Personal Project_04_v10_test1_3conv-layer_run40_advanced control 1_autorun

May 5, 2025

[2]:
```
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<0.87 and try_num<5):
    # DOE factors:
    learning_rate = 0.0005
    dropout_value = 0.5
    # n-conv_layers = 4
    n_units_last_layer = 4096
    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 = 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)
```

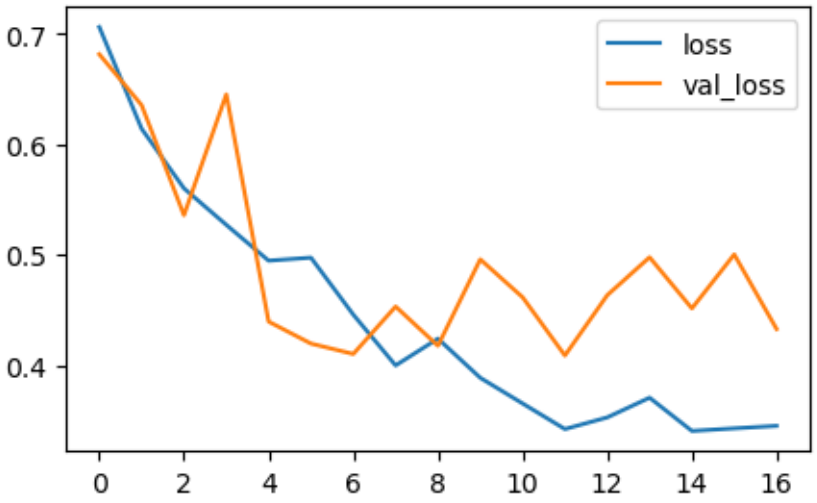
```
    # Get the first batch of images and labels
    for images, labels in training_dataset.take(1):
            example_batch_images = images
            example_batch_labels = labels
    max_pixel = np.max(example_batch_images)

    # 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 / 1.1
            return lr
        elif epoch < 15:
            if epoch % 5 == 0 and epoch > 0:
                return lr / 4
            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'),
```

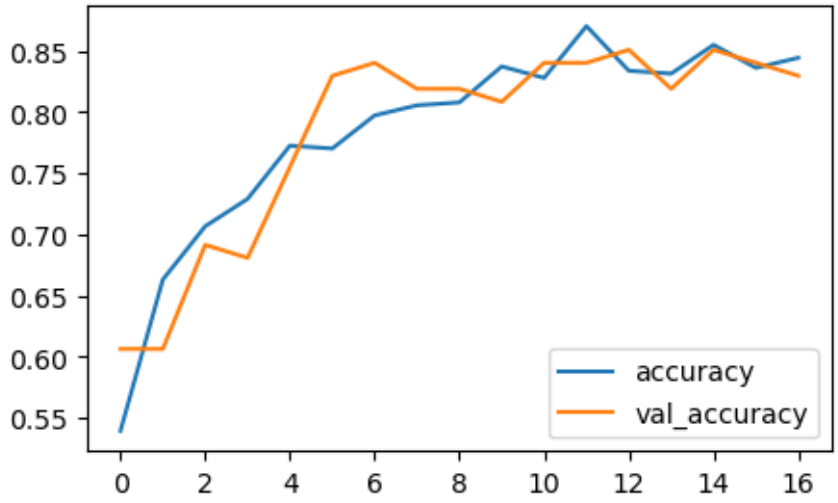
```
            tf.keras.layers.MaxPooling2D(2, 2),
            #####    CONV_LAYER_2:     #####
            tf.keras.layers.Conv2D(n_filters_12, (3, 3), activation = 'relu'),
            tf.keras.layers.MaxPooling2D(2, 2),
            #####    CONV_LAYER_3:     #####
            tf.keras.layers.Conv2D(64, (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_run40_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 - 4s - 158ms/step - accuracy: 0.5395 - loss: 0.7062 - val_accuracy: 0.6064
- val_loss: 0.6816 - learning_rate: 5.0000e-04
Epoch 2/17
27/27 - 2s - 60ms/step - accuracy: 0.6631 - loss: 0.6141 - val_accuracy: 0.6064
- val_loss: 0.6355 - learning_rate: 5.0000e-04
Epoch 3/17
27/27 - 2s - 59ms/step - accuracy: 0.7067 - loss: 0.5601 - val_accuracy: 0.6915
- val_loss: 0.5358 - learning_rate: 5.0000e-04
Epoch 4/17
27/27 - 2s - 59ms/step - accuracy: 0.7291 - loss: 0.5270 - val_accuracy: 0.6809
- val_loss: 0.6452 - learning_rate: 5.0000e-04
Epoch 5/17
27/27 - 2s - 60ms/step - accuracy: 0.7727 - loss: 0.4946 - val_accuracy: 0.7553
- val_loss: 0.4394 - learning_rate: 5.0000e-04
Epoch 6/17
27/27 - 2s - 60ms/step - accuracy: 0.7703 - loss: 0.4973 - val_accuracy: 0.8298
- val_loss: 0.4195 - learning_rate: 4.5455e-04
Epoch 7/17
27/27 - 2s - 62ms/step - accuracy: 0.7974 - loss: 0.4457 - val_accuracy: 0.8404
- val_loss: 0.4101 - learning_rate: 4.5455e-04
Epoch 8/17
27/27 - 2s - 60ms/step - accuracy: 0.8057 - loss: 0.3997 - val_accuracy: 0.8191
- val_loss: 0.4531 - learning_rate: 4.5455e-04
Epoch 9/17
27/27 - 2s - 59ms/step - accuracy: 0.8080 - loss: 0.4240 - val_accuracy: 0.8191
- val_loss: 0.4175 - learning_rate: 4.5455e-04
Epoch 10/17
27/27 - 2s - 59ms/step - accuracy: 0.8375 - loss: 0.3885 - val_accuracy: 0.8085
```

```
- val_loss: 0.4958 - learning_rate: 4.5455e-04
Epoch 11/17
27/27 - 2s - 59ms/step - accuracy: 0.8280 - loss: 0.3653 - val_accuracy: 0.8404
- val_loss: 0.4615 - learning_rate: 1.1364e-04
Epoch 12/17
27/27 - 2s - 59ms/step - accuracy: 0.8704 - loss: 0.3421 - val_accuracy: 0.8404
- val_loss: 0.4087 - learning_rate: 1.1364e-04
Epoch 13/17
27/27 - 2s - 59ms/step - accuracy: 0.8339 - loss: 0.3526 - val_accuracy: 0.8511
- val_loss: 0.4633 - learning_rate: 1.1364e-04
Epoch 14/17
27/27 - 2s - 60ms/step - accuracy: 0.8316 - loss: 0.3705 - val_accuracy: 0.8191
- val_loss: 0.4977 - learning_rate: 1.1364e-04
Epoch 15/17
27/27 - 2s - 59ms/step - accuracy: 0.8551 - loss: 0.3404 - val_accuracy: 0.8511
- val_loss: 0.4514 - learning_rate: 1.1364e-04
Epoch 16/17
27/27 - 2s - 59ms/step - accuracy: 0.8363 - loss: 0.3427 - val_accuracy: 0.8404
- val_loss: 0.5003 - learning_rate: 1.1364e-04
Epoch 17/17
27/27 - 2s - 59ms/step - accuracy: 0.8445 - loss: 0.3450 - val_accuracy: 0.8298
- val_loss: 0.4327 - learning_rate: 1.1364e-04

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
```

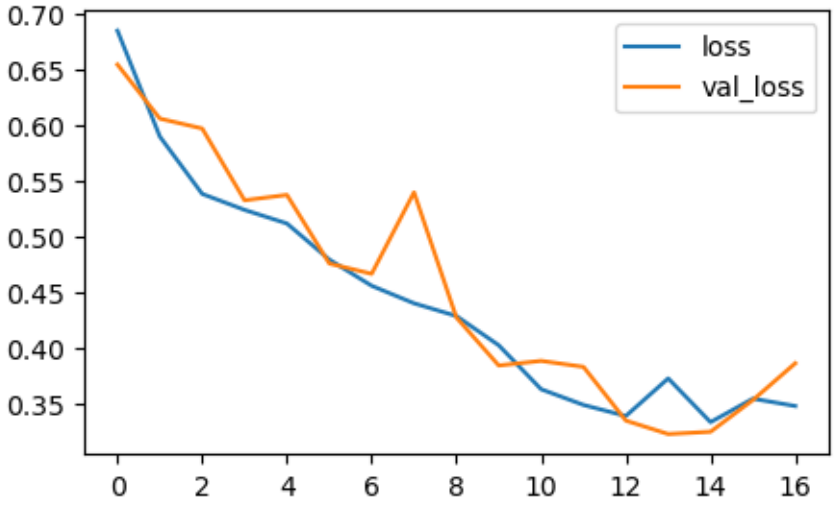


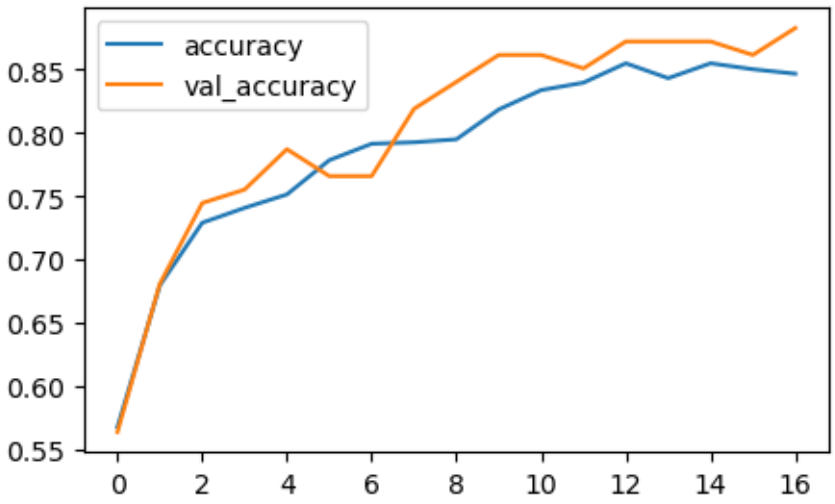



```
['loss', 'compile_metrics']
3/3             0s 21ms/step -
accuracy: 0.8211 - loss: 0.4622
[0.4327034652233124, 0.8297872543334961]
3/3             0s 58ms/step
Classification Report:
               precision    recall  f1-score   support

      Female       0.75      0.93      0.83        41
        Male       0.93      0.75      0.83        53

    accuracy                           0.83        94
   macro avg       0.84      0.84      0.83        94
weighted avg       0.85      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 1/17
27/27 - 4s - 149ms/step - accuracy: 0.5677 - loss: 0.6848 - val_accuracy: 0.5638
- val_loss: 0.6544 - learning_rate: 5.0000e-04
Epoch 2/17
27/27 - 2s - 60ms/step - accuracy: 0.6796 - loss: 0.5898 - val_accuracy: 0.6809
- val_loss: 0.6059 - learning_rate: 5.0000e-04
Epoch 3/17
27/27 - 2s - 61ms/step - accuracy: 0.7291 - loss: 0.5383 - val_accuracy: 0.7447
- val_loss: 0.5970 - learning_rate: 5.0000e-04
```

```
Epoch 4/17
27/27 - 2s - 64ms/step - accuracy: 0.7409 - loss: 0.5240 - val_accuracy: 0.7553
- val_loss: 0.5326 - learning_rate: 5.0000e-04
Epoch 5/17
27/27 - 2s - 73ms/step - accuracy: 0.7515 - loss: 0.5118 - val_accuracy: 0.7872
- val_loss: 0.5374 - learning_rate: 5.0000e-04
Epoch 6/17
27/27 - 2s - 60ms/step - accuracy: 0.7786 - loss: 0.4794 - val_accuracy: 0.7660
- val_loss: 0.4759 - learning_rate: 4.5455e-04
Epoch 7/17
27/27 - 2s - 58ms/step - accuracy: 0.7915 - loss: 0.4559 - val_accuracy: 0.7660
- val_loss: 0.4666 - learning_rate: 4.5455e-04
Epoch 8/17
27/27 - 2s - 60ms/step - accuracy: 0.7927 - loss: 0.4402 - val_accuracy: 0.8191
- val_loss: 0.5397 - learning_rate: 4.5455e-04
Epoch 9/17
27/27 - 2s - 59ms/step - accuracy: 0.7951 - loss: 0.4288 - val_accuracy: 0.8404
- val_loss: 0.4275 - learning_rate: 4.5455e-04
Epoch 10/17
27/27 - 2s - 60ms/step - accuracy: 0.8186 - loss: 0.4025 - val_accuracy: 0.8617
- val_loss: 0.3844 - learning_rate: 4.5455e-04
Epoch 11/17
27/27 - 2s - 59ms/step - accuracy: 0.8339 - loss: 0.3631 - val_accuracy: 0.8617
- val_loss: 0.3885 - learning_rate: 1.1364e-04
Epoch 12/17
27/27 - 2s - 59ms/step - accuracy: 0.8398 - loss: 0.3490 - val_accuracy: 0.8511
- val_loss: 0.3832 - learning_rate: 1.1364e-04
Epoch 13/17
27/27 - 2s - 60ms/step - accuracy: 0.8551 - loss: 0.3391 - val_accuracy: 0.8723
- val_loss: 0.3349 - learning_rate: 1.1364e-04
Epoch 14/17
27/27 - 2s - 59ms/step - accuracy: 0.8433 - loss: 0.3728 - val_accuracy: 0.8723
- val_loss: 0.3229 - learning_rate: 1.1364e-04
Epoch 15/17
27/27 - 2s - 60ms/step - accuracy: 0.8551 - loss: 0.3335 - val_accuracy: 0.8723
- val_loss: 0.3249 - learning_rate: 1.1364e-04
Epoch 16/17
27/27 - 2s - 61ms/step - accuracy: 0.8504 - loss: 0.3546 - val_accuracy: 0.8617
- val_loss: 0.3534 - learning_rate: 1.1364e-04
Epoch 17/17
27/27 - 2s - 62ms/step - accuracy: 0.8469 - loss: 0.3480 - val_accuracy: 0.8830
- val_loss: 0.3864 - learning_rate: 1.1364e-04

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 20ms/step -
accuracy: 0.8634 - loss: 0.4137
[0.38636478781700134, 0.8829787373542786]
3/3             0s 92ms/step
Classification Report:
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

      Female       0.80      0.98      0.88        41
        Male       0.98      0.81      0.89        53

    accuracy                           0.88        94
   macro avg       0.89      0.89      0.88        94
weighted avg       0.90      0.88      0.88        94
```

[2]:

	accuracy	loss	val_accuracy	val_loss	learning_rate
0	0.567727	0.684779	0.563830	0.654370	0.000500
1	0.679623	0.589792	0.680851	0.605881	0.000500
2	0.729093	0.538327	0.744681	0.597006	0.000500
3	0.740872	0.523966	0.755319	0.532588	0.000500
4	0.751472	0.511778	0.787234	0.537433	0.000500
5	0.778563	0.479400	0.765957	0.475871	0.000455
6	0.791519	0.455903	0.765957	0.466617	0.000455
7	0.792697	0.440169	0.819149	0.539729	0.000455
8	0.795053	0.428767	0.840426	0.427457	0.000455
9	0.818610	0.402508	0.861702	0.384373	0.000455
10	0.833922	0.363085	0.861702	0.388512	0.000114
11	0.839812	0.348974	0.851064	0.383158	0.000114
12	0.855124	0.339052	0.872340	0.334926	0.000114
13	0.843345	0.372764	0.872340	0.322862	0.000114
14	0.855124	0.333542	0.872340	0.324914	0.000114

[13]:

[ ]: