```
from google.colab import drive
drive.flush_and_unmount()
```

```
Drive not mounted, so nothing to flush and unmount.
```

```
!git clone https://github.com/threyareddya/dataset
!cd dataset
```

```
Cloning into 'dataset'...
remote: Enumerating objects: 173211, done.
remote: Total 173211 (delta 0), reused 0 (delta 0), pack-reused 1732
Receiving objects: 100% (173211/173211), 1.52 GiB | 16.95 MiB/s, don
Updating files: 100% (190335/190335), done.
```

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
#from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Conv2D, Dense, BatchNormalizati
from tensorflow.keras.models import Sequential
import os
import numpy as np
```

```
Start coding or generate with AI.
```

```
model = Sequential()

model.add(Conv2D(32,(3,3),input_shape = (200,200,3), activation = '
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
```

```
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(optimizer = 'adam', loss = 'binary_crossentropy', met
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolution
  super().__init__(activity_regularizer=activity_regularizer, **kwar
```

Model: "sequential"

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 200, 200, 32)
max_pooling2d (MaxPooling2D)	(None, 100, 100, 32)
dropout (Dropout)	(None, 100, 100, 32)
conv2d_1 (Conv2D)	(None, 100, 100, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 50, 50, 64)
dropout_1 (Dropout)	(None, 50, 50, 64)
conv2d_2 (Conv2D)	(None, 50, 50, 128)
max_pooling2d_2 (MaxPooling2D)	(None, 25, 25, 128)
dropout_2 (Dropout)	(None, 25, 25, 128)
flatten (Flatten)	(None, 80000)
dense (Dense)	(None, 128)
dropout_3 (Dropout)	(None, 128)
dense_1 (Dense)	(None, 1)

Total params: 10,333,505 (39.42 MB)
Trainable params: 10,333,505 (39.42 MB)
Non-trainable params: 0 (0.00 B)

```
base_dir = '/content/dataset'
train_dir = os.path.join(base_dir, 'Train')
val_dir = os.path.join(base_dir, 'Validation')
test_dir = os.path.join(base_dir, 'Test')

batch_size = 32
img_size = (128, 128)  # Reduced image size for faster processing
#target_size=(96,96)
#batch_size=64
#img_size = (200,200)
train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    rotation_range=20,
    zoom_range=0.2,
    shear_range=0.2
)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir, targ
val_generator = val_datagen.flow_from_directory(val_dir, target_siz
test_generator = test_datagen.flow_from_directory(test_dir, target_

mobilenet_base = MobileNetV2(weights='imagenet', include_top=False,
x = Flatten()(mobilenet_base.output)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
output = Dense(1, activation='sigmoid')(x)
mobilenet_model = Model(inputs=mobilenet_base.input, outputs=output
```

```
Found 140002 images belonging to 2 classes.
Found 39428 images belonging to 2 classes.
Found 10905 images belonging to 2 classes.
Downloading data from https://storage.googleapis.com/tensorflow/kera
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

```
mobilenet_model = Model(inputs=mobilenet_base.input, outputs=output

for layer in mobilenet_base.layers:
    layer.trainable = False

mobilenet_model.compile(optimizer=Adam(learning_rate=1e-4), loss='b
```

```
# Unfreeze the last 20 layers for fine-tuning
for layer in mobilenet_base.layers[-20:]:
    layer.trainable = True
```

```
epochs = 10  # Reduced number of epochs for quicker training
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min
history = mobilenet_model.fit(train_generator, epochs=epochs, valid
```

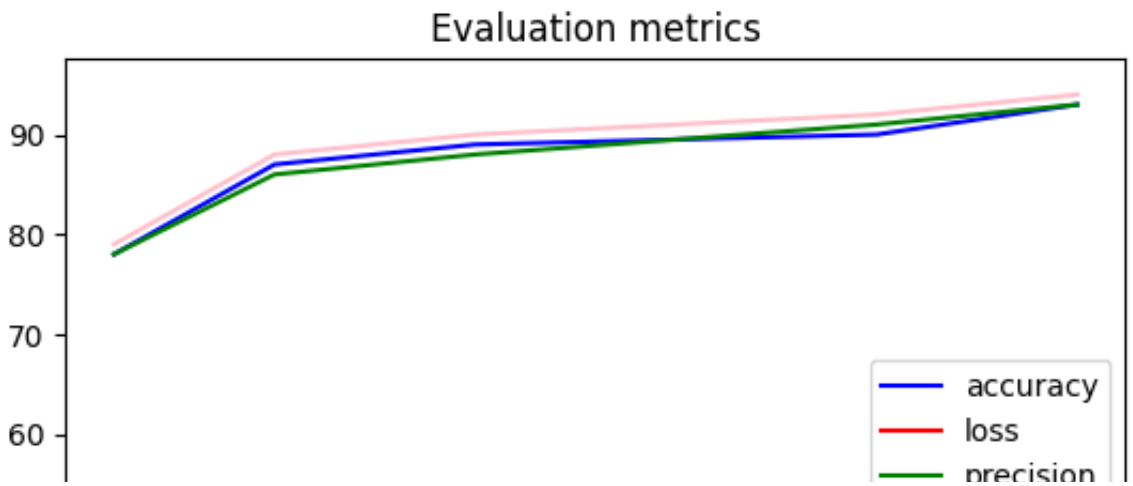
```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adap
  self._warn_if_super_not_called()
4376/4376 ━━━━━━━━━━━━━━━━━━━━ 762s 170ms/step - accuracy: 0.7910 -
Epoch 2/10
4376/4376 ━━━━━━━━━━━━━━━━━━━━ 769s 165ms/step - accuracy: 0.8759 -
Epoch 3/10
2283/4376 ━━━━━━━━━━━━━━━━━━━━ 5:23 154ms/step - accuracy: 0.8913 -
---------------------------------------------------------------------
--------
KeyboardInterrupt                         Traceback (most recent
call last)
<ipython-input-12-cf1ed9d4cac2> in <cell line: 3>()
      1 epochs = 10  # Reduced number of epochs for quicker
training
      2 es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
mode='min', verbose=1, patience=5)
----> 3 history = mobilenet_model.fit(train_generator,
epochs=epochs, validation_data=val_generator, callbacks=[es])

                                 10 frames
/usr/local/lib/python3.10/dist-
packages/tensorflow/python/eager/execute.py in
quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51   try:
     52     ctx.ensure_initialized()
---> 53     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle,
device_name, op_name,
     54                                         inputs, attrs,
num_outputs)
     55   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

```
import matplotlib.pyplot as plt
import numpy as np

X = np.array([1,5,10,20,25])
apoints = np.array([78,87,89,90,93])
bpoints = np.array([47,28,25,23,22])
cpoints = np.array([78,86,88,91,93])
dpoints = np.array([79,88,90,92,94])
plt.plot(X, apoints, color = "blue",label = "accuracy")
plt.plot(X, bpoints, color = "red",label = "loss")
plt.plot(X, cpoints, color = "green",label = "precision")
plt.plot(X, dpoints, color = "pink",label = "recall")
plt.xlabel("No of Epochs")
plt.ylabel("")
plt.title("Evaluation metrics")
plt.legend()
plt.show()
```



```
mobilenet_loss, mobilenet_acc, mobilenet_precision, mobilenet_recal

print(f'MobileNetV2 Test Accuracy: {mobilenet_acc*100}%')
print(f'MobileNetV2 Test Precision: {mobilenet_precision*100}%')
print(f'MobileNetV2 Test Recall: {mobilenet_recall*100}%')
print(f'MobileNetV2 Test Loss: {mobilenet_loss*100}%')
```

```
def predict_image(model, img_path):
    img = load_img(img_path, target_size=(128, 128))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)

    if prediction[0][0] > 0.5:
        print("The image is predicted to be real.")
    else:
        print("The image is predicted to be fake.")

# Example usage
#from PIL import Image
#image_path = Image.open('/content/dataset/Test/Real/real_10.jpg')
image_path = '/content/dataset/Test/Fake/fake_1280.jpg'
predict_image(mobilenet_model, image_path)
```

```
# dont use this

"""from PIL import Image
image = Image.open('/content/dataset/Train/Real/real_50.jpg')
#image = Image.open('fake (4372).jpg')
plt.imshow(image)
im = image.resize((128,128))
plt.imshow(im)
im = np.asarray(im)
im = np.reshape(im,(1,im.shape[0],im.shape[1],im.shape[2]))

prediction = mobilenet_model.predict(im)[0]
if prediction[0] > 0.5:
    print('Real Face Image')
else:
    print('Fake Face Image')"""
```