

به نام خدا



طراحی سامانه آشکارساز تصاویر ارقام فارسی در تصاویر محیطی

مبینا مصنفات

پاییز ۱۴۰۰

۴	بررسی علمی موضوع پروژه و انتخاب مدل نهایی
۴	۱. traditional image processing techniques
۴	۲. modern deep learning networks
۵	YOLOv5
۶	انتخاب/تولید مجموعه داده‌گان
۶	evaluation metrics
۷	mAP_0.5:0.95
۷	Precision
۷	Recall
۷	Box loss
۷	Class loss
۸	Custom Dataset1
۱۱	Hoda Dataset
۱۴	Custom Dataset2
۱۶	جمع بندی از مدل های train شده
۱۶	مدل نهایی custom dataset1
۱۷	مدل نهایی custom dataset2
۱۷	مدل نهایی hoda-dataset
۱۸	نتیجه گیری
۱۹	توضیحات

آموزش مدل	۱۹
پوشه بندی github	۱۹
پوشه بندی drive	۱۹
منابع	۲۱

## بررسی علمی موضوع پروژه و انتخاب مدل نهایی

مسئله "طراحی سامانه آشکارساز تصاویر ارقام فارسی در تصاویر محیطی" از جمله مسائل تشخیص شی<sup>۱</sup> محسوب میشود. الگوریتم های تشخیص شی به دو دسته تقسیم میشوند [1]:

۱. traditional image processing techniques

۲. modern deep learning networks

۱. traditional image processing techniques

این دسته از الگوریتم های برای آموزش نیاز به دیتاست آماده ندارند و ماهیتی unsupervised دارند، در نتیجه نیازی به برچسب زنی بر روی تصاویر وجود ندارد. همچنین به فاکتور های محدودی مانند unicolor background، clutter، illumination، occlusion و سایه ها حساس هستند.

۲. modern deep learning networks

این الگوریتم های عموماً آموزش supervised دارند و قدرت GPU بر عملکرد آن ها تاثیر بسزایی دارد. این روش ها در پاسخ به challenging illumination، occlusion و complex scenes بهتر عمل میکنند. هرچند که به دیتاست بزرگی نیازی دارند که عمل برچسب زنی زمان بری خواهد داشت. البته لازم به ذکر است که هم اکنون منابع زیادی همچون Kaggle برای دسترسی به دیتاست های برچسب خورده وجود دارد.

امروزه به دلیل پاسخگویی بهتر الگوریتم های deep learning و سرعت بالای آن ها، اغلب از این دسته الگوریتم ها در تحقیقات استفاده میشود، که به دو زیر گروه one-stage و two-stage تقسیم میشوند. از جمله الگوریتم های two-stage میتوان به RCNN، SPPNet، Fast RCNN، Mask R-CNN، Pyramid Networks/FPN و G-RCNN و برای الگوریتم های one-stage نیز به SSD، RetinaNet و YOLO اشاره کرد.

امروزه YOLO به دلیل عملکرد بسیار خوب و در عین حال کوچک بودن مدل برای استفاده ها real time و محیط های on-device deployment بسیار محبوب شده است. از این رو آخرین ورژن این مدل، YOLOv5 برای انجام این تسک انتخاب شده است که در ادامه بیشتر به آن میپردازیم.

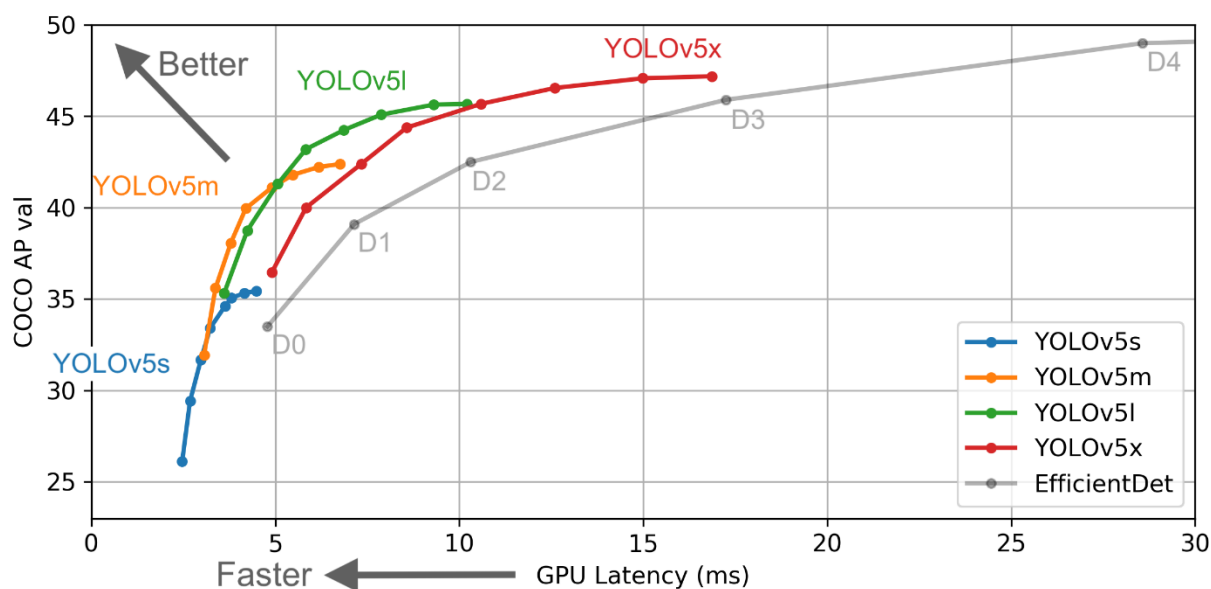
---

<sup>1</sup> Object detection

## YOLOv5

کمتر از ۵۰ روز پس از انتشار YOLOv4، YOLOv5 [2] منتشر شد که real time object detection را نسبت به نسخه قبلی خود بهبود بخشید. اولین نسخه رسمی YOLOv5 در ۲۹ ژوئن منتشر شد [3]. این نسخه دارای ۴ مدل است که عبارتند از: YOLOv5s (smallest)، YOLOv5m، YOLOv5l و YOLOv5x (largest).

YOLOv5 اولین نسخه از مدل های خانواده YOLO است که ابتدا به جای Darknet در PyTorch نوشته شده است. Darknet یک چارچوب تحقیقاتی فوق العاده انعطاف پذیر است، اما با در نظر گرفتن محیط های تولید ساخته نشده است به همی دلیل دارای تعداد کمتری از کاربران است. روی هم رفته، این امر باعث می شود تا تنظیمات و پیکرندی Darknet دشوارتر باشد و کمتر آماده تولید شود. از مزایای پیاده سازی YOLOv5 بر پایه PyTorch پیاده سازی میتوان به استفاده از اکوسیستم منتشر شده PyTorch اشاره کرد که باعث میشود پشتیبانی و استقرار آسان تر این شبکه آسان تر شود. این امر همچنین استقرار در دستگاه های تلفن همراه را ساده تر می کند، چرا که مدل را می توان به راحتی روی ONNX و CoreML کامپایل کرد. همچنین YOLOv5 بسیار سریع بوده و دارای accuracy بالایی میباشد. در انجام این تسک از سبک ترین مدل YOLOv5 یعنی YOLOv5s استفاده شده است. در تصویر زیر مقایسه بین سرعت و AP val ورژن های مختلف YOLOv5 بر روی دیتاست COCO [4] قابل مشاهده است.



## انتخاب/تولید مجموعه دادگان

مرحله بعدی و در واقع چالش برانگیز ترین مرحله انجام این تسک تهیه دیتاستی از ارقام فارسی بود!

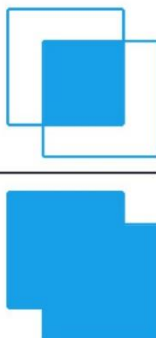
در زمینه تشخیص ارقام به زبان انگلیسی، به ویژه اعداد دست نویس پژوهش های فراوانی صورت گرفته است که نتیجه آن دیتاست های جامع و مناسب میباشد. اما چالش این بخش پیدا کردن دیتاست مناسب در محدوده زبان فارسی آن هم در تصاویر محیطی بود!

تنها دیتاست عددی ارائه شده در زبان فارسی، دیتاست هدی میباشد که به دو دلیل دست نویس بودن و تک رقم بودن هر فریم دیتاست قابل استفاده در این پژوهش نبود. طبق محدودیت های موجود و بررسی های انجام شده، جهت نتیجه گیری بهتر، این پژوهش بر روی سه دیتاست انجام شده است که در ادامه علت و نتایج هر یک را مشاهده خواهیم کرد.

## evaluation metrics

برای بررسی نتایج بدست آمده از توابع Metric و Loss استفاده شده است.

در ارزیابی مدل های تشخیص اجسام از IoU یا Intersection over Union استفاده می شود که میزان همپوشانی دو ground truth bounding box و predicted bounding box می باشد.


$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

همچنین یک threshold برای این پارامتر انتخاب شده که بعد از محاسبه IoU برای هر شیء اگر از آن میزان کمتر بوده به عنوان False Positive و اگر بزرگتر یا مساوی این مقدار باشد به عنوان True Positive شناخته می شود. بر همین اساس می توان metric های زیر را برای مدل تعریف و ارزیابی نمود.

## mAP\_0.5:0.95

برای محاسبه mAP مقدار IOU threshold رو از ۰.۵ تا ۰.۹۵ با بازه های ۰.۰۵ تغییر داده و از مقادیر آن ها میانگین می گیرد. این پارامتر بر روی تمام دیتاست محاسبه می شود.

$$\text{The mAP (mean average precision) = } \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

## Precision

از دیگر metric هایی که برای ارزیابی مدل های تشخیص اجسام به کار می روند، معیار precision می باشد. این معیار بیان کننده میزان دقت مدل است. در واقع نسبت تعداد object هایی که درست تشخیص داده شده اند به تعداد تمام تشخیص ها.

## Recall

این معیار بیانگر خلوص bounding box های predict شده می باشد. در واقع نسبت تعداد object هایی است که به درستی تشخیص داده شده اند بر تعداد تمام object های تصویر.

## Box loss

این تابع تفاوت بین prediction box و ground truth box را محاسبه می کند.

## Class loss

این تابع کلاس هایی که به اشتباه برای یک شی تشخیص داده شده اند را محاسبه می کند.

## Custom Dataset1

در ابتدا برای تهیه تصاویر دیتاست به کمک دوربین تلفن همراه از اعداد مندرج بر محصولات غذایی، کتاب ها و فلش کارت های آموزشی تصویر برداری شده و سپس تعدادی تصویر، مانند ساعت شامل اعداد فارسی نیز از اینترنت تهیه شده و به آن ها اضافه شد. نمونه هایی از این تصاویر در ادامه آمده است:



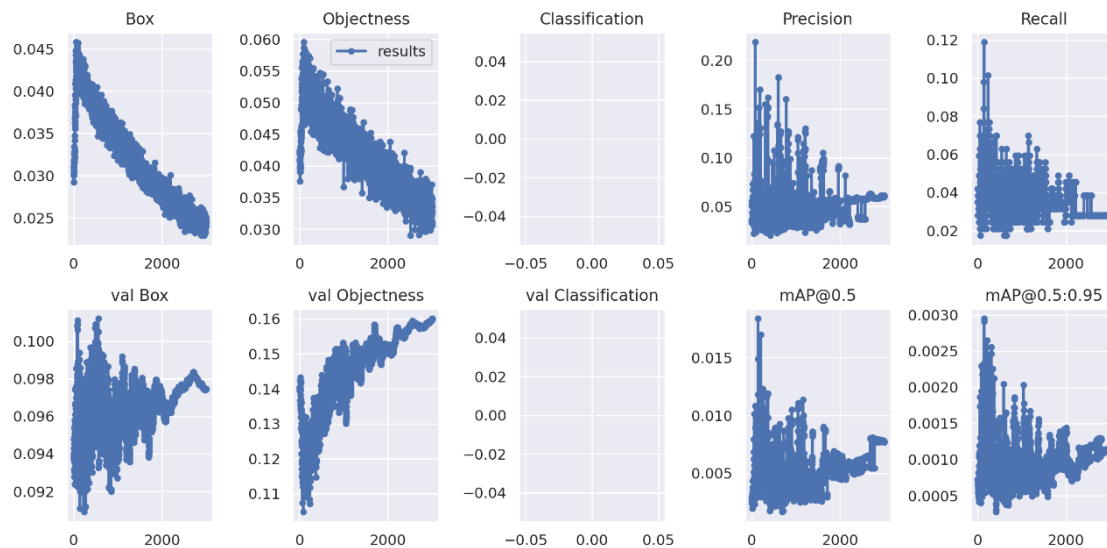
این تصاویر به کمک ابزار dark label برچسب گذاری شده و preprocessing شامل resize و normalize کردن تصاویر بر روی آن ها انجام است. تعداد کل این تصاویر ۱۰۱ فریم بود که با augment کردن (rotation و flipping) این تعداد به ۳۵۰ فریم تصویر رسید. ۸۰ درصد تصاویر برای train، ۱۰ درصد برای test و ۱۰ درصد برای validation در نظر گرفته شد.

عمل training به کمک YOLOv5s ابتدا در طی ۲۰۰۰<sup>۲</sup> اپوک صورت گرفت که نمودار های زیر نشان دهنده نتایج میباشند:

<sup>۲</sup> به طور تجربی ثابت شده است حداقل تعداد epoch های منتخب برای train، ۲۰۰۰ برابر تعداد کلاس هاست.







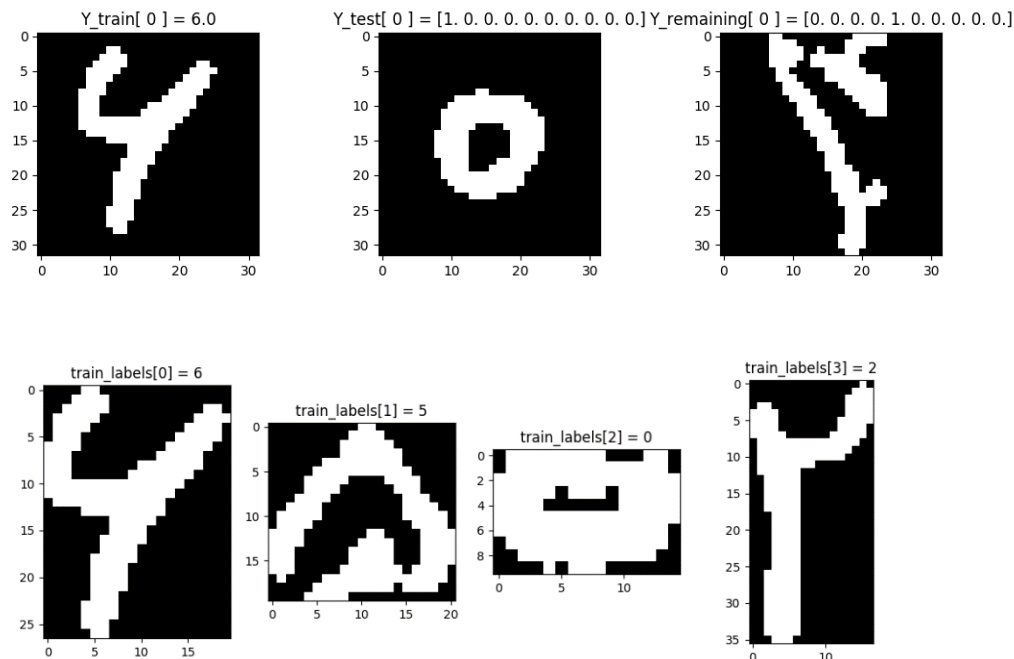
! برای بررسی دقیق تر نمودار ها میتوانید به فولدر results در فولدر hoda-dataset در درایو مراجعه کنید.

پس از بررسی نتایج این سوال پیش می آمد که " آیا مشکل فقط از دیتاست است؟ یا مدل انتخاب شده اشتباه است؟ "

برای بررسی این سوال، مسئله بر روی دیتاست دیگری نیز بررسی شد.

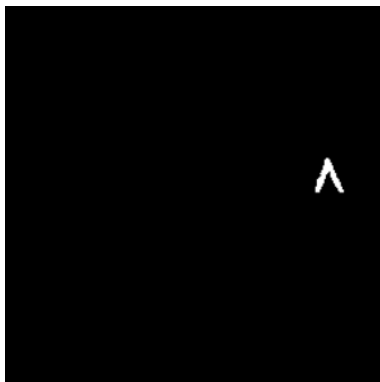
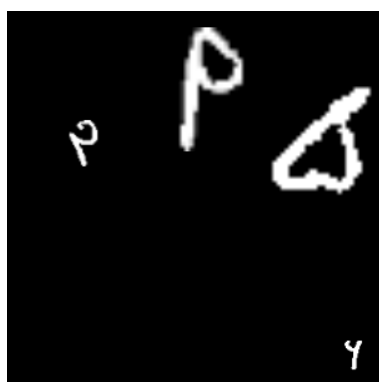
## Hoda Dataset

دیتاستی از ارقام دست نویس فارسی [5] میباشد که مشابه دیتاست معروف MNIST [5] است.



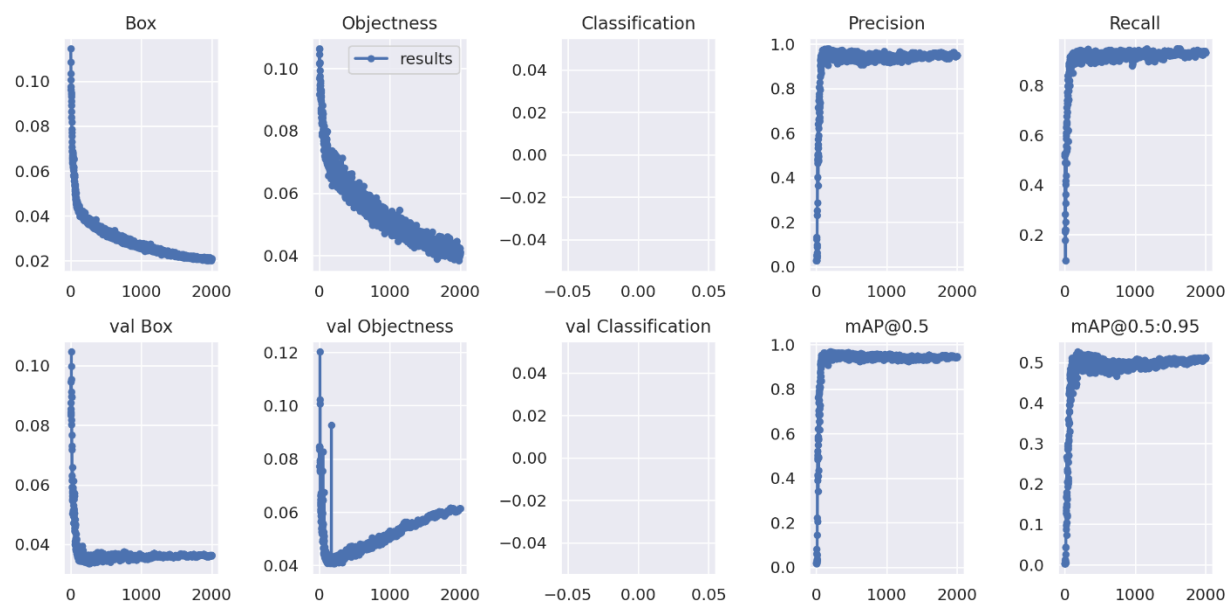
مشخص است به دلیل تک رقمه بودن هر فریم نمیشود از آن برای detection استفاده کرد، لذا به کمک کد پایتون دیتاست جدیدی از این دیتاست تهیه شد که هر فریم تصویر شامل تعدادی از فریم های تک رقمی اصلی میباشد.

در ادامه تعدادی از تصاویر دیتاست جدید قابل مشاهده است.

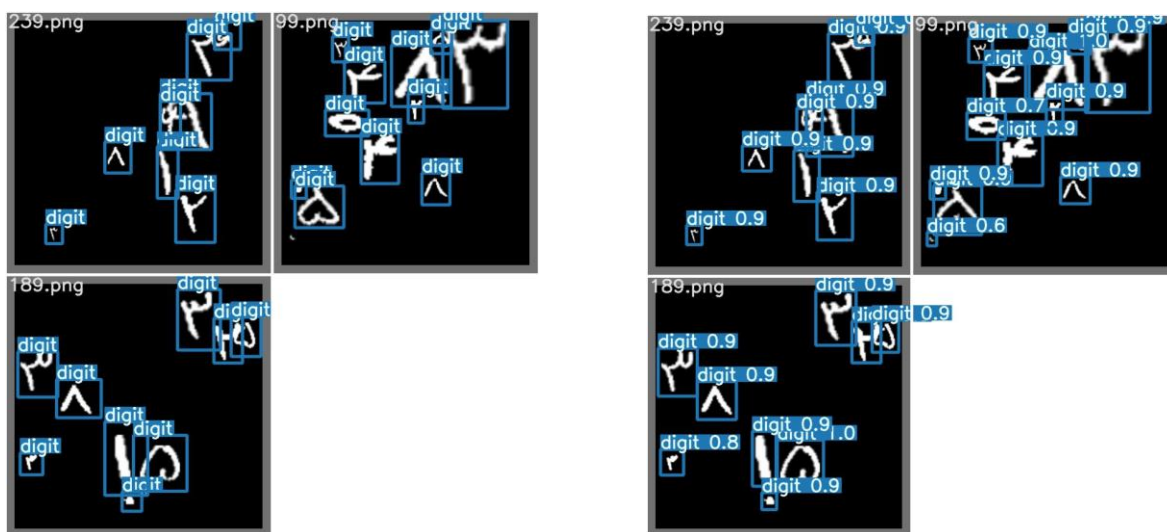


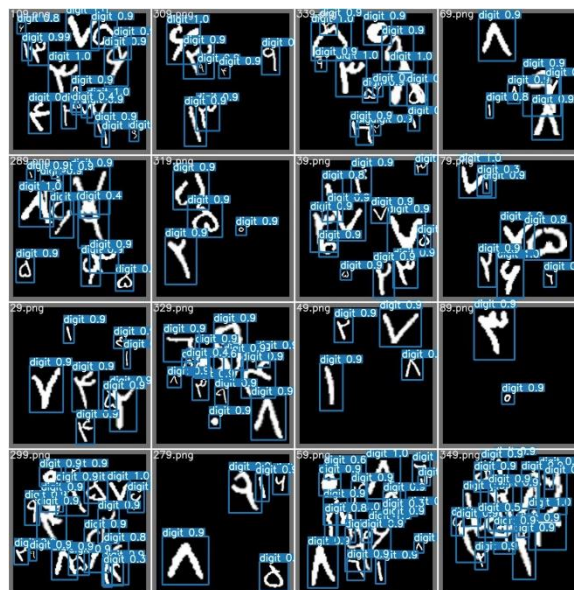
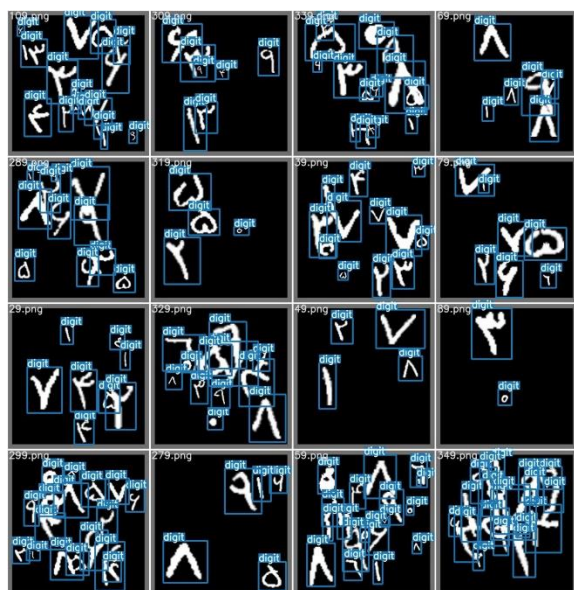
تعداد کل فریم های دیتاست ۳۵۱ میباشد که به کمک ابزار dark label برچسب گذاری شده اند و فقط عمل resizing روی آن صورت گرفته است. ۸۰ درصد تصاویر برای train، ۱۰ درصد برای test و ۱۰ درصد برای validation در نظر گرفته شد.

عمل training به کمک YOLOv5s ابتدا در طی ۲۰۰۰ اپوک صورت گرفت که نمودار های زیر نشان دهنده نتایج میباشد:



تصاویر زیر برای بررسی توانایی مدل در detect ارقام آمده است(تصاویر سمت چپ ground truth و تصاویر سمت راست predict شده هستند):





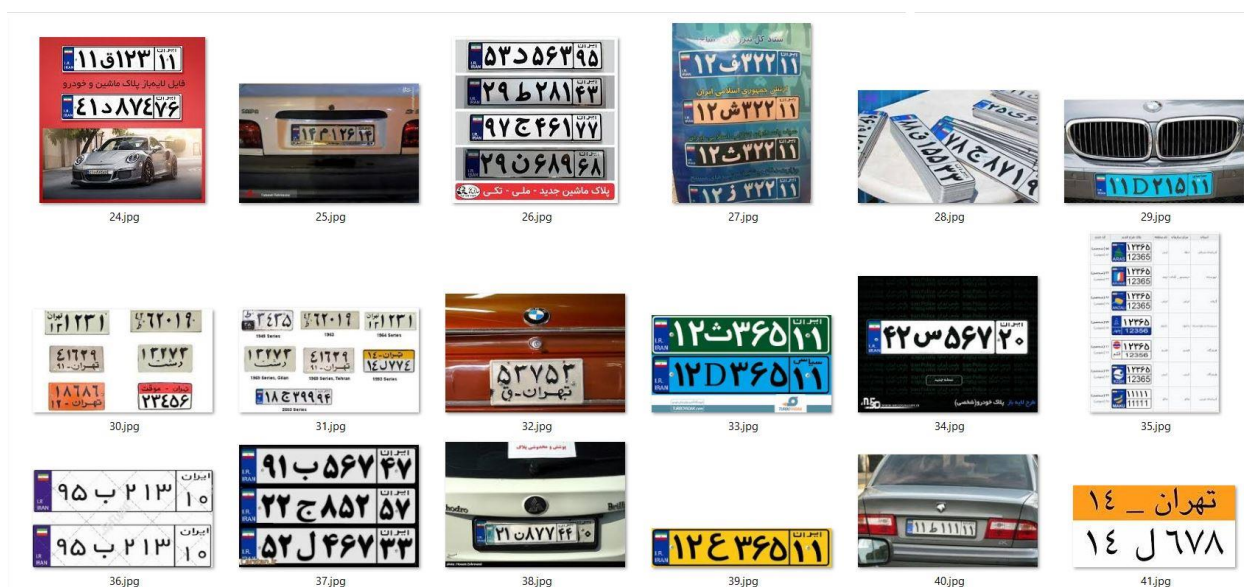
که اینبار بر خلاف اولین دیتاست به دلیل همگون بودن تصاویر نتایج بسیار خوبی قابل مشاهده است. مقادیر precision, recall و map بالا بوده و detect ارقام نیز با confidence بالایی صورت گرفته است. بنابراین میتوان نتیجه گرفت شبکه YOLOv5s به خوبی عمل میکند و مشکل از دیتاست ما میباشد.

**!** برای بررسی دقیق تر نمودار ها میتوانید به فولدر results در فولدر hoda-dataset در درایو مراجعه کنید.

به دلیل عدم دسترسی به تصاویر با کیفیت تر و همگون تر تصمیم بر آن گرفته شد که تصاویر محیطی در یکی از زیر شاخه ها بررسی شود تا مشخص شود در صورت بهبود دیتاست آیا میشود نتیجه مطلوب تری بدست آورد یا خیر؟

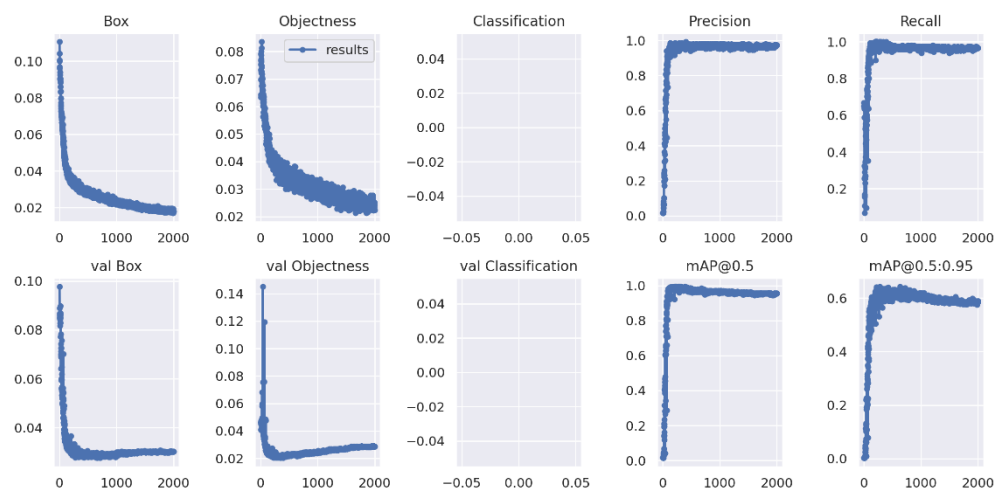
## Custom Dataset2

برای تهیه این دیتاست از تصاویر پلاک ماشین و موتور سیکلت های موجود در اینترنت استفاده شده است.



این تصاویر به کمک ابزار dark label برچسب گذاری شده و preprocessing شامل resize و normalize کردن تصاویر بر روی آن ها انجام است. تعداد کل این تصاویر 99 فریم بود که با augment کردن (rotation) این تعداد به 299 فریم تصویر رسید. 80 درصد تصاویر برای train، 10 درصد برای test و 10 درصد برای validation در نظر گرفته شد.

عمل training به کمک YOLOv5s ابتدا در طی 2000 اپوک صورت گرفت که نمودار های زیر نشان دهنده نتایج میباشند:

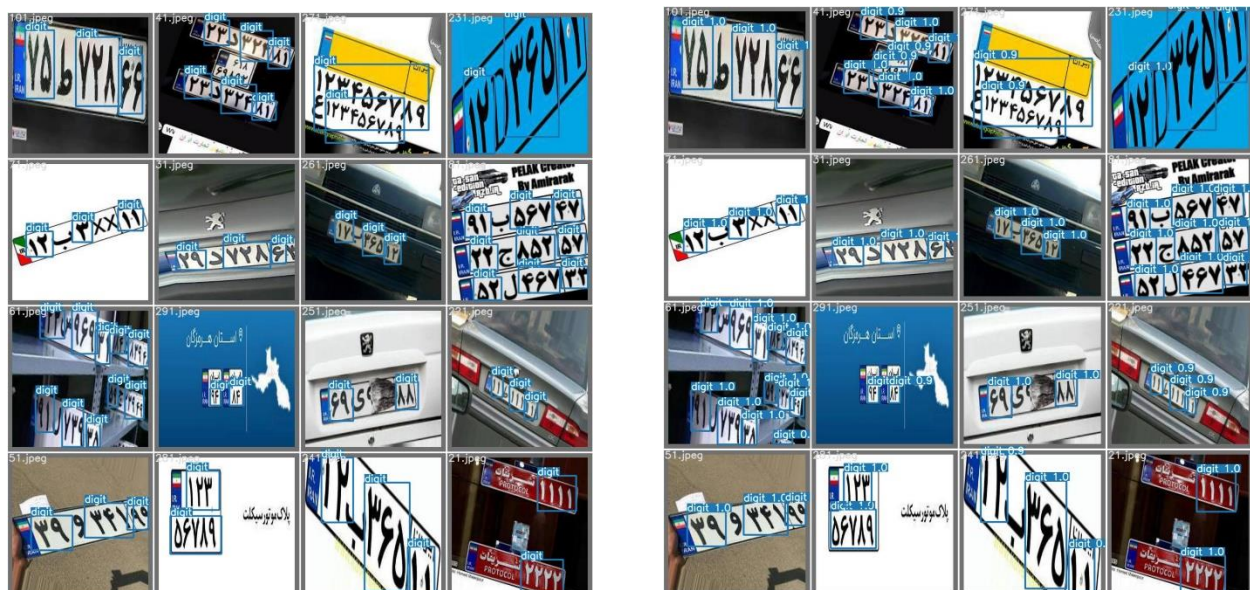




تصاویر زیر برای بررسی توانایی مدل در detect ارقام آمده است (تصویر سمت چپ ground truth و تصویر سمت راست، تصویر predict شده میباشد):



همانگونه که قابل مشاهده است معیارهای precision، recall و map مقادیر بالا و نزدیک به ۱ داشته اند و detetc در عمل YOLOv5 شبکه که نشان دهنده این است که کارایی مناسبی داشته و مشکل اصلی ما تهیه دیتاست همگون در زمینه ای میباشد که سیستم تشخیص برای آن قرار است ساخته شود (برای مثال سیستم تشخیص ارقام در پلاک ماشین ها میتواند در حوزه راهنمایی و رانندگی مورد استفاده قرار بگیرد). برای بررسی بیشتر فرایند آموزش ۳۰۰۰ اپوک دیگر با آخرین وزن یادگیری شده صورت گرفت که نتایج بهبود داده شد و دیگر مانند سری قبل حروف روی پلاک در برخی تصاویر رقم تشخیص داده نشدند.

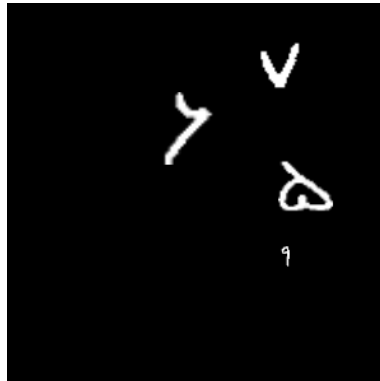
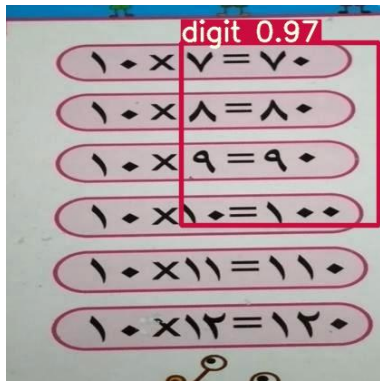


! برای بررسی دقیق تر نمودار ها میتوانید به فولدر results در DS2 در درایو مراجعه کنید.

### جمع بندی از مدل های train شده

در نهایت دیتاستی شامل ۱۵ فریم تصویر و فایل annotation آن ها در فولدر all-test از هر سه دیتاست (از هر دیتاست پنج تصویر انتخاب شده است) انتخاب شد تا نتیجه اعمال مدل های آموزش داده شده بر روی آن ها را ببینیم. در ادامه تعدادی از این نتایج را نمایش خواهیم داد اما برای مشاهده تمامی نتایج مطلوب است که به درایو مراجعه کنید.

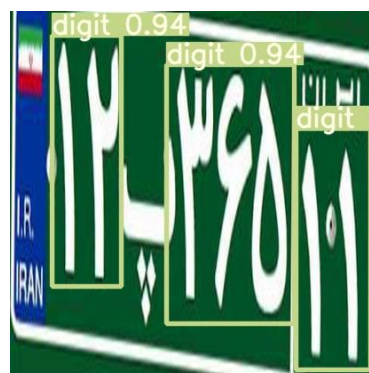
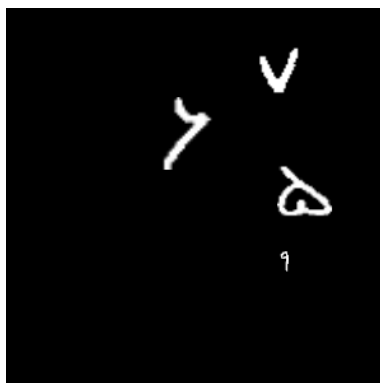
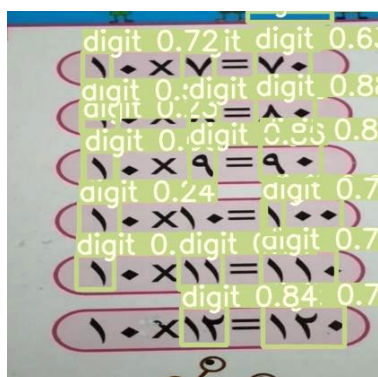
#### مدل نهایی custom dataset1



این مدل علاوه بر تصاویر دو دیتاست دیگر در تشخیص تصاویر دیتاستی که با آن آموزش داده شده است نیز ناتوان است!

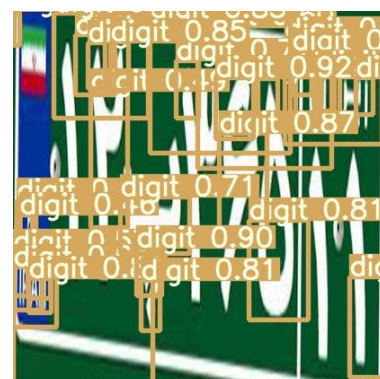
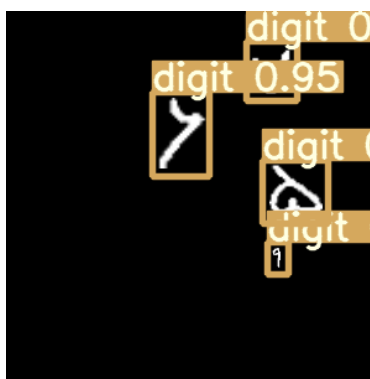
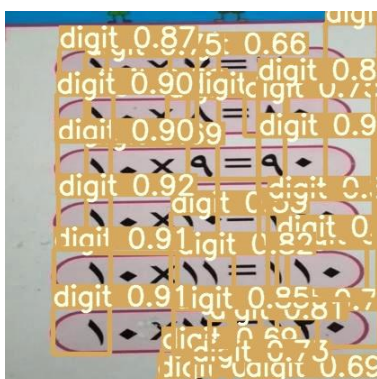


## مدل نهایی custom dataset2



این مدل در تشخیص ارقام دیتاست اول خوب عمل میکند اما در تشخیص ارقام دیتاست هدی ناتوان است.

## مدل نهایی hoda-dataset



همانگونه که مشخص است این مدل توانایی تشخیص اعداد تک رقمی در دو دیتاست دیگر را تا حد قابل قبولی دارد.

## نتیجه گیری

معماری استفاده شده از جمله معماری های سریع و دقیق در حوزه object detection بوده که برای تشخیص رقم نیز به خوبی عمل میکند. اما چالش مطرح در این مسئله انتخاب دیتاست مناسبی است که دارای diversity بالایی نباشد و ترجیحا از تصاویر موجود در حوزه ای که سیستم قرار است برای آن تعبیه شود استفاده شود تا عمل learning به خوبی انجام شده و قدرت detection مدل آموزش داده شده بالا برود.

## توضیحات

### آموزش مدل

به دلیل نیاز معماری YOLOv5 به GPU این مدل در محیط کلب و بصورت incremental آموزش داده شده است. از وزن اولیه ای استفاده نشده و batch size انتخاب شده نیز برابر ۱۶ میباشد.

### پوشه بندی github

رپازیتوری گیت هاب به اشتراک گذاشته شده شامل ۵ فولدر است:

- Notebook: این فولدر شامل نوت بوک کلبی است که در آن کد های preprocessing، augmentation و تقسیم داده ها به train، test و valid وجود دارد.
- hoda-detection-generator: این فولدر شامل سورس کد های مورد نیاز برای استفاده از دیتاست هدی و تبدیل این دیتاست به دیتاستی جهت detection میباشد. برای انجام این عملیات کافیت فایل generate\_data.py اجرا شود.
- فولدر های custom-dataset1، custom-dataset2 و hoda-dataset: هر یک از این فولدرها که هم نام یکی از سه دیتاست استفاده شده میباشد در بردارنده نتایج مربوط به آن دیتاست است. در این فولدر ها notebook مربوطه برای آموزش آن دیتاست (cell های notebook ها دارای دستورات یکسان هستند با path های متفاوت که جهت آسان تر شدن آموزش و مقایسه همزمان در سه notebook جدا پیاده سازی شده اند)، فولدر models شامل دو مدلی که یکی از آن ها بدون وزن اولیه و با ۲۰۰۰ اپوک و دیگری با وزن اولیه حاصل از مدل قبلی و با ۳۰۰۰ اپوک ترین شده است، میباشد. همچنین فولدر results شامل نتایج حاصل از دو مدل گفته شده در غالب نمودار ها و تصاویر detect شده میباشد.

### پوشه بندی drive

فولدر به اشتراک گذاشته شده خود شامل ۶ فولدر است:

- Notebook: این فولدر شامل نوت بوک کلبی است که در آن کد های preprocessing، augmentation و تقسیم داده ها به train، test و valid وجود دارد.

- `hoda-detection-generator`: این فولدر شامل سورس کد های مورد نیاز برای استفاده از دیتاست هدی و تبدیل این دیتاست به دیتاستی جهت `detetction` میباشد. برای انجام این عملیات کافیت فایل `generate_data.py` اجرا شود.
- `Test-All`: شامل یک دیتاست با ۱۵ تصویر از هر سه دیتاست میباشد (۵ تصویر از هر دیتاست). برای هر دیتاست فولدری با نام خودش وجود دارد که نتایج `detect` این ۱۵ تصویر با دو مدل آموزش داده شده با این دیتاست است.
- فولدر `hoda-dataset`: ۳ فولدر `train`, `test` و `valid` شامل تصاویر و فایل های `annotation` آن ها میباشد. در فولدر `yolov5` فایل های مورد نیاز این معماری در دسترس است که در فولدر `runs` دو فولدر مهم وجود دارد:
- ۱. `train`: برای هر بار اجرای `incremental` یک فولدر به فرمت `yolov5s_resultsn` ساخته شده است که `n` نشان دهنده `n` امین آموزش `incremental` میباشد. در این پوشه مدل نهایی و اطلاعات مربوط به نمودار ها و نتایج ذخیره شده است.
- ۲. `detect`: نتایج حاصل از اعمال دو مدل گفته شده (۲۰۰۰ اپوک بدون وزن اولیه و ۳۰۰۰ اپوک با وزن اولیه) بر روی تصاویر تست با دو `confidence` متفاوت ۰.۲ و ۰.۴ در این پوشه وجود دارد.
- فولدر `custom-datasets`: شامل دو فولدر `DS1` و `DS2` که شامل اطلاعات موجود در فولدر `hoda-dataset` اما اینبار برای `custom dataset1&2`، میباشد.

لینک google drive:

[https://drive.google.com/drive/folders/1ka5e9wZ\\_Am-MQSgfSzSUA07xrNG-ymjU?usp=sharing](https://drive.google.com/drive/folders/1ka5e9wZ_Am-MQSgfSzSUA07xrNG-ymjU?usp=sharing)

رفرنس ها:

- [1] "Object Detection in 2021: The Definitive Guide," [Online]. Available: <https://viso.ai/deep-learning/object-detection/>.
- [2] "yolov5," Ultralytics, [Online]. Available: <https://github.com/ultralytics/yolov5>.
- [3] "YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS," roboflow, [Online]. Available: <https://blog.roboflow.com/yolov5-is-here/>.
- [4] "COCO Dataset," [Online]. Available: <https://cocodataset.org/>.
- [5] "مجموعه ارقام دست نویس هدی," [Online]. Available: <https://farsiocr.ir/%d9%85%d8%ac%d9%85%d9%88%d8%b9%d9%87-%d8%af%d8%a7%d8%af%d9%87/%d9%85%d8%ac%d9%85%d9%88%d8%b9%d9%87-%d8%a7%d8%b1%d9%82%d8%a7%d9%85-%d8%af%d8%b3%d8%aa%d9%86%d9%88%db%8c%d8%b3-%d9%87%d8%af%db%8c/>.
- [6] "THE MNIST DATABASE of handwritten digits," [Online]. Available: <http://yann.lecun.com/exdb/mnist/>.