```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
# Define constants
img_height = 128
img_width = 128
batch_size = 32
epochs = 10
num_classes = 10
# Load the "CelebA" dataset
df = pd.read_csv('list_attr_celeba.csv')
```

```
df = df.sample(frac=1).reset_index(drop=True) # shuffle the dataset
df['image_id'] = df['image_id'].apply(lambda x: x[:-4]) # remove file extension from image IDs
df_train = df[:int(len(df)*0.8)] # 80% for training
df_val = df[int(len(df)*0.8):int(len(df)*0.9)] # 10% for validation
df_test = df[int(len(df)*0.9):] # 10% for testing

# Define data generators for training, validation, and testing sets
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_dataframe(
dataframe=df_train,
directory='img_align_celeba',
x_col='image_id',
y_col='Smiling',
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='binary')
val_generator = val_datagen.flow_from_dataframe(
dataframe=df_val,
directory='img_align_celeba',
x_col='image_id',
y_col='Smiling',
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='binary')
test_generator = test_datagen.flow_from_dataframe(
dataframe=df_test,
directory='img_align_celeba',
x_col='image_id',
y_col='Smiling',
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='binary')
# Define the neural network model
model = Sequential([
Conv2D(32, (3,3), activation='relu', input_shape=(img_height, img_width, 3)),
MaxPooling2D((2,2)),
Conv2D(64, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D((2,2)),
Flatten(),
Dropout(0.5),
Dense(512, activation='relu'),
Dense(num_classes, activation='sigmoid')
])
# Compile the model
model.compile(optimizer='adam',
```

```
loss='binary_crossentropy',
metrics=['accuracy'])
# Train the model
history = model.fit(train_generator,
epochs=epochs,
validation_data=val_generator)
# Evaluate the model on the test set
loss, accuracy = model.evaluate(test_generator)
print("Test accuracy:", accuracy)
# Predict the smiling attribute of a sample image
img = cv2.imread('sample_image.jpg')
img = cv2.resize(img, (img_height, img_width))
img = np.expand_dims(img, axis=0)
```