مستند پروژه‌ی Image Processing مسابقه‌ی ICT Challenge

https://github.com/amirphl/ICTChallenge_ImageProcessing_2019

تیم redhat

نسخه‌ی اولیه

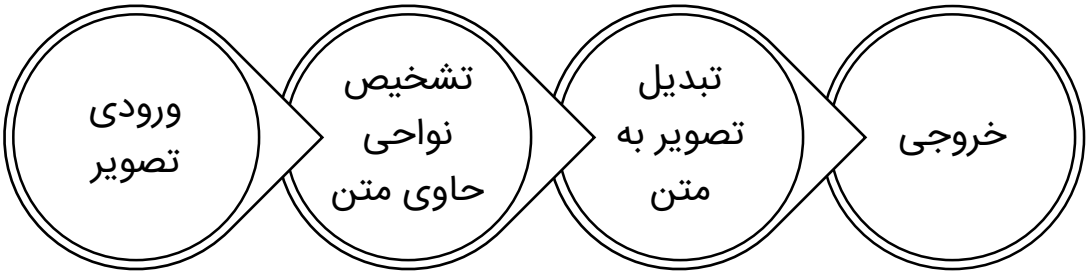
هدف

هدف از تولید این نرم‌افزار، تشخیص شماره‌ی کارت و تاریخ انقضای کارت دریافتی با کمترین خطای ممکن توسط دوربین می‌باشد.

چالش‌ها

- کیفیت پایین عکس‌ها
- وجود متن‌های فارسی و برجسته
- وجود نویز(رفلکشن و سایر عوامل روی کارت)
- چرخش کارت

روند کلی اجرا



مراحل الگوریتم

روند کلی اجرای الگوریتم نهایی به شکل زیر خواهد بود:

- گرفتن عکس توسط ورودی
- بریدن حاشیه‌های اضافه‌ی تصویر
- کاهش نویز
- تشخیص ناحیه‌های حاوی متن در تصویر
- جداسازی کاراکترهای هر ناحیه از تصویر
- تشخیص کاراکترها
- تصحیح کاراکترها
- تشخیص قسمت شماره‌ی کارت ۴ رقمی
- چسباندن قسمت های ۴ رقمی برای تولید شماره کارت ۱۶ رقمی
- تشخیص قسمت تاریخ انقضا
- جداسازی ماه و سال از قسمت تاریخ انقضا
- خروجی

نحوه‌ی اجرای کد

با استفاده از پایتون ۳٫۵، با وارد کردن دستوری به فرمت زیر می توانید خروجی را مشاهده کنید. در این نسخه خروجی به شکل پنجره‌ی عکس نمایش داده می‌شود و با زدن هر کلید سگمنت بعدی در شکل مشخص و متن تشخیص داده شده بالای آن نمایش داده می شود.

```
Python text_recognition.py -i image_address -east frozen_east_text_detection.pb
```

که `image_address` آدرس تصویر کارت است؛ برای مثال:

```
Python text_recognition.py -i images/example01.jpg -east frozen_east_text_detection.pb
```

توجه: اگر مایلید یک پوشه‌ی کامل را تست کنید از `script.sh` برای این کار کمک بگیرید. توجه کنید که آرگومان ورودی بر اساس نام فایل‌ها و پوشه باید تغییر کند.

فرمت خروجی

فرمت خروجی نهایی به شکل متن در ترمینال با فرمت زیر و تصویر کارت ها با متن تشخیص داده شده در آن نمایش داده می‌شود. با فشردن space کد بسته می‌شود. در صورت تشخیص ندادن هر کدامیک از کاراکترهای زیر، آن قسمت در ترمینال نمایش داده نمی‌شود.

رفتار شکست سیستم: اگر کد قادر به تشخیص هیچ کدامیک از بخش ها نباشد خروجی `failed to recognize` در ترمینال چاپ می‌شود.

```
Cart number:1111111111111111
Year: xx
Month:xx
```

کتابخانه‌های مورد استفاده

- Cv2 برای تشخیص سگمنت‌های متن با استفاده از الگوریتم‌های یادگیری
- Pytesseract برای تبدیل عکس به متن
- numpy برای نگهداری اطلاعات سگمنت‌های متن
- Imutils برای تشخیص لبه‌ها
- argparse برای پارس کردن تنظیمات ورودی

مشکلات و راه حل ها

در ابتدا لازم به ذکر است هنوز هیچ الگوریتم OCR با کارایی ۱۰۰% ساخته نشده است.

مشکل وجود نویز را با سیاه و سفید کردن و دینویز کردن تصاویر تا حدودی رفع کردیم.

مشکل چرخش ۹۰ درجه‌ی کارت را با تشخیص طول و عرض تصویر حل کردیم، اما قادر به حل مشکل چرخش جزئی بصورت کامل نبودیم.

توضیحات پیاده سازی

- **یادگیری Text Detection**

فایل frozen_east_text_detection.pb توسط openCV برای تشخیص نواحی حاوی متن از پیش آموزش داده شده است. در این پروژه از این روش برای تشخیص این نواحی استفاده شده است. با لود کردن این فایل در شبکه‌های عصبی، برنامه احتمال وجود متن در نواحی عکس را در یک لیست تولید می‌کند.

- **تبدیل تصویر به متن**

پس از تشخیص نواحی حاوی متن، نواحی با احتمال وجود متن بیش از p با استفاده از pytesseract تشخیص متن داده می‌شوند. این احتمال را ۰/۵ در نظر می‌گیریم. Tesseract موتورهای مختلفی برای تشخیص متن دارد که ما از LSTM + Legacy استفاده کردیم. برای حالت page segmentation هم از روش Fully automatic page segmentation, but no OSD استفاده می‌کنیم.

- **تصحیح کاراکترها**

در این مرحله اشتباهات متداول الگوریتم OCR حل می‌شود؛ برای مثال اگر در شماره‌ی کارت علامت ! داشتیم، آن را تبدیل به ۱ می‌کنیم و اگر شماره‌ی تاریخ ۵ رقمی شد و رقم وسط 7 بود، آن را تبدیل به / می کنیم.

ملاحظات

فایل `frozen_east` در پروژه ضمیمه شده است.

کتابخانه‌های نامبرده شده در بالا را قبل از اجرا نصب کنید.

برای استفاده از کتابخانه‌ی `pytesseract` حتما باید `tesseract` را در سیستم خود نصب کرده باشید. برای اطلاعات بیشتر در این مورد از لینک زیر کمک بگیرید:

https://github.com/tesseract-ocr/tesseract/wiki