مبانی بینایی کامپیوتر

پروژه پایانی

نیکی نزاکتی - سینا ضیائی

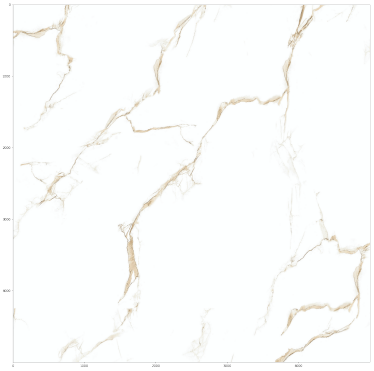
97521387 - 98522094

آماده سازی داده ها

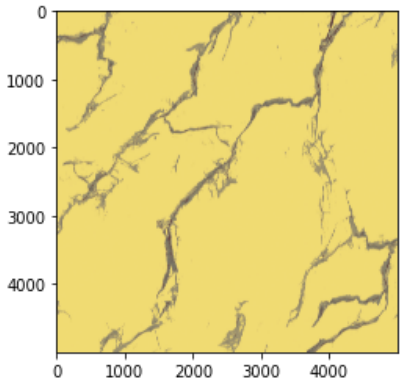
برای راحتی کار داده ها به درایو منتقل شده و از این طریق به آن ها دسترسی پیدا کردیم. سپس داده ها داخل آرایه ریخته شده اند:

```
for file in os.listdir("/content/Dataset/"):
  if file.endswith(".jpg"):
    images.append(cv2.imread(os.path.join("/content/Dataset/",file))
[74:1590,74:1590])
```

۷۴ پیکسل از هر طرف از تصاویر کاشی ها کم شده تا حاشیه تصاویر که نامرتبط هستند حذف شوند. در ادامه، بین پترن و تصاویر مربوطه آن ها، تطبیق هیستوگرام میزنیم که پترن های ما از نظر هیستوگرامی با تصاویر منطبق باشند. این مقادیر در دیکشنری حاوی هسیتوگرام منطبق شده و نام پترن می باشد.*



تصویر اولیه پترن



تصویر پترن بعد از تطبیق هیستوگرام

برای آماده سازی داده ها چهار تابع به صورت مستقیم روی آن ها صدا زده شده که به شرح زیر می باشند:

- degree = find_keypoints(image, pattern)

این تابع با گرفتن عکس و پترن مربوطه با استفاده از تابع sift.detectAndCompute نقاط کلیدی عکس ها را پیدا کرده و با استفاده از flann.knnMatch نقاط تطبیق یافته را پیدا می کند. از بین این نقاط، نقاطی که درصد تشابه آن ها ۸۰ و بیشتر است را برمیگزینیم. در نهایت این نقاط بهینه را با نقاط کلیدی دو عکس به تابع rotation_degree می دهیم.
در تابع rotation_degree، با استفاده از queryIdx و trainIdx نقاط کلیدی را بدست آورده و با pt به مختصات آن میرسیم. با داشتن مختصات این نقاط و مقایسه ی دو به دوی نقاط قرار گرفته در بالا، پایین، چپ و راست تصویر، درجه ی چرخش تصویر نسبت به پترن را بدست می آوریم.*

- rotated_pattern = rotate_image(pattern, degree)

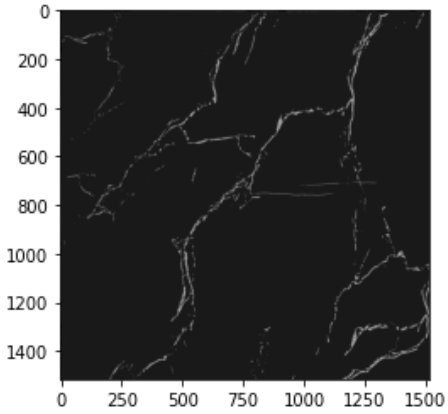
با دادن پترن و درجه ی چرخش، پترن را با توجه به تصویر با استفاده از cv2.rotate می چرخانیم.

- resized_pattern = resize_pattern(rotated_pattern, scale = 1516/5000)

پترن های چرخش یافته را با توجه به تصاویری که سایزشان تغییر کرده است با استفاده از cv2.resize تغییر سایز می دهیم.

- image_result = thresholding(image, resized_pattern)

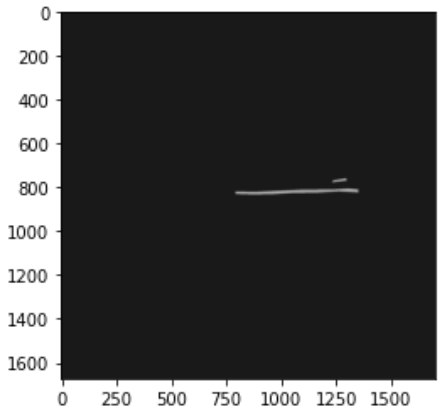
در این تابع تصویر و پترن ابتدا به سیاه و سفید تبدیل می شوند و سپس روی آن ها adaptiveThreshold پیاده می شود. در نهایت از این دو تصویر تفاوت گرفته می شود با استفاده از cv2.subtract.



تصویر پس از اعمال thresholding

*این قسمت از کد با کمک گرفتن از گروه های دیگر (بهکام کیا و سماوات) و همچنین از وبسایت https://pysource.com/2018/07/20/find-similarities-between-two-images-with-opencv-and-python/ پیاده سازی شده است.

در ادامه با توجه به باینری شدن عکس، لیبل های باینری تولید می کنیم. این کار با استفاده از تابع label = create_label(img_result,all_data[new_indx]) انجام می شود که در آن با توجه به نقاط موجود به عنوان مختصات ترک، لیبل ما یک تصویر سیاه با ابعاد تصویر کاشی اصلی خواهد بود که در آن ناحیه ترک سفید است.



تصویر لیبل جدید

در نهایت این مراحل روی دیتاست اعمال شده و داده ها برای آموزش مدل آماده می شوند.
داده های آموزش توسط random_split_dataset و با نسبت 0.15 به داده validation تقسیم شده اند. در آموزش از EarlyStopping با patience 2 استفاده شده است. داده ها را shuffle کرده و آموزش می دهیم.

برای آموزش از مدل U-Net استفاده شده است که در معماری آن از لایه های Conv2D، MaxPooling2D، Dropout و Concatenate استفاده شده است. از بهینه ساز Adam با learning rate مقدار 1e-4، تابع ضرر binary_crossentropy و متریک accuracy استفاده کرده ایم.*

در نهایت دقت مدل ما به 0.99 و ضرر 0.02 می رسد و عملکرد خوبی بر داده های val دارد.

- نحوه اجرای کد:

فایل run_me.ipynb را اجرا کرده و مسیر داده را بر اساس دیرکتوری خود تغییر دهید.

*این قسمت از کد با کمک گرفتن از گیت هاب بر پایه ی مقاله ای با ایده یافتن ترک در سیمان پیاده سازی شده است. https://github.com/arthurflor23/surface-crack-detection