تشخیص انواع کشتی توسط الگوریتم YOLO در تصاویر

برای بیستمین کنفرانس ملی سالانه انجمن کامپیوتر ایران

محمد خدمتی[1]، محمد علی کیوان راد[2]

[1] دانشجو کارشناسی ارشد، دانشگاه صنعتی مالک اشتر، تهران، ایران
Mohammad.khedmati2012@gmail.com

[2] استادیار، دانشگاه صنعتی مالک اشتر، تهران، ایران
keyvanrad@aut.ac.ir

چکیده

سیستم های تشخیص اشیاء نقش بسیار مهمی در ایمنی و امنیت دارند. در این مقاله از شبکه های عصبی کانولوشنال عمیق برای تشخیص ۵ مدل کشتی استفاده شده است. ابتدا مجموعه ای از تصاویر کشتی جمع آوری شده و با الگوریتم یولو نسخه ۸ آموزش داده شد تا تصاویر ورودی را تشخیص دهد، سپس الگوریتم یولو نسخه ۸ با الگوریتم mask-rcnn ترکیب شده است. در مجموع، دقت روش پیشنهادی به میانگین دقت متوسط ۷۵٪ رسید و میانگین سرعت تشخیص ۱۰/۷۹ میلی ثانیه است .

کلمات کلیدی

تشخیص اشیاء، تشخیص کشتی، Yolo

1- مقدمه

گسترش روزافزون فناوری در بسیاری از زمینه‌ها به کمک بشریت آمده‌است. در این بین قابلیت ترکیب فناوری‌های مختلف نتایج بسیار خوبی به همراه داشته‌است. محققان تلاش کردند که با ترکیب برخی از فناوری‌ها مشکلات و چالش‌های موجود در دنیای امروز را کنترل کنند و راه‌حل‌های مناسبی را برای رفع این مشکلات ارائه دهند. در این بین چالش‌های امنیتی همواره جایگاه خاصی دارند. یکی از فناوری‌هایی که به کمک رفع این چالش آمده است، فناوری فناوری تشخیص اشیاء در تصاویر است. خروجی این فناوری شناسایی اشیاء مورد نیاز سازمان ها و ارگان‌های مختلفی است که با تحلیل آن می‌توان موارد خطرناک را تشخیص داد.

در دنیای امروز قابلیت‌های روبه گسترشِ سیستم‌ها و تکنیک‌های پردازش تصویر، درهای تازه‌ای را به روی پژوهش در زمینه‌ی کاربردهای پردازش تصویر در ایمنی و امنیت گشوده است. امروزه با افزایش جرایم اینترنتی و فیزیکی، نیاز به ارائه تکنیک‌های تازه و بدیع برای کنترل آن‌ها در حوزه‌ی پردازش تصویر در ایمنی و امنیت، بیش از پیش احساس می شود. لازم به‌ذکر است ظهور تکنیک‌های یادگیری ماشین و پردازش تصویر منجر به بروز فرصت‌های پژوهشی جدیدی در این زمینه شده است. یادگیری ماشین، قابلیت استخراج خودکار و تحلیل

اطلاعات از تصاویر را به وجود آورده‌است و این همراهی یادگیری ماشین و پردازش تصویر با یک دیگر در کاربری‌های متعدد امنیتی مفید واقع شده‌است. پردازش تصویر نقش کلیدی‌ای را هم در امنیت دیجیتال و هم در امنیت فیزیکی ایفا می‌کند. از جمله کاربری‌های امنیت فیزیکیِ می توان به امنیت کشور، کاربری‌های نظارتی، احراز هویت اشاره کرد.امنیت دیجیتال نیز بر حفاظت از داده‌های دیجیتال تأکید دارد.

تکنیک‌هایی مثل واترمارک کردن دیجیتال، امنیت شبکه و پنهان‌نگاری(steganography) در بخش امنیت دیجیتال طبقه بندی می‌شوند. باید در نظر داشت در هر دو زمینه‌ی امنیت فیزیکی و دیجیتال، عملکرد بدون وقفه (Real-time) مسأله‌ای حیاتی و ضروری محسوب می شود. بدین معنی که در دسترس بودن اطلاعاتِ تصویر مرتبط و با کیفیت در زمان مناسب، آگاهی کامل و جامعی را نسبت به موقعیت موردنظر ما به ارمغان می‌آورد. تکنیک‌های پردازش تصویر بدون وقفه می‌توانند عملیات لازم را با تأخیری قابل قبول و اندک در بازه زمانی مد نظر ما انجام دهند. همچنین کاربری‌های امنیت فیزیکی مانند نظارت، احراز هویت بیومتریک، انجام واترمارک یا امنیت شبکه، مواردی هستند که دارای محدودیت زمانی بوده و صد البته به پردازش تصویر بدون وقفه نیاز دارند.

بر اساس موارد گفته شده در این تحقیق سعی می‌شود توضیحاتی در مورد شناسایی اشیاء در تصاویر و کاربردهای آن ارائه گردد، سپس یکی از روش‌های پرکاربرد در این زمینه را بر روی مجموعه دادگان انتخابی آزمایش کند و در انتها نتایج اجرای رویکردهای مختلف گزارش میشود.

2- کارهای پیشین

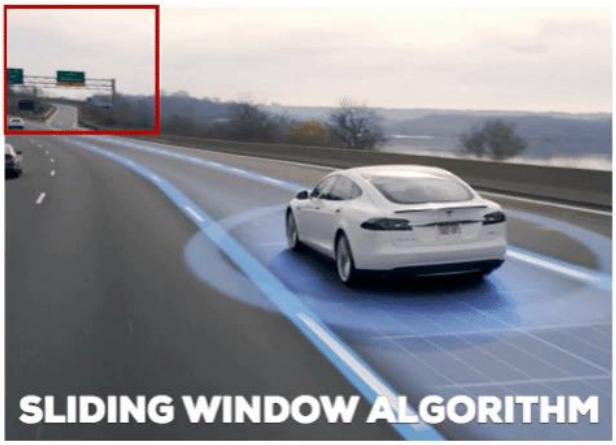
هر مقاله باید شامل این بخش‌های اصلی باشد: چکیده، کلمات کلیدی، مقدمه، مطالب اصلی، نتیجه، و مراجع. سایر بخش‌ها مثل سپاسگزاری، ضمایم و زیرنویس‌ها اختیاری است. این بخش‌ها باید در آخر مقاله و قبل از مراجع قرار گیرند، بجز بخش زیرنویس‌ها که پس از مراجع آورده می‌شود.

شماره‌گذاری بخش‌ها از مقدمه شروع می‌شود. مقدمه دارای شماره 1 است. آخرین شماره نیز مربوط به بخش نتیجه است. سایر بخش‌های قبل از مقدمه و پس از نتیجه، دارای شماره نیستند. هر بخش می‌تواند شامل چند زیربخش باشد. زیربخش‌ها نیز دارای شماره هستند که از 1 شروع می‌شود. هنگام شماره‌گذاری زیربخش‌ها دقت کنید که شماره بخش در سمت راست قرار گیرد. مثلاً برای شماره‌گذاری زیربخش 3 از بخش 2 بنویسید: 2-3. برای نوشتن عنوانِ یک بخش از سبک Heading 1، و اگر بخش دارای شماره نیست از سبک Heading 0 استفاده کنید. عنوان زیربخش‌ها (سطح 2) با سبک Heading 2 نوشته شوند. برای سطح 3 نیز از سبک Heading 3 استفاده کنید. معمولاً نیازی به زیربخش‌های سطوح بعدی وجود ندارد، با این حال اگر وجود داشت، آن زیربخش‌ها را بدون شماره و تنها بصورت متن پررنگ بنویسید.

در هر بخش یا زیربخش یک یا چند بند (پاراگراف) وجود دارد. دقت شود که جملات هر بند زنجیروار به هم مربوط باشند و یک موضوع را دنبال کنند. اولین بند هر بخش یا زیربخش بدون تورفتگی (Intend) است. برای نوشتن اولین بند، از سبک Text1 استفاده کنید. سایر بندها دارای تورفتگی به اندازه 0/5 سانتی‌متر است که برای نوشتن آن‌ها باید سبک Text را انتخاب کنید. سعی کنید از نوشتن بندهای طولانی پرهیز کنید. یک بند حداکثر می‌تواند 10 تا 15 سطر را از یک ستون، به خود اختصاص دهد.

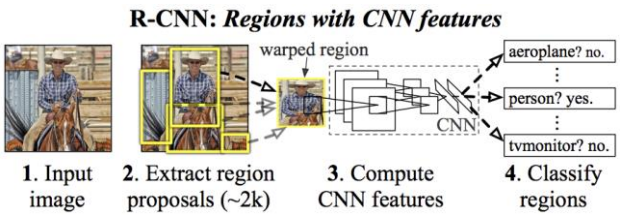
2-1- روش یولو (YOLO)

سیستم‌های تشخیص اشیای پیش از YOLO، از کلاسیفایرها در کار تشخیص اشیا استفاده می‌کردند. این سیستم‌ها برای تشخیص یک شی، یک کلاسیفایر را در موقعیت‌ها و مقیاس‌های مختلف به تصویر ورودی اعمال می‌کردند. به‌عنوان مثال، سیستم‌هایی مانند Deformable Part Models یا DPM از پنجره‌های لغزان (Sliding Window) بهره می‌برند که کلاسیفایر را به موقعیت‌های مختلف در سراسر تصویر اعمال می‌کنند. این اِعمال کلاسیفایر به موقعیت‌های مختلف تصویر، کار زمان‌بری است که البته شباهت چندانی هم به سیستم بینایی انسان در تشخیص اشیا ندارد. در تصویر 1 پنجره لغزان یولو، نمونه‌ای از الگوریتم‌های مبتنی بر پنجره لغزان را مشاهده می‌نمایید.



دسته دیگری از رهیافت‌ها که نسبت به DPM جدیدتر هستند، رهیافت‌های مبتنی بر پروپوزال ناحیه (Region Proposal) مانند R-CNN است. در تصویر RCNN 2 ، ساختار یک الگوریتم مبتنی‌بر پروپوزال ناحیه به‌نام R-CNN را مشاهده می‌نمایید. در این روش‌ها، ابتدا مجموعه زیادی پروپوزال یا همان باکس برای هر تصویر تولید می‌شوند (مثلا 2000 پروپوزال برای هر تصویر در مرحله 2 تصویر RCNN 2). سپس، هریک از پروپوزال‌ها به یک سایز مشخص ریسایز می‌شوند و برای استخراج ویژگی در اختیار شبکه‌های CNN قرار می‌گیرند (مرحله 3 در تصویر RCNN 2). درنهایت، یک کلاسیفایر برای کلاسیفای‌کردن این باکس‌های تولیدشده به‌کار برده می‌شود (مرحله 4 در تصویر RCNN 2). بنابراین، به‌همین دلیل است که گفتیم روش‌های تشخیص اشیای پیش از YOLO عمل تشخیص اشیا را با

کلاسیفایرها انجام می‌دهند. این مسیر نسبتا پیچیده سرعت پایینی دارد و بهینه‌سازی آن مشکل است، چون هریک از این اجزا که در شکل تصویر 2 RCNN مشاهده می‌کنید، باید به‌صورت جداگانه آموزش داده شوند.[1]



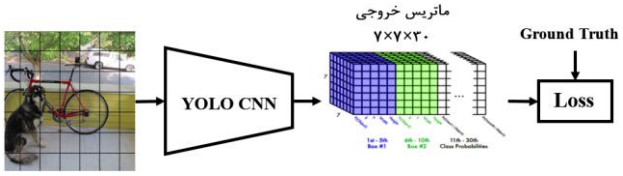
YOLO معماری سیستم‌های تشخیص اشیا را دستخوش تغییراتی کرده‌است و به مساله تشخیص اشیا به‌صورت یک مساله رگرسیون می‌نگرد که مستقیم از پیکسل‌های تصویر به مختصات باکس و احتمال کلاس‌ها می‌رسد. با استفاده از سیستم YOLO، برای تشخیص اشیای موجود در تصویر، به هر تصویر شما فقط یک بار می‌نگرید (You Only Look Once). YOLO بسیار ساده است (به شکل \ref{fig:yolo} نگاه کنید). تنها یک شبکه کانولوشنی وجود دارد که تصویر ریسایز ورودی را دریافت (مرحله 1) و سپس به‌صورت همزمان چندین باکس را به همراه احتمال کلاس‌ها پیش‌بینی می‌کند (مرحله 2). YOLO روی تصاویر کامل آموزش می‌بیند و مستقیما کارآیی تشخیص را بهبود می‌دهد.

اول، YOLO بسیار سریع است. در اینجا، تنها یک شبکه وجود دارد که خیلی ساده به آن ورودی تصویر داده می‌شود تا شبکه پیش‌بینی‌های تشخیص اشیا را به ما نشان دهد. دو نسخه شبکه YOLO شامل YOLO اصلی و YOLO سریع طراحی شده است. YOLO اصلی با کارت گرافیک Titan X با سرعت 45 فریم‌برثانیه اجرا می‌شود. نسخه سریع YOLO هم سرعتی بیش از 150 فریم‌برثانیه دارد. یعنی YOLO می‌تواند در یک ویدئوی 40 فریم‌برثانیه درحالت بلادرنگ به تشخیص اشیا بپردازد. YOLO نسبت به دیگر سیستم‌های تشخیص اشیای بلادرنگ، به mAP یا همان mean Average Precision دوبرابر دست یافته است. دقت کنید، عملکرد بهتر نسبت به سایر سیستم‌های بلادرنگ و نه سیستم‌های تشخیص اشیای قدرتمند مانند Faster R-CNN که بلادرنگ نیستند.

دوم، YOLO برای پیش‌بینی تشخیص، به صورت کلی (Global) به تصویر نگاه می‌کند. برخلاف تکنیک‌های پنجره‌های لغزان (اسلاید) و پروپوزال، YOLO به کل تصویر نگاه می‌کند.

سوم، YOLO تعمیم‌پذیری بالایی دارد. زمانی‌که تصاویر به شبکه آموزش داده می‌شوند و سپس شبکه آموزش‌دیده روی کارهای هنری تست می‌شود (در واقع منظورمان همان تغییر حوزه داده‌های ورودی است) شبکه YOLO با فاصله زیادی بهتر از شبکه‌هایی مانند DPM و R-CNN کار می‌کند. بنابراین، YOLO به شدت تعمیم‌پذیر هست و در مقابل حوزه‌های جدید و یا داده‌های ورودی غیرمنتظره با احتمال کمتری نسبت به بقیه سیستم‌ها با شکست مواجه می‌شود.

ساختار کلی الگوریتم YOLO در تصویر YOLO3 نشان داده شده‌است. تصویر ورودی با ابعاد 3×448×448 به یک Grid یا شبکه S×S تقسیم‌بندی می‌شود. این تصویر به شبکه YOLO داده می‌شود. خروجی شبکه کانولوشنی، ماتریسی به ابعاد S×S×30 خواهد بود. هریک از درایه‌های ماتریس S×S خروجی معادل با یک سلول در شبکه S×S ورودی است (به ورودی و خروجی در تصویر YOLO3 دقت کنید). خروجی S×S×30 شامل مختصات باکس‌ها و احتمال‌هاست. اگر در فرآیند آموزش (Train) باشیم، خروجی S×S×30 به‌همراه باکس‌های واقعی یا هدف (Ground Truth) به تابع اتلاف داده می‌شود. مقدار S در یولو نسخه 1، برابر با 7 درنظر گرفته شده است. اگر در فرآیند آزمایش (Test) باشیم، خروجی S×S×30 به الگوریتم حذف غیرحداکثرها (Non-maximum Suppression) داده می‌شود تا باکس‌های ضعیف از بین بروند و تنها باکس‌های درست در خروجی نمایش داده شوند. درادامه، درمورد طراحی شبکه YOLO، نحوه آموزش شبکه، تابع اتلاف، آزمایش شبکه و غیره توضیح خواهیم داد.



ابتدا، شبکه YOLO با پایگاه داده 1000 کلاسه ImageNet برای عمل کلاس‌بندی آموزش داده شده‌است. در این فرآیند آموزش، از 20 لایه کانولوشنی ابتدایی YOLO استفاده شده‌است. در انتهای این 20 لایه، یک لایه پولینگ میانگین (Average Pooling) و یک لایه فولی‌کانکتد قرار گرفته‌است. تصاویر ورودی در اندازه 3×224×224 به شبکه داده شده‌اند. این شبکه تقریبا به‌مدت یک هفته آموزش داده شده که درنهایت دقت 88٪ در top-5 در ImageNet حاصل شده‌است.

در مرحله دوم، برای کار تشخیص اشیا در ساختار مدل تغییراتی ایجاد شده است. تغییرات به این‌صورت است که چهار لایه کانولوشنی و دو لایه فولی کانکتد با وزن‌های تصادفی به انتهای 20 لایه شبکه اضافه شده‌است. در کار تشخیص اشیا اغلب به اطلاعات با جزئیات بیشتری نیاز است، به‌همین دلیل رزولوشن ورودی شبکه از 3×244×224 به 3×448×448 افزایش داده شده‌است. بنابراین، هدف از افزایش اندازه ورودی، بهره‌گیری از جزئیات بیشتر در تصویر است.

درمورد ورودی شبکه توضیح داده شد. حال نوبت به بررسی خروجی شبکه است. اندازه خروجی شبکه 30×7×7 است. ابتدا از اندازه 7×7 شروع کنیم؛ تصاویر ورودی به یک شبکه 7×7 تقسیم‌بندی می‌شوند (در تصویر YOLO3 نشان داده شده است). بنابراین، خروجی 7×7 متناظر با تصویر شبکه‌شده ورودی است. هر درایه در 7×7 خروجی، متناظر با یک سلول در تصویر شبکه‌شده ورودی است (تصویر YOLO3). هر درایه از این ماتریس 7×7 خروجی، یک بردار به‌طول 30 دارد (تصویر YOLO3). این بردار به‌طول 30 شامل اطلاعات پیش‌بینی احتمال‌ها و مختصات باکس است. اما چگونه؟ هر سلول از این آرایه 7×7 دو باکس می‌تواند رسم کند. برای رسم هر باکس به 5 پارامتر ({x,y,w,h,confidence}) نیاز است. پارامترهای x و y، مختصات سطر و ستون مبدا باکس (مرکز باکس) را نشان می‌دهند. مختصات w و h به‌ترتیب متناظر با پهنا و ارتفاع باکس هستند. با این چهار پارامتر می‌توانیم باکس را ترسیم کنیم، درحالی‌که گفتیم 5 پارامتر برای ترسیم باکس نیاز است. پارامتر پنجم چه کاربردی دارد؟ پارامتر پنجم confidence

هست؛ یک پارامتر احتمالاتی با مقدار بین 0 تا 1 که می‌گوید اصلا این باکس شامل شی هست یا این‌که پس‌زمینه تصویر است! طبیعتا ما باکس‌هایی را می‌خواهیم که مقدار confidence بزرگی داشته باشند که نشان می‌دهد این باکس شامل یک شی است. مقدار confidence از طریق رابطه IoU بین باکس پیش‌بینی و باکس واقعی محاسبه می‌شود.

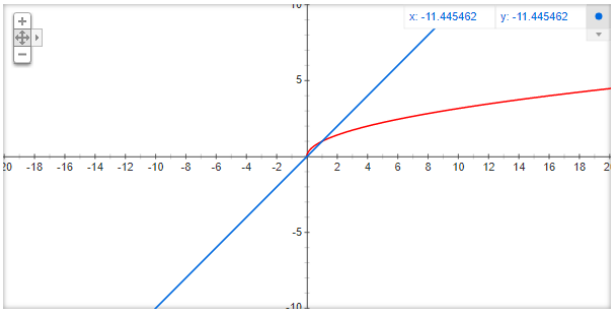
2-2- تابع اتلاف در الگوریتم YOLO

در YOLO از تابع اتلاف MSE یا Mean Squared Error استفاده شده‌است، چون بهینه‌سازی این تابع اتلاف آسان است و با مساله رگرسیون که در YOLO مطرح شده سازگار است. پلی به گذشته می‌زنیم و یادآوری می‌کنیم که در این مقاله بارها گفته شد که الگوریتم یولو به مساله تشخیص اشیا به‌صورت رگرسیون می‌نگرد. حال اینجا هم خروجی شبکه را مشاهده کردید و هم اینکه تابع اتلاف MSE نشان‌دهنده دلیل رگرسیون هست. اما لازم است در تابع اتلاف MSE تغییراتی ایجاد شود که بیشتر با خواسته ما برای تشخیص اشیا هم‌راستا باشد. تابع اتلاف نهایی در YOLO v1 به‌شکل زیر است:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

بیایید از ابتدا، خطبه‌خط تابع اتلاف الگوریتم یولو را بررسی کنیم؛ در خط اول، با استفاده از رابطه SSE موقعیت مبداهای دو باکس پیش‌بینی و واقعی (x,y) باهم مقایسه شده‌اند. اندیس‌های i و j به‌ترتیب نشان‌دهنده سلول‌ها (49 سلول داریم) و باکس‌ها (B) هستند. پشت سیگما یک متغیر obj1 لحاظ شده‌است؛ درصورتی 1 هست که باکس j در سلول i شامل یک شی باشد، درغیراینصورت صفر خواهد بود. سلولی که شامل باکسی باشد که شی در آن وجود ندارد و شامل پس‌زمینه باشد، چه ارزشی برای ما دارد؟ دو سیگما داریم که وظیفه‌شان بررسی تک‌تک سلول‌ها و باکس‌ها هست. پشت سیگماهاپیرپارامتر λ قرار دارد. این پارامتر را به ذهن بسپارید تا در آخر درباره آن توضیح دهیم.

در خط دوم، رابطه تقریبا مشابهی با خط اول می‌بینیم. اما بجای x و y از w و h استفاده شده‌است. یعنی در این‌جا می‌خواهیم پهنا و ارتفاع باکس پیش‌بینی را با باکس واقعی مقایسه کنیم. اما چرا w و h داخل یک √ قرار دارند؟ به این سوال فکر کنید و جواب آن را بیابید و بعد درادامه پاسخ آن را بخوانید. جواب آن بسیار ساده و جالب است؛ در تصویر، اشیای با اندازه‌های مختلف از خیلی کوچک تا خیلی بزرگ داریم. حالا وقتی‌که بخواهیم باکس‌های این اشیا را با باکس‌های واقعی مقایسه کنیم، همه باکس‌ها با هر اندازه‌ای را با یک معیار مقایسه می‌کنیم. درحالی‌که می‌دانیم خطا در باکس‌های بزرگ مانند خطا در باکس‌های کوچک نیست. به‌عبارت دیگر، یک پیکسل خطا در باکس بزرگ باید کمتر مجازات داشته باشد تا یک پیکسل خطا در باکس کوچک. با استفاده از √، ما باکس‌های بزرگ را کمتر از باکس‌های کوچک مجازات می‌کنیم. کافی‌است نمودار {y=x} و {y=√x} را در تصویر 4 تابع هزینه یولو باهم مقایسه کنید.



خط سوم و چهارم، ضریب اطمینان (Confidence) برای حضور یا عدم حضور یک شی در باکس هست. اول اینکه، خط سوم برای ضرایب اطمینان باکس‌هایی است که شامل شی هستند و خط چهارم متناظر با باکس‌هایی است که شامل هیچ‌گونه شی نیستند. پشت سیگماهای خط چهارم، یک‌هاپیرپارامتر λ قرار داده شده‌است. مقدار این پارامتر 0.5 درنظر گرفته شده‌است. چرا؟ چون، در هر تصویر بسیاری از باکس‌ها شامل شی نیستند و تعداد باکس‌های بدون شی بیشتر از با شی هست. برای این‌که مقدار اتلاف باکس‌های بدون شی بر باکس‌های با شی غلبه نداشته باشد، ضریب 0.5 پشت آن قرار داده شده تا مقدار اتلاف باکس‌های بدون‌شی کاهش یابد. درنهایت، مقدار احتمال کلاس‌ها باهم مقایسه شده‌اند.

2-3- دارک نت

دارک نت، یک فریمورک (framework) یا چارچوب متن باز برای شبکه عصبی است که به زبان C و CUDA نوشته شده‌است. دارک نت (Darknet) ، معماری زیربنایی و اساسی شبکه را تنظیم می‌کند و به عنوان فریمورک آموزش YOLO مورد استفاده قرار گرفته‌است. این پیاده سازی که توسط خود «ردمون» معرفی شده‌است، سریع و به سادگی قابل نصب بوده و پردازشگرهای CPU و GPU را پشتیبانی می‌کند. بعدها، یک ترجمه مبتنی بر کتابخانه (PyTorch translation) برای YOLO} v3 توسط «گلن جوچر» (Glenn Jocher) از شرکت Ultralytics معرفی شده‌است. YOLO v2} می تواند تصاویر را با سرعت 40 تا 90 فریم در ثانیه پردازش کند در حالی که YOLO v3 این امکان را به ما می‌دهد که به سادگی، تنها با تغییر اندازه مدل و بدون نیاز به آموزش مجدد، به یک توازن و مصالحه میان سرعت و دقت دست پیدا کنیم.

2-4- مدل yolo3

پیاده سازی اصلی YOLO که توسط «ردمون» صورت گرفته است، مبتنی بر دارک نت (Darknet) است. [2]

Model	Train	Test	mAP	FPS
YOLO	VOC 2007+2012	2007	63.4	45
Fast YOLO	VOC 2007+2012	2007	52.7	155
YOLOv2	VOC 2007+2012	2007	76.8	67
YOLOv2 544x544	VOC 2007+2012	2007	78.6	40
Tiny YOLO	VOC 2007+2012	2007	57.1	207
YOLOv2 608x608	COCO trainval	test-dev	48.1	40
Tiny YOLO	COCO trainval	-	-	200
YOLOv3-320	COCO trainval	test-dev	51.5	45
YOLOv3-416	COCO trainval	test-dev	55.3	35
YOLOv3-608	COCO trainval	test-dev	57.9	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	220
YOLOv3-spp	COCO trainval	test-dev	60.6	20

2-5- مدل yolo5

YOLO v5 با استفاده از جعبه‌های لنگر، جعبه‌های محدود کننده اشیاء در یک تصویر را پیش بینی می‌کند. این مدل با استفاده از جعبه‌های لنگر پیش‌بینی می‌کند که کدام یک از بسیاری از جعبه‌های از پیش تعریف‌شده با نسبت‌های مختلف، با آیتم موجود در تصویر مطابقت دارد. این‌ها جعبه‌های از پیش تعریف شده هستند. و آن‌ها YOLO v5 را قادر می سازند تا موارد موجود در یک تصویر را با دقت تشخیص دهد و پیدا کند. [3]

2-5-1- تقویت داده های موزاییکی

هنگام آموزش، YOLO v5 از روشی به نام موزاییک استفاده می‌کند افزایش داده‌ها. برای ایجاد تصاویر آموزشی تازه، مدل ما به طور تصادفی تکه‌هایی از چندین عکس را ترکیب می‌کند. در نتیجه، مدل انعطاف‌پذیرتر و قابل اعتمادتر می‌شود. از این رو، می توان به داده‌های جدید تعمیم داد و بیش از حد برازش را کاهش داد.

2-5-2- یک خط لوله آموزشی منحصر به فرد

یک خط لوله آموزشی منحصر به فرد که با نظارت و یادگیری بدون نظارت استفاده می‌شود.

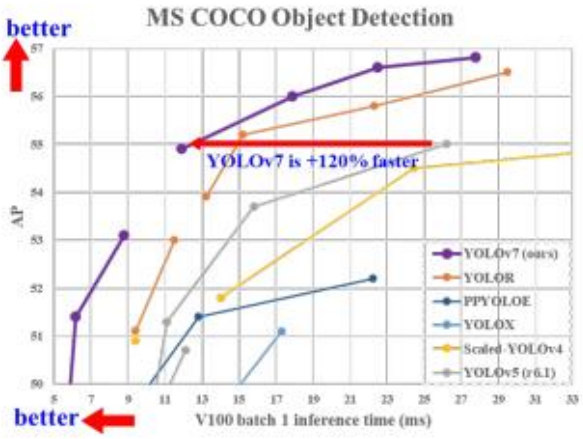
بنابراین، مدل از یک نمونه کوچک‌تر یاد می‌گیرد و از ورودی بدون برچسب به طور موثر استفاده می‌کند. این کار عملکرد مدل را افزایش می‌دهد و ظرفیت آن را برای تعمیم به ورودی‌های جدید افزایش می‌دهد.

معماری YOLO v5 لایه‌هایی را که باقی مانده و غیر باقی مانده هستند ترکیب می‌کند. با اجازه دادن به گرادیان‌ها برای جریان در سراسر لایه‌ها، لایه‌های باقیمانده به مدل در یادگیری ویژگی‌های دشوار کمک می‌کنند. همچنین، لایه‌های غیر باقیمانده، درک جامع‌تری از تصویر ورودی به مدل ارائه می‌دهند. در نتیجه، YOLO v5 می‌تواند دقیق تر و موثرتر عمل کند.

2-6- مدل yolo7

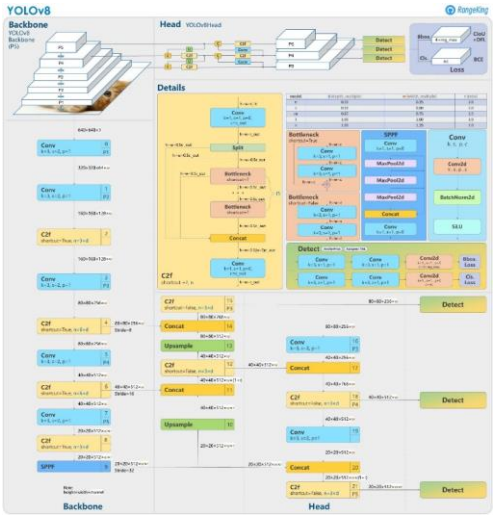
مدل YOLO V7 توسط WongKinYiu و AlexeyAB خالقان YOLO v4 Darknet ساخته شده است. این مدل بسیار شگفت انگیز که در فریمورک PyTorch ساخته شده، دارای عملکردی پیشرفته روی دیتاست MS COCO برای تشخیص اشیاء در زمان واقعی است . سرعت آن طبق تعریف، ۵ فریم در ثانیه یا سریع‌تر در GPU V100 است. ارزیابی مدل YOLO v7 نشان می‌دهد که این مدل سریع‌تر (محور x) و با دقت بیشتری (محور y) نسبت به سایر مدل‌های تشخیص اشیاء نزدیک به همزمان (real time object detection) است. یعنی با صرف زمان بسیار کوتاهی نسبت به تشخیص اشیا اقدام می‌نماید. این مدل روی دیتاست COCO آموزش داده شده‌است. این دیتاست دارای ۸۰ کلاس است، بدین معنی که این مدل قادر است ۸۰ شی(Object) مختلف تعریف شده را تشخیص دهد.

جهت استفاده مستقیم از مدل YOLO V7 برای تشخیص یک یا دو object ، نیاز است که مدل YOLO V7 روی دیتاست مورد نظر، که اصطلاحا custom data set نامیده می‌شود، مجددا آموزش ببیند تا وزن‌های مدل برای شی جدید منطبق (customize) شود. [4]



3- مدل پیشنهادی

پیشنهادی ما شامل بخش‌های مختلف از جمله پیش پردازش تصاویر، انتخاب مدل مناسب ، استخراج ویژگی، و دسته‌بندی است. ما در این پروژه از مدل از قبل آموزش دیده YOLO 8 استفاده کردیم فلوچارت کلی مدل پیشنهادی در **Error! Reference source not found.** نشان داده شده‌است.



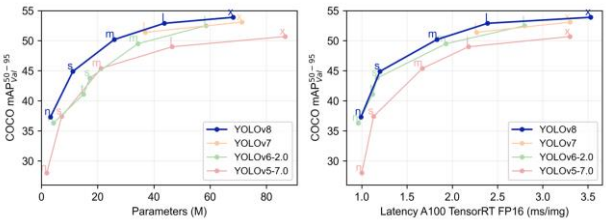
3-1- مدل yolo8

آخرین نسخه از تحسین برانگیز تشخیص شی در زمان واقعی و مدل تقسیم بندی تصویر. YOLO v8 بر اساس پیشرفت‌های پیشرفته در یادگیری عمیق و بینایی کامپیوتر ساخته شده‌است و عملکرد بی نظیری از نظر سرعت و دقت ارائه می‌دهد. طراحی ساده آن، آن را برای برنامه‌های مختلف مناسب می‌کند و به راحتی با پلتفرم‌های سخت افزاری مختلف، از دستگاه‌های لبه گرفته تا API های ابری، سازگار است. YOLO v8 یک مدل جدید بینایی کامپیوتری پیشرفته است که توسط Ultralytics خالق YOLO v5 ساخته شده است. مدل YOLO v8 شامل پشتیبانی خارج از جعبه برای وظایف تشخیص، طبقه‌بندی و بخش‌بندی است.

YOLO v8 آخرین نسخه از خانواده مدل‌های تشخیص اشیا YOLO است. در سال 2021 توسط Alexey Bochkovskiy و همکاران منتشر شد. YOLO v8 بر اساس موفقیت پیشینیان خود، YOLO v7 و YOLO v6 ساخته شده‌است و چندین پیشرفت در بینایی کامپیوتر را شامل می‌شود. یکی از ویژگی‌های کلیدی YOLO v8 افزایش دقت و سرعت آن نسبت به نسخه‌های قبلی است. این امر با استفاده از معماری شبکه عصبی بزرگ‌تر و عمیق‌تر آموزش داده شده بر روی یک مجموعه داده در مقیاس بزرگ به دست می‌آید.

علاوه بر این، YOLO v8 از تکنیکی به نام "فعال سازی چرخش" استفاده می‌کند که به بهبود همگرایی شبکه در طول آموزش معروف است.

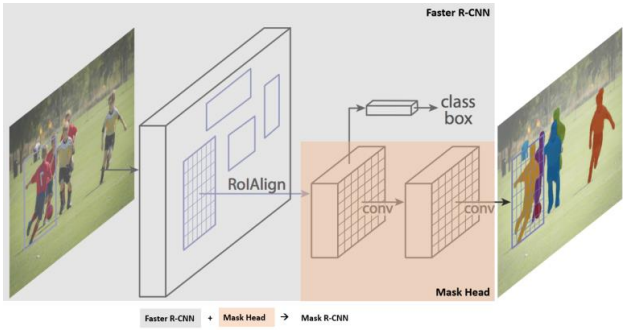
یکی دیگر از پیشرفت‌های مهم در YOLO v8 استفاده از یک معماری backbone جدید به نام CSPDarknet است. این معماری برای بهبود جریان اطلاعات از طریق شبکه با کاهش تعداد لایه‌های کانولوشنال و حفظ همان سطح دقت طراحی شده‌است. YOLO v8 همچنین دارای ویژگی‌هایی مانند افزایش داده‌ها، زمان بندی نرخ یادگیری و استراتژی‌های آموزشی بهبود یافته برای افزایش عملکرد است. به طور کلی، YOLO v8 یک الگوریتم تشخیص اشیاء پیشرفته است که به طور قابل توجهی دقت و سرعت را نسبت به نسخه‌های قبلی بهبود می‌بخشد و آن را به گزینه‌ای محبوب برای برنامه‌های مختلف بینایی رایانه تبدیل می‌کند. [5]



3-2- الگوریتم Mask-RCNN

Mask RCNN یک الگوریتم شناسایی شیء است که در سال 2018 توسط Kaiming He و همکارانش معرفی شد. این الگوریتم با استفاده از یک شبکه عصبی کانولوشنی عمیق (CNN) برای شناسایی اشیاء و همچنین تولید ماسک برای هر شیء استفاده می شود.[6]

رویکرد mask-rcnn به طور موثر اشیاء را در یک تصویر تشخیص می‌دهد در حالی که به طور همزمان یک ماسک تقسیم‌بندی با کیفیت بالا برای هر نمونه ایجاد می‌کند. این روش ، R-CNN سریعتر را با افزودن یک شاخه برای پیش‌بینی یک ماسک شی به موازات شاخه موجود برای تشخیص جعبه مرزی گسترش می‌دهد. آموزش Mask R-CNN ساده است و تنها مقدار کمی به R-CNN سریعتر اضافه می کند که با سرعت 5 فریم در ثانیه اجرا می شود. علاوه بر این، Mask R-CNN را به راحتی می توان به کارهای دیگر تعمیم داد، به عنوان مثال، به ما امکان می دهد موقعیت های انسانی را در همان چارچوب تخمین بزنیم.

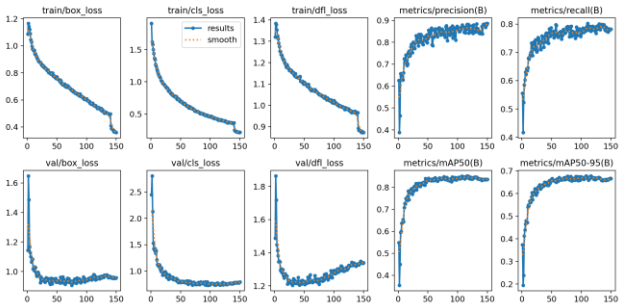


4- نتیجه

فرآیند آموزش بر روی سیستمی با یک گرافیک Nvidia RTX 1080 TI} و 32 گیگابایت حافظه رم انجام گردید. مدت زمان آموزش شبکه‌های مربوط به استخراج ویژگی با تعداد تکرار 100 مرحله ، با اندازه دسته‌های تصاویر 16، 32 و 64 تایی به ترتیب حدود 0/87±100، 0/87±130 و 0/87±160 دقیقه برای آموزش Yolo به تنهایی بود. زمان اجرای برای Mask-rcnn حدود 87، 99 و 120 دقیقه بود.

نتایج بدست‌آمده از هر تصویر و هر دسته بصورت جزئی در جداول مربوطه آمده است. نتایج بدست‌آمده مدل بر روی داده‌های آزمایش و محاسبه میانگین نتایج صحت با بازه اطمینان 75 ٪ است. میتوانید مقادیر بدست آمده را در تصاویر برای loss و auc مشاهده کنید. همانطوری که در تصویر

مشاهده میکنید برای این حجم از تصاویر میتوان با حدود ۵۰ – ۶۰ epochs
در آموزش یولو به نتایج مطلوبی رسید.



مراجع

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” 2013.

[2] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018.

[3] W. Wu et al., “Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image,” PLoS One, vol. 16, no. 10, p. e0259283, Oct. 2021.

[4] Y. Xiao et al., “A review of object detection based on deep learning,” Multimed. Tools Appl., vol. 79, no. 33–34, pp. 23729–23791, Sep. 2020.

[5] G. Jocher, A. Chaurasia, and J. Qiu, Ultralytics YOLOv8. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, ‘Mask R-CNN’, 2017.