

# گزارش پروژه

## فهرست مطالب

۲	مقدمه و مرور:
۴	بخش اول، پیش پردازش:
۴	بخش دوم، پیدا کردن object ها:
۵	بخش سوم، ذخیره اطلاعات box ها:
۵	بخش چهارم، تهیه برش های تصویر اصلی:
۵	بخش پنجم، مرحله پیش پردازش برای طبقه بندی:
۶	بخش ششم، طبقه بندی برش ها با classifier:
۶	classifier:
۹	بخش هفتم، ایجاد مستطیل دور اعداد در تصویر اصلی:
۱۰	راهنمای استفاده:
۱۳	Pytesseract:
۱۴	راهنمای استفاده:

## مقدمه و مرور:

در این پروژه قصد دارم تا اعداد فارسی موجود در یک تصویر داده شده را شناسایی کنم. نمونه ای از نتایج پروژه ام را میتوانید در تصویر زیر مشاهده کنید.



- **بخش اول:** پروژه مربوط می شود به این که روی عکس پیش پردازش انجام دهم.
- **بخش دوم:** تمام object های موجود در تصویر را شناسایی میکنم.
- **بخش سوم:** ذخیره اطلاعات مستطیل هر object در یک لیست است.

- **در بخش چهارم:** یک تصویر که در حقیقت برش خورده تصویر مرجع تا محدوده object مورد نظر است را ایجاد میکنم که به آن slice می گویم.
- **در بخش پنج :** این تصویر برش خورده را تغییر سایز میدهم و پیش پردازش روی آن انجام میدهم و سپس آن را به یک classifier می دهم.
- **بخش ششم :** یک شبکه عصبی است که وظیفه دارد تا عکس ورودی را دسته بندی کند که آیا یک تصویر غیر عدد فارسی است و یا یک عدد فارسی است. یعنی یک classifier دوتایی دارم. این classifier قبلا روی داده ای آموزش داده ام و در این قسمت فقط از آن برای پیش بینی استفاده میکنم.
- **بخش هفتم:** بعد از مشخص شدن label هر slice آن ها را در یک لیست دریافت میکنم. سپس در یک حلقه اطلاعات box هر slice ی که به عنوان عدد شناسایی شده است را به صورت مستطیل روی عکس مرجع اضافه میکنم.
- **در نهایت** عکس مرجعی که اطلاعات مستطیل ها به آن اضافه شده است را نمایش می دهم.

## بخش اول، پیش پردازش:

در این قسمت ابتدا با تابع زیر نویز های روی تصویر را میگیرم.

```
cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15)
```

بعد تصویر را از مد زنگی به مد خاکستری میبرم.

```
cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

بعد با کمک blur جزئیات زائد در زمینه را محو میکنم.

```
cv2.GaussianBlur(gray, (7,7), 1)
```

سپس با کمک تابع Sobel لبه های ناحیه ها را مشخص میکنم.

البته من تابع های دیگری نظیر Albert و یا Canny را هم امتحان کردم و برای من تفاوتی تو یافتن object ها ایجاد نشد. این را هم از کتابخانه OpenCV و هم از SkImage تست کردم.

## بخش دوم، پیدا کردن object ها:

بعد با کمک تابع regionprob محدوده هایی که از یک آستانه ای بزرگتر بودند را جدا میگردم:

```
for region in regionprops(label_image):
    # take regions with large enough areas
    if region.area >= 100:
```

## بخش سوم، ذخیره اطلاعات box ها:

و سپس اطلاعات مستطیل آنها یعنی نقطه شروع و پایان طول و عرض آنها را می‌گرفتم و در یک لیست به نام image\_boxes وارد می‌کردم.

```
minr, minc, maxr, maxc = region.bbox
image_boxes.append([minc,minr, maxc,maxr])
```

## بخش چهارم، تهیه برش های تصویر اصلی:

سپس عکس اصلی را از محدوده همین مستطیل برش می‌زدم و در یک لیست جداگانه به نام image\_slices ذخیره می‌کردم.

```
img=image[minr:maxr, minc:maxc]
img=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
image_slices.append(img)
```

## بخش پنجم، مرحله پیش پردازش برای طبقه بندی:

در اینجا باید برش های تصویر را برای ورود به classifier آماده کنم. لذا روی آنها پیش پردازش انجام می‌دهم. از مد رنگی به خاکستری می‌برم. و به سایز مناسب برای ورود به classifier تغییر سایز می‌دهم. سپس آن را تبدیل به np\_array می‌کنم و تغییر scale می‌دهم.

محصول این بخش را در لیستی به نام x\_img وارد می‌کنم.

```
x_img=[]
for image in image_slices:
```

```
#some preprocessing commands:
image = cv2.fastNlMeansDenoisingColored(image, None, 10, 10, 7
, 15)
image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
image = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,87,9)
image=cv2.resize(image,(32,32), interpolation=cv2.INTER_AREA)

image=img_to_array(image)
x_img.append(image)

x_img=np.array(x_img,dtype='float')/255.0
```

## بخش ششم، طبقه بندی برش ها با classifier:

در این قسمت برش های عکس که از مرحله پیش پردازش دریافت کردم و الان به شکل np\_array هستند را به شبکه عصبی ای که قبلاً آموزش دادم و ذخیره کرده ام، میدهم. و نتایجی که برای label های آن پیش بینی می شود را در labels ذخیره میکنم. از آنجایی که مقادیر گرفته شده از دستور predict مقادیر احتمال پیش بینی شده برای هر کلاس هست در نتیجه، با دستور زیر مشخص میکنم که اگر احتمال از ۰/۵ بیشتر بود به کلاس یک و اگر کمتر به کلاس صفر نسبت دهد. و حاصل را در dig\_l ذخیره میکنم.

```
dig_l=np.where(labels > .5, 1,0)
```

### classifier:

classifierی که اینجا استفاده کردم یک binary classifier است که با CNN ساختم. در ورودی یک تصویر در مد خاکستری و در ابعاد ۳۲ در ۳۲ میگیرد. یعنی ابعاد ورودی فارغ از تعداد تصاویر برابر با

$$32 \times 32 \times 1$$

است.

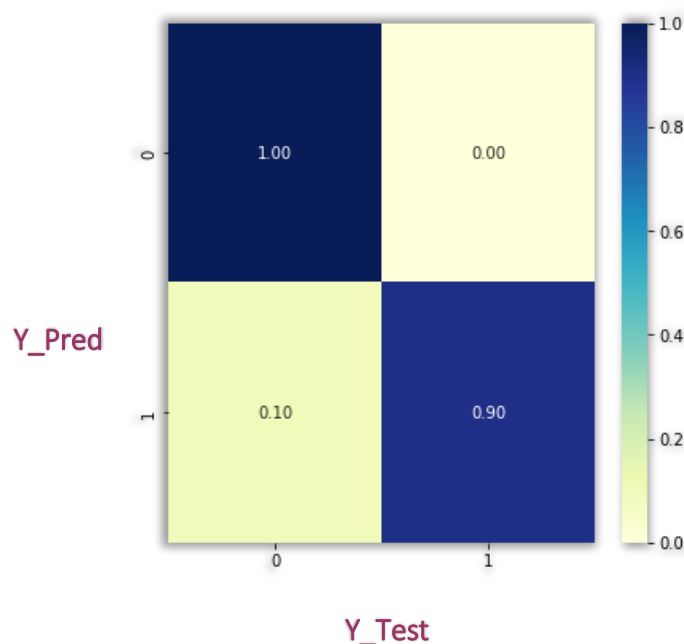
به عنوان داده ابتدا یک بار آن را با داده کمتر آموزش دادم. داده هایی که به آن دادم عبارت بودند از :

- اعداد فارسی به تعداد ۱۰۲۳۵۲
  - عکس های غیر متن که از دو مجموعه داده از GitHub دانلود کردم. یکی شامل عکس های متفرقه نظیر تصاویر داخل کتابها بود و دیگری شامل نماد های غیر متنی بود.
- بعد از اتمام یادگیری از این classifier استفاده کردم و متوجه شدم که متن فارسی را هم مثل اعداد label میکند.
- بنابراین یک بار دیگر ولی با مجموعه داده گسترده تری classifier را آموزش دادم . این بار داده های زیر را نیز اضافه کردم.
- به مجموعه اعداد دستنویس، اعداد چاپی با فونت های مختلف را که خودم با plt.text درست کرده بودم اضافه کردم. من قبلا یک مجموعه ۵۰۰ فونتی فارسی از اینترنت دانلود کردم. با آنها ابتدا برای تنها یک عدد عکس های عدد مربوطه را تولید کردم و متوجه شدم که matplotlib.pyplot با تعدادی از این فونت ها سازگار نیست و یا خروجی نمیدهد و یا خروجی تصویر سفید ذخیره میکند بنابراین فونت های مشکل دار را از فولدر فونت ها حذف کردم و تنها ۱۶۰ فونت سالم باقی ماند که به آنها ok\_fonts گفتم. سپس برای تمامی یونیکدهای اعداد فارسی روی این ۱۶۰ فونت تصویر اعداد را گرفتم و در فولدری ذخیره کردم. سائز این تصاویر بزرگ بود و بنابراین یادگیری شبکه روی آنها کند بود پس باید سائزشون را تغییر میدادم . از دستور resize در OpenCV استفاده کردم و دیدم که از کیفیت تصاویر کم میکنه و نتیجه accuracy خوب نیست( البته مربوط به classifier شناسایی اعداد نه عدد غیر عدد). پس تصاویر اعداد را دوباره تولید کردم و این بار با ترکیب دستور figsize و گزینه dpi سائز عکس ها را کاهش دادم و تا حدود یک پنجم قبل رساندم . بعد عکس ها را تا محدوده تنها object درون آنها برش دادم تا حاشیه زائد نداشته باشم و سپس سائز عکس ها را با pad به سائز بزرگترین برش رساندم و در انتها سائز آنها ۶۸ در ۶۳ بود. این مجموعه داده را قبلا ذخیره کرده بودم و حالا در این برنامه وارد کردم.
  - مجموعه از اسامی فارسی چاپی را از Kaggle گرفتم که اسم ان شتر بود و به عنوان داده غیر عدد به مجموعه داده ها اضافه کردم.
  - یک مجموعه از نوشته های دستنویس فارسی را هم به عنوان داده غیر عدد از سایت آزمایشگاه ابراهیم پور دانلود کردم و به Colab آپلود کردم. چون لینک دانلود مستقیم نداشت. این را هم به عنوان داده غیر عدد فارسی اضافه کردم. ( متن فارسی اضافه کردم چون متن فارسی را دفعه اول مثل اعداد label میکرد، بعدش بهتر شد)

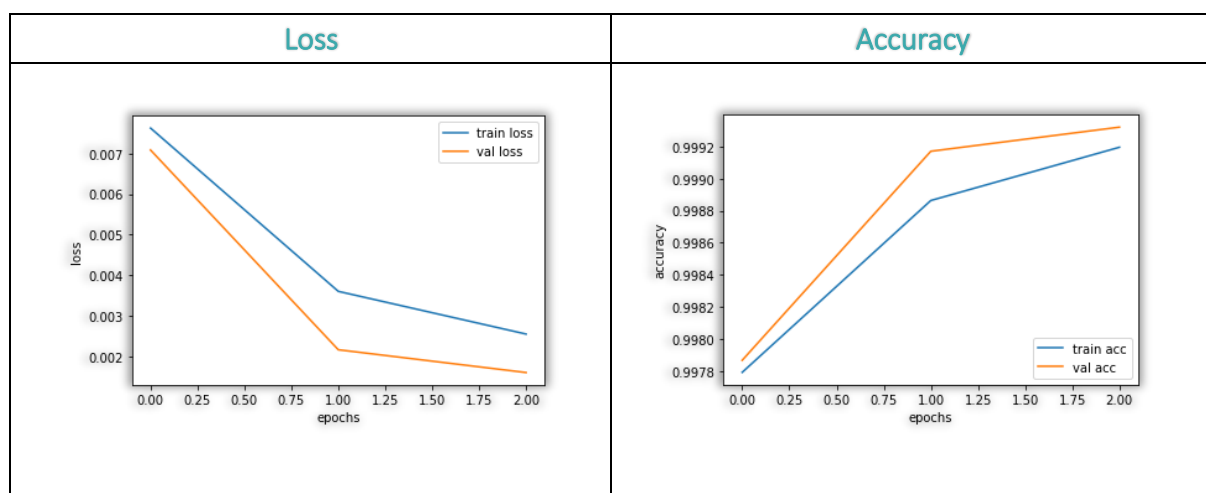
بعد تصاویر را خواندم و پیش پردازش کردم و تغییر سایز دادم به سایز ورودی شبکه و بعد به شکل np\_array در آوردم و سپس داده های هر دسته را با هم np.concatenate کردم و برای اعداد با np.ones ، label درست کردم و برای غیر اعداد در حلقه خواندن عکس ها label صفر را با لیست ایجاد کردم.

در نهایت هم آرایه اعداد فارسی را به آرایه غیر اعداد concatenate کردم و روش shuffle دادم. سپس با train\_test\_split به دو قسمت برای یادگیری و تست تقسیم کردم.

پس از اتمام یادگیری accuracy\_score روی داده test برابر با 95.8 بود. (متأسفانه اینجا یک بار اجرا کردم و نتیجه ام خیلی خوب بود ولی مدل save نشد بنابراین نتیجه را یک بار از دست دادم)



و نمودار های loss و accuracy برای داده train و validation به صورت زیر است.





## بخش هفتم، ایجاد مستطیل دور اعداد در تصویر اصلی:

از classifier که آموزش دادم برای پیش بینی label هر slice استفاده میکنم.

```
labels=model.predict(x_img)
labels=labels.reshape(-1)
labels=np.array(labels)
dig_l=np.where(labels > .5, 1,0)
```

سپس در یک حلقه روی تمام برش های تصاویر شرطی قرار میدهم که اگر label برابر با ۱ بود مستطیلی به ابعاد مستطیل آن object را بر روی تصویر اضافه کند. و پس از اتمام حلقه دستور نمایش تصویر همراه با مستطیل ها را قرار میدهم.

```
input_image_path=path
image=cv2.imread(input_image_path)
orig=image
i=0
for img in x_img:
    if dig_l[i]==1:
        startX = image_boxes[i][0]
        startY = image_boxes[i][1]
        endX = image_boxes[i][2]
        endY = image_boxes[i][3]
        cv2.rectangle(orig, (startX, startY), (endX, endY), (0, 255,
0), 3)
        i += 1
```

البته یک تابع درست کردم که در آن مسیر تصویر و مسیر فایل classifier را دریافت میکند و خروجی دو تصویر نمایش میدهد. در سمت چپ تصویر با object های که به عنوان عدد فارسی label کرده است و در سمت راست تصویر با تمامی object هایی که شناسایی شده است.

<http://kia-kahroba.ir/laureates/sites/default/files/covers/2-i-37-b.jpg>

object هایی که به عنوان عدد فارسی label شدند.

تمام object های شناسایی شده



در مورد عدد صفر یک مشکل وجود دارد ، چون هر نقطه یا دایره ای را هر جا میبینی به عنوان صفر و در نتیجه عدد فارسی میگیرد.

### راهنمای استفاده:

برای استفاده از این مدل، به فایل زیر مراجعه کنید:

Function\_Persian\_Digits\_Recognizer.ipyp

سپس فایل مدل یعنی فایل :

tx\_vs\_ntx\_clf\_big2

را در آدرس مورد نظر خود آپلود کنید.

به ترتیب آدرس تصویر مورد نظر و آدرس مدل را در تابع زیر

`dg_rec(path1, model_path1)`

وارد کنید. و تابع را اجرا کنید.

در خروجی دو تصویر میبینید که در سمت راست تمامی object های شناسایی شده در تصویر موجود است. و در سمت چپ object هایی را علامت زده که به عنوان عدد فارسی شناسایی کرده است.

---

## نکته پایانی این روش:

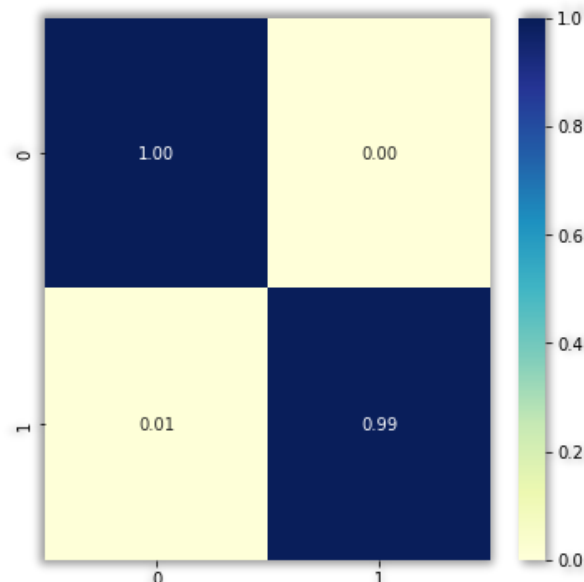
اگر بخواهم از این مدل برای هر مجموعه عکسی استفاده کنم، ابتدا باید classifier را روی آن مجموعه عکس fine tune کنم و سپس از مدل استفاده کنم. به عبارت دیگر باید عکسهایی از همان نوع را برای یادگیری دقیق تر به مدل بدهم و سپس از مدل برای عکس استفاده کنم.

همچنین میتوانیم هابیر پارامترهای مدل را نیز fine tune کنیم.

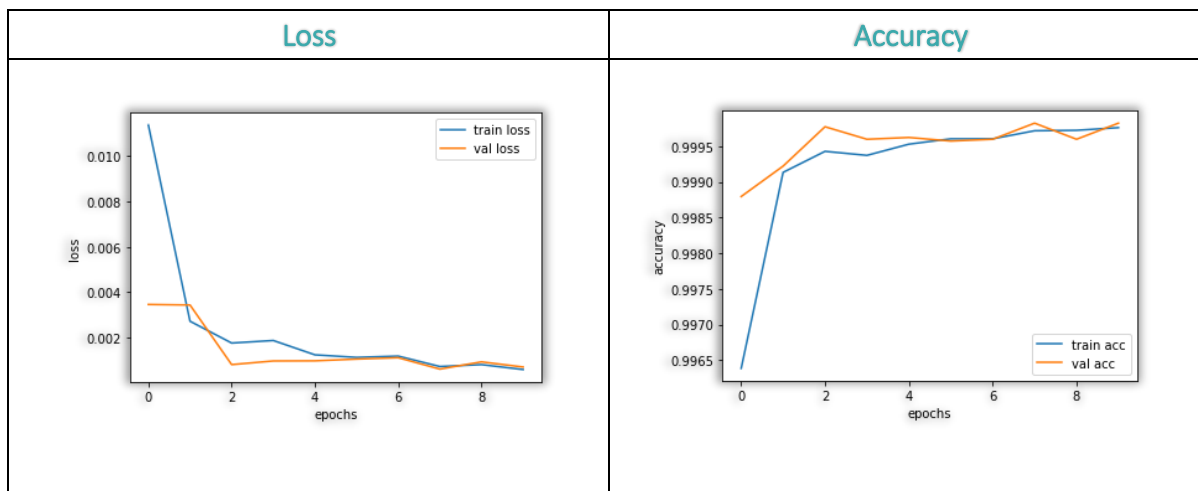
البته کارهای دیگری هم کردم که دیگه تو نتایج قرار ندادم. مثلاً classifier نوشتم که تشخیص بده هر عدد چند هست.

و یا مدل را برای epochs و batch\_size های متفاوت آموزش دادم.

برای مثال برای epochs = 10 و batch\_size = 128 (هم اجرا کردم که فایل مربوطه را نیز ضمیمه کردم). Accuracy=99.67 شد. Confusion matrix آن به صورت زیر در آمد.



نمودارهای loss و accuracy در زیر قرار دادم.



یک روش دیگر را هم مورد بررسی قرار دادم.

برای مطالعه آن لطفاً به **صفحه بعد** مراجعه

کنید. (در **Repository** دیگری موجود است)

## روش دوم

### Pytesseract

از این ابزار برای تبدیل نوشته های موجود در عکس به رشته های متنی استفاده می شود. این پکیج از East استفاده میکند و اساس کارش بر RCNN هست. (البته East را هم استفاده کردم ولی نتیجه ای نگرفتم برای همین در موردش صحبت نمیکنم)

بعد از نصب آن و load کردن بخش فارسی آن روی چند نمونه عکس استفاده کردم. به درستی ترجمه را انجام نداد. سپس از آن برای یافتن box های متناظر با اعداد استفاده کردم. به این ترتیب که pattern متناظر با اعداد را با تصویر مقایسه کند و مکان هایی که مربوط به اعداد فارسی است را مشخص کند.

```
def PDG_rec(img_path):
    import re
    import cv2
    import pytesseract
    from pytesseract import Output
    import matplotlib.pyplot as plt

    img = cv2.imread(img_path)
    d = pytesseract.image_to_data(img, output_type=Output.DICT, lang='fas')
    keys = list(d.keys())

    digits=['1','2','3','4','5','6','7','8','9','0','.','۰','۱','۲','۳','۴','۵','۶','۷','۸','۹']

    for digit_pattern in digits:

        n_boxes = len(d['text'])
        for i in range(n_boxes):
            if int(d['conf'][i]) > 0:
                if re.match(digit_pattern, d['text'][i]):
                    (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])
                    img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.axis('off')
```

در زیر نمونه ای از نتیجه کار بر روی یک تصویر را مشاهده میکنید.



همانطور که میبینید اعداد را تشخیص نداده است.

و در مورد یک عکس دیگر:



حتی توانایی کمتری برای تشخیص محدوده object ها و تشخیص سایر اعداد موجود در تصویر دارد.

### راهنمای استفاده:

برای استفاده از این مدل، به فایل زیر مراجعه کنید.

Persion\_Digit\_Finder\_using\_Pytesseract.ipynb

ابتدا بخش زیر عنوان

## How to install Pytesseract for persian language:

را کامل اجرا کنید تا Pytesseract و پکیج فارسی آن برای شما نصب شود.

سپس آدرس تصویر مورد نظر را به صورت string در تابع زیر

```
PDG_rec(path)
```

به جای path وارد کنید. و تابع را اجرا کنید.

در خروجی یک تصویر میبینید که تمامی object هایی که توسط Pytesseract با pattern اعداد فارسی مطابقت داشته باشد، با مستطیل سبز علامت زده شده است.