گزارش تمرین ششم بینایی کامپیوتر سها کرد – 983212058

در این تمرین با داشتن یک تصویر با بافت های مختلف باید توسط یک CNN این بافت ها را جدا می‌کردیم. مجموعه دیتا را باید از تصویر خودمان جدا می‌کردیم.

خواندن تصویر اصلی

```
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.show()
```

ساخت ماسک

با استفاده از کد زیر ماسکی مناسب با 5 کلاس مختلف از textureهای موجود در عکس را می‌سازیم. شامل 4 مثلث و یک دایره. همینطور از رنگ‌های مجزایی برای هر کلاس استفاده شده است.

این کد یک تصویر را با استفاده از کتابخانه های cv2 و numpy و matplotlib ایجاد میکند و نمایش میدهد. در این کد، ابتدا یک ماسک سه کاناله RGB ایجاد میشود که به صورت پیش فرض مقادیر آن صفر است. سپس یک مربع با پارامترهای مشخص شده (مختصات بالا چپ و سایز) رسم میشود. سپس چهار مثلث با مربع مورد نظر به عنوان مرکز وجود در کوادرانتهای مختلف به وسیله آرایه های numpy ایجاد میشوند و روی ماسک رنگی متفاوت پر میشوند. در نهایت یک دایره سفید روی مرکز مربع رسم میشود. در نهایت تصویر ساخته شده با استفاده از matplotlib نمایش داده می‌شود.

کد:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Create an image with three color channels (RGB)
mask = np.zeros((image.shape[0], image.shape[1], 3), dtype=np.uint8)

# Define the square's parameters
x, y = 0, 0
```

```
size = image.shape[0]

# Draw the square
cv2.rectangle(mask, (x, y), (x + size, y + size), (255, 255, 255), 2)

# Calculate the coordinates for the triangle vertices
x_mid = x + size // 2
y_mid = y + size // 2

# Draw the triangles
triangle1 = np.array([[x, y], [x + size, y], [x_mid, y_mid]])
triangle2 = np.array([[x + size, y], [x + size, y + size], [x_mid, y_mid]])
triangle3 = np.array([[x, y + size], [x + size, y + size], [x_mid, y_mid]])
triangle4 = np.array([[x, y], [x, y + size], [x_mid, y_mid]])

# Draw the triangles on the mask with different colors
cv2.fillPoly(mask, [triangle1], (255, 0, 0))
cv2.fillPoly(mask, [triangle2], (0, 255, 0))
cv2.fillPoly(mask, [triangle3], (0, 0, 255))
cv2.fillPoly(mask, [triangle4], (255, 255, 0))

# Draw a white circle with a desired radius and center
radius = 90
cv2.circle(mask, (x_mid, y_mid), radius, (255, 255, 255), -1)

# Display the mask using matplotlib
plt.imshow(mask)
plt.axis('off')
plt.show()
```

خروجی:

همانطور که می‌بینید ماسک خروجی اندازه‌ای مشابه با تصویر اصلی دارد.

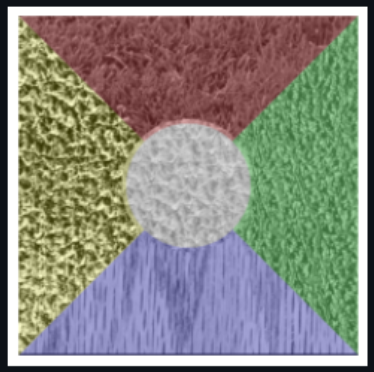
```
image.shape, mask.shape
✓ 0.1s
((491, 491), (491, 491, 3))
```

با اضافه کردن وزن‌دار تصویر بافت‌ها و ماسک ساختگی، می‌توانید مشاهده کنید که ناحیه هر کلاس دقیقا منطبق با ماسک است.

در اینجا، تصویر اصلی به صورت خاکستری COLOR_GRAY2RGB تبدیل شده و سپس به عنوان تصویر اول به addWeighted ارسال میشود. ماسک رنگی mask نیز به عنوان تصویر دوم وارد می‌شود. وزندهی مختلفی برای هر دو تصویر مشخص شده است (0.8 برای تصویر اول و 0.2 برای تصویر دوم) و پارامتر آخر (`0`) نیز برای تعیین کردن عدد ثابتی است که به تصویر نهایی اضافه می‌شود.

ر نهایت، تصویر نهایی به نام blended تولید شده و با استفاده از matplotlib نمایش داده میشود.

```
blended = cv2.addWeighted(cv2.cvtColor(image, cv2.COLOR_GRAY2RGB), 0.8, mask , 0.2, 0)
plt.imshow(blended)
plt.axis('off')
plt.show()
[6] ✓ 0.1s
```



تقسیم‌کردن تصویر به منظور ساخت دیتاست

تابع تقسیم‌بندی بر اساس تعداد سطر و ستون ورودی:

این تابع ماسک و تصویر اصلی را به عنوان ورودی می‌گیرد و آن‌ها را تکه‌تکه می‌کند و مجموعه‌ای از تصاویر ماسک و تصویر بافت را خروجی می‌دهد.

این کد یک تابع به نام break_image_into_sub_images تعریف می‌کند که یک تصویر و ماسک مربوط به آن را به تعداد سطرها و ستونهای مشخص شده تقسیم میکند و تصاویر کوچکتر را در لیست sub_imagesو ماسک های مربوطه را در لیست sub_masksذخیره می‌کند.

ابتدا ابعاد تصویر اصلی و مقدار تعداد سطرها و ستونها به عنوان ورودی دریافت میشود. سپس اندازه هر تصویر کوچکتر را براساس تعداد سطرها و ستونها محاسبه میکند.

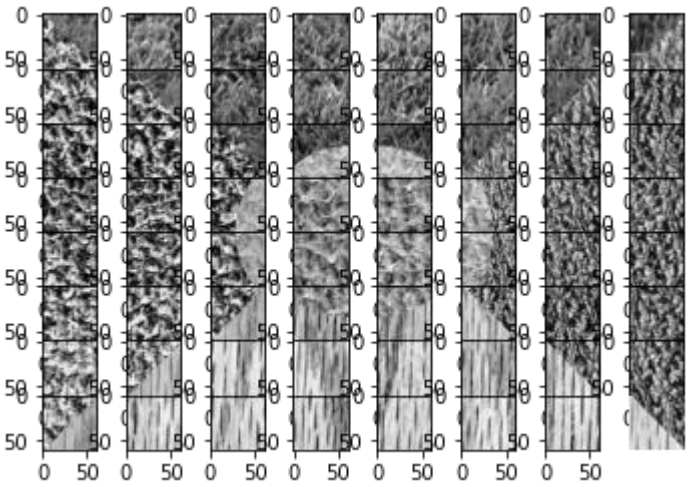
در دو حلقه forتعداد سطرها و ستونها را پیمایش میکند. برای هر مقدار از rowو col، یک بخش کوچک از تصویر اصلی و ماسک استخراج میشود. محدوده ی مورد نیاز برای استخراج بخش مورد نظر با استفاده از محاسبات ساده از y_start تا y_endو x_start تا x_end تعیین می‌شود. سپس بخش استخراج شده از تصویر و ماسک در لیست های sub_images و sub_masks قرار می‌گیرند.

در نهایت، لیست های sub_images و sub_masks به عنوان خروجی تابع برگردانده می‌شوند.

```
def break_image_into_sub_images(image, mask, rows, cols):
    # Get the size of each smaller image
    height, width = image.shape[0:2]
    sub_image_height = height // rows
    sub_image_width = width // cols

    # Store the smaller sub_images and sub_masks in lists
    sub_images = []
    sub_masks = []
    for row in range(rows):
        for col in range(cols):
            # Extract a small section from the image
            y_start = row * sub_image_height
            y_end = y_start + sub_image_height
            x_start = col * sub_image_width
            x_end = x_start + sub_image_width
            sub_image = image[y_start:y_end, x_start:x_end]
            sub_mask = mask[y_start:y_end, x_start:x_end, :]
            sub_images.append(sub_image)
            sub_masks.append(sub_mask)

    return sub_images, sub_masks
rows = 8
cols = 8

sub_images, sub_masks = break_image_into_sub_images(image, mask, rows, cols)
images = []
masks = []
fig, axs = plt.subplots(rows, cols)
```

```
fig.subplots_adjust(hspace=0, wspace=0)
for i, sub_image in enumerate(sub_images):
    ax = axs[i // cols, i % cols]
    ax.imshow(sub_image, cmap='gray')
    images.append(sub_image)
    masks.append(sub_masks[i])

images = np.array(images)
masks = np.array(masks)

plt.show()
```

تصاویر تکه‌تکه شده‌ی حاصل:



تصاویر حاصل شده و ابعاد آن‌ها:

این قطعه کد اندازه های تصاویر و ماسک ها را نمایش میدهد و سپس اندازه های آنها را تغییر میدهد.
در ابتدا، اندازه های اصلی تصاویر و ماسکها با استفاده از shape نمایش داده میشود.
سپس، اندازه های تصاویر و ماسک ها را تغییر میدهد. این تغییر بر روی بعد اول (تعداد تصاویر) و دو بعد دیگر (ارتفاع و عرض) اعمال میشود. با استفاده از عملگر تقسیم صحیح (`//`)، اندازه هر بعد را به نصف تقسیم میکند و سپس با استفاده از عملگر (`:`) و تقسیم صحیح دوباره، اندازه های جدید را تعیین میکند.

سپس، اندازه های جدید تصاویر و ماسکها با استفاده از `shape`` مجدداً نمایش داده میشود.

```
# Original shapes
print("Original image shape:", images.shape)
print("Original mask shape:", masks.shape)
```

```
# Drop a pixel if dimensions are odd
images = images[:, :images.shape[1] // 2 * 2, :images.shape[2] // 2 * 2]
masks = masks[:, :masks.shape[1] // 2 * 2, :masks.shape[2] // 2 * 2, :]

# New shapes
print("New image shape:", images.shape)
print("New mask shape:", masks.shape)
```

خروجی کد بالا:

```
Original image shape: (64, 61, 61)
Original mask shape: (64, 61, 61, 3)
New image shape: (64, 60, 60)
New mask shape: (64, 60, 60, 3)
```

به دست آوردن لیبل هر یک از تصاویر

با استفاده از کد زیر هر تصویر بافت و ماسک آن که به صورت ورودی گرفته شود، شماره‌ی کلاس آن بر اساس رنگ غالب در ماسک آن، محاسبه می‌شود.

کلاس رنگ‌های مختلف(بافت‌های مختلف) به صورت زیر است:

سفید: 0

زرد: 1

قرمز: 2

سبز: 3

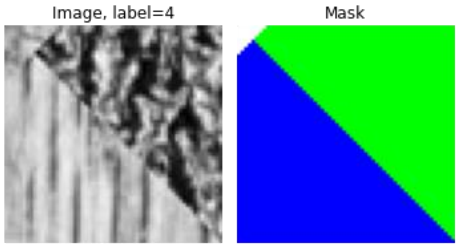
آبی: 4

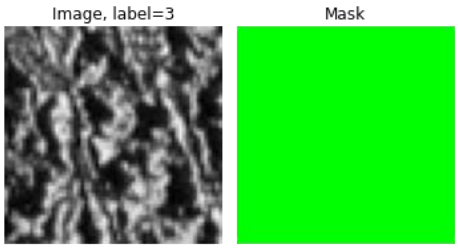
```
color_labels = ['blue', 'green', 'red', 'yellow', 'white']

def get_labels(images, masks):
    labels = np.zeros(images.shape[0])

    # Iterate through each mask image
    for i, mask in enumerate(masks):
        pixel_counts = {label: 0 for label in color_labels}

        white_pixels = np.all(mask == [255, 255, 255], axis=-1)
        count = np.sum(white_pixels)
```

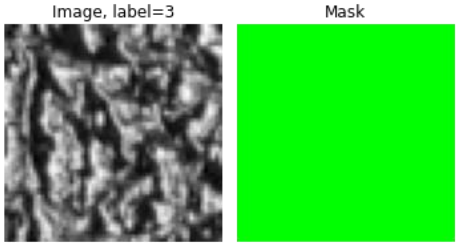
```
        pixel_counts['white'] = count

        yellow_pixels = np.all(mask == [255, 255, 0], axis=-1)
        count = np.sum(yellow_pixels)
        pixel_counts['yellow'] = count

        red_pixels = np.all(mask == [255, 0, 0], axis=-1)
        count = np.sum(red_pixels)
        pixel_counts['red'] = count

        green_pixels = np.all(mask == [0, 255, 0], axis=-1)
        count = np.sum(green_pixels)
        pixel_counts['green'] = count

        blue_pixels = np.all(mask == [0, 0, 255], axis=-1)
        count = np.sum(blue_pixels)
        pixel_counts['blue'] = count

        # Determine the label with the maximum pixel count
        max_label = max(pixel_counts, key=pixel_counts.get)

        # Assign the label value based on the color
        if max_label == 'white':
            labels[i] = 0
        elif max_label == 'yellow':
            labels[i] = 1
        elif max_label == 'red':
            labels[i] = 2
        elif max_label == 'green':
            labels[i] = 3
        elif max_label == 'blue':
            labels[i] = 4
    labels = labels.astype('uint8')
    return labels
```

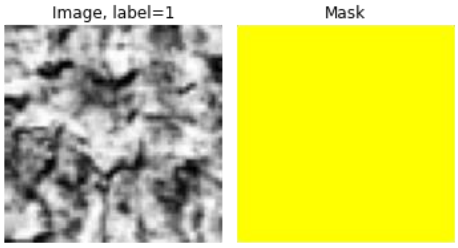
رسم تعدادی از تصاویر و کلاس آن‌ها:

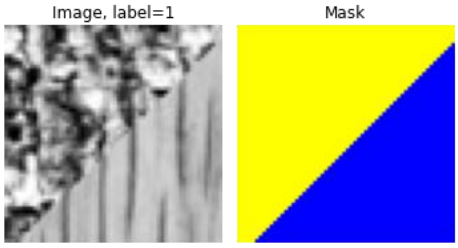
(در صفحه بعدی آورده شده)


Image, label=4
Mask


Image, label=3
Mask


Image, label=3
Mask


Image, label=1
Mask


Image, label=1
Mask

تقسیم داده‌ها به train و test:

```
import numpy as np
from sklearn.model_selection import train_test_split

test_size = 0.2
train_images, test_images, train_masks, test_masks = train_test_split(images, masks, test_size=test_size, random_state=42)
train_labels = get_labels(train_images, train_masks)
test_labels = get_labels(test_images, test_masks)

print("Train set shapes: ", train_images.shape, train_masks.shape)
print("Test set shapes: ", test_images.shape, test_masks.shape)
```

[15] ✓ 0.0s

```
Train set shapes:  (51, 60, 60) (51, 60, 60, 3)
Test set shapes:  (13, 60, 60) (13, 60, 60, 3)
```

افزودن داده‌ها(augmentation):

```
import imgaug.augmenters as iaa

augmentation_seq = iaa.Sequential([
    iaa.Flipud(0.5),             # Flip images vertically
    iaa.Affine(rotate=(-10, 10)), # Rotate images within the range -10 to 10 degrees
    iaa.GaussianBlur(sigma=(0, 1.0)),  # Apply Gaussian blur with sigma between 0 and 1.0
])

augmented_train_images = []
augmented_train_labels = []

augmentation_factor = 4

for _ in range(augmentation_factor):
    augmented_data = augmentation_seq.augment(images=train_images)
    augmented_train_images.extend(augmented_data)
    augmented_train_labels.extend(train_labels)

# Convert the augmented data to NumPy arrays
augmented_train_images = np.array(augmented_train_images)
augmented_train_labels = np.array(augmented_train_labels)

print("Augmented images shape: ", augmented_train_images.shape)
print("Augmented labels shape: ", augmented_train_labels.shape)
```

[16] ✓ 2.5s

```
Augmented images shape:  (204, 60, 60)
Augmented labels shape:  (204,)
```

همانطور که می‌بینید 51 تصویر train با ضریبا مشخص‌شده‌ی 4، 4 برابر شد و حالا 204 تصویر برای train داریم.

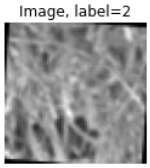
تغییر مقیاس تصاویر به [0,1]:

```
augmented_train_images = augmented_train_images / 255.0
test_images = test_images /255.0
```

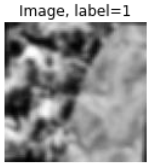
رسم تعدادی از تصاویر augment شده:

(در صفحه بعد رسم شده)

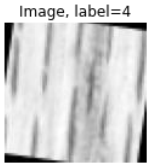
Image, label=2



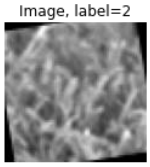
Image, label=1



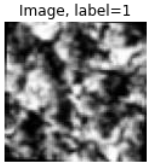
Image, label=4



Image, label=2

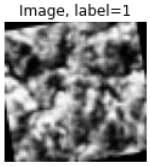


Image, label=1

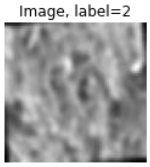


Image, label=1

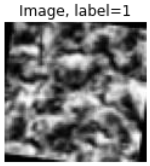
Image, label=1



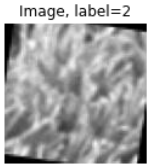
Image, label=2



Image, label=1



Image, label=2



تعریف مدل CNN

```
from tensorflow import keras
from tensorflow.keras import layers

# Define the input shape
input_shape = (60, 60, 1)

# Define the number of classes
num_classes = 5

# Create the CNN model
model = keras.Sequential([
    layers.Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=input_shape),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])

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

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

این مدل لایه‌های مختلفی دارد که قابل مشاهده است. از بهینه‌ساز Adam استفاده کردم. همینطور برای محاسبه‌ی loss از فرمول sparse_categorical_crossentropy استفاده کردم چون تصاویر لیبل به صورت hot-ones خواهد بود. به این صورت که اگر تصویری از کلاس 1 بود، ورودی این تصویر به مدل به صورت [0,1,0,0,0] خواهد بود.

آموزش مدل

```
# Define the number of epochs and batch size
epochs = 25
batch_size = 32

# Fit the model to the training data
history = model.fit(augmented_train_images, augmented_train_labels, batch_size=batch_size, epochs=epochs, verbose=1)
```

✓ 15.1s

```
Epoch 1/25
7/7 [==============================] - 1s 68ms/step - loss: 1.5342 - accuracy: 0.2451
Epoch 2/25
7/7 [==============================] - 0s 66ms/step - loss: 1.3183 - accuracy: 0.4853
Epoch 3/25
7/7 [==============================] - 1s 98ms/step - loss: 1.0989 - accuracy: 0.6176
Epoch 4/25
7/7 [==============================] - 1s 107ms/step - loss: 0.9585 - accuracy: 0.6176
Epoch 5/25
7/7 [==============================] - 1s 115ms/step - loss: 0.8464 - accuracy: 0.6912
Epoch 6/25
7/7 [==============================] - 1s 109ms/step - loss: 0.7140 - accuracy: 0.8137
Epoch 7/25
7/7 [==============================] - 0s 63ms/step - loss: 0.5961 - accuracy: 0.8039
Epoch 8/25
7/7 [==============================] - 0s 64ms/step - loss: 0.5058 - accuracy: 0.8088
Epoch 9/25
7/7 [==============================] - 1s 71ms/step - loss: 0.4455 - accuracy: 0.8431
Epoch 10/25
7/7 [==============================] - 1s 72ms/step - loss: 0.4637 - accuracy: 0.8284
Epoch 11/25
7/7 [==============================] - 1s 82ms/step - loss: 0.3453 - accuracy: 0.8873
Epoch 12/25
7/7 [==============================] - 1s 85ms/step - loss: 0.2633 - accuracy: 0.9020
Epoch 13/25
...
Epoch 24/25
7/7 [==============================] - 0s 70ms/step - loss: 0.0919 - accuracy: 0.9608
Epoch 25/25
7/7 [==============================] - 0s 69ms/step - loss: 0.1471 - accuracy: 0.9314
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

ارزیابی مدل بر روی مجموعه test

```
loss, accuracy = model.evaluate(test_images, test_labels)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

✓ 0.3s

```
1/1 [==============================] - 0s 170ms/step - loss: 0.4387 - accuracy: 0.7692
Test Loss: 0.4387425184249878
Test Accuracy: 0.7692307829856873
```

دقت این مدل فقط با 200 تصویر و 25 epoch به 93 درصد در داده‌های train و 77 درصد در داده‌های test رسیده است.

نمایش داده‌های test و کلاس پیشبینی شده برای هر یک از تصاویر

کلاس‌های واقعی هر یک از تصاویر test و کلاس‌های پیشبینی‌شده‌ی آن‌ها به صورت زیر است:

```
predicted_classes = np.argmax(predicted_test_labels, axis=1).astype(np.uint8)
predicted_classes
```
✓ 0.0s

```
array([0, 4, 1, 0, 2, 2, 1, 2, 1, 4, 1, 1, 1], dtype=uint8)
```

```
true_classes = get_labels(test_images, test_masks)
true_classes
```
✓ 0.0s

```
array([4, 4, 1, 4, 2, 0, 1, 2, 1, 4, 1, 1, 1], dtype=uint8)
```

نمایش داده‌های test به همراه کلاس آن‌ها در ادامه آمده است:

(در صفحه بعد آمده است)

Image, class=4

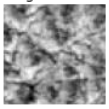
Image, class=4

Image, class=1

Image, class=4

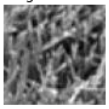
Image, class=1

Image, class=4

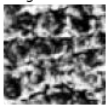


Image, class=1

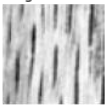
Image, class=1



Image, class=1



Image, class=4



Image, class=1

Image, class=1

Image, class=1