گزارشکار مدل های متن باز OCR انگلیسی :

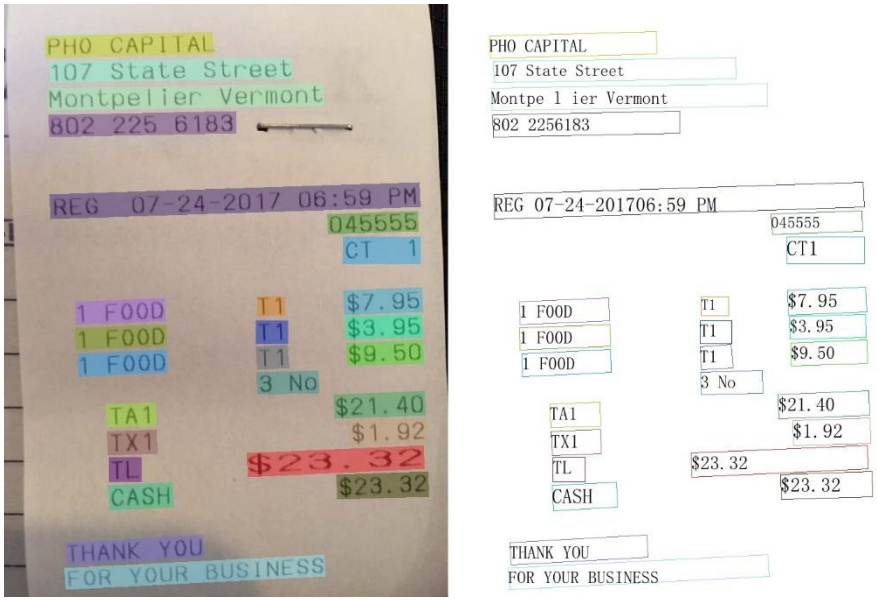
ای گزارش بر اساس خروجی از 12 تصویر داری متن دست نویس و غیر دست نویس انگلیسی ایجاد شده است

مدل های متن باز بررسی شده که خروجی خوبی داده اند (به ترتب عملکرد) :

PaddleOCR: با بررسی نمونه های مختلف از متن هایی که دستنویس هستند یا ورودی دارای نویز یا کجی است این کتابخانه بدون هیچ پیش پردازش اضافه ای و با confidence بالایی متن ها را اسکن کرده و در خروجی ذخیره کرده است ، سرعت اجرای این کتابخانه هم بسیار خوب بوده و ازین نظر بین ocr های مبتنی بر شبکه عصبی مانند کراس و easy ocr قرار می گیرد .

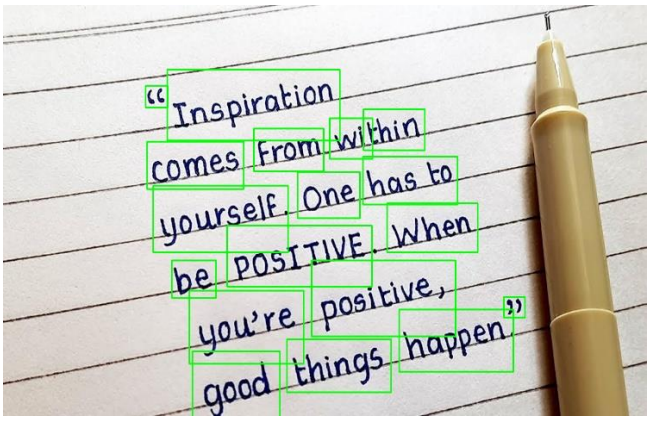
مستندات این کتابخانه به زبان چینی است که به انگلیسی هم موجود است ، قابلیت شخصی سازی این کتابخانه آموزش مدل های یادگیری عمیق روی دیتاست خودمان را هم دارد ، این کتابخانه از زبان فارسی پشتیبانی نمی کند

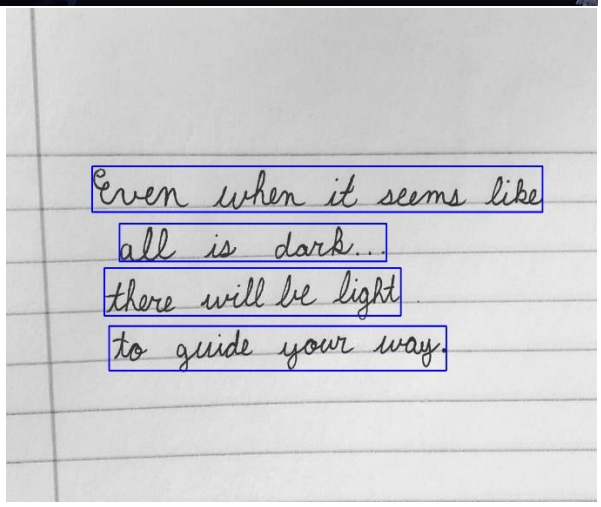
این کتبخانه برای OCR از دو مرحله line detection , text recognition استفاده می کند و بر روی ورودی های دستنویس همراه با کجی هم دقت بسیار بالایی دارد.



این کتابخانه یک پردازش انتها به انتها انجام می دهد و قابلیت کامپایل و پیاده سازی روی سیستم های نهفته مانند nvidia jetson , arm based ها را نیز دارد .(قبل از استقرار نیاز به یک فشرده سازی مدل وجود دارد)

نمونه خروجی های مدل:





خروجی 8 : این مدل بهتر از همه مدل های دیگر برای اسکن متن دست نویس عمل کرده است همچنین خروجی های از conf بسیار خوبی برای همچین فونتی (بالای 85%) بر خوردار هستند -> هر چند عملکرد کلی آن برای متن های دستنویش شکسته قابل اتکا نیست اما میتوان مدل را فاین تیون کرد)

خروجی 9 : متن به صورت کامل و میانگین conf بالایی اسکن شده است (بالای 95%)

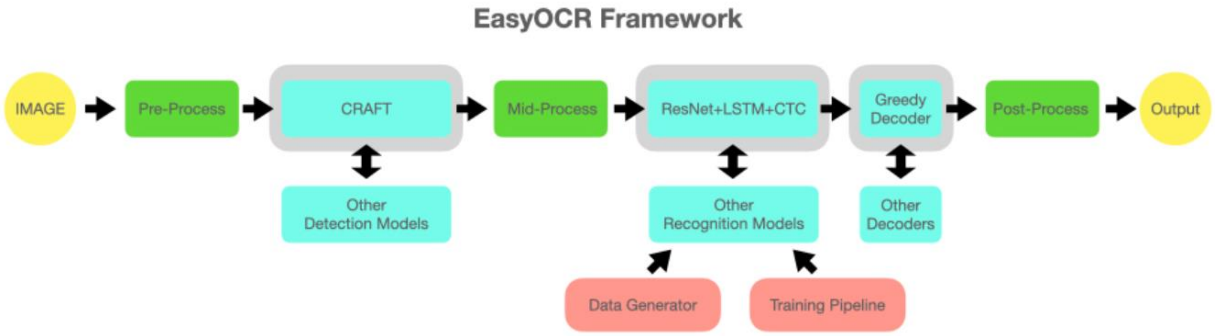
خروجی 7: متن به صورت کامل و میانگین conf بالایی اسکن شده است (بالای 95%) همچنین ترتیب اسکن بالا به پایین و چپ به راست رعایت شده است

خروجی 3: متن به صورت کامل و میانگین conf بالایی اسکن شده است (بالای 97%) همچنین ترتیب اسکن بالا به پایین و چپ به راست رعایت شده است اما در بعضی جاها کلمات چسبیده به هم اسکن شده است که احتمالا حساسیت مدل به بکگراند را می رساند.

easyOCR: این کتابخانه بدلیل راحت و سریع بودن معروف است و استفاده و کانفیگ آن پیچیدگی خاصی ندارد ، از زبان فارسی نیز پشتیبانی میکند) هر چند که دقتش در ورودی های معمولی مثال زدنی است اما در ورودی های دست نویس انگلیسی دقتش مناسب است (البته هر چه دست خط شکسته باشد عملکرد بد تر می شود)

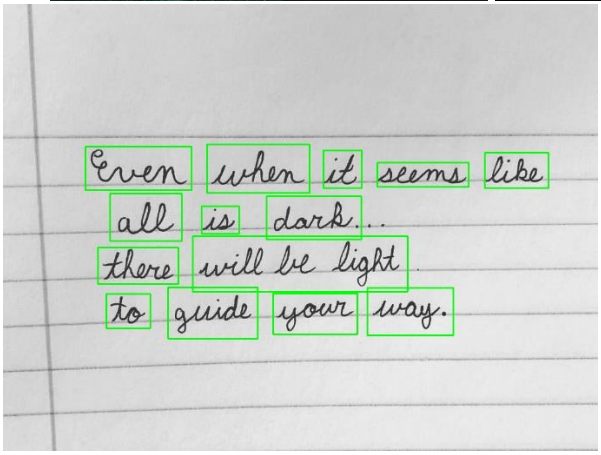
معماری این کتبخانه مبتنی بر شبکه های عصبی است و قابلیت آموزش بر روی دیتاست خودمان را دارد (قبل از آموزش با استفاده از این کتبخانه TextRecognitionDataGenerator **باید دیتاستمان آماده سازی کنیم) ،**

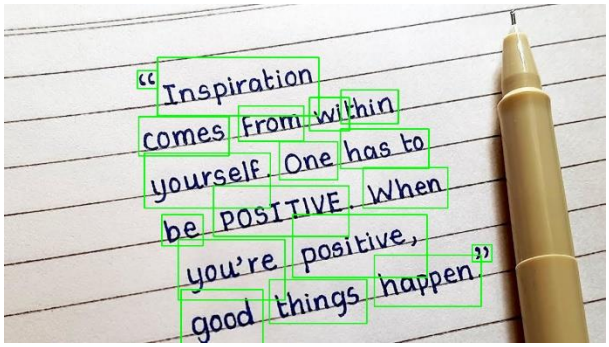
همچنین می توان از مدل شخصی سازی شده خودمان برای تشخیص متن استفاده کنیم (که پیشنهاد سازنده استفاده از CNN ها است که بتواند خروجی هایی با سایز متفاوت ایجاد کند) ، بعد تعریف مدل شخصی سازی شده باید فایل تعریف معماری RECOGNITION MODEL و فایل کانفیگ مدل را نیز ایجاد کنیم)



پایپلاین انتها به انتها این کتابخانه ورودی هایی دارای کجی و ... را اصلاح کرده و نیاز خاصی به پیش پردازش از سمت برنامه نویس ندارد (یادآوری :این مدل برای حذف بکگراند هم استعداد بسیار خوبی از خود نشان داد)

خروجی مدل :





خروجی 9 : کلمات تا حدودی خیلی زیادی درست اسکن شده اند اما ترتیب اسکن آنها برای ایجاد جمله مناسب نیست همچنین ربای یک پیش بینی درست، CONF پایین در حدود 50 -60 % نیز وجود دارد .

خروجی 8 : کلمات اصلا درست اسکن نشده اند همچنین در بعضی موارد که CONF=90% وجود دارد یک کلمه کاملا غلط اسکن شده است (برای مثال guide به اشتباه quide اسکن شده) -> راهکار بهبود استفاده از فونت های شکسته برای آموزش مجدد مدل است یا اینکه برای خروجی هایی با اطمینان بالا از روش های NLP برای پسش پردازش متن استفاده کنیم)

خروجی 7 : مدل در پیش بینی 4 5 کلمه دچار خطا شده است (حرفی را جا انداخته یا اتباه اسکن کرده) ، همچنین میانگین اطمینان متن های اسکن شده نسبت به paddle ocr که خروجی کاملا درستی داشت کمتر است)

خروجی 3 : مدل تمام جملات را بخوبی اسکن کرده است اما اطمینان آن پایین است برای مثال خط اول را با conf=56% و خط دوم را با conf=78% اسکن کرده است که برای یک نتیجه کاملا درست یک متن دیجیتال بدون کجی ، کم بنظر می رسد

OCR_Doctr:این کتابخانه متن باز که محصول 2021 شرکت مایکروسافت است ، مبتنی بر شبکه های عصبی عمیق و معماری ترنسفورمر است ، میتوان از vgg16 , resnet برای تشخی متن در آن استفاده کرد ، بهترین عملکرد این کتبخانه روی متن دیجیتال است اما برای تشخیص متن دستنویس هم عملکرد مناسبی دارد.

مدل تشخیص متن (localization): از معماری resnet18,resnet50 و معماری های زیر برای قطعه بندی معنایی استفاده میکند :

db_mobilenet_v3_large , db_resnet50 , fast_base (https://arxiv.org/pdf/2111.02394.pdf)

مدل شناسایی متن : از معماری های زیر پیشتیبانی می کند :

ViTSTR-Base

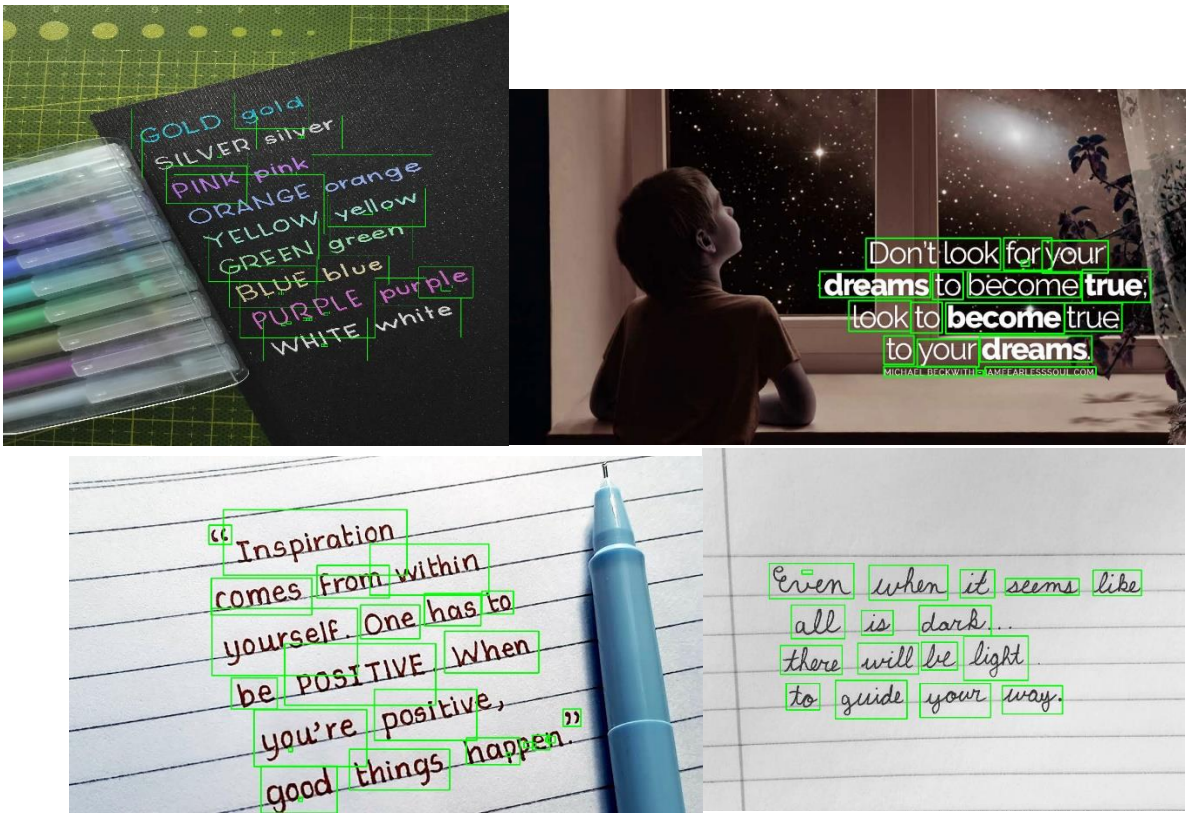
MASTER architecture (https://arxiv.org/pdf/1910.02562.pdf)

CRNN with MobileNet V3, CRNN with VGG16, SAR with resnet-31

این کتابخانه پیش پردازش های ابتدایی مانند تغییر سایز ، نرمال کردن و باینری کردن آن را انجام می هد اما خودش از تسک هایی مانند deskewing کاملا پشتیبانی نمی کند و بهتر است با opencv پیش پردازش شود

این کتابخانه نسبت به paddle ocr کند تر اما نسبت به keras ocr سریعتر است و برای اجرا نیاز به دانلود فایل های حجیم وزن های شبکه و ... دارد ، همچنین عملکرد بسیار خوبی در تحلیل layout تصویر دارد و اجزا مختلف تصویر را بخوبی اسکن میکند

https://mindee.github.io/doctr/latest/index.html

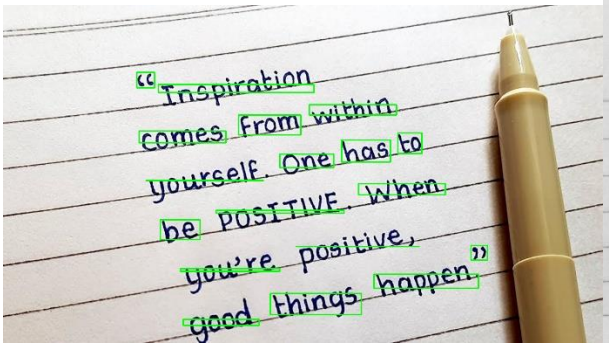


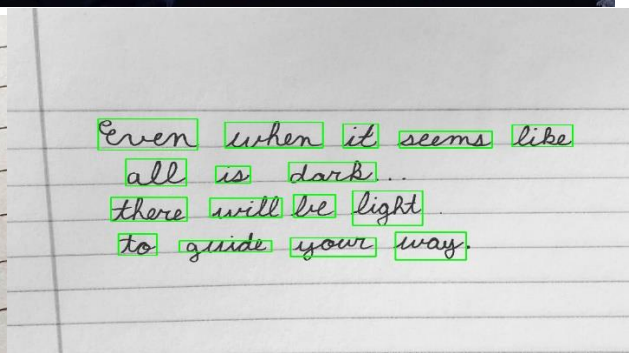
Keras_OCR: این روش ازمدل های **CRAFT** و شبکه **RCNN** استفاده میکند و علاوه برکند بودن نسب به سایر روش ها اجرای آن به حدود **8** گیگ رم در حال اجرا نیاز دارد ،

این مدل مستندات زیادی ندارد اما مطابق لینک زیر میتواند مدل های **recognition , detector** ان را جداگانه با داد های خودمان فاین تیون کرد تا مدل شخصی سازی شود.

https://github.com/faustomorales/keras-ocr

https://keras-ocr.readthedocs.io/en/latest/examples/index.html





خورجی 3و7و8و9 : این مدل قابلیت ایجاد معیار conf برای پیش بینی هایش ندارد همچنین در توجه کمی به بحث معنایی واژه های اسکن شده دارد (بخش recognition مدل باید حتما فاین تیون شده باشد) .

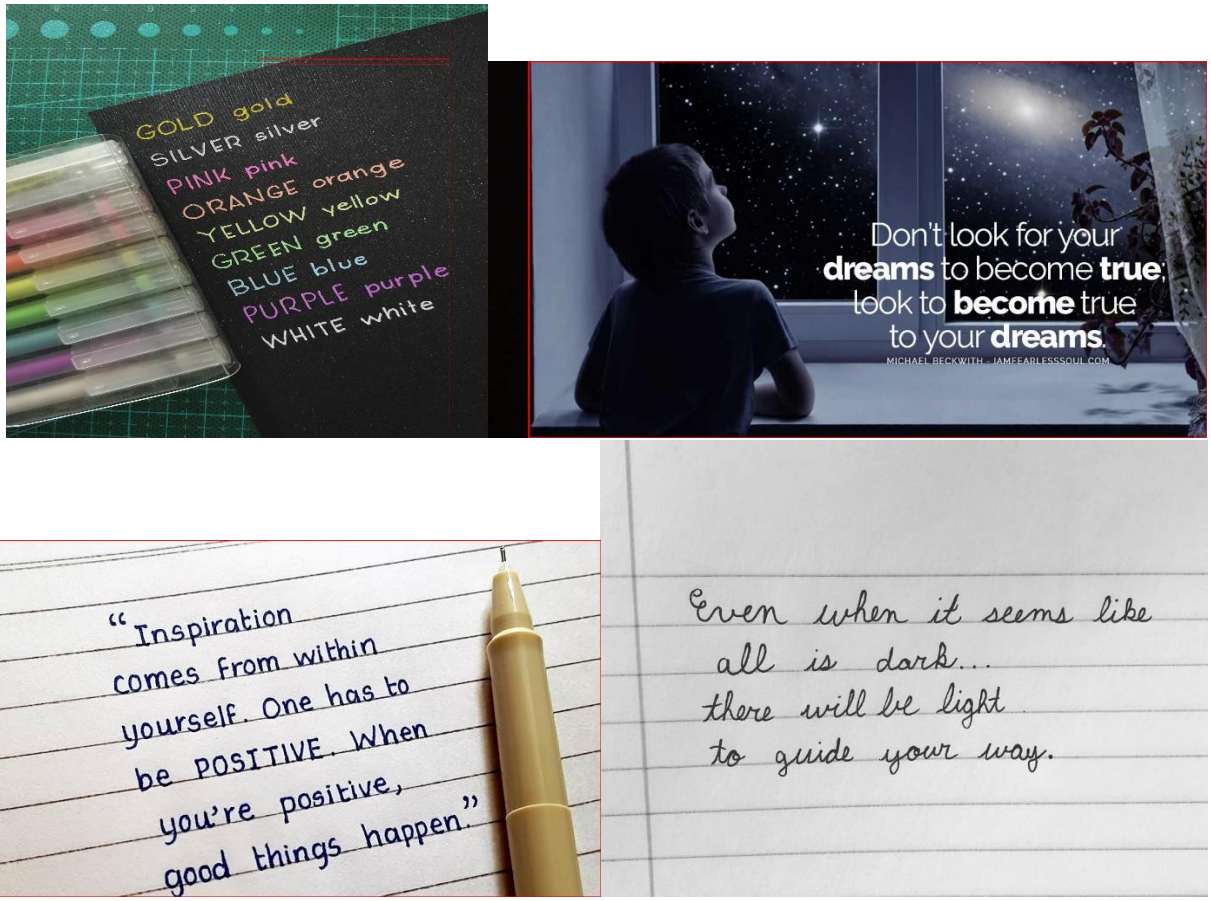
model	latency	precision	recall
AWS	719ms	0.45	0.48
GCP	388ms	0.53	0.58
keras-ocr (scale=2)	417ms	0.53	0.54
keras-ocr (scale=3)	699ms	0.5	0.59

عملکرد مدل بر روی GPU P4

Pytesseract: این کتابخانه سرعت بالایی در پیش بینی هایش دارد اما خیلی دقیق نیست برای پیش پردازش داخلی از کتابخانه Leptonica استفاده میکند که مواردی مثل skewness,distortion را در ورودی بررسی و حل میکند همچنین برای متن های کم رنگ که bold نیستند پیش پردازش هایی اعمل میکند و با اعمال آستانه گیر ادپتیو روی ورودی تصویر را باینری میکند .

این کتابخانه مبتنی بر LSTM است و میتوان بر روی دیتاست خودمان آموزش داد .

این مدل از فارسی پشتیبانی می کند و هایپر پارامتر های زیادی برای تنظیم کردن دارد که همین موضوع کار باعث می شود در تلاش اول نتوان متن را به خوبی اسکن کرد و بسته به هر نوع متن (دست نویس یا غیر دست نویس و ..) بهتر است قبل اسک این پاامتر ها تنظیم شود



مطابق خروجی های بالا عملکرد این مدل در تشخیص و اسکن متن (تقریبا هر دو) بسیار بد بوده است و پیشنهاد توسعه دهنده به انجام پیش پردازش هایی قبل اعمال مدل OCR است .

در پروژه Pytesseract-opencv پیش پردازش هایی مانند باینری کردن تصویر اعمال کرده ایم که عملکرد مدل را در بعضی بخش ها ارتقا داده است (اما همچنان در تشخیص متن کج و دست نویس دچار مشکل است)

بر اساس این نتایج باید پیش پردازش هایی برای حذف بکگراند ، نویز و رفع کجی متن قبل از دادن تصویر به مدل اعمال کرد.

***توجه :** در حالت عادی این مدل برای اسکن متن های غیر دست نویس که کج نباشند (مانند کتاب های دیجیتال pdf) مناسب است*

مدل های تجاری بررسی شده که از انها خروجی گرفته نشده :

google_vision_ocr
aws_textract_ocr
azure_ocr

مدل های متن باز بررسی شده که کد آنها خروجی نداده است :

calamari_ocr
kraken library
GOCR ocrad_ocr
cuneiform_ocr
microsoft TrOCR<- خروجی رندوم تولید کرده و نیاز به فاین تیون شدن دارد