دانشکده مهندسی مهندسی کامپیوتر

ردیابی همزمان خودروها در بزرگراه

پایان‌نامه یا رساله برای دریافت درجه کارشناسی

در رشته مهندسی کامپیوتر گرایش نرم‌افزار

نام دانشجو

فاطمه اکبری

استاد راهنما:

دکتر محمد رضا محمدی

فرودین ماه 1399

دانشکده مهندسی کامپیوتر

ردیابی همزمان خودروها در بزرگراه

پایان‌نامه یا رساله برای دریافت درجه کارشناسی

در رشته مهندسی کامپیوتر گرایش نرم افزار

نام دانشجو

فاطمه اکبری

استاد راهنما:

دکتر محمدرضا محمدی

اساتید مشاور:

دکتر محسن سریانی

فرودین ماه 1399


بسم الله الرحمن الرحيم

تأییدیه‌ی هیأت داوران جلسه‌ی دفاع از پایان‌نامه/رساله

نام دانشکده:

نام دانشجو:

عنوان پایان‌نامه یا رساله:

تاریخ دفاع:

رشته:

گرایش:

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضا
1	استاد راهنما				
2	استاد راهنما				
3	استاد مشاور				
4	استاد مشاور				
5	استاد مدعو خارجی				
6	استاد مدعو خارجی				
7	استاد مدعو داخلی				
8	استاد مدعو داخلی				

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب فاطمه اکبری به شماره دانشجویی 94521027 دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پایان‌نامه/رساله حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری‌شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. درصورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض درخصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسؤولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذی‌صلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسؤولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: فاطمه اکبری

امضا و تاریخ: 1399/2/13

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر تعیین می‌شود، بلامانع است:

☐ بهره‌برداری از این پایان‌نامه/ رساله برای همگان بلامانع است.

☐ بهره‌برداری از این پایان‌نامه/ رساله با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه/ رساله تا تاریخ ..... ممنوع است.

نام استاد یا اساتید راهنما: دکتر محمدرضا محمدی

تاریخ:

امضا:

چکیده

افزایش روزافزون شبکه‌های جاده‌ای و شهری معاصر در طول سه دهه‌ی گذشته نیازمند نظارت و مدیریت کارآمد ترافیک جاده‌ای است به همین دلیل تحلیل صحنه‌های ترافیکی و تشخیص خودروها از جهات بسیاری دارای اهمیت است و می‌توان برای آن کاربردهای مختلفی در سیستم‌های حمل و نقل هوشمند از جمله کنترل ترافیک و ایجاد سیستم‌های همکار راننده با هدف کاهش سوانح ترافیکی نام برد.

هدف ما بررسی یک سیستم نظارت ترافیکی است که مکان‌یابی و ردیابی خودروها را در بصورت بلادرنگ و همزمان انجام می‌دهد.

در این پایان‌نامه به پیاده‌سازی شبکه های شناسایی معروف پرداخته شده است. این شبکه ها تنها در مکان‌ها و اندازه‌های مشخص دنبال اجسام می‌گردند. این مکان‌ها و اندازه‌ها به گونه‌ی انتخاب می‌شوند تا بیش‌تر حالات ممکن را پوشش دهند. الگوریتم‌های این گروه معمولا عکس را به چند بخش با اندازه‌ی مشخص تقسیم می‌کنند. سپس در نظر می‌گیرند که در هر بخش، تعداد مشخصی اجسام با اشکال و اندازه‌های از پیش تعیین شده وجود دارد. لایه های ابتدایی این شبکه ها برای استخراج ویژگی به کار می رود و خروجی نهایی نیز شامل کلاس و مکان هر شی در تصویر است. این الگوریتم‌ها را روش تک مرحله ای نیز نامیده می‌شوند. آن‌ها به الگوریتم ردیابی داده می‌شود الگوریتم ردیابی سعی می‌کند با توجه به اختلاف مکانی بین خودروهای شناسایی شده در فریم فعلی و فریم قبلی به خودرو یک شناسه یکتا تا زمانیکه خودرو از محدوده دوربین خارج می‌شود اختصاص دهد.

ما سه شبکه شناسایی و یک الگوریتم ردیابی مورد را بررسی و میزان دقت آنها را ارائه کرده‌ایم دقت شناسایی تمامی آن‌ها بالای 80 درصد و دقت ردیابی نزدیک به 30 درصد همانطور که مشخص است الگوریتم ها از نظر دقت بر یکدیگر برتری ندارد و می توان از هریک از آن‌ها را به عنوان یک سیستم نظارت ترافیکی مورد استفاده قرار داد.

واژه‌های کلیدی: شناسایی و ردیابی اشیاء, نظارت هوشمند, کنترل ترافیک.

فهرست مطالب

فصل 1: مقدمه **1**

1-1- مقدمه 2

فصل 2: مروری بر منابع **3**

2-1- تعاریف، اصول و مبانی نظری..... 4

2-2- کارهای مرتبط..... 5

فصل 3: روش تحقیق **6**

3-1- جمع‌آوری داده 7

3-2- معرفی اجمالی شبکه‌های شناسایی اشیاء 7

3-3- شبکه‌ی YOLO 8

3-4- شبکه‌ی SSD..... 13

3-5- شبکه‌ی RetinaNet 15

3-6- الگوریتم ردیابی 17

فصل 4: نتایج و تفسیر آنها **20**

فصل 5: جمع‌بندی و پیشنهادات **25**

5-1- مقدمه 26

مراجع **27**

فهرست اشکال

شکل 3-1 نحوه‌ی قرارگیری انکر روی تصاویر 4
شکل 3-1 معماری شبکه‌ی R-CNN 8
شکل 3-2 معماری شبکه‌ی YOLO 9
شکل 3-3 featureهای پیش‌بینی شده با مقیاس‌های متفاوت که در هریک خانه قرمز مسئول شناسایی تصویر سگ می‌باشد 10
شکل 3-4 تبدیلات لگاریتمی بر روی کادر پیش‌بینی شده 11
شکل 3-5 معماری شبکه‌ی SSD 13
شکل 3-6 استخراج ویژگی در FPN 15
شکل 3-7 معماری شبکه‌ی RetinaNet 16
شکل 4-1 نمودار خطای شبکه‌ی YOLO 21
شکل 4-2 خروجی شبکه‌ی YOLO 21
شکل 4-3 نمودار خطای شبکه‌ی SSD 22
شکل 4-4 خروجی شبکه‌ی SSD 22
شکل 4-5 نمودار خطای شبکه‌ی RetinaNet 23
شکل 4-6 خروجی شبکه‌ی RetinaNet 23

فهرست جداول

جدول 4-1 مقایسه‌ی سایز دسته در زمان آموزش شبکه‌ها 23
جدول 4-2 مقایسه‌ی زمان شناسایی هر فریم 24
جدول 4-3 مقایسه‌ی نحوه عملکرد شبکه‌ها بر روی داده‌های آموزشی 24
جدول 4-4 مقایسه نحوه عملکرد شبکه‌ها بر روی داده‌های تست 24
جدول 4-5 مقایسه‌ی دقت ردیابی 24

فصل 1:

مقدمه

1-1- مقدمه

در این پایان‌نامه ما به پیاده سازی سیستمی برای شناسایی خودروها و ردیابی آن‌ها در جاده‌ها و بزرگراه‌ها پرداخته‌ایم. با تغییرات صورت گرفته در سبک زندگی و توسعه بیشتر شهرها و خطوط جاده‌ای و همچنین تعداد اتومبیل‌ها, نیاز به یک سیستم برای نظارت و ثبت تخلفات امری ضروری است.

ما به سیستمی نیاز داریم که بتواند بر تعداد بالای خودروها بصورت همزمان نظارت داشته باشد و بصورت بلادرنگ آنها را شناسایی و ردیابی کند.

فناوری‌های متعارف برای اندازه‌گیری ترافیک، مانند حلقه‌های القایی، سونار یا ردیاب‌های مایکروویو، اشکالات فراوانی مانند هزینه‌ی بالای نصب و نگهداری, قابل حمل نبودن و عدم تشخیص درست خودروهایی که ساکن یا سرعت آنها کم است به همراه دارد. در ازای آن سیستم‌های مبتنی بر فیلمبرداری نصب و نگهداری آسان و قابلیت حمل دارند. علاوه بر آن قابلیت ارتقا الگوریتم آن و اضافه کردن امکاناتی همچون کلاس‌بندی خودروها, شمارش و غیره را نیز دارد.

تعداد زیادی سیستم مبتنی بر ویدئو و پردازش تصویر وجود دارد که با روش‌های گوناگون سعی در شناسایی خودروها دارند.

سیستم بررسی شده در این پایان‌نامه بر اساس بینایی ماشین و هوش مصنوعی است. ما متداول‌ترین الگوریتم‌های شناسایی اشیاء همچون YOLO[1], SSD[2], RetinaNet را انتخاب کرده‌ایم. هر یک از الگوریتم‌های شناسایی اشیاء از دو بخش شناسایی موقعیت مکانی اشیاء و همچنین کلاس آن تشکیل شده است.

در ادامه برای ردیابی آن‌ها از الگوریتم SORT[3] استفاده کرده‌ایم. این الگوریتم با استفاده از فیلتر کالمن, وظیفه‌ی ردیابی را انجام می‌دهد. به دلیل محاسبات و استفاده کم از حافظه و سرعت بالا یکی از بهترین الگوریتم‌های ردیابی برای نیازهای همزمانی و بلادرنگی است. در ادامه ما به بررسی دقیق نحوه‌ی کار هر یک از الگوریتم‌های شناسایی که نام برده‌ایم می‌پردازیم و آن‌ها را با دیتاست UA-DETRAC آموزش می‌دهیم و نتایج آن را ارائه می‌کنیم.

[1] You Only Look Once

[2] Single Shot Detector

[3] Simple, Online and Realtime Tracking

فصل 2:

مروری بر منابع

2-1- تعاریف، اصول و مبانی نظری

در الگوریتم‌های شناسایی اصطلاحاتی هستند که بصورت گسترده مورد استفاده قرار می‌گیرند که موارد مورد نیاز را در اینجا بیان می‌کنیم.

کادر هدف: بخشی از تصویر که شامل شی است و خروجی مورد انتظار شبکه نیز تشخیص آن است.

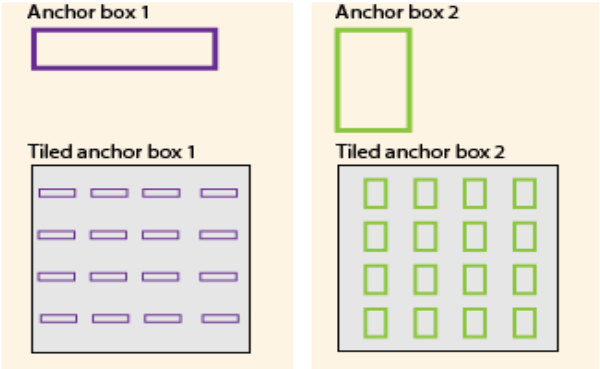
ژاکارد: حاصل تقسیم اشتراک دو محیط بر اجتماع آن که عددی بین صفر تا یک است.

سرکوب غیر حداکثر: راه‌حلی است برای اینکه هر شی تنها یک بار شناسایی شود اگر میزان ژاکارد دو کادر پیش‌بینی شده از یک مقدار آستانه بیشتر باشد تنها یکی از آن‌ها را در نظر می‌گیریم.

انکر: انکرها مجموعه‌ای از کادرهای از پیش تعریف شده با طول و عرض ثابت هستند. سایز آن‌ها با توجه به سایز اشیا موجود در تصاویر دیتاستی است که می‌خواهیم با آن شبکه را آموزش دهیم, مشخص می‌شود و سعی می‌شود به گونه‌ای تعریف شوند که اشیا عمودی و افقی را در بربگیرند.

استفاده از انکرها باعث ایجاد پیشرفت زیادی در سرعت و دقت شبکه‌های شناسایی اشیا شد. قبل از تعریف انکرها, الگوریتم‌های شناسایی از پنجره‌ی لغزنده استفاده می‌کردند. پنجره‌ی لغزنده تمام عکس را برای استخراج ویژگی‌ها پیمایش می‌کند که این عمل هزینه و سربار بالایی را برای الگوریتم‌ها ایجاد می‌کند علاوه بر آن قدرت تشخیص همزمان اشیا را نیز از ما سلب می‌کند و برای کارهای بلادرنگ نمی‌تواند گزینه مناسبی باشد.

نحوه کار انکر به این صورت است که هر انکر قسمت خاصی از تصویر را نشان می‌دهد البته بین آنها اشتراک نیز وجود دارد. شبکه در زمان آموزش میزان ژاکارد و انحراف هر انکر را نسب به کادر هدف محاسبه می‌کند و با توجه به خطا سعی به تصحیح



شکل 3-1 نحوه قرارگیری انکر روی تصاویر

پیش‌بینی‌ها می‌کند. استفاده از انکرها باعث می‌شود شبکه بتواند چندین شی با سایزهای مختلف, همچنین اشیاءای که همپوشانی دارند را پیدا کند.

در شبکه‌های شناسایی اشیا بدلیل استفاده از شبکه‌های کانولوشنال. خروجی شبکه به مکان منتاظر در تصویر نگاشت می‌شود مکان هر انکر با نگاشت کردن خروجی به تصویر اصلی مشخص می‌شود این عمل هر بار برای هر خروجی شبکه تکرار می‌شود و انکرهایی که احتمال وجود شی در آن‌ها از یک مقدار آستانه کمتر باشد به پس‌زمینه نگاشت می‌شوند. سپس با اعمال الگوریتم سرکوب غیرحداکثر بر روی باقی‌مانده‌ی آن‌ها, انکرهای نهایی انتخاب می‌شوند.

شبکه‌هایی که از انکر استفاده می‌کنند قدرت تشخیص اشیا بصورت همزمان را دارند و می‌توان از آن در کاربردهای بلادرنگ استفاده نمود. همچنین نیاز ما به یک پنجره لغرنده جهت پیمایش عکس را برطرف می‌کند و از هزینه‌های ناشی از استفاده پنجره لغزان اجتناب می‌شود.

باید توجه نمود برای شناسایی اشیا با سایزهای متفاوت باید انکرهایی با سایزهای متفاوت تعریف کنیم و این سایز وابسته به دیتاستی است که از آن استفاده می‌کنیم.

2-2- کارهای مرتبط

مطالعات و روش‌های گوناگون بسیاری در زمینه ردیابی خودرو صورت گرفته است از جمله آن‌ها می‌توان وضوح چندگانه,ردیابی لبه‌ها و تفریق‌پس‌زمینه نام برد.

وضوح چندگانه روشی است[1] که با توجه به اطلاعات پیکسل‌ها سعی در استخراج شی و ردیابی آن می‌کند یکی از معایب این روش در نظر گرفتن قسمتی از خطوط جاده‌ای یا سایه که در رنج رنگی خودرو قرار می‌گیرند به عنوان خودرو است. مشکل اساسی دیگر آن در مواجهه با چشم‌انداز است .

روش‌های مبتنی بر استخراج لبه, لبه‌های اشیاء را شناسایی و مورد ردیابی قرار می‌دهند[2] . مهم‌ترین مزیت آن این است که در مقیاس‌های گوناگون و تفاوت روشنایی نیز قادر به انجام اینکار است اما اگر ازدحام زیاد باشد شناسایی لبه‌ها برای استخراج اشکال وسلیه نقلیه سخت است.

تفریق پس‌زمینه موثرترین روشی است که استفاده می‌شود[3] و نسبت به روش‌های دیگر دقت بسیار مناسب‌تری را ارائه نموده است در این روش تصویری را به عنوان پس‌زمینه انتخاب می‌کنیم که ممکن است تصویری از جاده باشد که خالی از وسایل نقلیه است یا از دنباله‌ای از فریم‌های پی در پی استخراج شده است. سپس در هنگام تردد خودروها, تصویر ورودی را با پس زمینه مقایسه می‌کنند و از طریق بدست آوردن اختلاف آن‌ها, اشیا متحرکت را شناسایی می‌کنند.

فصل 3:

روش تحقیق

3-1- جمع‌آوری داده

داده‌های استفاده شده در این تحقیق از دیتاستUA-DTRAC می‌باشد [4] که شامل ده‌ها ساعت ویدئو در 24 موقعیت متفاوت در شهرهای چین است. سرعت ویدئو برای ضبط تصاویر 25 فریم بر ثانیه بوده است و رزولوشن تصاویر 960x540 است.

بیش از 140 هزار تصویر در این دیتاست موجود است و 8250 خودرو در آن برچسب‌گذاری شده است.

به دلیل زمان بر بودن آموزش شبکه با تمام داده‌های این دیتاست و با توجه به منابع محدود, ما تنها از قسمتی از داده‌های آموزشی استفاده نموده‌ایم و آن را به دو قسمت آموزش و تست تقسیم کرده و با آن شبکه را آموزش داده‌ایم.

3-2- معرفی اجمالی شبکه‌های شناسایی اشیاء

یکی از پایه‌ای‌ترین مباحث موجود در شبکه‌های عصبی و بینایی ماشین را میتوان شناسایی اشیاء نام برد که شامل دو بخش تشخیص مکان شی و تشخیص کلاس شی می‌باشد شبکه‌های عصبی کانوولوشنال با استخراج مرحله به مرحله‌ی ویژگی‌هایی مانند لبه, گوشه, اشکال گوناگون و ... توانایی شناسایی اشیا را بدست می‌آورند.

شبکه‌های عصبی گوناگونی برای شناسایی پیاده‌سازی شده است یکی از معروف‌ترین آن‌ها را می‌توان R-CNN[1] بیان کرد که در ادامه به توضیح مختصرآن می‌پردازیم [5].

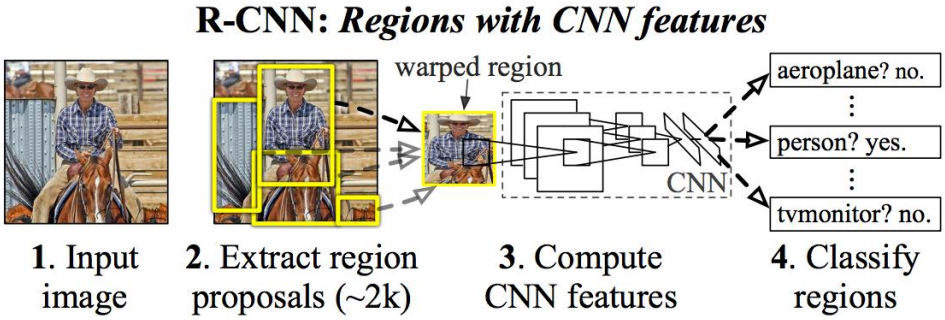
در سال 2012 آقای Krizhevs با معرفی شبکه‌ی کانوولوشنالی برنده مسابقات ImageNet شد این شبکه بصورت گسترده‌ای برای طبقه‌بندی اشیاء استفاده می‌شود و یک استاندارد در این زمینه است. بعد از آن تیمی کوچک در دانشگاه برکلی با رهبری پروفسور Jitendra Malik سؤالی تحت عنوان "چگونه میتوان نتایج Krizhevs را برای تشخیص اشیاء تعمیم داد؟" بیان کردندکه نتیجه پاسخ به سوال فوق شبکه R-CNN بود.

هدف R-CNN گرفتن تصویری به عنوان ورودی و تشخیص اشیاء در آن است. R-CNN تعدادی ناحیه پیشنهادی با روش سرچ انتخابی تولید می‌کند و آن‌ها را به عنوان ورودی به شبکه AlexNet تحویل می‌دهد. در لایه‌ی آخر نیز ماشین بردار پشتیبان را به عنوان طبقه‌بند استفاده می‌کند.

همانطور که در شکل 3-1 نیز مشاهده می‌شود R-CNN شامل دو مرحله است.

1. در اولین گام به کمک الگوریتم‌هایی مانند سرچ انتخابی مکان‌های کاندید برای وجود اشیاء انتخاب می‌شود.
2. این کاندیدها به شبکه‌ی کانوولوشنالی برای طبقه‌بندی پاس داده می‌شوند تا کلاس شی مورد نظر پیش‌بینی شود.

[1] Regional Convolutional Neural Network(CNN)



شکل 3-1 معماری شبکه‌ی R-CNN

مهم‌ترین مشکل چنین شبکه‌هایی سرعت پایین آن در کاربردهای همزمانی است. در نسخه‌ی بعدی این شبکه که Fast R-CNN است پیشرفت خوبی در زمینه دقت و سرعت انجام شد [**6**]. ولی همچنان مشکل دو مرحله‌ی بودن وجود داشت. در نسخه‌ی بعدی این شبکه که Faster R-CNN است تغییرات بیشتری رخ داد و الگوریتم سرچ انتخابی حذف و به جای آن از شبکه‌های پیشنهاددهنده استفاده شد.

بعد از آن شبکه‌هایی مانند YOLO و SSD معرفی شدند که تشخیص مکان و کلاس شی را بصورت همزمان انجام می‌دهند این الگوریتم‌ها به دلیل سرعت بالا تحول بزرگی در زمینه شناسایی اشیاء ایجاد کردند بطوریکه که YOLO می‌تواند با سرعت 45 فریم بر ثانیه در GPU کار کند. در ادامه به بررسی ساختار و پیاده‌سازی سه شبکه معروف YOLO, SSD و RetinaNet می‌پردازیم.

3-3- شبکه‌ی YOLO

YOLO الگوریتم شناسایی اشیاء یک مرحله‌ای یا بلادرنگ است [**7**]. که در سال 2015 توسط آقای Readmon ارائه شد. سرعت آن 45 فریم بر ثانیه بر روی GPU است و در نسخه‌های کوچک شده‌ی آن به سرعت بالای 100 فریم بر ثانیه نیز دست یافته است. اگر چه دقت آن نسبت به R-CNN کمتر است اما تفاوت سرعت آن‌ها باعث محبوبیت بیشتر YOLO شده است.

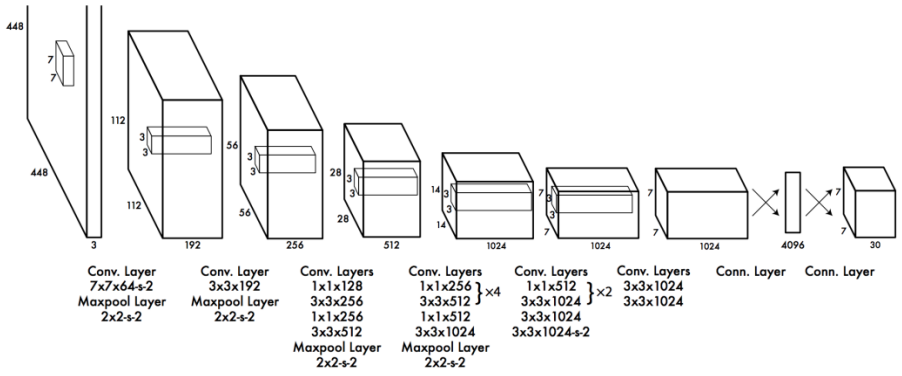
YOLO سه نسخه دارد که در هر نسخه‌ی جدیدتر سعی شده مشکلات نسخه‌ی قبلی مرتفع گردد نسخه اول آن دارای محدودیت در شناسایی اشیاء کوچک بود و همچنین وابسته با سایز شی در زمان آموزش بود بطور مثال اگر شبکه را با شی‌ای آموزش می‌دادند سپس همان شی را با سایز متفاوت با شی آموزش داده شده به شبکه ارائه می‌کردند شبکه در شناسایی آن مشکل داشت.

نسخه دوم که با نام YOLO9000 در سال 2016 ارایه شد [**8**] نسبت به نسخه‌ی قبلی سرعت و دقت بیشتری داشت تغییرات این نسخه شامل استفاده از نرمال‌سازی دسته, افزایش وضوح تصویر با تغییر سایز تصویر ورودی از 224 به 448 استفاده از انکرها, شناسایی اشیاء کوچکتر و غیره است.

سپس روی نسخه دوم قدم به قدم تغییراتی مانند شناسایی اشیاء در مقیاس‌های متفاوت, تغییر تابع‌های هزینه, استفاده از شبکه Darknet53 به جای Darknet23 و غیره رخ داد که این تغییرات را تحت عنوان YOLOV3 یا همان نسخه سوم معرفی می‌شود [9].

در ادامه به معرفی به معماری نسخه سوم این شبکه می‌پردازیم و هر کجا از YOLO استفاده کرده‌ایم, YOLOV3 مد نظر است.

معماری YOLO:



شکل 3-2 معماری شبکه‌ی YOLO

YOLO از شبکه‌ی Darknet به عنوان شبکه‌ی پایه استفاده می‌کند Darknet یک شبکه عصبی قوی و سریع برای شناسایی است. YOLO شامل یک شبکه عصبی با 24 لایه کانوولوشنالی برای استخراج ویژگی و همچنین 2 لایه کاملاً متصل برای پیش‌بینی کادرها است. معماری شبکه کانولوشنالی YOLO را در شکل 3-2 مشاهده می‌کنید.

همچنین یک نسخه سریع از YOLO نیز طراحی شده است. YOLOی سریع، یک شبکه عصبی با تعداد لایه‌های کانولوشنالی کمتر است که در آن از 9 لایه کانولوشنالی بجای 24 لایه کانولوشنالی استفاده شده و البته تعداد فیلترهای هر لایه در YOLO سریع نسبت به YOLO اصلی کمتر است.

برای ساخت و مشاهده کامل لایه‌ها YOLO میتوان از فایل **yolo3.cfg** که توسط نویسندگان این شبکه ارائه شده است استفاده نمود. این فایل شامل 5 بلاک متفاوت است که هر یک از آن‌ها لایه‌های شبکه را توصیف می‌کند. بلاک‌ها به عبارت زیر هستند

- Convolutional: توصیف‌کننده‌ی لایه‌ی کانوولوشنال است
- Upsample: سایز نمونه‌گیری از لایه‌ی قبل را مشخص می‌کند باعث کوچکتر شدن سایز feature map می‌شود.
- Shortcut: یک لایه‌ی میانی است و مشخص می‌کند لایه‌ای فعلی از اضافه کردن دو لایه‌ی مشخص به یکدیگر بدست می‌آید.
- Route: این بلاک مشخص می‌کند لایه‌ای فعلی از یک لایه‌ی مشخص یا از پیوست دو لایه‌ی مشخص در بعد عمق بدست می‌آید.
- YOLO: این بلاک مربوط به لایه‌ی شناسایی است که پیش‌بینی انکرها و همچنین محاسبه خطا در آن انجام می‌گیرد.و تعیین کننده سایز انکرها می‌باشد.

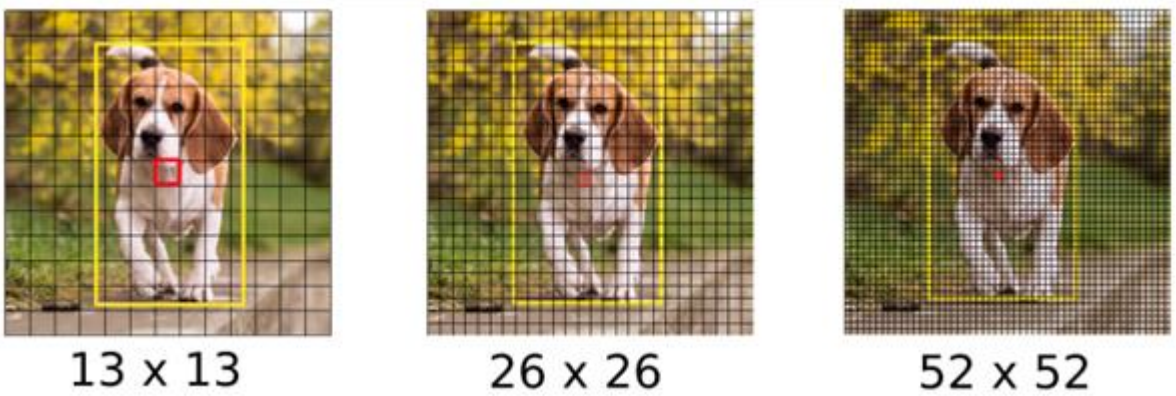
الگوریتم YOLO

شبکه‌ی YOLO تصاویر را به صورت فرضی به یک صفحه‌یNxN تقسیم می‌کند که اصطلاحاً به آن گرید نیز گفته می‌شود در YOLO سه گرید با سایزهای 56x56, 26x26 و 13x13 وجود دارد که قابلیت شناسایی اشیاء با سایزهای متفاوت را داشته باشد. هر خانه در گرید تنها مسئول شناسایی یک شی است. در ازای هر خانه سه انکر با سایزهای متفاوت در نظر گرفته می‌شود تا شبکه احتمال وجود شی و کلاس شی و همچنین مختصات آن را پیش‌بینی کند پس به ازای هر یک از این انکرها یک آرایه (C+1+4) پیش‌بینی می‌کند که ۴ نشان‌دهنده مختصات شی شناسایی شده‌است و یک مقدار نیز احتمال مشاهده شدن یک شیء در آن خانه را نشان می‌دهد و بردار C نیز احتمال تعلق شیء شناسایی شده به هر کدام از کلاس‌های از پیش تعریف شده را در اختیار قرار می‌دهد.

YOLO مستقیماً این آرایه را مورد استفاده قرار نمی‌دهد بلکه نیازمند اعمال تغییراتی برای بدست آوردن خروجی نهایی است.

درادامه تبدیلات مورد نیاز را بررسی می‌کنیم

- اعمال تابع سیگموئید[1] بر روی مقدار احتمال وجود شی که آن را به عددی بین 0 تا 1 تبدیل می‌کند کادرهایی که مقدار احتمال آن‌ها از مقدار آستانه که معمولاً 0.5 در نظر گرفته می‌شود بیشتر شود به عنوان کاندیدهایی که شامل شی هستند انتخاب می‌شوند.
- اعمال تابع سیگموئید بر روی بردار C نیز آن را به عددی بین 0 تا 1 تبدیل می‌کند. در نسخه‌های اول و دوم YOLO از تابع بیشینه هموار[2] استفاده می‌شد اما استفاده از این تابع ما را برای انتخاب کلاس شی محدود به یک کلاس می‌کرد بطور مثال حالتی را در نظر بگیرید که شی مورد نظر یک مرد می‌باشد این شی میتواند عضو کلاس انسان نیز باشد.



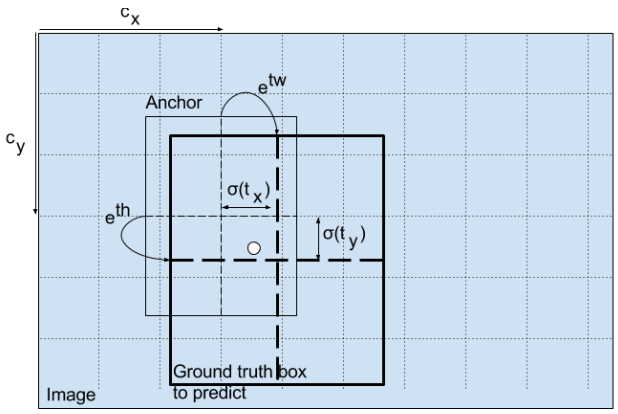
شکل 3-3 featuremapهای پیش‌بینی شده با مقیاس‌های متفاوت که در هریک خانه‌ی قرمز مسئول شناسایی تصویر سگ می‌باشد

- تبدیلات مورد نیاز برای کادر به صورت زیر است:

[1] sigmoid
[2] softmax

$$\hat{x} = S(t_x) + c_x$$
$$\hat{y} = S(t_y) + c_y$$
$$\hat{w} = p_w e^{t^w}$$
$$\hat{h} = p_w e^{t^h}$$

t_x, t_y, t_w, t_h مختصات مرکز و طول و عرض کادرهای پیش‌بینی شده را مشخص می‌کنند. YOLO مرکز هر شی را نسبت به همان خانه‌ای که مرکز کادر در آن قرار دارد پیش‌بینی می‌کند .برای اینکه در تصویر بتوانیم مکان شی را مشخص کنیم نیاز است تا این اعداد را با مختصات خانه‌ای که در آن قرار دارند جمع کنیم بطور مثال اگر تصویر سمت چپ شکل 3-3 را در نظر بگیریم مرکز تصویری که در خانه‌ی قرمز پیش‌بینی شده است بصورت (0.4,0.7) باشد باید آن را به مختصات (7,7) که مختصات سمت چپ و بالای خانه‌ی قرمز را مشخص می‌کند جمع کنیم پس خروجی در نهایت به شکل (7.4,7.7) نمایش داده می‌شود. سپس تبدیلات لگاریتمی بر روی طول و عرض کادر پیش‌بینی شده جهت تطبیق مناسب‌تر با کادر هدف صورت می‌گیرد.



شکل 3-4 تبدیلات لگاریتمی بر روی کادر پیش‌بینی شده

اکنون دو فیلتر بر روی کادرهای پیش‌بینی شده اعمال می‌کنیم

- کادرهایی که احتمال وجود شی در آن‌ها از مقدار آستانه که معمولاً 0.5 در نظر گرفته می‌شود کمتر باشد حذف می‌شوند
- احتمال شناسایی یک شی توسط چندین کادر وجود دارد به همین دلیل از روش سرکوب غیر حداکثر استفاده می‌کنیم و کادرهایی که ژاکارد آن‌ها از یک مقدار آستانه بیشتر باشد یکی را به عنوان کاندید و بقیه کادرها را در نظر نمی‌گیریم.

محاسبه‌ی خطا و تابع هزینه

تابع هزینه برای شبکه‌ی YOLO اشیا از سه مؤلفه تشکیل شده‌است:

- زیان محلی‌سازی: این مؤلفه، در صورتی که کادر پیش‌بینی شده, مسئول تشخیص شیء در تصویر باشد، مربع خطای مرتبط با اندازه و مکان آن رانسبت به کادر هدف محاسبه می‌کند.

$$\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2]$$
$$+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\widehat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$

λ_{coord} تاثیر زیان محلی سازی را افزایش می‌دهد.

1_{ij}^{obj} زمانی مقدار یک می‌گیرد که انکر jام خانه iام مسئول شناسایی شی باشد

$(\hat{x}_i,\hat{y}_i,\widehat{w}_i,\hat{h}_i)$ مختصات jامین کادر پیش‌بینی شده خانه‌ی iام است

$$x_i=g_x\ ,\ x_i=g_y$$
$$h_i=\log\frac{g_w}{a_w},$$
$$w_i=\log\frac{g_h}{a_h},$$

a_w و a_h ابعاد انکر jام

معادله 3-1 تابع زیان محلی سازی شبکه‌ی YOLO

- زیان دسته‌بندی: در صورتی که یک شیء در تصویر تشخیص داده شده باشد، این مؤلفه، مربع خطای احتمال شرطی کلاس را محاسبه می‌کند. بنابراین تنها در صورتی که یک شیء در یک خانه از گرید وجود داشته باشد، برای خطای انجام شده در دسته‌بندی، جریمه در نظر گرفته می‌شود. با توجه به اینکه شبکه‌ها برای آموزش هم به داده‌های مثبت و هم داده‌ی منفی نیاز دارند و باید تعادلی در تعداد آن‌ها برقرار باشد YOLO ضریبی را برای محاسبه کادرهایی که شامل تصویر پس زمینه هستند قرار می‌دهد تا تاثیر آن کمتر کند.

$$\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{obj}[(c_i-\hat{c}_i)^2]+\ \lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B}1_{ij}^{noobj}[(c_i-\hat{c}_i)^2]$$

$\widehat{c_i}$ میزان احتمال کلاس شی در کادر پیش بینی شده jام خانه i

معادله 3-2 تابع زیان کلاس بندی شبکه‌ی YOLO

- زیان ضریب اطمینان: این مؤلفه، مربع خطای ضریب اطمینان کادر پیش‌بینی شده را محاسبه می‌کند.

$$\sum_{i=0}^{s^2}1_{ij}^{obj}\sum_{c\in classes}(p_i(c)-\ \hat{p}_i(c))^2$$

معادله 3-3 تابع زیان اطمینان شبکه‌ی YOLO

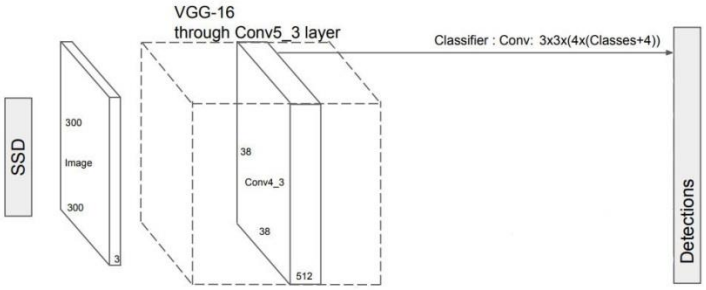
خطای شبکه از جمع کردن خطاهای فوق بدست می‌آید.

در بخش فوق نحوه کارکرد الگوریتم YOLO را توضیح دادیم **[10]**. در ادامه به بررسی دو الگوریتم دیگر می‌پردازیم.

3-4- شبکه‌ی SSD

SSD در سال 2016 توسط آقای C. Szegedy و دوستانش قبل از نسخه‌ی سوم YOLO ارائه شد[**11**] . سرعت این شبکه 59 فریم بر ثانیه می‌باشد. SSD را می توان اینگونه معنا کرد, یک الگوریتم شناسایی که شناسایی مکان و کلاس هر شی را در یک حرکت پیشرو و در یک گام انجام می‌دهد و توانایی شناسایی چندین شی را بصورت همزمان دارد.
SSD شباهت بسیاری در نحوه شناسایی اشیا به YOLO دارد. در ادامه به معرفی معماری این شبکه می‌پردازیم و با توجه به توضیحاتی که در قسمت YOLO داده شد بخش‌های مشابه را بطور مختصر بیان می‌کنیم.

معماری SSD



شکل 3-5 معماری شبکه‌ی SSD

پایه‌ی معماری SSD شبکه‌ی VGG16 می‌باشد که SSD از آن برای استخراج ویژگی از عکس و سپس از فیلتر کانوولوشنالی برای شناسایی اشیا در مقایس‌های متفاوت استفاده می‌کند.

الگوریتم SSD

SSD نیز مانند YOLO تصاویر را به نواحی NxN تقسیم می‌کند. و در ازای هر خانه‌ی آن مجموعه‌ای از انکرها را در نظر می‌گیرد. SSD از6 feature map با سایزهای 38x38, 19x19, 10x10, 5x5, 3x3 و 1x1 برای پیش‌بینی مختصات کادرها و کلاس شی احتمالی موجود در هر کادر استفاده می‌کند با توجه به اینکه هر کادر برای مشخص شدن به ۴ متغیر نیاز دارد. برای پیش‌بینی کلاس شی نیز به تعداد کلاس‌ها که در اینجا, شی ما تنها خودرو می باشد پس ۱ متغیر نیاز دارد. همانطور که مشخص است SSD پارامتری برای تعیین احتمال وجود یک شی در نظر نگرفته است در ازای آن کلاسی تحت عنوان پس زمینه به کلاس‌ها اضافه می‌کندکه وظیفه آن شناسایی مکان‌های از تصویر است که هیچ شی‌ای وجود ندارد پس اگر مقدار کلاس پس‌زمینه برای کادری زیاد باشد آن کادر شامل شی نیست.
نحوه تولید انکر در SSD کمی متفاوت با YOLO می‌باشد. تعداد انکرها به ازای هر خانه در feature mapها متفاوت و به ترتیب 4, 6, 6, 6, 4, 4 می‌باشد. برای تعیین ابعاد انکرها برای هر feature map دو پارامتر min و max و آرایه‌ای تحت

عنوان aspect ratios که شامل یک یا دو مقدار است در نظر گرفته می‌شود و انکرهایی با ابعاد زیر برای هر feature map در نظر گرفته می‌شود.

- (min,min)
- $(\sqrt{\min * max}),\sqrt{\min * max})$
- $(\min * \sqrt{ar}, \max / \sqrt{ar})$, $(\min / \sqrt{ar}, \max * \sqrt{ar})$, ar ∈ aspect ratios

تعداد کل انکرهایی که SSD تعریف می‌کند برابر با 8732 می‌باشد که به سادگی با محاسبه‌ی زیر مشخص می شود.

38*38*4 + 19*19*6 + 10*10*6 + 5*5*6 + 3*3*4 + 1*1*4 = 8732

همانطور که گفته شد خروجی SSD مجموعه‌ای از کادرها و احتمالات مربوط به کلاس‌ها می‌باشد.

محاسبه‌ی خطا و تابع هزینه:

- زیان محلی‌سازی: خطای مربوط به تفاوت مکانی کادر هدف و کادرهای پیش بینی شده که تنها برای کادرهایی که درست یک شی را تشخیص داده‌اند در نظر گرفته می‌شود و با تابع زیان هابر[1] محاسبه می‌گردد.

$$\sum_{i,j} \sum_{m\in\{x,y,w,h\}} 1_{ij}^{match} L_1^{smooth}\left(d_i^m - \hat{g}_j^m\right)^2$$

$$\hat{g}_j^{cx} = \left(g_j^{cx} - p_i^{cx}\right)/a_i^w$$

$$\hat{g}_j^{cy} = \left(g_j^{cy} - p_i^{cy}\right)/a_i^h$$

$$\hat{g}_j^{w} = \log\frac{g_j^w}{a_i^w}$$

$$\hat{g}_j^{wh} = \log\frac{g_j^h}{a_i^h}$$

1_{ij}^{match} مشخص می‌کند i امین انکر به مختصات $(a_i^{cx}, a_i^{cy}, a_i^w, a_i^h)$ با کادر هدف jام به مختصات $(g_i^{cx}, g_i^{cy}, g_i^w, g_i^h)$ تطبیق دارد

d_i^m مختصات کادری که پیش بینی شده است.

معادله 3-4 تابع زیان محلی سازی شبکه SSD

- زیان دسته‌بندی: زیان دسته‌بندی برای کلاس‌ها است و از تابع بیشینه هموار برای محاسبه آن استفاده می‌شود. SSD برای تولید داده برای کلاس پس‌زمینه از روش استخراج داده‌های منفی استفاده می‌کند نحوه کار آن به این صورت است که نواحی از تصویر را که شامل شی نیست را انتخاب کرده و کلاس آن را نیز پس‌زمینه انتخاب می‌کند.

$$\sum_{i\in pos} 1_{ij}^k \log \hat{c}_i^k - \sum_{i\in neg} \log \hat{c}_i^0$$

$$\hat{c}_i^k = \sigma(c_i^k)$$

k کلاس مورد نظر است

معادله 3-5 تابع زیان کلاس بندی شبکه SSD

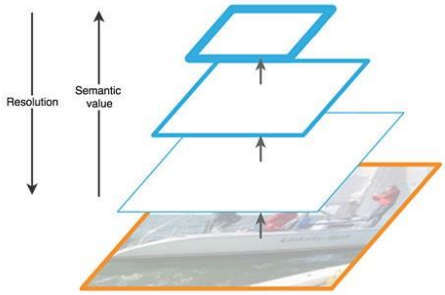
[1] Smooth L1

زیان نهایی از جمع دو زیان بالا محاسبه می‌شود.

3-5- شبکه‌ی RetinaNet

این شبکه در سال 2018 توسط آقای Lin معرفی شد[**12**] . RetinaNet از دو بخش FPN[1] و Focal Loss تشکیل شده است. برای استخراج ویژگی از شبکه FPN استفاده می‌کند که بر پایه‌ی شبکه‌ی ResNet می‌باشد.

همانطور که از معنای FPN نیز مشخص است شامل هرمی از feature mapها است. در شکل 3-6 استخراج ویژگی در FPN مشاهده می‌شود لایه‌های پایین‌تر هرم دارای وضوح بالاتری هست اما ساختارهای معنی‌دار کمتری در آن یافت می‌شود در این لایه‌ها شبکه توانایی شناسایی اشیاء کوچکتر را دارد. به سمت لایه‌های بالاتر که حرکت می‌کند وضوح کاهش می‌یابد اما ساختارهای بیشتر و پیچیده‌تری را شبکه می‌تواند تشخیص بدهد.



شکل 3-6 استخراج ویژگی در FPN

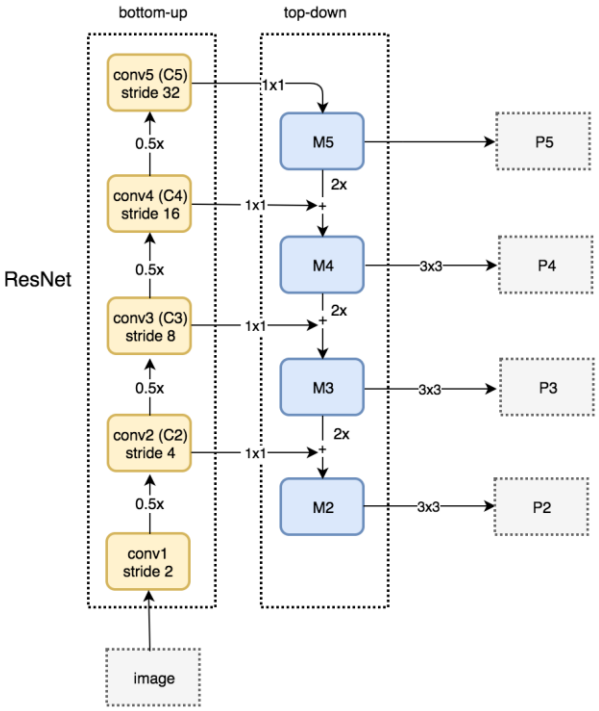
RetinaNet دارای دو حرکت پایین به بالا و بالا به پایین است که در ادامه هدف هریک از آنها را ذکر می‌کنیم

در حرکت رو به بالا از شبکه‌ی کانوولوشنالی برای استخراج ویژگی استفاده شده است و ساختارهای پیچیده‌تر نمایان می‌شود همانطور که در شکل 3-7 مشخص است در حرکت رو به بالا از شبکه ResNet برای ساخت هرم استفاده شده است و تعدادی واحد داریم که دارای تعداد زیادی لایه‌ی کانوولوشنال هستند و در هر یک سایز feature map با گام‌هایی با سایز 0.5 کاهش می یابد به خروجی هر یک از این واحدها C_i گفته می‌شود و هدف آن استخراج ویژگی است و در حرکت رو به پایین مورد استفاده قرار می‌گیرند.

در حرکت رو به پایین یک فیلتر 1x1 روی آخرین لایه‌ی FPN که C_5 است اعمال می‌شود و عمق آن را به نصف کاهش می‌یابد این اولین feature mapای است که برای پیش‌بینی استفاده می‌شود و نام آن نیز M_5 است و به همین ترتیب برای ساخت M_4 نیز یک فیلتر 1x1 روی M_5 اعمال می‌کنیم و عمل بیتی[2] را روی خروجی آن و لایه‌ی FPN متناظر آن اعمال می‌کنیم هدف از این عمل افزایش دقت پیش بینی مکان اشیا است زیرا در حرکت‌های رو به بالا و پایین مکان اشیا دیگر دقیق نیست. به

[1] Feature Pyramid Network
[2] bitwise

همین ترتیب ادامه می‌دهیم تا به پایین‌ترین لایه‌ی FPN برسیم باید توجه داشت ما تا P_2 ادامه می‌دهیم زیرا سایز C_1 برزگ است و سرعت را به شدت کاهش می‌دهد.



شکل 3-7 معماری شبکه‌ی RetinaNet

همانطور که گفتیم FPN به ذات یک شناسایی‌کننده‌ی شی نیست بلکه یک استخراج‌کننده‌ی ویژگی است اکنون برای اینکه اشیا را شناسایی کنیم از یک شبکه پیشنهاددهنده ناحیه یا RPN[1] استفاده می‌کنیم. در شبکه RetinaNet نیز ما از انکرها استفاده می‌کنیم مانند SSDیک کادر با توجه به سایز feature map و آرایه‌ای تحت عنوان aspect ratios داریم در ازای هر خانه feature map ما تعداد 9 انکر در نظر می‌گیریم, RetinaNet نزدیک به 100 هزار انکر تولید می‌کند.

سپیس RPN یک فیلتر 3x3 بر روی feature map اعمال می‌کند و در ادامه دو فیلتر 1x1که یک از آن‌ها برای استخراج کلاس و فیلتر دوم برای استخراج ابعاد شی است مورد استفاده قرار می‌گیرد.

محاسبه‌ی زیان

RetinaNet از یک تابع زیان جدید به نام Focal loss استفاده می‌کندهمانطور که گفته شد باید تعادلی بین دو نوع داده مثبت و منفی نیاز برای آموزش شبکه دو نوع داده مثبت و منفی نیاز برای آموزش شبکه باشد.

در YOLO هنگام محاسبه خطا از یک پارامتر برای کمتر کردن تاثیر داده نادرست در تابع زیان استفاده می‌شود. SSD نیز از روش استخراج داده منفی برای تولید داده پس‌زمینه یا منفی استفاده می‌کند.

RetinaNet نیز از تکنیک SSD استفاده می‌کند اما اگر توازن در تعداد داده‌های مثبت و منفی وجود نداشته باشد توابعی مانند آنتروپی متقاطع[2] بدرستی نمی‌توانند عمل کنند به همین دلیل RetinaNet از یک تابع زیاد جدید به نام Focal استفاده

[1] Region Proposal Network
[2] Crosse entropy

می‌کندکه از اضافه کردن یک ضریب به تابع آنتروپی متقاطع بدست می‌آید برای اینکه نسبت خطا را در داده‌های اشتباه نسبت به درست کاهش دهیم.

تشخیص داده‌های منفی راحت‌تر از مثبت است و با توجه به اینکه تعداد آن بیشتر است در توابعی مانند آنتروپی متقاطع بر مقدار گرادیان تاثیر بیشتری می‌گذارد در حالیکه که تفاوتی بین داده‌های مثبت و منفی نیست به همین دلیل از ضریب زیر برای کاهش وزن آن استفاده می‌کنیم

$$(1 - p_t)^{\gamma}$$

p_t= 1 if y=1 otherwise $p_t = 1 - p$

p مقدار احتمال وجود شی

معادله 3-6 تابع زیان Focal

اکنون سه الگوریتم شناسایی اشیا را بررسی کرده‌ایم در ادامه می‌خواهیم روش ردیابی را بررسی کنیم خروجی الگوریتم های فوق مجموعه ای از کادرها می‌باشد که احتمال وجود خودرو در آن بالا است سپس این خروجی به الگوریتم ردیابی داده می‌شود.

3-6- الگوریتم ردیابی

الگوریتمی که برای ردیابی استفاده شده است SORT است[**13**] . این الگوریتم از کالمن فیلتر برای تعیین مکان احتمالی خودرو استفاده می‌کند. هر جایی که اطلاعات نامعینی درباره‌ی یک سیستم دینامیکی داشته باشیم، می‌توانیم با استفاده از فیلتر کالمن تخمین مناسبی از تغییرات سیستم در آینده ارائه کنیم. فیلترهای کالمن برای سیستم‌هایی که مدام در حال تغییرند، ایده‌آل هستند. مزیت فیلترهای کالمن این است که به حافظه کمی نیاز دارند، زیرا به حافظه‌ای جز برای نگهداری اطلاعات وضعیت‌های قبلی نیاز ندارند. همچنین این فیلترها بسیار سریع هستند و برای مسائل زمان حقیقی و سیستم‌های تعبیه‌ای مناسب هستند.

در فیلتر کالمن برای هر متغیر یک توزیع گوسی در نظر گرفته می شود هر متغیر یک مقدار میانگین دارد که مرکز توزیع است و محتمل ترین حالتی است که رخ میدهد و یک مقدار واریانس که معرف نامعینی است.

یک ماتریس کواریانس یا همبستگی نیز داریم که درایه ij آن میزان همبستگی متغیر حالت i ام و متغیر حالت j ام را نشان می دهد البته این برای سیستمی است که دو متغیر داشته باشد سیستم ما می تواند به تعداد دلخواه ای وابسته به نیاز ما متغیر داشته باشد فیلتر کالمن از دو بخش پیش بینی و به روزرسانی تشکیل شده است.

- پیش بینی: شامل تخمین حالت فعلی سیستم بر اساس حالت قبلی آن است که در مسئله ما پیش‌بینی مکان فعلی ماشین بر اساس مکان قبلی آن است.
- به روز رسانی: شامل تصحیح پیش‌بینی‌ها و به روزرسانی ماتریس همبستگی بر اساس خطا است.

اکنون برای مسئله ردیابی خودرو متغیر میانگین را بصورت زیر تعریف می‌کنم که شامل مرکز و طول و عرض و همچنین سرعت جا به جای کادر در فریم فعلی نسبت به همان کادر در فریم قبلی است.

$$X = [c_x, c_y, w, h, v_x, v_y, v_w, v_h]$$

ماتریس همبستگی که آن را P می نامیم نیز با مقادیری دلخواه مقداردهی می‌کنیم مقادیر بزرگ‌تر نشان‌دهنده عدم قطعیت بیشتر هستند.

روند کار را به صورت زیر خلاصه می‌کنیم

1. در زمان صفر شبکه سه کادر را پیش‌بینی کرده است کالمن برای هر سه کادر ماتریس‌های را مقداردهی می‌کند.
2. در زمان یک شبکه سه کادر دیگر پیش‌بینی می‌کند سپس با معیار ژاکارد آن‌ها را به کادرهای قبلی که توسط شبکه پیش‌بینی شده است تطبیق می‌دهیم
3. کالمن نیز مکان کادرها را در زمان ۱ با معادله زیر پیش‌بینی می‌کند که ماتریس F فاصله زمانی بین دو فریم فعلی و قبلی است

$$\acute{x} = Fx$$
$$\grave{P} = FPF^T$$

4. اکنون زمان به روزرسانی است ابتدا خطا را که شامل تفاوت بین کادرهای پیش‌بینی شده توسط کالمن و شبکه در زمان یک است را محاسبه می‌کنیم

$$y = z - H\acute{x}$$

سپس پارامتری تحت عنوان Kalman Gain که آن را K می‌نامیم محاسبه می‌کنیم که نشان‌دهنده‌ی اهمیت خطا است و با آن ماتریس همبستگی را به روزرسانی می‌کنیم

$$S = H\acute{P}H^T$$
$$K = \acute{P}H^T S^{-1}$$

ماتریس H یک ماتریس برای ساده تر کردن عملیات های ضرب ماتریسی است.

اکنون عملیات به روزرسانی را انجام می‌دهیم

$$x = \acute{x} + Ky$$
$$P = (I - KH)\acute{P}$$

ما از الگوریتم فوق برای ردیابی خودروها استفاده می‌کنیم. این الگوریتم به دلیل سرعت خوب و حافظه ی کمی که دارد بسیار مناسب کارهای بلادرنگ است.

برای اندازه گیری دقت از معیار MOTA استفاده شده است این دقت اشسا ردیابی شده را به سه قسمت همواره ردیابی شده, قسمتی ردیابی شده و همواره از دست رفته تقسیم می کند. یک شی همواره ردیابی شده در نظر گرفته می شود اگر دز طئل دیده شدن در فریم ها 80 درصد درست ردیابی شده باشد همچنین یک شی همواره از دست رقته در نظر گرفته می شود اگر در سپس با معادله زیر دقت را می سنجیم:

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t}$$

FN_t تعداد کادرهایی که به اشتباه ردیابی شده اند

FP_t تعداد کادرهای که شامل خودرو بوده اند اما ردیابی نشده‌اند

IDS_t تعداد شناسه‌هایی که در زمان t تغییر کرده اند

GT_t کل تعداد کادرهای درست

معادله 3-7 نحوه محاسبه معیار MOTA

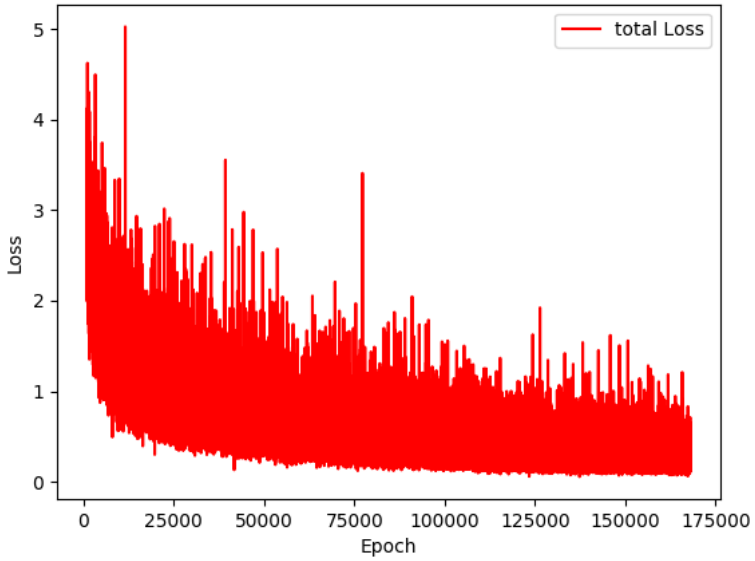
فصل 4:

نتایج و تفسیر آنها

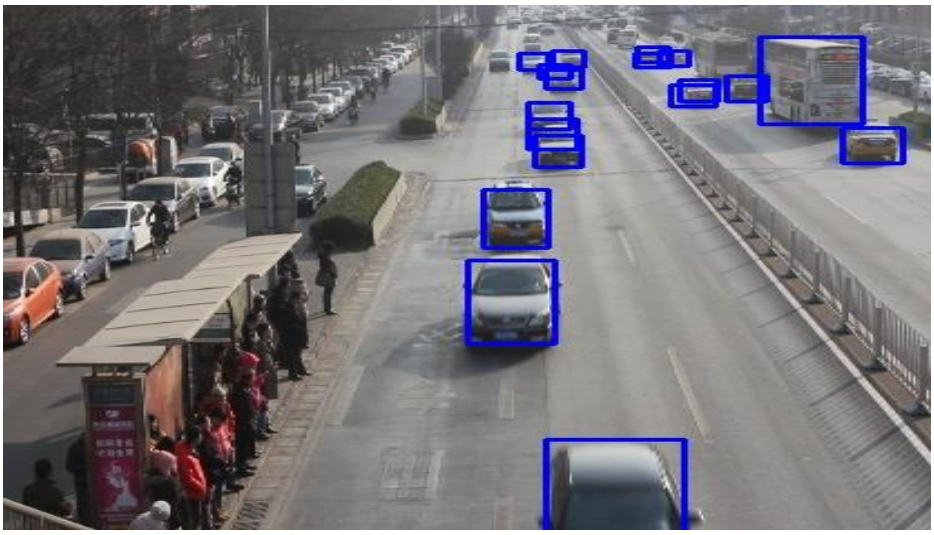
همانطور که گفته شد ما تنها قسمتی از دیتاست UA-TRAC که حدود 16000 تصویر است را مورد استفاده قرار داده‌ایم که 0.75 آن برای آموزش و 0.25 درصد آن برای تست شبکه استفاده شده است. برای پیاده‌سازی شبکه‌ها از زبان پایتون و فریم‌ورک پایتورچ استفاده شده است. در ادامه نمودارهای مربوط به زیان و دقت هر یک از شبکه‌ها ارائه شده است.

با توجه به اینکه شبکه‌ها روی Colab اجرا شده‌اند و محدودیت استفاده GPU و حافظه موجود است شبکه‌ها در سایز دسته با یک دیگر متفاوت هستند. همچنین اضافه کردن داده ارزیابی نیز مدت زمان آموزش را چندین برابر می‌کند به همین دلیل از اضافه کردن آن خوداری شده است.

خروجی شبکه‌ی YOLO:

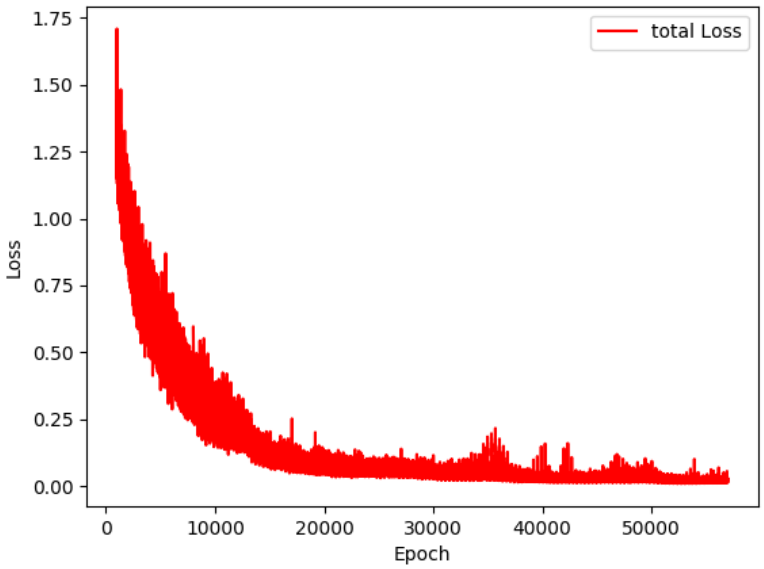


شکل 4-1 نمودار خطای شبکه‌ی YOLO

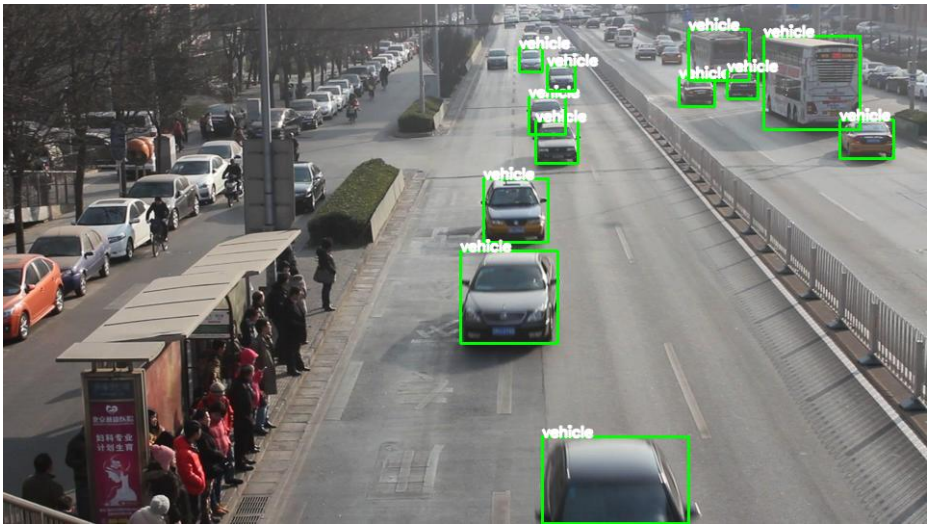


شکل 4-2 خروجی شبکه‌ی YOLO

خروجی شبکه‌ی SSD:

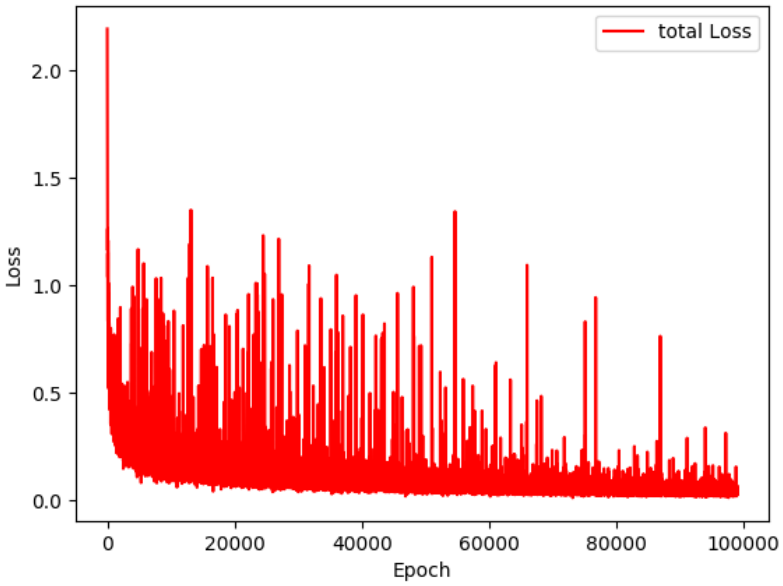


شکل 4-3 نمودار خطای شبکه‌ی SSD

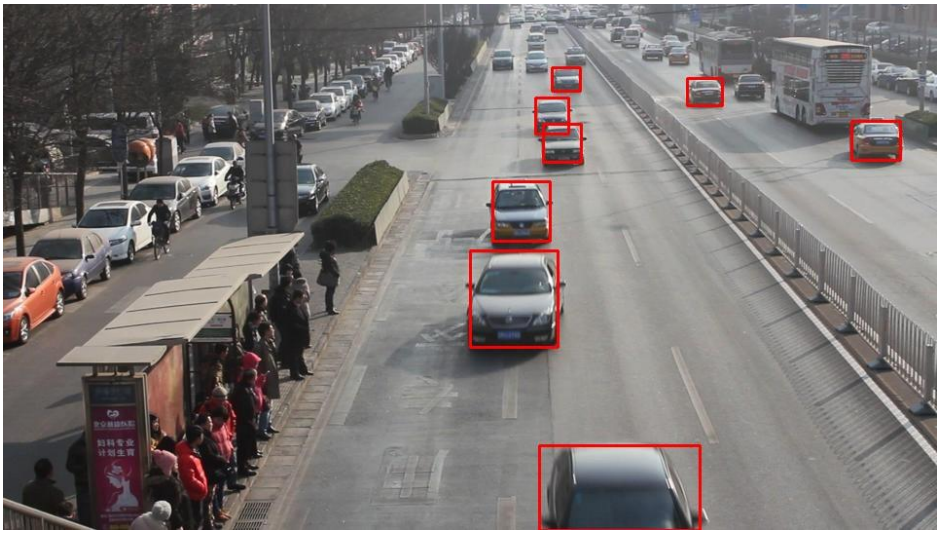


شکل 4-4 خروجی شبکه‌ی SSD

خروجی شبکه‌ی RetinaNet



شکل 4-5 نمودار خطای شبکه‌ی RetinaNet



شکل 4-6 خروجی شبکه‌ی RetinaNet

جدول 4-1 مقایسه‌ی سایز دسته در زمان آموزش شبکه‌ها

	YOLO	SSD	RetinaNet
سایز دسته	8	32	4

جدول 4-2 مقایسه‌ی زمان شناسایی هر فریم

	YOLO	SSD	RetinaNet
زمان به ثانیه	0.035	0.026	0.105

دقت شبکه‌ها در شناسایی

جدول 4-3 مقایسه‌ی نحوه عملکرد شبکه‌ها بر روی داده‌های آموزشی

	YOLO	SSD	RetinaNet
Recall	0.99	0.97	0.97
precision	0.90	0.97	0.86
Average precision	0.91	0.97	0.97
F1 score	0.94	0.97	0.91

جدول 4-4 مقایسه‌ی نحوه عملکرد شبکه‌ها بر روی داده‌های تست

	YOLO	SSD	RetinaNet
Recall	0.90	0.78	0.93
precision	0.83	0.77	0.78
Average precision	0.81	0.74	0.90
F1 score	0.86	0.78	0.85

دقت ردیابی الگوریتم SORT

جدول 4-5 مقایسه‌ی دقت ردیابی

	YOLO	SSD	RetinaNet
accuracy	0.31	0.31	0.33
precision	0.69	0.77	0.73

نحوه برچسب‌گذاری داده‌های UA-DETRAC به این صورت است که مکان هر خودرو و یک شماره به آن خودرو اختصاص داده شده است و سپس در فریم‌های بعدی اگر آن خودرو وجود داشته باشد همان شماره به آن اختصاص دارد.

فصل 5:

جمع‌بندی و پیشنهادات

5-1- مقدمه

ما در این تحقیق سه الگوریتم شناسایی اشیا را بررسی کرده و آموزش دادیم و دقت هر یک را ارائه نمودیم. همانطور که مشاهده شد دقت همه‌ی آن‌ها در شناسایی بالای 80 بود ولی در ردیابی تنها حدود 30 درصد اشیا شناسایی شده به درستی ردیابی شده بودند.

یکی از روش‌های موثری که می‌توان برای افزایش دقت انجام داد پیش پردازش تصاویر است که ما در این تحقیق آن را اعمال نکرده‌ایم همچنین دقت شبکه‌های عصبی تا حدود زیادی بستگی به میزان داده‌هایی که با آن آموزش می‌بینید دارد یکی محدودیت‌های پروژه نبود سیستم مناسب برای آموزش شبکه بود.

یک راهکار دیگر با توجه به این که دقت شناسایی خودروها مناسب است و دقت ردیابی پایین است بررسی روش های دیگر ردیابی بود که ما تنها یک روش را به کار برده‌ایم.

مراجع

مراجع

[1] Tony Lindeberg Kth, "Scale-space: A framework for handling image structures at multiple scales," 1996.

[2] Dieter Koller et al, "Robust multiple car tracking with occlusion reasoning," 1994.

[3] Nicholas A.Mandellos et al, "A background subtraction algorithm for detecting and tracking vehicles," 2011.

[4] Lyu et al, "New Benchmark and Protocol for Multi-Object Detection and Tracking," 2020.

[5] Ross Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[6] Ross Girshick, "Fast R-CNN," 2015.

[7] Joseph Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection," 2016.

[8] Ali Farhadi Joseph Redmon, "YOLO9000: Better, Faster, Stronger," 2016.

[9] Ali Farhadi Joseph Redmon, "YOLOv3: An Incremental Improvement," 2018.

[10] AYOOSH KATHURIA. (2018) blog.paperspace. [Online]. **https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/**

[11] Wei Liu et al, "SSD: Single Shot MultiBox Detector," 2016.

[12] Tsung-Yi Lin et al, "Focal Loss for Dense Object Detection," 2017.

[13] Alex Bewley et al, "Simple Online and Realtime Tracking," 2016.

1