پروژه بینایی کامپیوتر - تشخیص کارت

قدم اول: جداسازی کارت از تصویر

برای انجام قدم اول از توابع پردازش تصویر استفاده کردیم.

1. خاکستری کردن تصویر: برای این که بتوانیم قدم های بعدی مانند لبه یابی، عملگرهای مورفولوژی، باینری کردن تصویر و ... را روی تصویر انجام بدهیم.
2. مات کردن تصویر: این کار برای این است که مرزهای کوچک از بین بروند. مثلا اگر پس زمینه کارت دارای کمی نویز است، با این کار نویز پس زمینه از بین میرود.
3. باینری کردن تصویر: تصویر را با استفاده از متد های آداپتیو باینری کردیم. این کار کمک میکند تا لبه کارت بهتر شناسایی شود.
4. عملگرمورفولوژی: از این عملگر برای پیوند دادن مرز کارت استفاده شده. در برخی نقاط، مرز کارت دچار ناپیوستگی میشود که کار پیدا کردن مرز کارت را خراب میکند. با استفاده از این عملگر نقاط گسستگی را پوشش دادیم
5. پیدا کردن خطوط: کانتور های تصویر با استفاده از توابع اماده شناسایی شدند.
6. در این قسمت به جای این که خود کانتور ها را به عنوان مرز انتخاب کنیم، چندضلعی محدب آن را به عنوان مرز انتخاب کردیم. با این کار اگر یک ضلع از چهار ضلع کارت در مرز نیفتد، میتوان باز هم تصویر کارت را بدون از دست دادن بخشی از آن به دست آورد.
7. پیدا کردن نقاط گوشه: برای استفاده از تبدیلات تصویر، لازم است که نقاط گوشه را پیدا کنیم
8. تبدیل پرسپکتیو: این تبدیل میتواند کارت را صاف و افقی روی تصویر بیاندازد

به این ترتیب کارت را استخراج کردیم.

اما استخراج کارت با مشکلاتی مواجه شد:

1. در برخی تصاویر، کارت با پس زمینه رنگ مشابهی داشت و مرز بین کارت و پس زمینه دقیق مشخص نبود.
2. در برخی تصاویر نویز آنقدر زیاد بود که نمیشد مرز کارت را به خوبی استخراج کرد.

دو پارامتراساسی که این دو مشکل را کنترل میکنند میزان مات کردن تصویر و پارامتر های ترشولد باینری هستند. با بالا و پایین کردن این پارامتر ها میتوان هر کدام ازین مشکل ها را به تنهایی حل کرد اما مشکل اصلی این است که این دو حالت، حالات مقابل هم هستند و با اصلاح یک حالت، حالت دیگر خراب میشود.
بنابراین تابع را طوری تغییر دادیم که بتوان از آن در حالات مختلف استفاده کرد.

```python
def cut_card(im, config):
    im = cv2.resize(im, config['resize'])
    original = im.copy()

    im = make_gray(im, False)
    im = make_clahe(im , config['clahe']['limit'], config['clahe']['size'], False)
    im = make_blur(im, config['blur']['size'], config['blur']['eps'], False)
    # im = make_median(im, config['median'], True)
    im = make_thresh(im, config['thresh']['size'], config['thresh']['c'], False)
    im = make_morph(im, config['morph'], False)
    # edges = make_canny(im, False)
    contours = make_contours(im, False)
    corners = make_corners(contours[1])
    width, height = config['outsize']
    src_points = sort_corners(corners)
    result = make_prespective(original, src_points, width, height, False)
    return result
```

این تابع طوری است که میتوان به آن، کانفیگ های مختلف پاس داد تا رفتار متفاوتی داشته باشد.

سپس برای هر کدام از این دو حالت یک کانفیگ متفاوت ایجاد کردیم:

```python
high_contrast_config = {'resize':(1280, 960),
                        'clahe':{'limit':2.0, 'size':(8,8)},
                        'blur':{'size':(15,15), 'eps':11},
                        'median':5,
                        'thresh':{'size':31, 'c':3},
                        'morph': (5,5),
                        'outsize':(260,153)}

low_contrast_config = {'resize':(1280, 960),
                        'clahe':{'limit':2.0, 'size':(8,8)},
                        'blur':{'size':(5,5), 'eps':5},
                        'median':5,
                        'thresh':{'size':31, 'c':3},
                        'morph': (5,5),
                        'outsize':(260,153)}
```

همچنین یک کانفیگ ایجاد کردیم که تقریبا میتوانست هر دو حالت را به خوبی برآورده کند:

```
average_config = {'resize':(1280, 960),
                  'clahe':{'limit':2.0, 'size':(8,8)},
                  'blur':{'size':(15,15), 'eps':11},
                  'median':5,
                  'thresh':{'size':35, 'c':2},
                  'morph': (5,5),
                  'outsize':(512,328)}
```

به این ترتیب عملیات کات کردن تصاویر به اتمام رسید.

قدم دوم: تشخیص کارت بانکی از ملی

در این مرحله برای تشخیص کارت بانکی از کارت ملی، با استفاده از روش template matching و دو الگو یا template در کارت ملی استفاده کردیم:

این ساختار ها تنها در کارت ملی وجود دارند.

برای تشخیص کارت ملی، هر کدام از این template ها را به صورت جداگانه با تصویر اصلی match میکنیم و سپس میانگین مقدار پیک که حاصل template matching هر کدام از این ساختار ها با تصویر اصلی است را به دست میاوریم و با مقایسه با یک threshold تصمیم میگیریم که تصویر ورودی کارت ملی است یا خیر:

```
def identify(self, img):
  img = cv2.resize(img, (self.width, self.height))
  res1 = cv2.matchTemplate(cv2.cvtColor(img, 0), cv2.cvtColor(self.template1, 0), cv2.TM_CCOEFF_NORMED)
  res2 = cv2.matchTemplate(cv2.cvtColor(img, 0), cv2.cvtColor(self.template2, 0), cv2.TM_CCOEFF_NORMED)
  if (np.max(res1) + np.max(res2))/2 > self.threshold:
    return 0
  return 1
```

قدم سوم: استخراج شماره

برای این کار ابتدا سعی کردیم با استفاده از روش های پردازش تصویر و بدون مدل های هوش مصنوعی عمل کنیم.

روش های متفاوتی را برای پیدا کردن محل شماره کارت تست کردیم که در تصمیم گرفتیم که تراکم پیکسل ها را سطر به سطر در تصویر کارت بررسی کنیم. در محل هایی که شماره کارت وجود دارد، لبه های افقی بیشتری پیدا میشود.

برای مثال نتیجه به این صورت بود:

قدم چهارم: استخراج شماره با مدل

برای آموزش مدل به یک دیتاست مناسب نیاز داریم. ابتدا با استفاده از template و تصویر لایه باز هر کارت بانکی، یک سری دیتا درست میکنیم.

https://drive.google.com/file/d/1zep11cpOXDk0N2RjHyiWs-7Z0AKFm9eT/view?usp=drive_link

برای ساخت دیتا اینگونه پیش رفتیم که برای هر کارت بانکی بر روی تصویر لایه باز آن در جاهای مختلف، شماره کارت و 2cvv رندوم با رنگ های مختلف مینویسیم. به عنوان مثال یک دیتایی که درست کردیم اینگونه است:

در آخر فرم لیبل ها مانند YOLO میباشند یعنی:

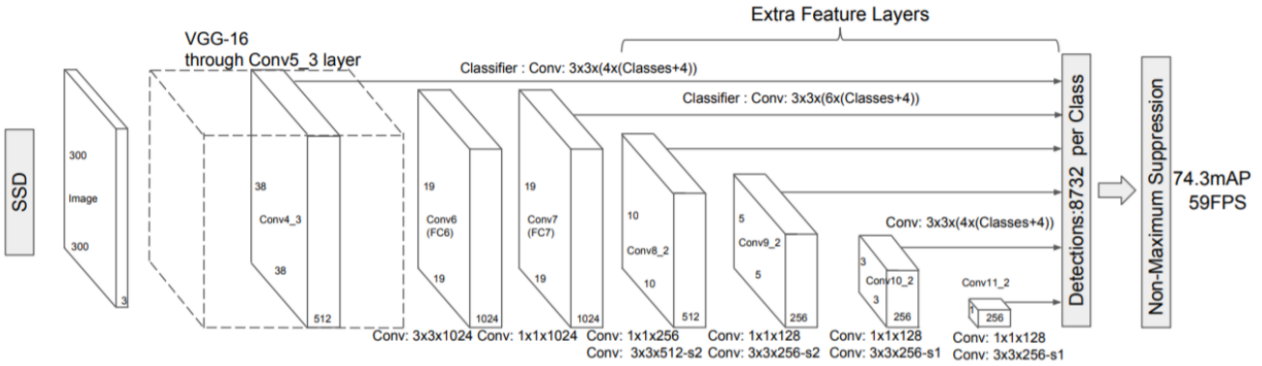
class_id x_center_norm y_center_norm width_norm height_norm

چون دو بخش شماره کارت و 2cvv را میخواهیم شناسایی کنیم پس داخل هر فایل لیبل، دو خط به فرم بالا میباشند.

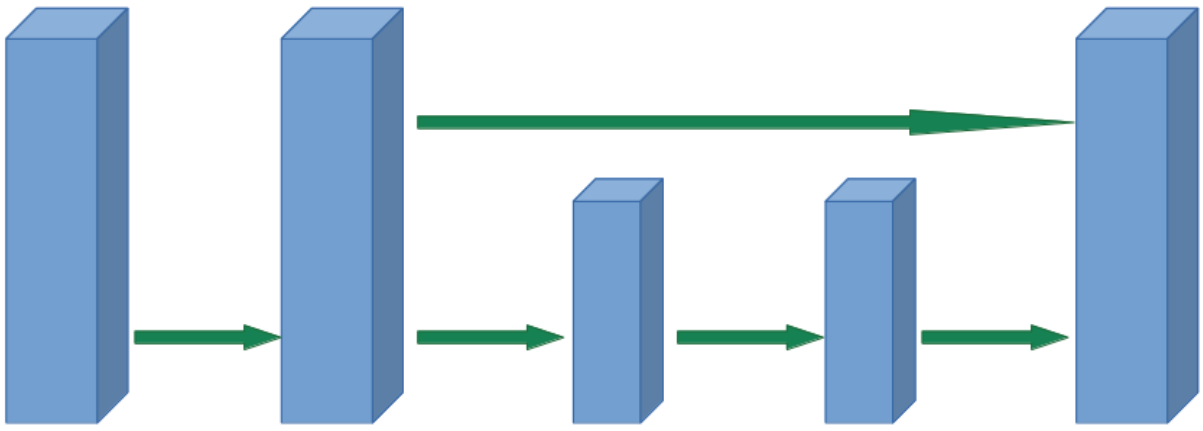
```
0 0.5 0.7032967032967034 0.7586206896551724 0.10439560439560439
1 0.19310344827586207 0.5934065934065934 0.1793103448275862 0.054945054945054944
```

در قدم بعدی سعی شد که این دیتا ها را با مدل YOLO , ssd و یک مدل custom آموزش دهیم.

برای پیاده سازی SSD از اسلاید زیر استفاده شده است:



در مدل custom مدل را به بخش classifier و regression تقسیم کردیم. برای classifier از سه لایه Conv و یک لایه Dense استفاده کردیم. برای بخش regression از لایه های Conv و Dense و BatchNormalization استفاده شده است. دید معماری این بخش به این صورت است که مدل ابتدا به صورت کلی تر به تصویر نگاه میکند و یک سری ویژگی های نه خیلی سطح بالا و نه خیلی سطح پایین کشف میکند تا برای پیدا کردن بخش شماره کارت که تقریبا یک بخش بزرگی از کارت را شامل میشود از آنها استفاده کند. سپس مدل عمیق تر میشود و با استفاده از ویژگی های سطح بالاتر، بخش 2cvv که قسمت نسبتا کوچکی از کارت را شامل میشود را شناسایی میکند.



مدل نهایی ترکیب classifier و regression میباشد.

در backbone classifier مدل 50resnet و در backbone regression مدل 16VGG میباشد.