0.0025
13	0.775029	0.469428	0.787234	0.482959	0.0025
14	0.779741	0.476103	0.797872	0.451276	0.0025

[ ]:

[ ]:

[ ]: