

# Personal Project\_04\_v10\_test2\_2conv-layer\_run58\_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<0.88 and try_num<10):
    # DOE factors:
    learning_rate = 0.0005
    dropout_value = 0.2
    # n-conv_layers = 2
    n_units_last_layer = 4096
    n_filters_l1 = 32
    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)

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
    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),
        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_run58_advanced_control.h5')
# Restart script
print("Resetting 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 - 9s - 327ms/step - accuracy: 0.4865 - loss: 1.8072 - val\_accuracy: 0.4574  
- val\_loss: 0.6976 - learning\_rate: 5.0000e-04

Epoch 2/17

27/27 - 6s - 225ms/step - accuracy: 0.6243 - loss: 0.6319 - val\_accuracy: 0.7660  
- val\_loss: 0.5540 - learning\_rate: 5.0000e-04

Epoch 3/17

27/27 - 6s - 230ms/step - accuracy: 0.7633 - loss: 0.5121 - val\_accuracy: 0.7234  
- val\_loss: 0.5842 - learning\_rate: 5.0000e-04

Epoch 4/17

27/27 - 6s - 229ms/step - accuracy: 0.7373 - loss: 0.5266 - val\_accuracy: 0.7872  
- val\_loss: 0.4985 - learning\_rate: 5.0000e-04

Epoch 5/17

27/27 - 6s - 227ms/step - accuracy: 0.7680 - loss: 0.4783 - val\_accuracy: 0.7979  
- val\_loss: 0.4392 - learning\_rate: 5.0000e-04

Epoch 6/17

27/27 - 6s - 218ms/step - accuracy: 0.7703 - loss: 0.4648 - val\_accuracy: 0.7766  
- val\_loss: 0.5009 - learning\_rate: 5.0000e-04

Epoch 7/17

27/27 - 6s - 216ms/step - accuracy: 0.8045 - loss: 0.4497 - val\_accuracy: 0.7979  
- val\_loss: 0.4225 - learning\_rate: 5.0000e-04

Epoch 8/17

27/27 - 6s - 238ms/step - accuracy: 0.8033 - loss: 0.4166 - val\_accuracy: 0.7128  
- val\_loss: 0.5338 - learning\_rate: 5.0000e-04

Epoch 9/17

27/27 - 6s - 227ms/step - accuracy: 0.8186 - loss: 0.4103 - val\_accuracy: 0.8298  
- val\_loss: 0.4455 - learning\_rate: 5.0000e-04

Epoch 10/17

27/27 - 6s - 236ms/step - accuracy: 0.8257 - loss: 0.3752 - val\_accuracy: 0.7872  
- val\_loss: 0.5103 - learning\_rate: 5.0000e-04

Epoch 11/17

27/27 - 6s - 230ms/step - accuracy: 0.8280 - loss: 0.3854 - val\_accuracy: 0.8298  
- val\_loss: 0.4768 - learning\_rate: 5.0000e-04

Epoch 12/17

27/27 - 6s - 226ms/step - accuracy: 0.8492 - loss: 0.3703 - val\_accuracy: 0.8191  
- val\_loss: 0.5547 - learning\_rate: 5.0000e-04

Epoch 13/17

27/27 - 6s - 225ms/step - accuracy: 0.8433 - loss: 0.3523 - val\_accuracy: 0.8298  
- val\_loss: 0.4243 - learning\_rate: 5.0000e-04

Epoch 14/17

27/27 - 6s - 228ms/step - accuracy: 0.8563 - loss: 0.3423 - val\_accuracy: 0.8191  
- val\_loss: 0.4870 - learning\_rate: 5.0000e-04

Epoch 15/17

27/27 - 10s - 374ms/step - accuracy: 0.8410 - loss: 0.3605 - val\_accuracy:  
0.8404 - val\_loss: 0.3864 - learning\_rate: 5.0000e-04

Epoch 16/17

27/27 - 6s - 218ms/step - accuracy: 0.8575 - loss: 0.3229 - val\_accuracy: 0.8511  
- val\_loss: 0.3451 - learning\_rate: 5.0000e-04

Epoch 17/17

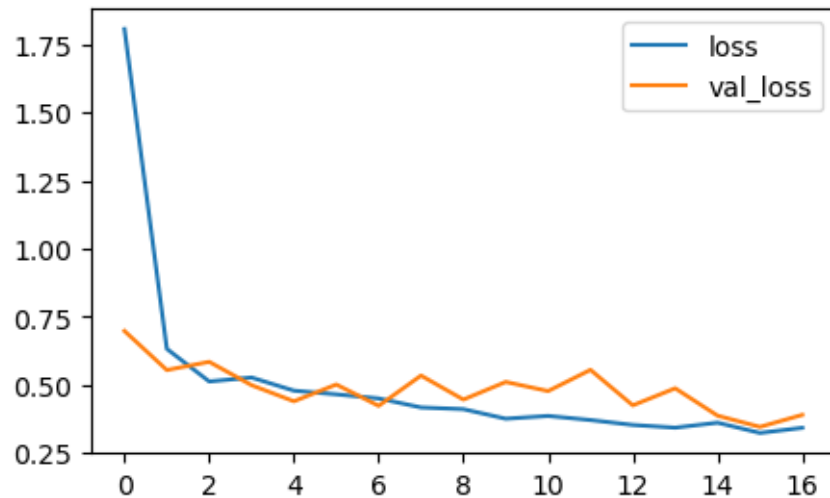
27/27 - 6s - 224ms/step - accuracy: 0.8551 - loss: 0.3417 - val\_accuracy: 0.8617  
- val\_loss: 0.3893 - learning\_rate: 5.0000e-04

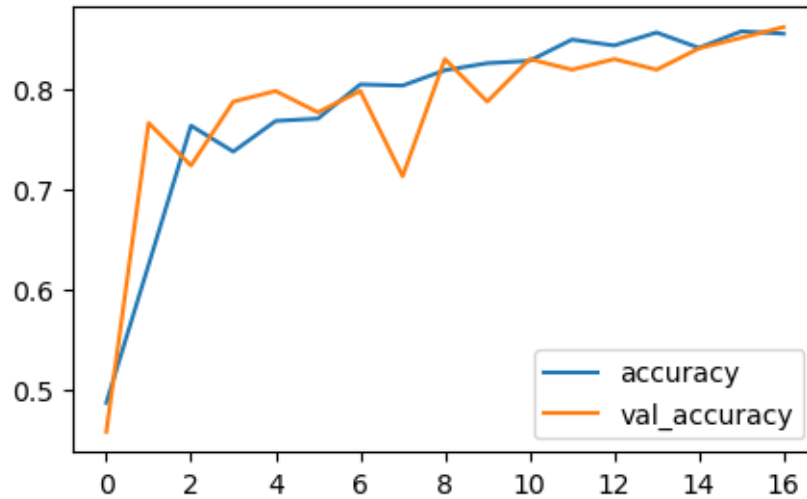
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or  
`keras.saving.save\_model(model)`. This file format is considered legacy. We  
recommend using instead the native Keras format, e.g.  
`model.save('my\_model.keras')` or `keras.saving.save\_model(model,  
'my\_model.keras')`.

Current validation accuracy: 0.8617021441459656

Reseting all weights...

Current number of trials: 1





```
['loss', 'compile_metrics']
```

```
3/3          0s 33ms/step -
```

```
accuracy: 0.8566 - loss: 0.4332
```

```
[0.3892681896686554, 0.8617021441459656]
```

```
3/3          0s 95ms/step
```

```
Classification Report:
```

	precision	recall	f1-score	support
Female	0.77	0.98	0.86	41
Male	0.98	0.77	0.86	53
accuracy			0.86	94
macro avg	0.87	0.87	0.86	94
weighted avg	0.89	0.86	0.86	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 - 8s - 304ms/step - accuracy: 0.5512 - loss: 0.8595 - val_accuracy: 0.6915
```

```
- val_loss: 0.6598 - learning_rate: 5.0000e-04
```

```
Epoch 2/17
```

```
27/27 - 6s - 224ms/step - accuracy: 0.7138 - loss: 0.5817 - val_accuracy: 0.7979
```

```
- val_loss: 0.4885 - learning_rate: 5.0000e-04
```

```
Epoch 3/17
```

```
27/27 - 6s - 224ms/step - accuracy: 0.7644 - loss: 0.5252 - val_accuracy: 0.7553
```

```
- val_loss: 0.5569 - learning_rate: 5.0000e-04
```

```
Epoch 4/17
```

```
27/27 - 6s - 223ms/step - accuracy: 0.7703 - loss: 0.4966 - val_accuracy: 0.7872
```

```

- val_loss: 0.4773 - learning_rate: 5.0000e-04
Epoch 5/17
27/27 - 6s - 238ms/step - accuracy: 0.7750 - loss: 0.4630 - val_accuracy: 0.8404
- val_loss: 0.4429 - learning_rate: 5.0000e-04
Epoch 6/17
27/27 - 7s - 247ms/step - accuracy: 0.8057 - loss: 0.4551 - val_accuracy: 0.8404
- val_loss: 0.4191 - learning_rate: 5.0000e-04
Epoch 7/17
27/27 - 6s - 232ms/step - accuracy: 0.8127 - loss: 0.4131 - val_accuracy: 0.8298
- val_loss: 0.4041 - learning_rate: 5.0000e-04
Epoch 8/17
27/27 - 6s - 241ms/step - accuracy: 0.8339 - loss: 0.3945 - val_accuracy: 0.8511
- val_loss: 0.3852 - learning_rate: 5.0000e-04
Epoch 9/17
27/27 - 7s - 261ms/step - accuracy: 0.8257 - loss: 0.3812 - val_accuracy: 0.8298
- val_loss: 0.3987 - learning_rate: 5.0000e-04
Epoch 10/17
27/27 - 6s - 236ms/step - accuracy: 0.8174 - loss: 0.3851 - val_accuracy: 0.8191
- val_loss: 0.4423 - learning_rate: 5.0000e-04
Epoch 11/17
27/27 - 6s - 236ms/step - accuracy: 0.8457 - loss: 0.3518 - val_accuracy: 0.8723
- val_loss: 0.3877 - learning_rate: 5.0000e-04
Epoch 12/17
27/27 - 6s - 225ms/step - accuracy: 0.8481 - loss: 0.3532 - val_accuracy: 0.9043
- val_loss: 0.3441 - learning_rate: 5.0000e-04
Epoch 13/17
27/27 - 6s - 222ms/step - accuracy: 0.8492 - loss: 0.3464 - val_accuracy: 0.8723
- val_loss: 0.4132 - learning_rate: 5.0000e-04
Epoch 14/17
27/27 - 6s - 219ms/step - accuracy: 0.8422 - loss: 0.3372 - val_accuracy: 0.9149
- val_loss: 0.3146 - learning_rate: 5.0000e-04
Epoch 15/17
27/27 - 6s - 218ms/step - accuracy: 0.8516 - loss: 0.3443 - val_accuracy: 0.8936
- val_loss: 0.3922 - learning_rate: 5.0000e-04
Epoch 16/17
27/27 - 6s - 218ms/step - accuracy: 0.8787 - loss: 0.2784 - val_accuracy: 0.8936
- val_loss: 0.3861 - learning_rate: 5.0000e-04
Epoch 17/17
27/27 - 6s - 220ms/step - accuracy: 0.8704 - loss: 0.3045 - val_accuracy: 0.8936
- val_loss: 0.3293 - learning_rate: 5.0000e-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')`.

```

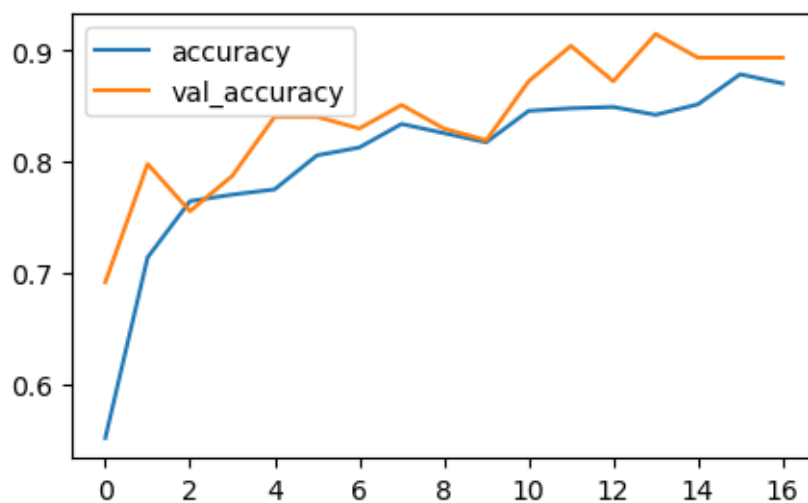
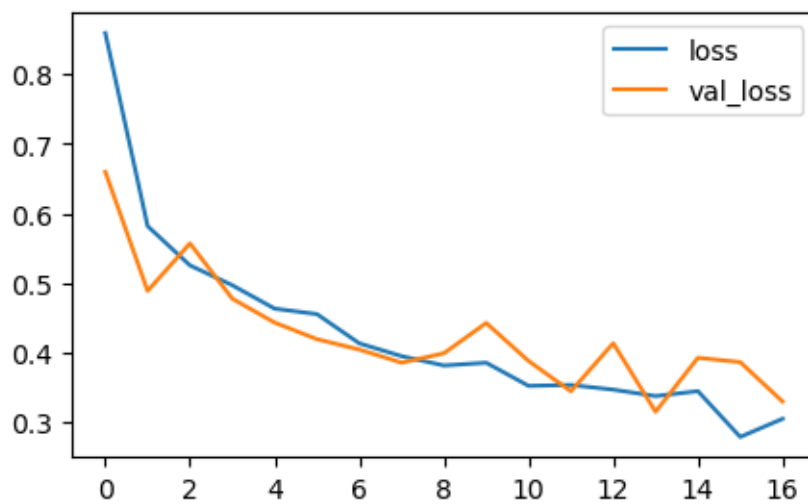
```

Current validation accuracy: 0.8936170339584351
Reseting all weights...

```



Current number of trials: 2



```
['loss', 'compile_metrics']
3/3      0s 31ms/step -
accuracy: 0.8843 - loss: 0.3854
[0.32934555411338806, 0.8936170339584351]
```

```
3/3      0s 75ms/step
```

Classification Report:

	precision	recall	f1-score	support
Female	0.86	0.90	0.88	41
Male	0.92	0.89	0.90	53

accuracy			0.89	94
macro avg	0.89	0.89	0.89	94
weighted avg	0.89	0.89	0.89	94

```
[1]: accuracy      loss  val_accuracy  val_loss  learning_rate
0  0.551237  0.859462    0.691489  0.659786    0.0005
1  0.713781  0.581653    0.797872  0.488451    0.0005
2  0.764429  0.525197    0.755319  0.556878    0.0005
3  0.770318  0.496641    0.787234  0.477290    0.0005
4  0.775029  0.462957    0.840426  0.442902    0.0005
5  0.805654  0.455089    0.840426  0.419053    0.0005
6  0.812721  0.413070    0.829787  0.404122    0.0005
7  0.833922  0.394536    0.851064  0.385247    0.0005
8  0.825677  0.381184    0.829787  0.398740    0.0005
9  0.817432  0.385108    0.819149  0.442344    0.0005
10 0.845701  0.351849    0.872340  0.387667    0.0005
11 0.848057  0.353173    0.904255  0.344100    0.0005
12 0.849234  0.346435    0.872340  0.413227    0.0005
13 0.842167  0.337172    0.914894  0.314608    0.0005
14 0.851590  0.344277    0.893617  0.392248    0.0005
```

```
[13]: 
```

```
[ ]: 
```

```
[ ]: 
```

```
[ ]: 
```