سامانه تشخیص لبخند در ویدیو

در دیتاست خام 4000 تا عکس داریم که 2162 تای اول لبخند و بقیه غیر لبخند است.

برای split و preprocess دیتاست به این صورت عمل می کنیم که در پوشه dataset دو پوشه test و train می سازیم و کل دیتاها را به نسبت 20 به 80 تقسیم می کنیم.

هر کدوم از این پوشه ها خود شامل smile و non-smile است.

لیست package های استفاده شده :

```
[7]
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
import pickle
```

سپس مسیر های پوشه ها را تعیین می کنیم.

```
[8]
train_dir = 'dataset/train'
valid_dir = 'dataset/test'
```

سپس توسط image generator تمام عکس ها رو خونده. عکس های train در train generator و عکس های test در test generator ذخیره می شوند

```
batch_size = 8
target_size = (256, 256)  # Adjust the target size according to your requirements
color_mode = 'grayscale'
class_mode = 'binary'

train_data_generator = ImageDataGenerator(rescale=1.0 / 255)
valid_data_generator = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_data_generator.flow_from_directory(
    train_dir,
    target_size=target_size,
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode=class_mode
)

valid_generator = valid_data_generator.flow_from_directory(
    valid_dir,
    target_size=target_size,
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode=class_mode
)
```

[9]

```
Found 3201 images belonging to 2 classes.
Found 799 images belonging to 2 classes.
```

مزیت استفاده از image generator این است که همه عکسای خوانده شده را به فرمت (256, 256) در میاورد و grayscale می کند بنابراین فرمت هر عکس به صورت (256, 256, 1) می شود.

و در batch های 8تایی به مدل داده می شود.

```
# each batch
print(images.shape)
# every image in each batch
print(train_generator.image_shape)
```

[11]

```
(8, 256, 256, 1)
(256, 256, 1)
```

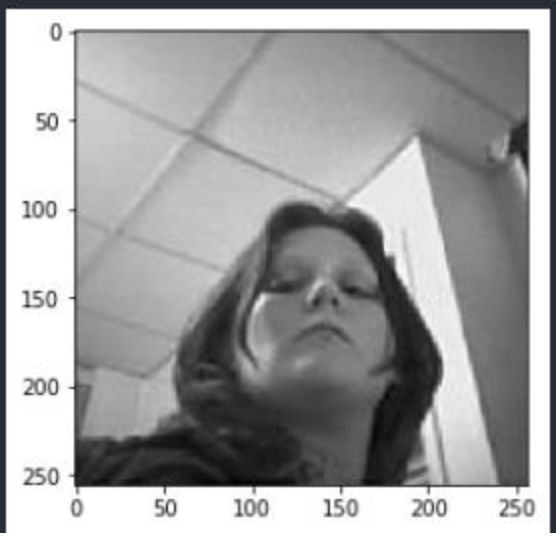
نمایش یک batch رندوم 8تایی.

```
# Generate a random batch of images and labels
images, labels = next(train_generator)

# Display the images from the batch
num_images = images.shape[0]

for i in range(num_images):
    plt.imshow(images[i], cmap='gray')
    plt.show()
```

[10]



در مرحله بعد مدل CNN رو میسازیم:

```
[12]
# Define the CNN model
model = models.Sequential()

# Convolutional layers
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 1)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the feature maps
model.add(layers.Flatten())

# Dense layers
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))  # Binary classification

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Display the model summary
model.summary()
```

مدل ما دارای 9 لایه است که به ترتیب convolution, max pooling, convolution, max pooling, convolution, max pooling, flatten, dense, dense است

.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 254, 254, 32)      320

 max_pooling2d_3 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_4 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_4 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 115200)            0

 dense_2 (Dense)             (None, 128)               14745728

 dense_3 (Dense)             (None, 1)                 129

...
Total params: 14,838,529
Trainable params: 14,838,529
Non-trainable params: 0
```

سپس باید مدل را fit کنیم.

برای اینکه overfit نداشته باشیم ابتدا epoch 6 می زنیم و دقت را اندازه گیری می کنیم.

```
model.fit(train_generator, steps_per_epoch = 401, epochs = 6,
          validation_data = valid_generator, validation_steps = 401, verbose = 1)
```
[13]

```
Epoch 1/6
401/401 [==============================] - ETA: 0s - loss: 0.6995 - accuracy: 0.5380WARNING:tensorflow:Your input ran out of data; interrupting
401/401 [==============================] - 226s 561ms/step - loss: 0.6995 - accuracy: 0.5380 - val_loss: 0.6793 - val_accuracy: 0.5569
Epoch 2/6
401/401 [==============================] - 219s 547ms/step - loss: 0.6781 - accuracy: 0.5720
Epoch 3/6
401/401 [==============================] - 219s 546ms/step - loss: 0.6700 - accuracy: 0.5964
Epoch 4/6
401/401 [==============================] - 212s 529ms/step - loss: 0.6573 - accuracy: 0.6154
Epoch 5/6
401/401 [==============================] - 217s 540ms/step - loss: 0.5973 - accuracy: 0.6813
Epoch 6/6
401/401 [==============================] - 232s 580ms/step - loss: 0.4542 - accuracy: 0.7779

<keras.callbacks.History at 0x1da642c6ef0>
```

سپس دیتای test را به مدل می دهیم و loss و accuracy را اندازه گیری می کنیم.

```
loss, accuracy = model.evaluate(valid_generator)

print("Test loss:", loss)
print("Test accuracy:", accuracy)
```
[14]

```
100/100 [==============================] - 13s 127ms/step - loss: 0.8458 - accuracy: 0.6070
Test loss: 0.8458103537559509
Test accuracy: 0.6070087552070618
```

سپس دو epoch دیگر می زنیم.

```
model.fit(train_generator, steps_per_epoch = 401, epochs = 2,
          validation_data = valid_generator, validation_steps = 401, verbose = 1)
```
[15]

```
Epoch 1/2
401/401 [==============================] - ETA: 0s - loss: 0.2245 - accuracy: 0.9044WARNING:tensorflow:Your input ran out of data; interrup
401/401 [==============================] - 257s 640ms/step - loss: 0.2245 - accuracy: 0.9044 - val_loss: 1.1443 - val_accuracy: 0.5920
Epoch 2/2
401/401 [==============================] - 244s 608ms/step - loss: 0.0750 - accuracy: 0.9738

<keras.callbacks.History at 0x1da6429f9a0>
```

```
loss, accuracy = model.evaluate(valid_generator)

print("Test loss:", loss)
print("Test accuracy:", accuracy)
```
[16]

```
100/100 [==============================] - 14s 137ms/step - loss: 1.8660 - accuracy: 0.5907
Test loss: 1.8659706115722656
Test accuracy: 0.5907384157180786
```

مشاهده می کنیم با اینکه دقت train بیشتر شده دقت test کمتر شده.

در این مرحله مدل به دست اومده رو توسط کتابخونه pickle اون رو سیو می کنیم.

```
# Save the trained model
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)
```
[36]

در یک نوتبوک دیگه داریم:

لیست package های استفاده شده :

```
import pickle
import cv2
from matplotlib import pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
```
[48] ✓ 0.0s

مدل به دست اومده رو لود می کنیم.

```
# Load the saved model
with open('model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)
```
[37]

در مرحله بعد مدل های آماده رو لود می کنیم.

first step is Load the face detection model

```
hog_face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
print(hog_face_cascade)
lbp_face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'lbpcascade_frontalface.xml')
print(lbp_face_cascade)
```

[18]

در مرحله بعد دیتا ورودی گرفته می شود.

وبکم شروع به کار کرده و زمانی که دکمه esc زده شود فریم سیو میشود.

```
# in this step we should receive a video and then we should extract picture and then checking
# capture frames from a camera
cap = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret, image = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = hog_face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        # To draw a rectangle in a face
        cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]

    # Display an image in a window
    cv2.imshow('image', image)

    # Wait for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
# at first we try that on a simple picture
```

[20]

در اینجا با استفاده از مدل های آماده ای که لود کرده بودیم صورت (های) تصویر را تشخیص می دهیم.

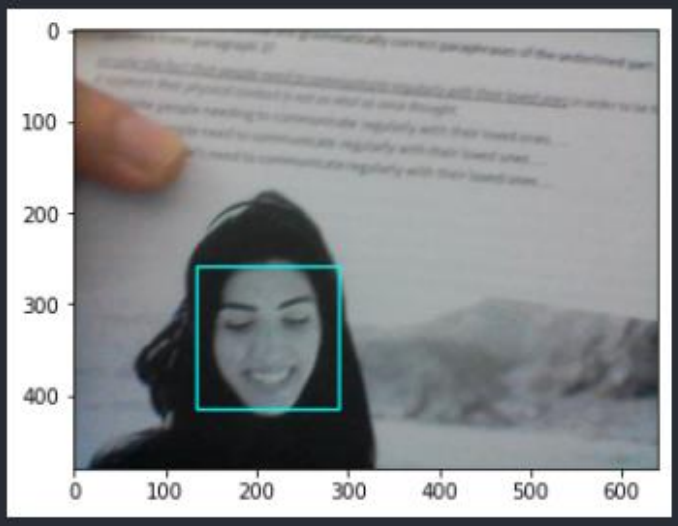
سپس دور اونا مستطیل می کشیم و جداشون می کنیم.

```
for (x,y,w,h) in faces:
    # To draw a rectangle in a face
    cv2.rectangle(image,(x,y),(x+w,y+h),(255,255,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
```

تصویر اولیه را نمایش می دهیم :

```
    # display the original pic
    image_show = cv2.cvtColor(image , cv2.COLOR_BGR2RGB)
    plt.imshow(image_show)
    plt.show()
```

[52] ✓ 0.3s



صورت های دیتکت شده را نمایش میدهیم :

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

ax1.axis('off')
ax1.imshow(roi_gray, cmap=plt.cm.gray)
ax1.set_title('grayscaled detected face')

ax2.axis('off')
ax2.imshow(roi_color)
ax2.set_title('colored detected face')
plt.show()
```

[25]

برای اینکه عکس ورودی را به فرمت مناسب در بیاوریم:

```
print(roi_color.shape)
```

[54] ✓ 0.0s

```
(156, 156, 3)
```

```
resized_image = cv2.resize(roi_color, (256, 256))
gray_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)
print(gray_image.shape)
final_image = gray_image.reshape((256, 256, 1))
print(final_image.shape)
```

[55] ✓ 0.0s

```
(256, 256)
(256, 256, 1)
```

```
expanded_image = np.expand_dims(final_image, axis=0)
print(expanded_image.shape)
```

[56] ✓ 0.0s

```
(1, 256, 256, 1)
```

```
predictions = loaded_model.predict(expanded_image)
print(predictions)
```

[62] ✓ 0.1s

```
1/1 [==============================] - 0s 81ms/step
[[1.]]
```

چون مساله binary classification است بنابراین 1 به نشانه لبخند بودن است.

همچنین می توان عکس تست را سیو کرد و با image generotor خواند و به مدل داد

```
# save image
status = cv2.imwrite('./test_image_folder/test/test_image.png', roi_color)
```

[57]

```
test_image_dir = './test_image_folder'
```

[58]

```
batch_size = 8
target_size = (256, 256)  # Adjust the target size according to your requirements
color_mode = 'grayscale'
class_mode = 'binary'

train_data_generator = ImageDataGenerator(rescale=1.0 / 255)
valid_data_generator = ImageDataGenerator(rescale=1.0 / 255)

test_image_generator = train_data_generator.flow_from_directory(
    test_image_dir,
    target_size=target_size,
    color_mode=color_mode,
    batch_size=batch_size,
    class_mode=class_mode
)
```

[59]

```
Found 1 images belonging to 1 classes.
```