```
# !unzip /content/drive/MyDrive/gender_dataset_face-20220319T023558Z-001.zip -d /content/drive/MyDrive/gender_dataset_
```

Start coding or generate with AI.

```
import os
os.makedirs("Model_weights", exist_ok=True)
```

```
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import img_to_array
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
```

```
from keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Activation
from keras.layers import Flatten

from keras.layers import Dropout
from keras.layers import Dense
from tensorflow.keras import datasets, layers, models
from keras import backend as K
from keras import models
from keras.applications.vgg16 import VGG16
```

Double-click (or enter) to edit

```
class SmallerVGGNet:
    @staticmethod
    def vgg_net(width, height, depth, classes):

            conv_base = VGG16(weights='imagenet',
                          include_top=False,
                          input_shape=(height, width, depth))
            model = models.Sequential()

            model.add(conv_base)
            model.add(Flatten())

            model.add(Dense(1024, activation='relu'))
            model.add(BatchNormalization())
            model.add(Dropout(0.5))
            model.add(Dense(classes, activation='sigmoid'))
            return model
```

```
import matplotlib
matplotlib.use("Agg")  # Only needed in headless environments like servers
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import cv2
import os
import glob
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD, RMSprop
from tensorflow.keras.utils import img_to_array, to_categorical, plot_model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model

# If you have a custom model, make sure the import is correct
# from model.smallervggnet import SmallerVGGNet  # Uncomment if you have this file and model



from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

class SmallerVGGNet:
    @staticmethod
    def build(width, height, depth, classes):
        model = Sequential()

        # First CONV => RELU => POOL layer
        model.add(Conv2D(32, (3, 3), padding="same", activation="relu", input_shape=(height, width, depth)))
        model.add(MaxPooling2D(pool_size=(3, 3)))

        # Second CONV => RELU => POOL layer
        model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        # Third CONV => RELU => POOL layer
        model.add(Conv2D(128, (3, 3), padding="same", activation="relu"))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        # Flatten and add Fully Connected layers
        model.add(Flatten())
        model.add(Dense(256, activation="relu"))
        model.add(Dropout(0.5))  # Prevent overfitting
        model.add(Dense(classes, activation="softmax"))  # Output layer

        return model

model = SmallerVGGNet.build(width=img_dims[0], height=img_dims[1], depth=img_dims[2], classes=2)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
import glob
import os
import random
import numpy as np
import cv2
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
#from SmallerVGGNet import SmallerVGGNet

epochs = 50
lr = 1e-3
batch_size = 32
img_dims = (96, 96, 3)

data = []
labels = []

# Load image files
image_files = [f for f in glob.glob('/content/drive/MyDrive/ARCHIVE/gender_dataset_face' + "/**/*", recursive=True)
               if not os.path.isdir(f)]
random.seed(42)
random.shuffle(image_files)
```

```
    # Debug: Check if images are found
    if len(image_files) == 0:
        raise ValueError("No images found! Check dataset path.")

    # Process images
    for img in image_files:
        image = cv2.imread(img)
        if image is None:
            print(f"Skipping unreadable image: {img}")
            continue  # Skip corrupted images

        image = cv2.resize(image, (96, 96))
        image = img_to_array(image)
        data.append(image)

        label = os.path.basename(os.path.dirname(img))  # Extract folder name
        label = 1 if label.lower() == "woman" else 0
        labels.append([label])

    # Check dataset size before splitting
    if len(data) == 0:
        raise ValueError("No valid images loaded! Check dataset.")

    # Preprocessing
    data = np.array(data, dtype="float32") / 255.0
    labels = np.array(labels)

    # Train-test split
    trainX, testX, trainY, testY = train_test_split(data, labels, test_size=0.2, random_state=2)
    trainY = to_categorical(trainY, num_classes=2)
    testY = to_categorical(testY, num_classes=2)

    # Data augmentation
    aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
                             height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                             horizontal_flip=True, fill_mode="nearest")

    # Build and compile model
    model = SmallerVGGNet.build(width=img_dims[0], height=img_dims[1], depth=img_dims[2], classes=2)
    opt = Adam(learning_rate=lr, decay=lr / epochs)
    model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

    # Model checkpoint
    filepath = "epochs:{epoch:03d}-val_accuracy:{val_accuracy:.3f}.keras"  # Use .keras instead of .hdf5
    checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,
                                 save_best_only=True, mode='max')
    callbacks_list = [checkpoint]


    # Train model
    history = model.fit(aug.flow(trainX, trainY, batch_size=batch_size),
                        validation_data=(testX, testY),
                        steps_per_epoch=len(trainX) // batch_size,
                        epochs=epochs, verbose=1, callbacks=callbacks_list)
```

```
Epoch 41/50
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9243 - loss: 0.1793
Epoch 41: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 10s 97ms/step - accuracy: 0.9243 - loss: 0.1794 - val_accuracy: 0.9350 - val_loss: 0.
Epoch 42/50
 1/92 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8750 - loss: 0.2290
Epoch 42: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8750 - loss: 0.2290 - val_accuracy: 0.9242 - val_loss: 0.19
Epoch 43/50
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step - accuracy: 0.9258 - loss: 0.1902
Epoch 43: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 8s 79ms/step - accuracy: 0.9258 - loss: 0.1902 - val_accuracy: 0.9337 - val_loss: 0.1
Epoch 44/50
 1/92 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 1.0000 - loss: 0.0636
Epoch 44: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0636 - val_accuracy: 0.9350 - val_loss: 0.19
Epoch 45/50
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step - accuracy: 0.9358 - loss: 0.1780
Epoch 45: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 9s 97ms/step - accuracy: 0.9357 - loss: 0.1782 - val_accuracy: 0.9378 - val_loss: 0.1
Epoch 46/50
 1/92 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.9375 - loss: 0.1278
Epoch 46: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9375 - loss: 0.1278 - val_accuracy: 0.9337 - val_loss: 0.15
Epoch 47/50
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.9304 - loss: 0.1788
Epoch 47: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 9s 97ms/step - accuracy: 0.9305 - loss: 0.1787 - val_accuracy: 0.9364 - val_loss: 0.1
Epoch 48/50
 1/92 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.8438 - loss: 0.3173
Epoch 48: val_accuracy did not improve from 0.94317
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8438 - loss: 0.3173 - val_accuracy: 0.9378 - val_loss: 0.15
Epoch 49/50
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 131ms/step - accuracy: 0.9275 - loss: 0.1671
Epoch 49: val_accuracy improved from 0.94317 to 0.94993, saving model to epochs:049-val_accuracy:0.950.keras
92/92 ━━━━━━━━━━━━━━━━━━━━ 14s 138ms/step - accuracy: 0.9276 - loss: 0.1670 - val_accuracy: 0.9499 - val_loss: 0
Epoch 50/50
 1/92 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.9062 - loss: 0.1510
Epoch 50: val_accuracy did not improve from 0.94993
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9062 - loss: 0.1510 - val_accuracy: 0.9486 - val_loss: 0.12
```

```python
#from google.colab import drive
#drive.mount('/content/drive')   if drive is not mounteded
```

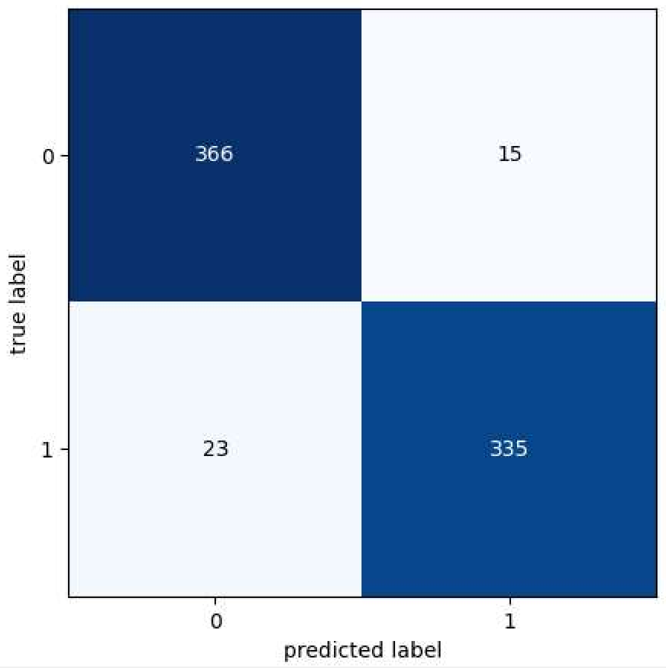
```python
import matplotlib.pyplot as plt

# Ensure training history exists before plotting
if history and hasattr(history, 'history'):
    # Check available keys
    keys = history.history.keys()

    # Determine correct accuracy keys
    acc_key = 'accuracy' if 'accuracy' in keys else 'acc'
    val_acc_key = 'val_accuracy' if 'val_accuracy' in keys else 'val_acc'
    loss_key = 'loss'
    val_loss_key = 'val_loss'

    # Plot Accuracy
    plt.figure(figsize=(10, 5))
    plt.plot(history.history[acc_key], label='Train Accuracy')
    plt.plot(history.history[val_acc_key], label='Validation Accuracy')
    plt.title('Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Plot Loss
    plt.figure(figsize=(10, 5))
    plt.plot(history.history[loss_key], label='Train Loss')
    plt.plot(history.history[val_loss_key], label='Validation Loss')
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Optional: Save the plots
    plt.savefig('/content/drive/MyDrive/ARCHIVE/result.png')

else:
    print("Error: Training history is empty or model training was not completed.")
```

from google.colab import drive drive.mount('/content/drive')

```
from sklearn.metrics import confusion_matrix
pred = model.predict(testX)
pred = np.argmax(pred,axis = 1)
y_true = np.argmax(testY,axis = 1)
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step
```

```
CM = confusion_matrix(y_true, pred)
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=CM ,  figsize=(5, 5))
# plt.show()
fig
```



```
!pip install cvlib
```

```
Requirement already satisfied: cvlib in /usr/local/lib/python3.11/dist-packages (0.2.7)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from cvlib) (1.26.4)
Requirement already satisfied: progressbar in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.5)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.32.3)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from cvlib) (11.1.0)
Requirement already satisfied: imageio in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.37.0)
Requirement already satisfied: imutils in /usr/local/lib/python3.11/dist-packages (from cvlib) (0.5.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib) (3.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib
```

```
# from keras.preprocessing.image import img_to_array
from tensorflow.keras.utils import img_to_array
from keras.models import load_model
# from keras.utils import get_file
import numpy as np
import argparse
import cv2
import os
!pip install cvlib
import cvlib as cv
```

```
Requirement already satisfied: cvlib in /usr/local/lib/python3.11/dist-packages (0.2.7)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from cvlib) (1.26.4)
Requirement already satisfied: progressbar in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.5)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.32.3)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from cvlib) (11.1.0)
Requirement already satisfied: imageio in /usr/local/lib/python3.11/dist-packages (from cvlib) (2.37.0)
Requirement already satisfied: imutils in /usr/local/lib/python3.11/dist-packages (from cvlib) (0.5.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib) (3.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->cvlib
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_
```

```
img= '/content/drive/MyDrive/ARCHIVE/man1.jpg'
image = cv2.imread(img)
# cv2.imread('/conte')
```

```
image = cv2.imread('/content/drive/MyDrive/ARCHIVE/women2.jpeg')  #give any image from computer

if image is None:
    print("Could not read input image")
    exit()

# load pre-trained model
model = load_model("/content/drive/MyDrive/ARCHIVE/epochs_049-val_accuracy_0.950.keras")   # check most accuracy and los

# detect faces in the image
face, confidence = cv.detect_face(image)

classes = ['man','woman']
# loop through detected faces
for idx, f in enumerate(face):

     # get corner points of face rectangle
    (startX, startY) = f[0], f[1]
    (endX, endY) = f[2], f[3]

    # draw rectangle over face
    cv2.rectangle(image, (startX,startY), (endX,endY), (0,255,0), 2)

    # crop the detected face region
    face_crop = np.copy(image[startY:endY,startX:endX])

    # preprocessing for gender detection model
    face_crop = cv2.resize(face_crop,(96,96))
    face_crop = face_crop.astype("float") / 255.0
    face_crop = img_to_array(face_crop)
    face_crop = np.expand_dims(face_crop, axis=0)

    # apply gender detection on face
    conf = model.predict(face_crop)[0]
    print(conf)
    print(classes)

    # get label with max accuracy
```

```
        idx = np.argmax(conf)
        label = classes[idx]

        label = "{}: {:.2f}%".format(label, conf[idx] * 100)

        Y = startY - 10 if startY - 10 > 10 else startY + 10

        # write label and confidence above face rectangle
        if conf[idx] * 100 > 50.0:
            cv2.putText(image, label, (startX, Y),  cv2.FONT_HERSHEY_SIMPLEX,
                    0.7, (0, 255, 0), 1)
    # cv2.imwrite('/content/',image)
    # display output
    # cv2.imshow("gender detection", image)

    # # press any key to close window
    # cv2.waitKey(0)

    # # save output
    cv2.imwrite("gender_detection.jpg", image)

    # # release resources
    # cv2.destroyAllWindows()
```

```
WARNING:tensorflow:6 out of the last 8 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_ste
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 303ms/step
[2.3005035e-07 9.9999976e-01]
['man', 'woman']
True
```

Thank you...

Double-click (or enter) to edit