

الله
لله
لله



دانشگاه صنعتی شریف

دانشکده مهندسی برق

درس یادگیری عمیق

گزارش پروژه نهایی

عنوان:

Joint Object Detection and Depth Estimation in Images

نگارش:

زهرا مس کار

محسن شیرکرمی

استاد راهنما:

دکتر فاطمی زاده

بهمن ۱۴۰۰

چکیده

توانایی تخمین فاصله با اشیاء اطراف توسط سیستم، به آن کمک می کند تا درک بهتری از محیط اطراف داشته باشد و توانایی سنجش موقعیت خود نسبت به اجسام مختلف را داشته باشد. در این مقاله می خواهیم به طراحی یک شبکه مت Shankل از دو زیرشبکه تشخیص اشیاء و تخمین عمق پردازیم که قادر به تشخیص اشیاء و عمق هر کدام از آنها در یک تصویر باشد.

واژه‌های کلیدی:

تشخیص اشیاء، تخمین عمق، یادگیری عمیق، شبکه عصبی

فهرست

۱.....	فصل اول: مقدمه	۱
۲.....	ساخтар پژوهش	۱-۱
۳.....	فصل دوم: داده‌ها.....	۲
۳.....	مقدمه	
۳.....	داده ها.....	۲-۲
۷.....	جمع‌بندی	۳-۲
۸.....	فصل سوم: شبکه‌های تشخیص اشیاء و تخمین عمق	۳
۸.....	مقدمه	۱-۳
۹.....	شبکه‌های عصبی پیچشی	۲-۳
۱۲.....	تشخیص اشیاء با استفاده از شبکه عصبی	۱-۲-۳
۱۳.....	شبکه های بررسی شده ی تشخیص اشیاء	۲-۲-۳
۲۹.....	شبکه ها و مقالات بررسی شده برای مسئله ی تخمین عمق	۳-۲-۳
۳۲.....	جمع‌بندی	۳-۳
۳۳.....	فصل چهارم: پیاده‌سازی عملی شبکه تشخیص همزمان اشیاء و عمق	۴
۳۳.....	مقدمه	۱-۴
۳۳.....	پیاده سازی	۲-۴

۳۴.....	روش اول: آموزش شبکه تخمین عمق ۱-۲-۴
۴۴.YOLOV5	۴-۲-۲ روشن دوم: شبکه تخمین عمق pre-trained به همراه تشخیص اشیاء با ۴
۴۹.....	۴-۲-۳ روشن سوم: مدل pre-trained تخمین عمق و مدل تشخیص اشیاء YOLOV2 ۴
۵۰.....	۳-۴ جمع‌بندی ۳-۴
۵۱.....	۵ فصل پنجم: روش‌های ارزیابی عملکرد شبکه ۵
۵۱.....	۱-۵ مقدمه ۱-۵
۵۲.....	۲-۵ ارزیابی شبکه تشخیص اشیاء ۲-۵
۵۳.....Mean Average Precision	۵-۲-۱
۵۵.....	۳-۵ ارزیابی شبکه تخمین عمق ۳-۵
۵۶.....	۴-۵ ترکیب شبکه‌ها ۴-۵
۵۷.....	۵-۵ سایر مقالات بررسی شده ۵-۵
۶۰.....joint	۶-۵ معیار ارزیابی شبکه‌ی joint ۶-۵
۶۲.....	۷-۵ پیشنهادها ۷-۵
۶۳.....	۸-۵ جمع‌بندی ۸-۵
۶۴.....	۶ فصل ششم: طراحی رابطه‌ای کاربری ۶
۶۴.....	۱-۶ مقدمه ۱-۶
۶۴.....	۲-۶ ربات تلگرام ۲-۶

۶۷.....	وب اپلیکیشن	۳-۶
۶۹.....	جمع بندی	۴-۶
۷۰.....	مراجع.....	۷

۱ فصل اول: مقدمه

هر موجود هوشمندی برای حرکت در محیط اطرافش نیاز به فهم درستی از آن محیط دارد. یکی از مهم‌ترین موارد در درک درست از محیط اطراف، دانستن نوع اشیاء و مقدار فاصله آنها با دقت نسبتاً مناسبی است. این موضوع در مورد انسان نیز صادق است. انسان در هر زمانی به خصوص هنگامی که در حال حرکت است به صورت ناخودآگاه عمق حدودی هر قسمت و یا شیء از محیط اطراف را می‌سنجد و با توجه به آن تصمیم می‌گیرد.

بنابرین تشخیص اشیاء اطراف و تخمین عمق آن‌ها یکی از ابزارهایی است که به کامپیوتر امکان داشتن دید بهتر نسبت به محیط اطرافش را می‌دهد و جزء جدایی ناپذیری از سیستم‌های هوشمند است. این مشخصه، کاربرد بسیار زیادی در حوزه‌هایی همچون رباتیک و اتومبیل‌های خودران دارد.

حال اگر بخواهیم با استفاده از یک تصویر دو بعدی که از یک دوربین ثبت شده است، تخمین عمق را انجام دهیم باید به این نکته توجه کنیم که سنسور دوربین سنسوری غیرفعال است، که اطلاعات سه بعدی محیط را به صورت دوبعدی ضبط می‌کند. بنابرین اطلاعات عمق در خروجی این سنسور از بین می‌روند. یک از راههای حل این مشکل، استفاده از چند دوربین به عنوان سنسور تشخیص

عمق است (multi view geometry). به طور مثال، ایده تخمین عمق با استفاده از دو دوربین یا بیشتر که به stereo vision معروف است را می‌توان معرفی کرد که در اتومبیل‌های خودران معمول است که از دو دوربین به عنوان جفت برای تخمین عمق استفاده می‌کنند. به دلیل هزینه‌های محاسباتی بالا در این روش، می‌توان روش تخمین عمق روی تمام پیکسل‌های تصویر را مثال زد که در آن، به هر نقطه از محیط اطراف یک عمق نسبت داده می‌شود.

علاوه بر مسئله تخمین عمق، راه حلی برای تشخیص اشیاء در یک تصویر ارائه کنیم. تشخیص اشیاء خود از ترکیب دو مسئله طبقه‌بندی شیء و همچنین تعیین موقعیت برای تمام اشیاء موجود در تصویر تشکیل شده است. راه حل ما باید قادر باشد تا در یک تصویر، اشیاء را شناسایی کرده و نوع هر کدام (از قبیل فرد، ماشین و...) تشخیص دهد.

در این پژوهه می‌خواهیم که با بهره‌گیری از دو شبکه تشخیص اشیاء و همچنین تخمین عمق، یک سیستم را طراحی کنیم که قادر به تشخیص همزمان این دو مورد در یک تصویر است.

1-1 ساختار پژوهش

پژوهش حاضر دارای شش فصل می‌باشد که در فصل دوم داده‌هایی که در اختیار ما قرار گرفته را بررسی خواهیم کرد. در فصل سوم، مروری بر شبکه‌های موجود در تشخیص اشیاء و تخمین عمق خواهیم داشت و در فصل چهارم، به پیاده‌سازی عملی این شبکه‌ها و بررسی نتایج خواهیم پرداخت. در فصل پنجم، در مورد شیوه‌های ارزیابی این شبکه‌ها و به طور کلی مسائل مربوط به شبکه‌های تشخیص شیء و تخمین عمق پرداخته و در مورد اتصال این شبکه‌ها به یکدیگر صحبت خواهیم کرد. در فصل ششم نیز، دو پلتفرم متفاوت از پیاده‌سازی رابط کاربری این پژوهش را خواهیم دید.

۲ فصل دوم: داده‌ها

2-1 مقدمه

برای داشتن یک طراحی درست، باید با ابزارهای در اختیار، کاملاً آشنا بود. داده، یک از مهم‌ترین ابزارهای مورد استفاده در شبکه‌های عمیق است. در این فصل به بررسی دیتاست داده شده برای شبکه تخمین عمق می‌پردازیم.

2-2 داده‌ها

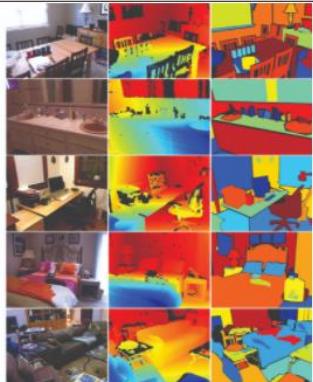
دیتاستی که برای این پروژه مورد استفاده قرار گرفته دیتاست NYU-Depth V2 می‌باشد. این دیتاست از رشته‌های ویدیو و از مناظر تشکیل شده که به وسیله‌ی هر دو نوع دوربین تعیین عمق و دوربین RGB

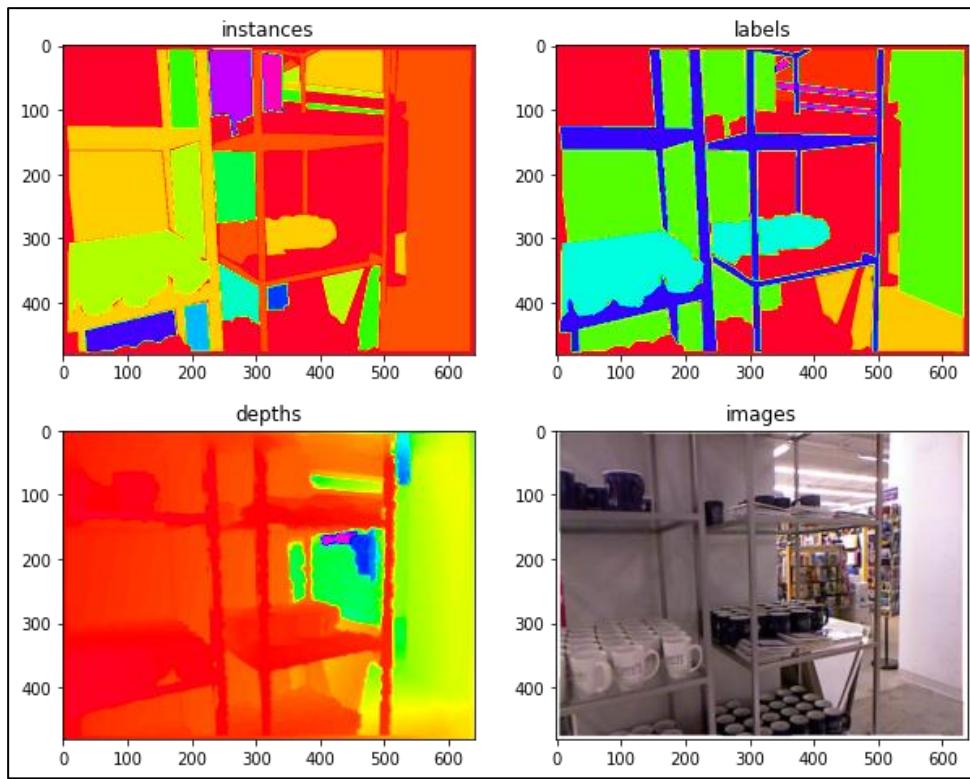
ضبط شده اند. این دیتاست متشکل از ۱۴۴۹ جفت تصویر RGB و عمق متناظر با آن است و در مجموع در آن ۴۶۴ صحنه از ۳ شهر گرفته شده است. هر شی با یک کلاس و یک شماره مختص به آن نام گذاری شده است. دیتاست برچسب گذاری شده زیر مجموعه ای از دیتاست خام اصلی است که در آن علاوه بر عمق تک تک پیکسل ها یک مجموعه ای از عمق های پیش پردازش شده نیز قرار دهد که در آن مقادیر از دست روده توسط [colorization scheme of Levin et al](#) مقداردهی شده اند.

نمونه هایی از تصاویر این دیتاست به همراه سایر ویژگی های آن در شکل زیر آورده شده است.
این ماتریس حاوی دادگان accelData:
چه زمانی گرفته شده. هر ستون نشان دهنده ی یکی از پارامتر های roll,yaw,pitch و tilt angle می باشد.
این ماتریس نشان دهنده ی عمق هر پیکسل بر حسب متر می باشد.

این ماتریس حاوی تصاویر RGB Images:
این ماتریس حاوی ماسک های هر شی می باشد.
این ماتریس حاوی ماسک های هر شی می باشد.

ماتریسی متشکل از این که هر پیکسل به چه کلاسی تعلق دارد. Labels:
ماترسی که نام هر کلاس را در بر دارد Names:

NYU Depth V2									
 					<ul style="list-style-type: none"> • 464 different indoor scenes • 26 scene types • 407,024 unlabeled frames • 1449 densely labeled frames • 1000+ Classes • Inpainted and raw depth available • Both object and instance labels 				



```
print(names)
[["book"], ["bottle"], ["cabinet"], ["ceiling"], ["chain"], ["cone"], ["counter"], ["dishwasher"], ["faucet"], ["fire extinguisher"], ["floor"], ["garbage bin"], ["microwave"], ["paper towel disp
```

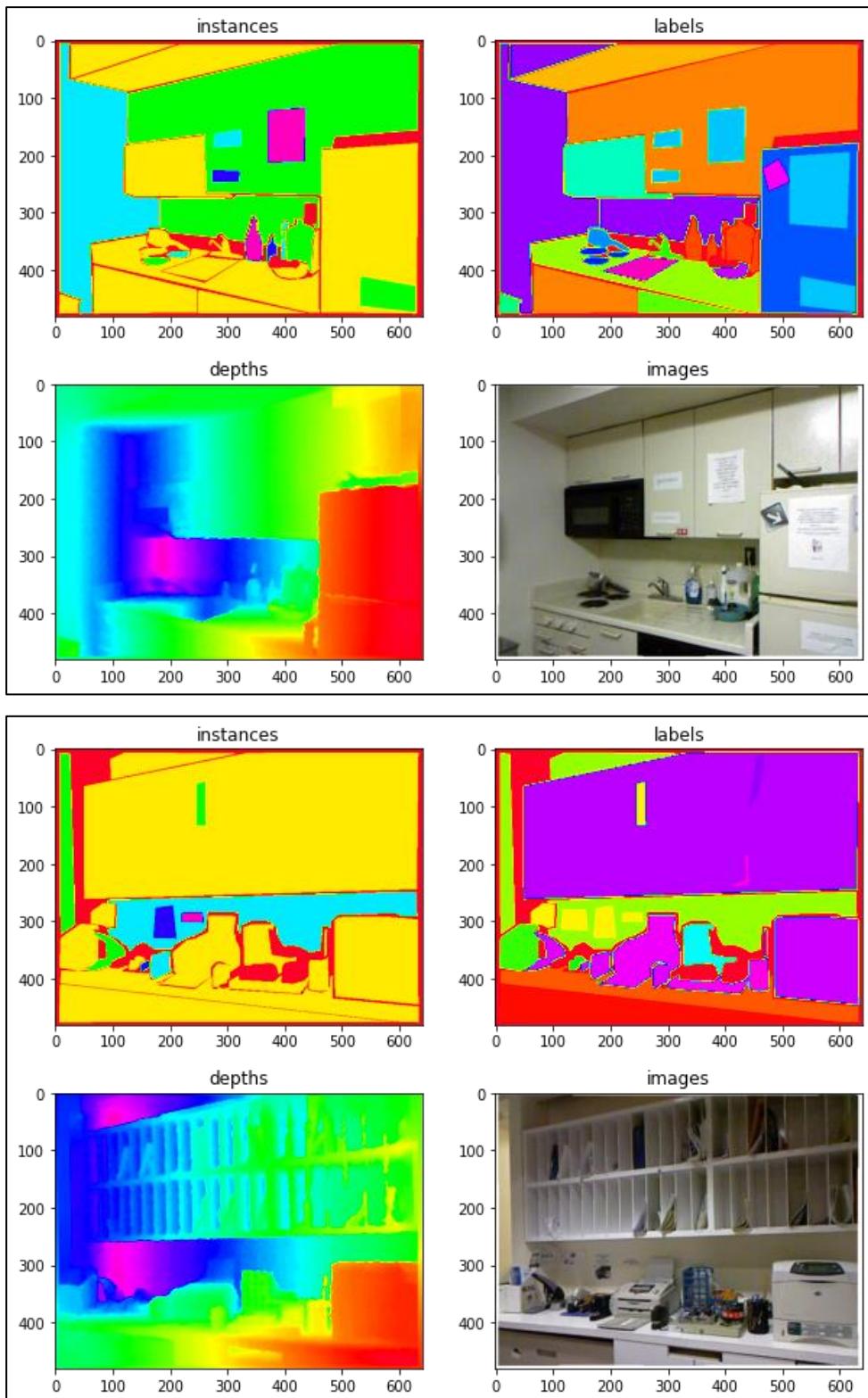
پس label ها نشان می دهند که هر شیء کدام یک از اشیاء دسته بندی بالاست.

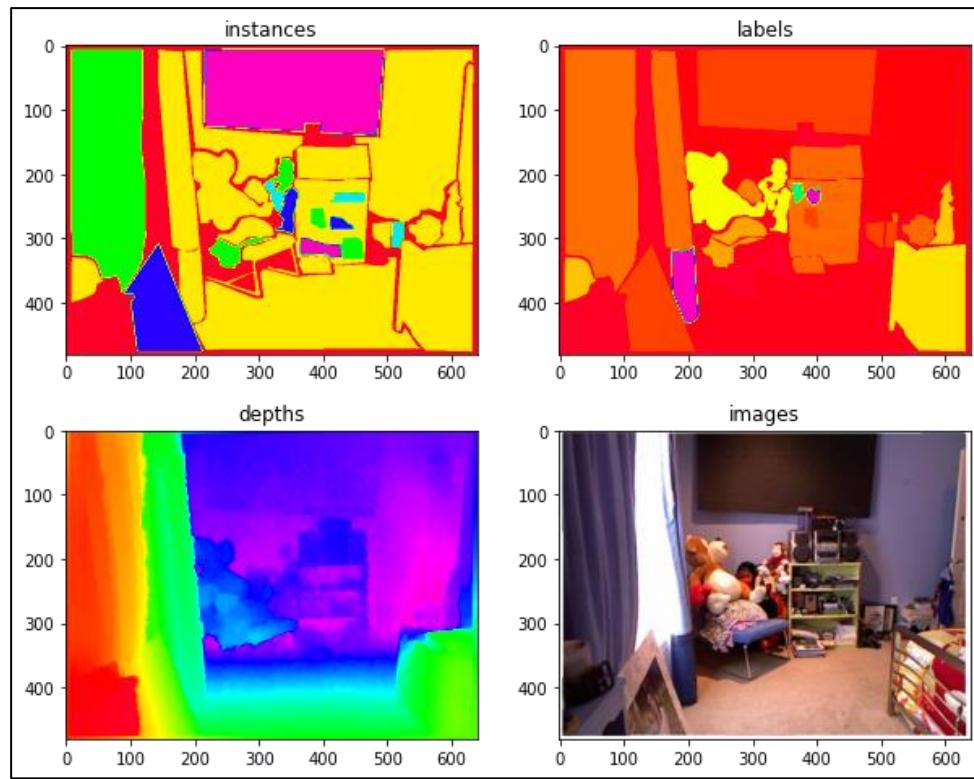
یک نگاشتی است از برچسب ها به نام هر کدام از کلاس ها. namesTolds

ماتریس خام عمق های پیکسل ها که حاوی عمق هاست قبل از این که مقادیر از دست

رفته مقدار دهی شوند. در این ماتریس غیر خطی بودن های دیوايس Kinect حذف شده اند و مقادیر به

متر داده شده اند.





2-3 جمع‌بندی

در این فصل دیتاست NYU-Depth V2 بارگذاری و به بررسی جزئیات آن پرداخته و در نهایت چند

نمونه از تصاویر آن نمایش داده شد.

۳ فصل سوم: شبکه‌های تشخیص اشیاء و تخمین عمق

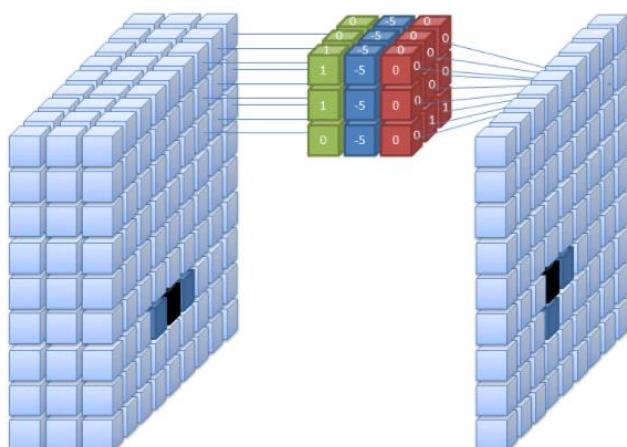
3-1 مقدمه

در این فصل قسمت اولیه سیستم مورد نظر برای تخمین عمق، یعنی آموزش شبکه‌ای برای تشخیص اشیاء را بررسی می‌کنیم. ویژگی‌ای که در نهایت به عنوان مشخصه‌ی یک شیء انتخاب می‌شود باید رابطه مناسبی با عمق اشیاء در تصویر داشته باشند. برای این منظور با توجه به دقت بالای YOLO [4] از این شبکه استفاده می‌کنیم.

در ادامه پس از آشنایی با مقدمات و تاریخچه به بررسی معماری های توسعه داده شده برای انجام وظیفه تشخیص شئ^۱ می پردازیم و درنهایت در مورد معماری شبکه YOLO و مزایا و معایب آن بحث می کنیم.

3-2 شبکه های عصبی پیچشی

شبکه های عصبی پیچشی^۲ برای پردازش تصاویر گسترش یافتهند که اساس کارکرد آنها آموزش فیلترهایی در هر لایه است بدین ترتیب که در هر لایه فیلترهایی روی عکس اعمال می شوند و نتیجه به لایه بعدی منتقل می شود. باروی کار آمدن این شبکه ها دقت پردازش تصاویر با استفاده از شبکه عصبی benchmark قابل توجهی داشت به صورتی که دقت روی ساختمان داده imagenet که یکی از رشد قابل توجهی این حوزه است به ۸۰ درصد رسید، که با توجه به اینکه قبل از این روی این دقت در ۴۰ درصد به اشباح رسیده بود انقلابی در این حوزه پدید آورد.



شکل ۳-۱ عملکرد لایه های کانولوشنال

Object detection^۱

² Convolutional neural networks

در این شبکه‌ها از دو نوع لایه مختلف استفاده می‌شود که

۱- لایه‌های پیچشی^۱ ۲- لایه‌های پولینگ^۲

لایه‌های پیچشی:

همانطور که در بالا گفته شد اساس کارکرد شبکه‌های پیچشی به این صورت است که در هر لایه

فیلترهایی روی تصاویر اعمال شوند و ویژگی خاصی را از آن استخراج کنند.

ابرپارامترهایی که در این لایه‌ها دخیل هستند علاوه بر سایز فیلتر، تعداد آن‌ها در هر لایه(عمق) گام

حرکت را نیز شامل می‌شود که با انتخاب‌های متفاوت این موارد به نتایج متفاوتی می‌رسیم. برای مثال عمق

خروجی به تعداد فیلترهای انتخابی بستگی دارد و همچنین سایز آن به گام حرکت فیلتر انتخابی بستگی

دارد.

لایه‌های پولینگ:

به تجربه ثابت شد که وجود لایه‌هایی که از مجموعه چند پیکسل، یک پیکسل را با توجه به یک

ویژگی(feature) برگرداند تأثیر زیادی در بهبود دقت شبکه‌های پیچشی دارد و همین موضوع باعث به

وجود آمدن لایه‌های پولینگ شد.

به صورت کلی دو نوع لایه پولینگ داریم که عبارت‌اند از:

(۱) Max pooling: در این لایه‌ها از هر ناحیه دربگرفته شده مقدار ماکریم در بین پیکسل‌های آن

ناحیه برگردانده می‌شود.

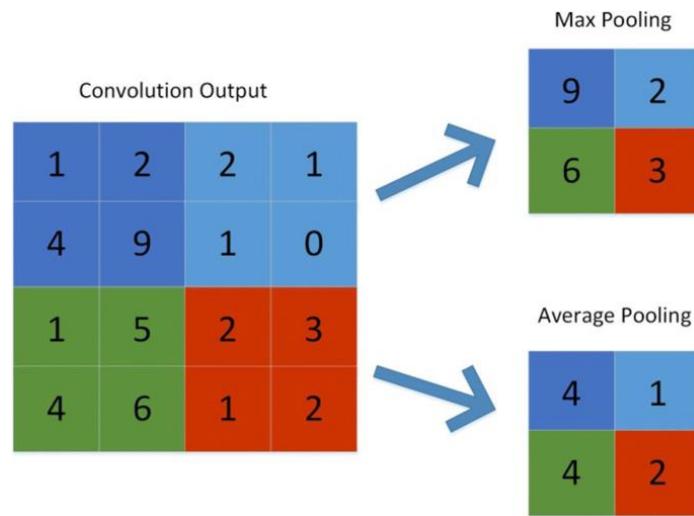
(۲) Average pooling: این لایه‌ها نیز مانند لایه‌های Max pooling خروجی واحد برای هر

ناحیه بر می‌گردانند اما با این تفاوت که خروجی این لایه‌ها میانگین مقدار پیکسل‌های دربگرفته

Convolutional layers^۱

Pooling layers^۲

شده است.



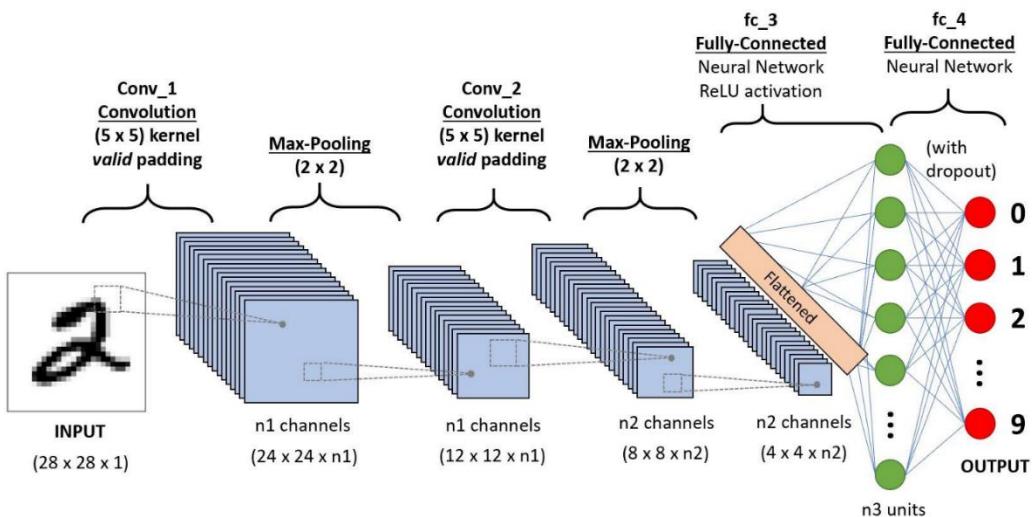
شکل ۳-۲ مقایسه لایه های max pooling و average pooling

با استفاده از ترکیب های متفاوت لایه های پیچشی و پولینگ می توان شبکه های متفاوتی را ایجاد کرد که هر کدام به دقت های متفاوتی برای دیتاهای متفاوت می رستند. اما بیشتر معماری های متداول مانند VGG16, VGG19, Resnet, Mobilenet تا اینجا در مورد اجزای شبکه های پیچشی صحبت کردیم. اما این لایه ها چگونه کار می کنند و خروجی به چه شکل است؟ به صورت کلی در اکثر موارد شبکه های پیچشی برای دسته بندی (classification) عکس ها مورد استفاده قرار می گیرند به این صورت که پس از اعمال فیلتر های مختلف همراه با لایه های pooling در نهایت ویژگی هایی از عکس ها استخراج می شود که با استفاده از لایه Flattened بردار تبدیل می شوند و سپس این بردار ویژگی ها به یک شبکه MLP داده می شوند تا با خروجی موردنظر را از آنها استخراج کند.

- اخیراً بجای استفاده از لایه های flatten که تعداد پارامتر های بسیار زیادی به شبکه اضافه می کنند، از شبکه های تماماً پیچشی^۱ استفاده می شود و درنتیجه پارامتر های شبکه بسیار کمتر می شوند و

Fully convolutional¹

فرایند جمع‌آوری دیتا و آموزش راحت‌تر می‌شود.



شکل ۳-۳- نمونه‌ای از شبکه عصبی کانولوشنال

۱-۲-۳- تشخیص اشیاء با استفاده از شبکه عصبی

تا کنون در مورد شناسایی و دسته بندی تصاویر با استفاده از شبکه های عصبی پیچشی صحبت کردیم

بدین صورت که عکس مورد نظر به شبکه داده می شود و عملیات پیش بینی و شناسایی روی کل تصویر صورت می گرفت. اما در بسیاری از موارد نیاز داریم که یک شیء را داخل یک تصویر پیدا کنیم و محدوده ای را برای آن تعیین کنیم این عمل در پردازش تصویر به object detection معروف است. در واقع عمل object detection ترکیبی از دو عمل رگرسیون^۱ و دسته بندی^۲ است. رگرسیون برای تشخیص مختصات باکس و دسته بندی برای تشخیص شیء مورد نظر استفاده می شوند.

¹ regression
² classification

برای تشخیص شئ در تصویر نیاز داریم که که احتمال وجود شئ مورد نظر در آنها وجود داشته باشد بدست آوریم و سپس با استفاده از یک شبکه عصبی پیچشی هر کدام از این باکس ها را دسته بندی^۱ کنیم تا احتمال وجود شئ در آنها را بسنجدیم. همچنین در برخی از این شبکه ها پس از انتخاب باکس مناسب مجددا روی آن عملیات رگرسیون انجام می شود تا مختصات آن نیز با دقت خوبی بدست بیایند. قبل از ابداع این شبکه ها با استفاده از الگوریتم های کلاسیک مانند: هیستوگرام گرادیان ها یا روش R-CNN های رأی دهنده^۲ عملیات تشخیص شئ انجام می شد اما با ارائه اولین شبکه در این زمینه یعنی دقت این تسلیک بسیار بالا رفت.

برای این وظیفه^۳ نیز معماری های مختلفی به وجود آمدند. برای مثال می توان از Faster R-CNN، YOLO^۴ و ... نام برد.

۳-۲-۳-۲ شبکه های بررسی شده ی تشخیص اشیاء

۳-۲-۲-۱ MTCNN [6]

تشخیص چهره یکی از پیچیده ترین موارد تشخیص شئ است زیرا چهره افراد متفاوت می تواند تفاوت های بسیاری باهم داشته باشند و علاوه بر آن با توجه به حالت فرد چهره او می تواند حالت های متفاوتی داشته باشد که این امر کار تشخیص را بسیار دشوار می کند. به همین دلیل آموزش شبکه های که بتواند به دقت مناسب در این زمینه برسد کاری دشوار است. تحقیق های زیادی در این زمینه صورت گرفته اند ولی در این میان یکی از شبکه هایی که توانسته به دقت بسیار مناسب در این زمینه برسد شبکه

classify^r

voting^r

task^r

Single Shot Detector⁴

You only Look Once^۵

عصبی پیچشی چندوظیفه‌ای^۱ است. که به اختصار MTCNN نامیده می‌شود. ویژگی بارز این شبکه کم عمق^۲ بودن و در عین حال دقت بالای آن است.

اما شبکه ای است که تنها برای تشخیص یک شئ(چهره) استفاده شده است و نمی‌توان از آن برای تشخیص چندین نوع شئ متفاوت استفاده کرد. و در صورت نیاز باید برای این امر شخصی سازی شود. قبل از ورود عکس به شبکه در ابتدا از تصویر موجود یک image pyramid ساخته می‌شود یعنی تصویر به سایزهای متفاوتی resize می‌شود.(این قسمت جزو قسمت آماده‌سازی دیتا است و قسمتی از شبکه نیست.)

این شبکه از سه شبکه کوچک‌تر تشکیل شده است که عبارت‌اند از:

(P-Net)Proposal Net - ۱

ورودی‌های این شبکه عکس‌های resize شده با اندازه $3 \times 12 \times 12$ هستند. همانطور که در شکل شکل ۸-۲ در این شبکه در سه لایه فیلترهایی روی تصویر اعمال می‌شوند و سپس باکس‌هایی که به عنوان کاندید برای وجود چهره مطرح هستند را به عنوان خروجی برمی‌گردانند. بنابراین این شبکه که اولین قسمت از شبکه MTCNN است به سرعت و با هزینه محاسباتی بسیار کم باکس‌هایی که امکان دارد که چهره‌ای در آن‌ها باشد را شناسایی می‌کند.

(R-Net)Refinement Net - ۲

- ورودی‌های این شبکه عکس‌های resize شده با اندازه $3 \times 24 \times 24$ هستند. در این

قسمت باکس‌هایی که در شبکه P-Net به دست آمده‌اند بررسی(classify) می‌شوند. این شبکه در خروجی باکس‌هایی که چهره در آن‌ها وجود دارد را برمی‌گرداند. با توجه به اینکه

Multi task convolutional neural network^۱
shallow^۲

باکس‌های پیش‌بینی شده در شبکه P-Net دقت مناسبی دارند بنابراین پس خروجی شبکه R-

Net به دقت بسیار مناسبی می‌رسد و با دقت بسیار بالایی چهره‌های تصویر را تشخیص

می‌دهد.

(O-Net)Output Net - ۳

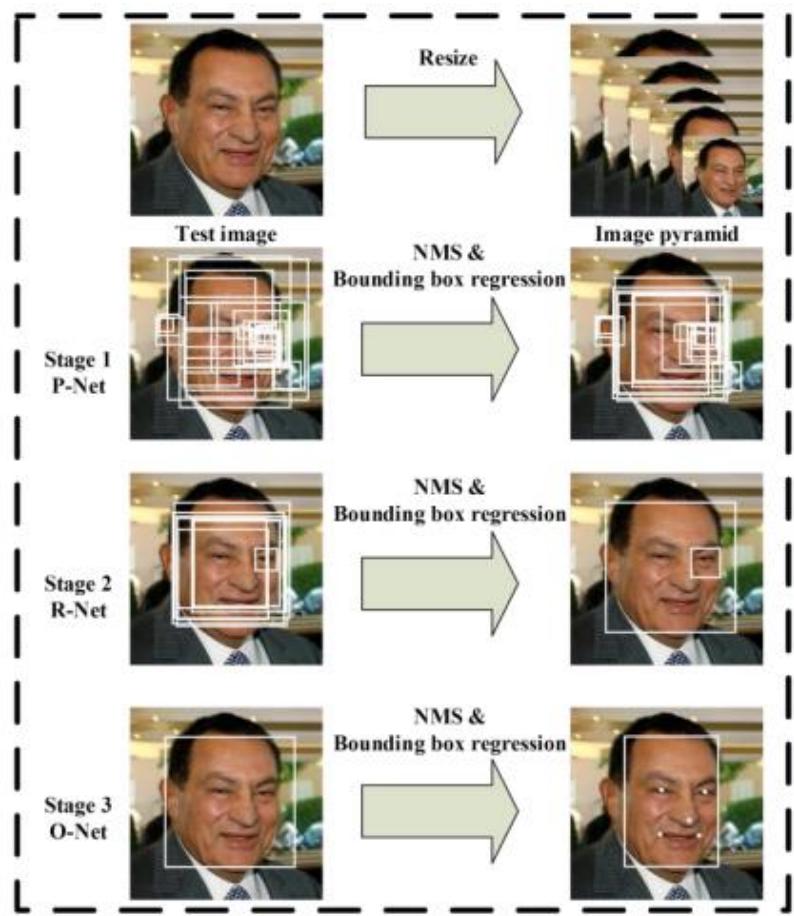
ورودی‌های این شبکه عکس‌های resize شده با اندازه $3 \times 48 \times 48$ هستند. همانطور که گفته

شد در قسمت‌های قبلی به دقت مناسبی برای تشخیص چهره رسیدیم در این شبکه مختصات

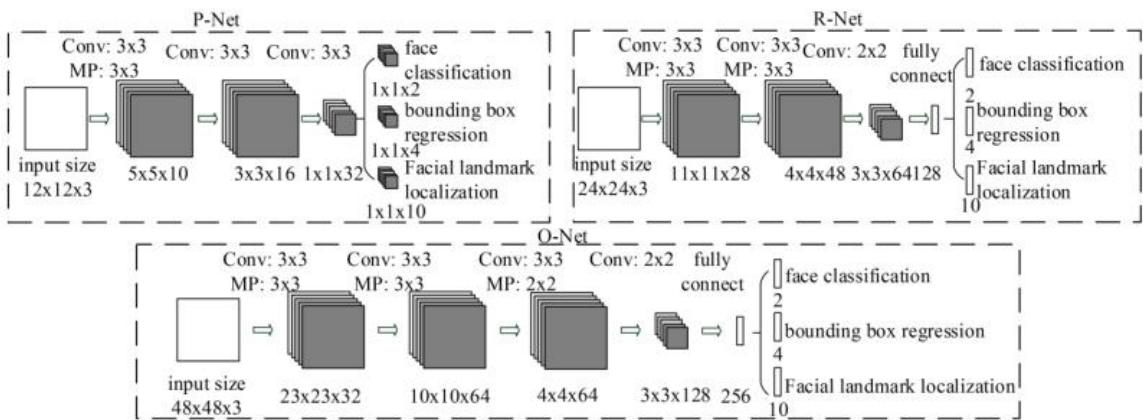
باکس‌های قبلی به صورتی تغییر می‌کنند که تصویر را به صورت کامل در برگیرد و بیشتر

امتیاز(score) وجود چهره به دست آید. همچنین این شبکه کار تشخیص landmark یا همان

نقاط عطف صورت را انجام می‌دهد.



شکل ۳-۴ نحوه‌ی تشخیص چهره توسط شبکه‌ی MTCNN



شکل ۳-۵ ساختار شبکه MTCNN

در برخی از موارد پس از لایه اول یک مرحله عملیات non maximum suppression روی باکس^۱های که با هم همپوشانی دارند اعمال می‌شود و با این عمل تعداد از باکسها که اسکور آنها کمتر است حذف شده و کار تشخیص سریعتر انجام می‌شود. البته این موضوع در مواردی باعث می‌شود که صورتهایی که کوچک هستند تشخیص داد نشوند که این موضوع بسیار کم اتفاق می‌افتد و قابل چشم پوشی است.

بنابرین همانطور که در بالا توضیح داده شد شبکه MTCNN شبکه‌ایست که از سه زیرشبکه تشکیل شده است که هر کدام وظیفه خاصی را انجام می‌دهند به همین دلیل به این شبکه، شبکه چند وظیفه‌ای یا multi task گویند.

یک ویژگی مثبت دیگر این شبکه این است که سه قسمت آن از هم مستقل هستند و می‌توان هر کدام را جداگانه آموزش داد این موضوع باعث می‌شود در زمان آموزش آزادی عمل بیشتری داشته باشیم و همچنین زمان کمتری تلف شود و به نتیجه بهتری برسیم.

همچنین اگر به شکل بالا توجه کنیم ورودی‌های سه قسمت شبکه سایزهای متفاوتی دارند این سایزها برای شبکه‌های P-Net, O-Net, R-Net به ترتیب $3 \times 24 \times 24 \times 3$, $48 \times 48 \times 3$, $12 \times 12 \times 3$ می‌باشند. بنابرین برای هر عکس باید عملیات interpolation سه بار انجام شود که این موضوع باعث می‌شود که فرایند پیش پردازش دیتا افزایش یابد اما با توجه به shallow بودن شبکه و سرعت آن این قسمت قابل چشم پوشی است.

¹Bounding box

عملیات درونیابی^۱ نیز زمانی استفاده می شود که نیاز داشته باشیم که سایز یا رزولوشن تصویر را افزایش دهیم در چنین مواردی نیاز داریم که پیکسل هایی را به عکس اضافه کنیم برای این مورد روش های متنوع وجود دارد که عبارتند از:

Nearest neighbor (۱)

در این روش با توجه به مکان پیکسل جدید در مورد مقدار آن تصمیم گرفته می شود. به عبارتی پیکسل جدید مقداری برابر با نزدیکترین پیکسل موجود از لحاظ مکانی می گیرد.

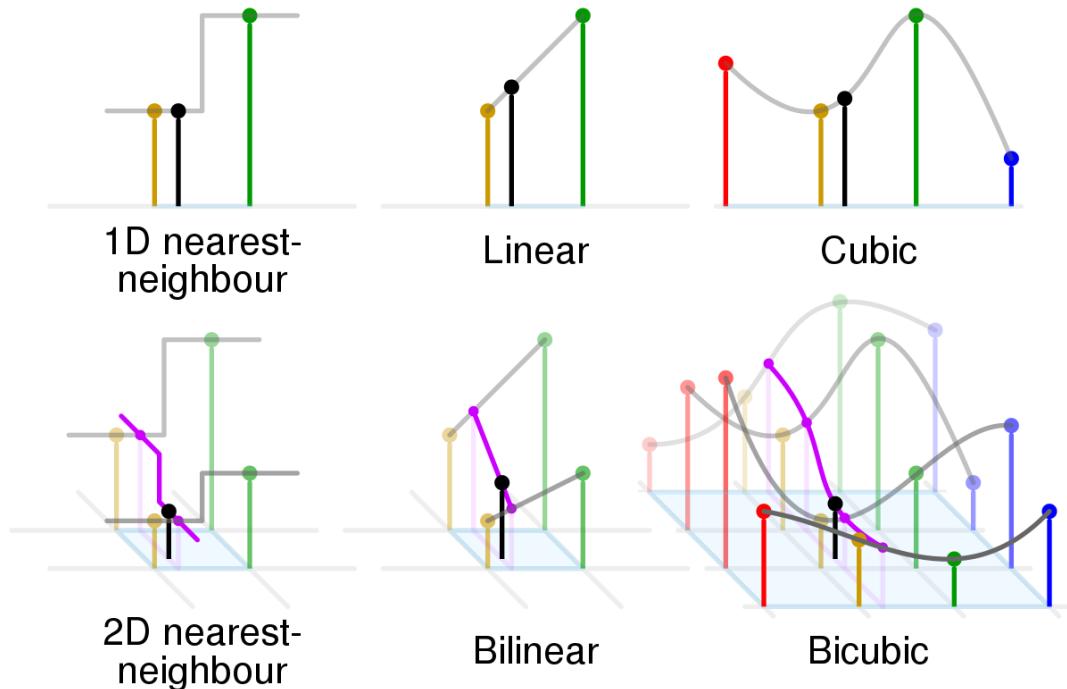
Bilinear (۲)

در این روش مقدار پیکسل جدید با درونیابی بین پیکسل های اطرافش بدست می آید. این درونیابی به صورت خطی صورت می گیرد و عملکرد بهتری نسبت به روش قبل دارد. اما هزینه محاسباتی آن به نسبت بالاتر است.

Bicubic (۳)

این تبدیل نیز مانند تبدیل قبل از درونیابی استفاده می کند اما این درونیابی غیر خطی است، بنابرین نیاز است که مقدار چندین پیکسل اطراف در نظر گرفته شود تا بتوان به تابع غیرخطی مورد نظر رسید. این تبدیل نیز عملکرد بهتری از دو روش قبلی دارد. اما هزینه محاسباتی آن به طرز قابل توجهی نسبت به آنها بیشتر است.

interpolation^۱



شکل ۳-۶ انواع درونیابی

۳-۲-۲-۲ تشخیص دهنده های شی کلاسیک

از جمله BOX، EDGE، SIFT، HOG،^۱ ویژگی استخراج می کنند و بعد روی این ویژگی ها طبقه بندی انجام می دهند تا نوع شی مشخص شود.

۳-۲-۲-۳ تشخیص اشیا مبتنی بر CNN

روش های تشخیص اشیاء مبتنی بر CNN دو دسته هی یک مرحله ای و دو مرحله ای دارند.
 - یک گام: که در آن به طور مستقیم احتمال تعلق به کلاس ها و bounding box ها حساب می شوند.
 مثال آن ها SSD^۱ و YOLO^۲ هستند که در اینجا ما از این ها استفاده می کنیم. روش های تک مرحله ای، با هدف حل چالش سرعت پایین در روش های دو مرحله ای پشنهاد شدند. روش های تک مرحله ای

¹ Single Shot MultiBox Detector

² You Only Look Once

شبیه این هستند که بلوک RPN در روش‌های دو مرحله‌ای را حذف کنیم. بنابراین در این دسته، مطابق

شکل زیر، بخش RPN وجود ندارد و فیچر مپ خروجی CNN مستقیماً به Detection Head می‌رود.

در اصل روش‌های تک‌مرحله‌ای تشخیص شی را مشابه با دسته‌بندی حل می‌کنند. چون در دسته‌بندی

هم تنها براساس فیچر مپ خروجی عمل دسته‌بندی/شناسایی اشیا انجام می‌شود. شبکه معروف YOLO

از جمله روش‌های تک‌مرحله‌ای است که پایه‌گذار روش‌های تک‌مرحله‌ای بود

-دو گام: object proposal RCNN/FAST RCNN/ FASTER RCNN که در آن‌ها ابتدا

حساب می‌شوند.

روش‌های دو مرحله‌ای، فرآیند تشخیص اشیا را در دو مرحله پیشنهاد ناحیه (Region Proposal

و Detection Head) Network انجام می‌دهند. طبق شکل زیر، این دو مرحله عبارتند از:

- ابتدا، بخش Region Proposal Network یا RPN چندین کاندیدای شی را به عنوان نواحی

موردنظر علاقه (RoI) یا Region of Interest (RPN در تصویر زیر)

- در بخش دوم (Detection Head)، مناطق پیشنهادی بررسی می‌شوند. تعدادی حذف می‌شوند و

تعدادی هم دقیق‌تر تنظیم می‌شوند و نام شی درون چارچوب هم تعیین می‌شود.

پس، تشخیص اشیای دو مرحله‌ای، یک حدس اولیه از محل احتمالی اشیا می‌زنند (همان RPN). این

حدس‌ها به شکل چارچوب نمایش داده می‌شود. این چارچوب‌ها خیلی دقیق نیستند و همچنین ممکن

هست بسیاری از آنها به شی خاصی اشاره نکند (پس زمینه باشد). به تصویر وسطی در شکل زیر نگاه

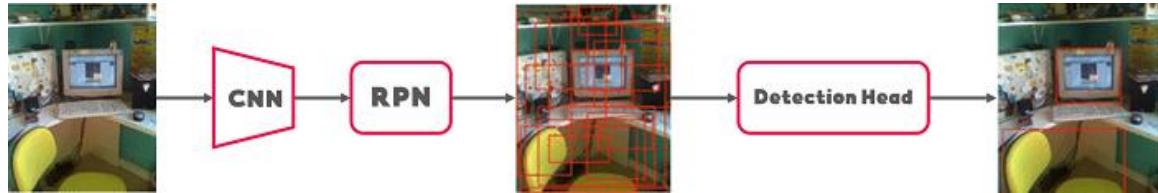
کنید؛ RPN یک عالمه چارچوب رسم کرده که خیلی از آنها کیفیت مطلوبی ندارند. در مرحله دوم، به

دبای حذف اشتباه‌ها هستیم. همچنین، چارچوب‌های غیردقیق را بهتر تنظیم می‌کنیم و نام شی موجود

در چارچوب را هم پیش‌بینی کنیم.

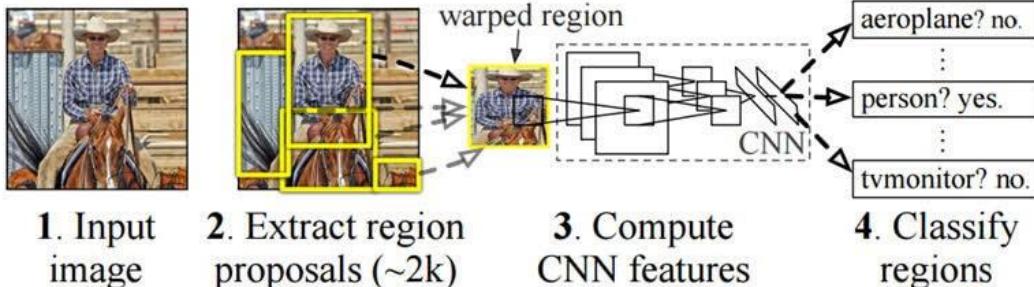
روش دو گامه از لحاظ محاسباتی برای کاربردهای واقعی مناسب نیست چرا که محدودیت محاسبه و حافظه داریم و در روش یک گامه این tradeoff بین سرعت و دقیقت بهتر است به همین دلیل ما روش اول را انتخاب کردیم. مثالی از این شبکه ها Faster R-CNN است و محبوبیت و قدرت بالایی دارد.

روش های دو مرحله ای، معمولاً دقیقت بالایی دارند، اما معمولاً نسبت به تک مرحله ای ها کندتر هستند.



RCNN ۳-۲-۲-۴

R-CNN: Regions with CNN features



معیار شباخت: وجه تشابه میان نواحی بر اساس معیارهای زیر به دست می آیند.

رنگ | بافت | اندازه | سازگاری شکل

Fast RCNN ۳-۲-۲-۵

نسخه پیشرفته RCNN که برخی از معایب RCNN را برطرف کرده است.

مزایا:

کیفیت تشخیص بالاتر (mAP) از SPPnet، R-CNN

زمان محاسبه کمتر به دلیل تک مرحله ای بودن

بی نیازی از فضای حافظه بیشتر برای ذخیره سازی ویژگی ها میانی.

پارامترهای کمتر در مقایسه با SPPnet و rcnn

۱- ایجاد نگاشت ویژگی: کل عکس به همراه پیشنهاد اشیا به Convnet داده می‌شود. با گذر از

لایه‌های Conv و لایه‌های ادغام بیشینه (Max Pooling)، نگاشت ویژگی به دست می‌آید.

۲- ROI Pooling: ناحیه مورد نظر (ROI) در نگاشت ویژگی با مختصات (y, r, c, h, w) بدست

می‌آید. این ناحیه از لایه ادغام ROI عبور می‌کند تا نگاشت ویژگی $H \times W$ به دست آید.

۳- لایه‌های کاملاً متصل: این نگاشت ویژگی استخراج شده و از لایه‌های FC عبور داده می‌شود.

هدف از این کار، پیش‌بینی احتمال و رگرسور برای خروجی‌های رگرسیون قادر محصور‌کننده

است.

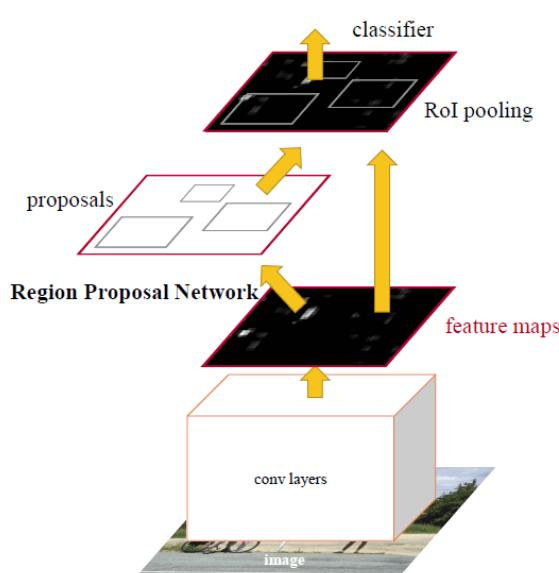
۳-۲-۲-۶ Faster RCNN

مزایا:

پیشنهاد نواحی در عکس را آسان می‌کند.

از یک شبکه عصبی کاملاً پیچشی برای این منظور استفاده می‌کند.

RPN که در این مقاله پیشنهاد شد، ویژگی‌ها را به خوبی با ابزار تشخیص اشیاء به اشتراک می‌گذارد.



ایجاد نگاشت ویژگی: عکس به درون لایه‌های Conv فرستاده می‌شود که خروجی آنها یک نگاشت ویژگی است.

شبکه پیشنهاد ناحیه: از یک پنجره لغزان در RPN برای هر بخش پیرامون نگاشت ویژگی استفاده می‌شود.

کادر: برای هر بخش، $k=9$ کادر محوری برای پیشنهاد ناحیه استفاده می‌شوند.

طبقه‌بندی: لایه cls از $k=2$ امتیاز خروجی، برای تعیین اینکه این k کادر شامل شی مورد نظر هست یا خبر استفاده‌هم می‌کند.

رگرسیون: لایه رگرسیون از $k=4$ خروجی (مختصات مرکز کادر، طول و عرض باکس) جهت تعیین k کادر استفاده می‌کند.

شبکه تشخیص: به جز بخش مربوط به RPN، شبکه تشخیص مثل rcnn سریع عمل می‌کند. آموزش دیگر: RPN و تشخیص به صورت متناوب آموزش داده می‌شوند و ویژگی‌های یاد گرفته شده را با هم به اشتراک می‌گذارند.

۳-۲-۲-۷ SSD [5]

در این ساختار از feature map های با میدان دید متفاوت برای تعیین و تشخیص اشیا استفاده می‌کنیم

۳-۲-۲-۸ DSSD

در این ساختار از deconvolution استفاده شده و ماژول پر迪کشن آن نسبت به SSD پیچیده تر است. به همین دلیل دقت ها بالا ترند.

۳-۲-۲-۹ retinaNet

مشکل روش های یک گامه را که class imbalance بود رو حل کرده و این کار رو با طراحی یک تابع هزینه‌ی جدید انجام داده.

۳-۲-۲-۱۰ YOLO [4]

الگوریتم‌های مختلفی برای پیاده‌سازی سیستم تشخیص اشیا در نظر گرفته شدند، اما در نهایت، الگوریتم YOLO به عنوان الگوریتم اصلی بر پیاده‌سازی این سیستم در نظر گرفته شد. دلیل انتخاب الگوریتم YOLO، سرعت بالا و قدرت محاسباتی آن و همچنین، وجود منابع آموزشی زیاد برای راهنمایی کاربران هنگام پیاده‌سازی این الگوریتم است. به دلیل محدودیت‌های محاسباتی و زمانی، تصمیمات گرفته شد تا از YOLO V2 و YOLO V5 از پیش آموزش داده شده برای شناسایی اشیاء به دلیل محدودیت سخت افزار استفاده شود.

مشخصات ورودی‌های مدل YOLO برای تشخیص اشیا

الگوریتم YOLO برای اینکه بتواند اشیاء موجود در تصویر را تشخیص دهد، ورودی‌های خاصی را می‌پذیرد:

ابعاد تصاویر ورودی: شبکه YOLO به گونه‌ای طراحی شده است تا با تصاویری که ابعاد مشخصی دارند آموزش ببیند. ابعاد تصویر استفاده شده برای آموزش شبکه YOLO، برابر با 608×608 است. تعداد کلاس‌ها: برابر ۴۳ کلاس است. تعداد کلاس‌ها، پارامتری است که برای تعریف ابعاد خروجی‌های شبکه YOLO مورد نیاز است.

پارامترهای Anchor Box: تعداد کادرهای محصور کننده یا Bounding Box و همچنین ابعاد بیشینه و کمینه آن‌ها؛ به مجموعه این اطلاعات، پارامترهای Anchor Box گفته می‌شود.

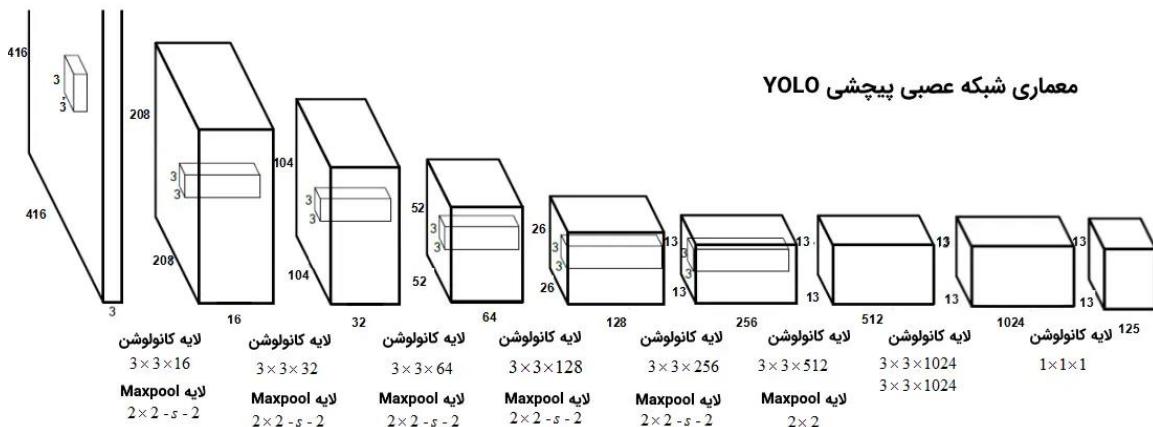
حد آستانه برای معیارهای IoU (معیار Intersection over Union) و ضریب اطمینان (Confidence): این حد آستانه به این دلیل تعریف شده است تا مشخص شود کدام یک از کادرهای محصور کننده، باید

به عنوان کادرهای در بر گیرنده اشیاء موجود در تصویر انتخاب شوند (انتخاب میان کادرهای محصور کننده).

اسامی تصاویر به همراه اطلاعات پارامترهای Anchor Box: به ازاء هر تصویر، نیاز است تا اطلاعاتی همانند اطلاعات تعبیه شده در شکل زیر در اختیار شبکه YOLO قرار گرفته شود.

معماری شبکه YOLO V2

معماری شبکه YOLO V2 در شکل زیر نمایش داده شده است. این شبکه، از ۲۳ لایه کانولوشن (پیچشی) تشکیل شده است؛ هر کدام از این لایه‌ها، واحد «نرمال‌سازی دسته‌ای» (Batch Normalization)، تابع فعال‌سازی Leaky RELU و واحد Max Pooling مختص به خود را دارد.



هدف لایه‌های تعریف شده، استخراج چندین ویژگی مهم از تصاویر دیجیتالی است تا این طریق، سیستم تشخیص اشیا قادر باشد اشیاء مختلف موجود در تصویر را تشخیص دهد و آن‌ها در کlassen‌های متناظر دسته‌بندی کند. با هدف تشخیص اشیا موجود در تصویر، الگوریتم YOLO تصویر را به یک گردید (Grid) مت Shankل از سلول‌های 19×19 تقسیم‌بندی می‌کند؛ به ازاء هر سلول تشکیل شده در گردید، پنج کادر محصور کننده با ابعاد متفاوت تعریف می‌شود. سپس شبکه YOLO تلاش می‌کند تا کlassen اشیاء موجود

در سلول‌های گریدی را تشخیص دهد؛ به عبارت دیگر، احتمال تعلق هر کدام از اشیاء شناسایی شده

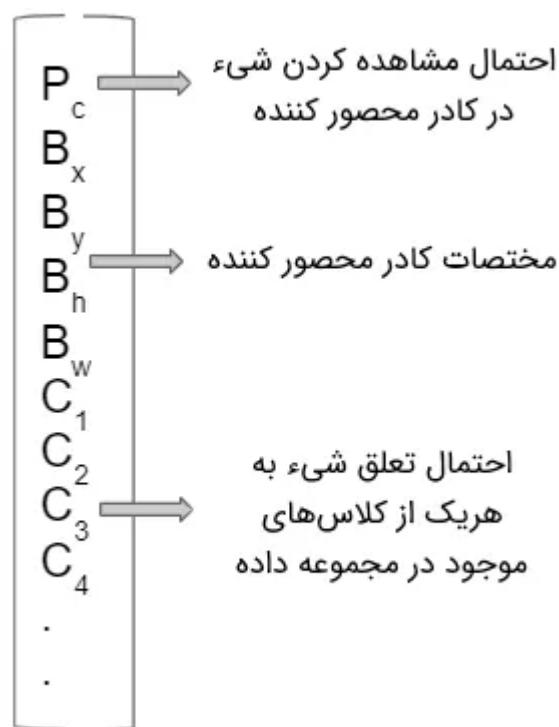
(درون کادرهای محصور کننده هر سلول) به کلاس‌های موجود در مجموعه داده محاسبه می‌شود.

هر کدام از کادرهای محصور کننده، ابعاد و اشکال متفاوتی نسبت به یکدیگر دارند و در اصل ،

برای تشخیص دادن اشیاء گوناگون (با شکل‌ها و ابعاد مختلف) در هر یک از سلول‌های گریدی طراحی

شده‌اند. خروجی الگوریتم YOLO ماتریسی به شکل زیر است؛ به ازاء هر کدام از کادرهای محصور

کننده تعریف شده (در هر کدام از سلول‌های گریدی)، ماتریسی مشابه شکل زیر تولید خواهد شد.



از آنجایی که شبکه YOLO با استفاده از تصاویر ۴۳ کلاس آموزش داده می‌شود، ابعاد ماتریس

خروجی به شکل زیر محاسبه می‌شود.



این ماتریس اطلاعات بسیار مهمی نظیر «احتمال مشاهده شدن یک شیء در هر یک از کادرهای

محصور کننده» (Probabilities of Observing an Object for Each Anchor Box) و احتمال تعلق شیء

شناسایی شده در کارد محصور کننده، به هر کدام از کلاس‌های از پیش تعریف شده را، در اختیار قرار

می‌دهد. برای اینکه کادرهای بدون شیء (کادرهایی که هیچ شیء خاصی در آنها وجود ندارد)، کادرهایی

که شیء شناسایی شده در آنها در هیچ کلاسی دسته‌بندی نمی‌شود یا کادرهایی که شیء شناسایی شده در

آنها با کادرهای دیگر هم‌پوشانی دارند، فیلتر شوند ار دو حد آستانه زیر استفاده می‌شود:

حد آستانه IoU: برای فیلتر کردن کادرهایی به کار می‌رود که یک شیء واحد و یکسان در آنها

شناسایی شده است.

حد آستانه ضریب اطمینان (Confidence): برای فیلتر کردن کادرهایی به کار می‌رond، احتمال تعلق

آنها به کلاس‌های مختلف بسیار پایین است.

به عنوان نمونه، شبکه عصبی از پیش آموزش داده شده که در این مطلب استفاده شده است، چیزی

حدود ۵۰ میلیون پارامتر وزن دارد. یادگیری چنین مدلی در پلتفرم‌های قدرتمند محاسبات ابری نظیر

Google Cloud، حداقل به ۴ الی ۵ روز زمان نیاز دارد. برای اینکه از مفهوم یادگیری انتقال، با موفقیت،

برای آموزش دوباره شبکه عصبی (روی داده‌های جدید) استفاده شود، لازم است که برخی از پارامترها و

تنظیمات مدل به روز رسانی شوند تا مدل آموزش از پیش داده شده بتواند خود را با داده‌های جدید وفق

دهد:

ابعاد تصاویر ورودی: ابعاد تصاویر ورودی در مدل YOLO از پیش آموزش داده شده، برابر با 416×416 است. از آن جایی که تصاویر استفاده شده در این مطلب، ابعاد بزرگتری نسبت به تصاویر استفاده شده در مدل YOLO از پیش آموزش داده شده دارند و همچنین، ابعاد برخی از اشیائی که قرار است شناسایی شوند بسیار کوچک است (نظیر کفش، پرنده و سایر موارد)، منطقی نیست که ابعاد تصاویر جدید تا ابعاد 416×416 کاهش پیدا کنند. به همین خاطر، ابعاد تصاویر جدید برای آموزش دوباره مدل YOLO، برابر با 608×608 در نظر گرفته شده است.

اندازه گرید (Grid): در مدل YOLO از پیش آموزش داده شده، ابعاد سلول‌های گرید برابر با 13×13 است. در این مطلب، ابعاد سلول‌های گرید، جهت آموزش دوباره مدل YOYO، به 19×19 تغییر پیدا کرده است.

لایه خروجی: تعداد کلاس‌ها در مدل YOLO از پیش آموزش داده شده، برابر با ۸۰ است. در حالی که برای آموزش دوباره مدل، از داده‌های ۴۳ کلاس استفاده شده است. بنابراین، لازم است تغییراتی در ابعاد لایه خروجی ایجاد شود تا سیستم بتواند خود را با کلاس‌های جدید، داده‌های آن‌ها و الگوهای موجود در آن‌ها وفق دهد.

برای اینکه بتوان مدل YOYO را روی داده‌های تصویری جدید آموزش داد، نیاز است تا پارامترهای وزن لایه کانولوشن (پیچشی) آخر از دوباره مقدار دهی اولیه شوند؛ چنین کاری به سیستم اجازه می‌دهد تا تصاویر متعلق به کلاس‌های خاص (و جدید) را به درستی دسته‌بندی کند. قطعه کد زیر، برای آموزش دوباره شبکه عصبی YOLO مورد استفاده قرار می‌گیرد.

تابع «هزینه» (Cost) یا «هزینه» (Loss)

در مسائل مرتبط با تشخیص اشیا در تصویر، هدف شناسایی درست کلاس اشیاء، با بیشترین احتمال یا ضریب اطمینان (Confidence) ممکن است. تابع هزینه برای یک سیستم تشخیص اشیا از سه مؤلفه تشکیل شده است:

هزینه دسته‌بندی (Classification Loss): در صورتی که یک شیء در تصویر تشخیص داده شده باشد، این مؤلفه، «مربع خطای» (Squared Error) احتمال شرطی کلاس را محاسبه می‌کند. بنابراین، تابع هزینه تنها در صورتی که یک سلول گردیدی وجود داشته باشد، برای خطای انجام شده در دسته‌بندی، جریمه در نظر می‌گیرد.

هزینه محلی‌سازی (Localization Loss): این مؤلفه، در صورتی که کاردهای محصور کننده مسئول تشخیص شیء در تصویر باشند، مربع خطای مرتبط با «اندازه و مکان کادر محصورسازی مسئول تشخیص شیء» را نسبت به «اندازه و مکان کادر محصور کننده در «پاسخ‌های صحیح مبنای» (Ground Truth)» می‌محاسبه می‌کند. جهت مشخص کردن جریمه خطاهای حاصل از مختصات مکانی کاردهای محصور کننده، از یک پارامتر regularization استفاده می‌شود.

هزینه ضریب اطمینان (Confidence Loss): این مؤلفه، مربع خطای ضریب اطمینان کادر محصور کننده را محاسبه می‌کند. از آنجایی که تمامی کاردهای محصور کننده تعریف شده، در تشخیص یک شیء در تصویر دخالت ندارند، معادله محاسبه هزینه در این مؤلفه به دو بخش تقسیم می‌شود؛ بخش اول، برای کاردهای محصور کننده‌ای که مسئول تشخیص اشیا موجود در تصویر هستند و بخش دوم، برای بقیه کاردهای محصور کننده.

مزیت عمده مدل YOLO نسبت به مدل‌های مشابه تشخیص اشیا این است که خطاهایی توسط این مدل محاسبه می‌شوند که به راحتی توسط «توابع بهینه‌سازی» (Optimization Functions) معروف نظیر «گرادیان کاهشی تصادفی» (Stochastic Gradient Descent | SGD)، گرادیان کاهشی تصادفی با پارامتر Adam و بهینه‌ساز Momentum قابل بهینه‌سازی هستند

۳-۲-۳ شبکه‌ها و مقالات بررسی شده برای مسئلهٔ تخمین عمق

DispNet [8] ۳-۲-۳-۱

A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation

Nikolaus Mayer, Eddy Ilg, P. Häusser, Philipp Fischer, D. Cremers, Alexey Dosovitskiy, Thomas Brox
IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016



Available Data

- Stereo RGB
 - "Clean" and "Final" pass images
 - Lossless PNG and efficient WebP
- Optical flow
 - Left and right view
 - Forwards and backwards in time
- Disparity
 - Left and right view
 - Full float32 precision
- Disparity change
 - Left and right view
 - Forwards and backwards in time
 - No occluded regions
- Segmentation
 - Left and right view
 - Object and Material level
- Motion boundaries
 - Left and right view
 - Forwards and backwards in time
- + Full camera intrinsics / extrinsics

39049 frames

Train your networks for:

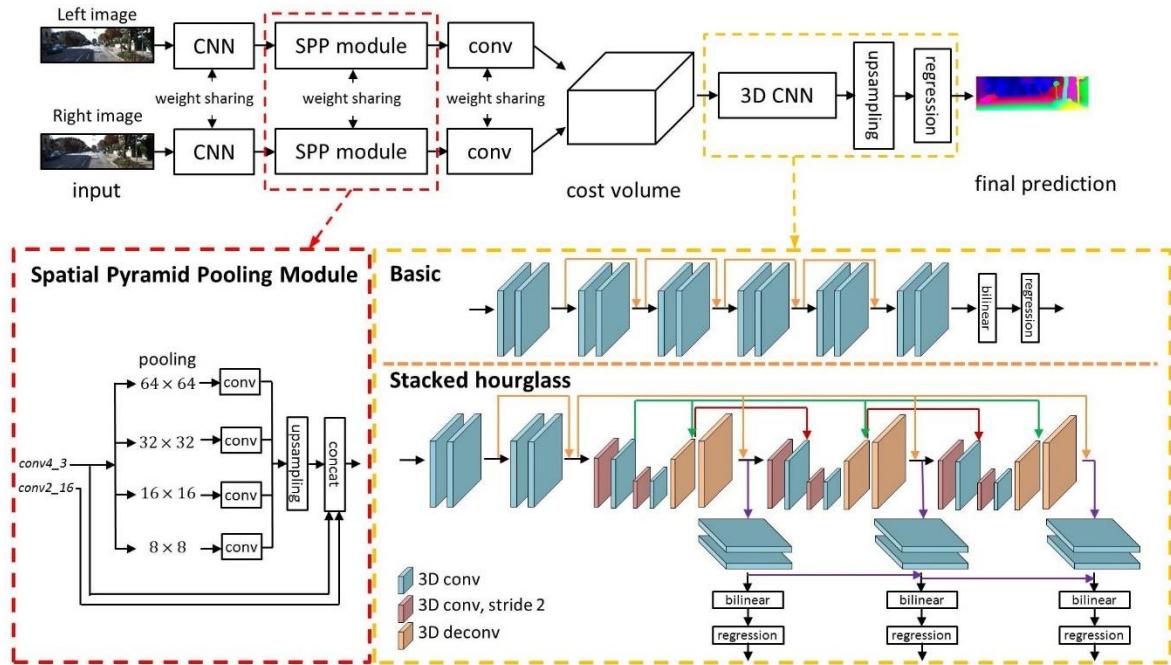
- Optical flow
- Disparity/depth from stereo
- Scene flow
- Depth from single image
- Segmentation
- Motion de-blurring

Get the Datasets:
<http://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html>

در این مقاله دیتای لازم برای انجام تلسک‌هایی که در عکس بالا مشاهده می‌شود فراهم شد.

۳-۲-۳-۲ PSMNet [7]

که در آن مسئله‌ی تخمین عمق به عنوان یک مسئله‌ی ناظارت شده در نظر گرفته شده بود. و روی تصاویر استریوو و چپ و راست یک صحنه انجام شده بود.



۳-۲-۳-۳ AnyNet[3]

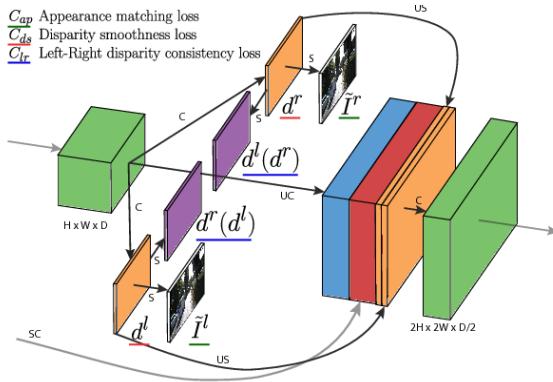
در این ساختار در مراحل مختلف عمق تخمین زده می‌شود و در طول این مراحل می‌توان به مدل query زد و بهترین خروجی اش را در آن لحظه گرفت. در مرحله‌ی اول یک نسخه‌ی downscale شده از تصویر را پردازش می‌کند تا به یک تخمین اولیه با دقت پایین بر سر سپس نتیجه با جزئیات بیشتر و بیشتر و در مراحل بعدی اصلاح می‌شود.

۳-۲-۳-۴ MonoDepth [9]

در این روش برخلاف روش‌های مرسوم از یک تصویر استفاده می‌شود. این روش ناظارت شده نیست ولی روی دیتابیست KITTI نتایج بهتری حتی نسبت به مدل‌های ناظارت شده کسب کرده است.

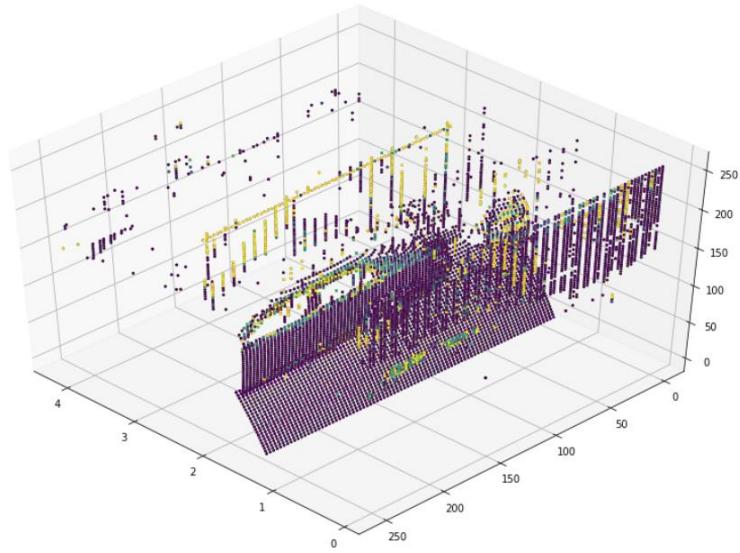
Method	Dataset	Resolution	Abs Rel	Sq Rel	RMSE	RMSE log	$D1-all$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Ours with Deep3D [51]	K	128×384	0.412	16.37	13.693	0.512	66.85	0.690	0.833	0.891
Ours with Deep3Ds [51]	K	128×384	0.151	1.312	6.344	0.239	59.64	0.781	0.931	0.976
Ours without LR	K	256×768	0.110	1.774	5.617	0.188	20.88	0.896	0.962	0.982
Ours	K	256×768	0.098	1.218	5.265	0.175	19.49	0.899	0.964	0.984
Ours	CS	256×512	0.651	9.128	13.770	0.516	93.90	0.064	0.431	0.890
Ours	CS + K	256×512	0.097	1.265	5.243	0.175	18.55	0.904	0.966	0.985
Ours with post-processing	CS + K	256×512	0.090	1.000	4.895	0.164	18.05	0.909	0.969	0.987
Ours Stereo	K	256×768	0.118	3.155	5.746	0.197	10.58	0.931	0.967	0.980

Lower is better
Higher is better



۳-۲-۳-۵ سایر روش ها

هم چنین به عنوان سایر روش ها برای تخمین عمق می توان از ماسک های دو بعدی عمق در هر پیکسل استفاده کرد. مقاله های [1] و [2] مربوط به این موضوع هستند. در این مقاله ها برای هر تصویر حالتی از segmentation را انجام می دهند که تصاویر را بر اساس عمق طبقه بندی می کنند و بدین ترتیب می توان برای هر تصویر با استفاده از این روش طبقه بندی ای از عمق بدست آورد.



شکل ۳-۷ تخمین عمق و نمایش سه بعدی آن روی یک تصویر تست از دیتاست DIODE

می توانیم برای این که حدود خطای سیستم را داشته باشیم و ببینیم که در چه فواصل و عمق هایی داریم خطای بیشتری تولید می کنیم هیستوگرام^۱ خطای خروجی را رسم کنیم.

3-3 جمع‌بندی

در این فصل به بررسی شبکه‌های موجود برای تشخیص اشیاء و همچنین تخمین عمق پرداختیم و با جزئیات هر کدام از آن‌ها تا حد خوبی آشنا شدیم.

histogram^۱

۴ فصل چهارم: پیاده‌سازی عملی شبکه تشخیص همزمان اشیاء و عمق

4-1 مقدمه

با توجه به موارد مطرح شده در فصل‌های قبل، همانطور که بیان شد، میخواهیم به پیاده‌سازی یک شبکه تشخیص اشیاء با قابلیت تخمین عمق هر شئ بپردازیم که این شبکه، از یک زیر شبکه تشخیص اشیاء و همچنین یک زیر شبکه تخمین عمق تشکیل شده است.

4-2 پیاده سازی

تصویر ورودی به شبکه کلی، در ابتدا به شبکه تشخیص اشیاء داده شده و طی آن، اشیاء موجود در تصویر به همراه سایر اطلاعات تکمیلی در مورد آن‌ها، در قالب خروجی شبکه مشخص می‌شوند (مثلاً در شبکه YOLO، نام و شماره کلاس هر شئ، مختصات مربوط به bounding-box و همچنین عدد اطمینان آن شئ به عنوان خروجی مشخص می‌شود). سپس خروجی شبکه تشخیص اشیاء، به یک شبکه

تخمین عمق داده‌می شود تا عمق اشیاء موجود در تصویر مشخص شوند؛ بدین صورت که برای پیکسل‌های موجود در **bounding-box** مربوط به هر شیء، میانگین عمق گرفته شده و به عنوان عمق شیء ذکر می‌شود. در نهایت شبکه قادر به گزارش اشیاء و عمق آن‌ها در تصویر می‌باشد. برای درک بصری، یک تصویر با مشخص کردن اشیاء با **bounding-box** مربوط به خود به همراه میزان اطمینان از تشخیص و عمق هر شیء در خروجی نمایش داده می‌شود.

بر اساس انتخاب‌هایی که برای انتخاب زیرشبکه‌ها انجام داده‌ایم، به بررسی حالت‌های مختلف پیاده‌سازی شبکه کلی می‌پردازیم.

4-2-4 روش اول: آموزش شبکه تخمین عمق

در روش اول تخمین عمق برای هر پیکسل از منبع [11] استفاده کردیم. این ساختار روی DIODE آموزش داده شده بود. به همین دلیل ما دوباره روی NYU، آن را آموزش می‌دهیم. پایپلاین دیتا در این شبکه به صورت زیر می‌باشد:

ورودی یک تصویر RGB می‌باشد که بعد از خواندن `resize` می‌شود. همچنین داده‌ی خروجی یک `depth map` با سایز تصویر ورودی می‌باشد که به ازای هر پیکسل یک عمق به آن نسبت داده شده است.

برای پیاده‌سازی این شبکه سه کلاس یا بلوک تعریف شده که در هر کدام از این کلاس‌ها عملیات خاصی روی تصویر انجام می‌شود. این سه بلوک `Downscale`, `bottleneck` و `upscale` هستند.

این شبکه مبتنی بر ساختار U-net است و از `skip connection`‌ها برای اتصالات لایه‌های upscale به downscaling استفاده کرده است. در ابتدا داده‌ها وارد بلوک `downscaling` می‌شوند که در

آن ها ابعاد فیچر مپ ها با کرنل های پیش فرض $(3,3)$ کاهش داده می شوند. هر لایه از این بلوک به ترتیب $f = [16, 32, 64, 128, 256]$ فیلتر دارد. سپس خروجی کانولوشن ها از تابع leakyReLU با ضریب 0.2 گذشته و بعد batch normalization اعمال می شود. در نهایت نیز maxpooling با کرنل $2,2$) انجام می شود. کلاس downscale block علاوه بر constructor متدهای دیگری به نام call دارد که ورودی را به ساختار construct شده اعمال می کند. در قسمت میانی شبکه یک بلوک bottleneck داریم که در این بلوک دو لایه ای کانولوشن به شبکه اعمال شده. در این بلوک دو لایه ای کانولوشنی با تعداد فیلتر های 256 به خروجی بلوک قبلی اعمال می شود. در طرف دیگر کلاس upscale block را داریم که این کلاس عملیات عکس بلوک downscale را انجام می دهد. به این ترتیب که در ابتدا یک upsampling با کرنل $(3,3)$ و در نهایت اعمال همان تابع فعالیت leakyReLU با ضریب 0.2 و batch normalization با این جا تعداد فیلتر ها به ترتیب 256 و 128 و 64 و 32 و 16 می باشد. در نهایت نیز یک لایه conv $(1,1)$ به شبکه اعمال می شود که صرفا بعد خروجی را حذف می کند. تابع فعالیت این لایه tanh می باشد.

حال بعد از پیاده سازی شبکه مدل و پارامتر های آن را تعریف می کنیم. کلاسی به نام DepthEstimationModel در نظر می گیریم که خروجی آن یک مدل است. این مدل شبکه ای با ساختاری که پیش تر شرح داده شد دارد و پارامتر های آن در ادامه توضیح داده خواهند شد.

٤-٢-١-١ LOSS FUNCTION

برای تابع هزینه سه نوع هزینه در نظر گرفته شده که سعی در مینیمم کردن هر کدام داریم.

SSIM ۴-۲-۱-۲

$$SSIM(I, \hat{I}) = \left(\frac{2\mu_I\mu_{\hat{I}} + C_1}{\mu_I^2 + \mu_{\hat{I}}^2 + C_1} \right)^\alpha \cdot \left(\frac{2\sigma_I\sigma_{\hat{I}} + C_2}{\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2} \right)^\beta \cdot \left(\frac{\sigma_I\sigma_{\hat{I}} + C_3}{\sigma_I\sigma_{\hat{I}} + C_3} \right)^\gamma$$

SSIM یک معیار مقایسه‌ای ساختاری دو تصویر است که براساس ساختار تصاویر طبیعی ارائه شده است. ساختار تصاویر طبیعی به این گونه است که پیکسل‌ها وابستگی زیادی به پیکسل‌های مجاور خود دارند و این وابستگی اطلاعات مهمی را درباره ساختار اشیاء در تصویر در بردارد. با محاسبه SSIM میزان مشابهت ساختاری در همسایگی هر پیکسل جداگانه محاسبه می‌شود.

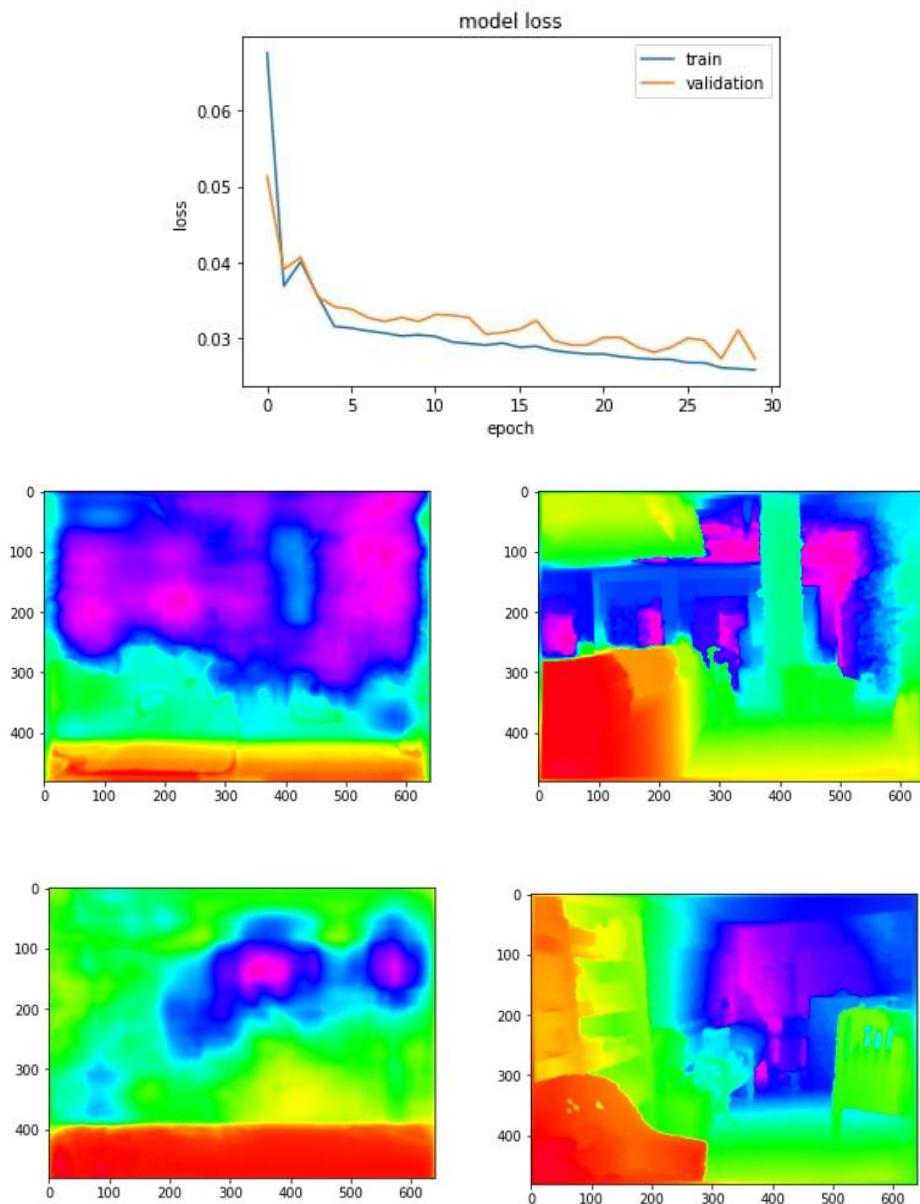
L1 ۴-۲-۱-۳

یکی دیگر از توابع هزینه که خواص جالبی دارد، میانگین قدر مطلق خطای مطلق (Mean Absolute Error) است که به اختصار MAE نیز نامیده می‌شود. این تابع هزینه، به مانند MSE از فاصله بین مقدار پیش‌بینی و واقعی به عنوان معیار استفاده کرده ولی جهت این تفاضل را در نظر نمی‌گیرد. بنابراین در محاسبه خطای MAE فقط میزان فاصله و نه جهت فاصله به کار می‌رود. البته گاهی در مباحث آماری، به این تابع، هزینه L1 نیز گفته می‌شود.

EDGE LOSS ۴-۲-۱-۴

```
depth_smoothness_loss = mean(abs(smoothness_x)) + mean(abs(smoothness_y))
```

به این منظور تابع calculate_loss را نوشتیم که از تصویر خروجی و target گرادیان در راستای y, x می‌گیرد و به نوبی لبه‌های تصاویر depth map را پیدا می‌کند. بعد از ۳۰ اپیک آموزش می‌بینیم که نتایج روی دادگان تست به صورت زیر می‌باشند:



پس متوجه می شویم که نواحی دارند حدودا درست تشخیص داده می شوند اما مشکل در تعیین

لبه هاست که smoothness در تصاویر پیش بینی شده بالاتر از انتظار است. پس سعی می کنیم مقدار

بهینه برای ضریب مربوط به آن در تابع loss smooth نهایی پیدا کنیم. و به loss هزینه بیشتری

اختصاص دهیم. با تغییر دادن ضرایب مربوط به هر loss می توان نتیجه را خیلی تغییر داد.

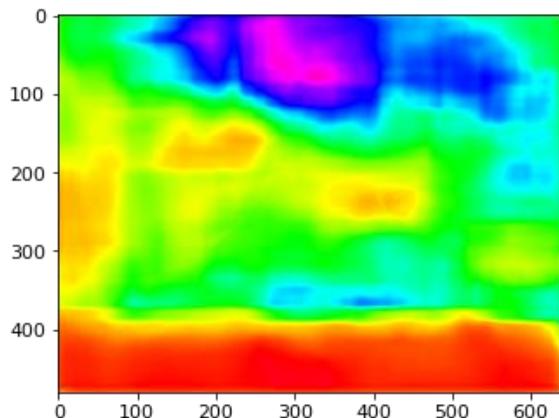
شبکه را با تقسیم داده ها به train, test و ولیدیشن اموزش دادیم.

X_train, X_test, Y_train, Y_test = train_test_split (images, depths, test_size=0.2)

```
(1159, 480, 640, 3) (290, 480, 640, 3) (1159, 480, 640) (290, 480, 640)
```

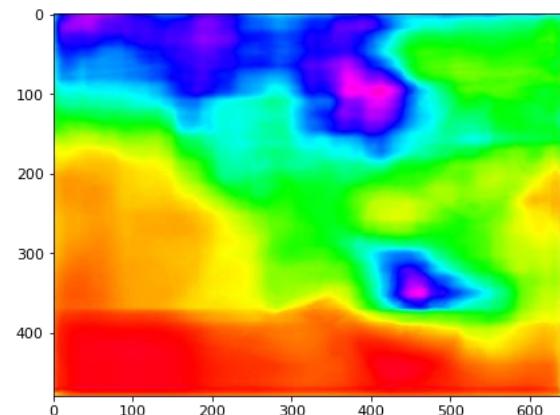
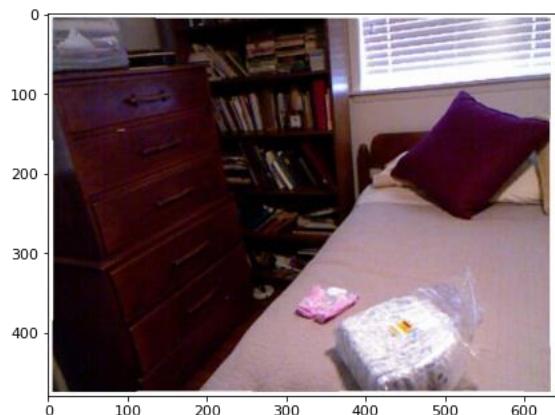
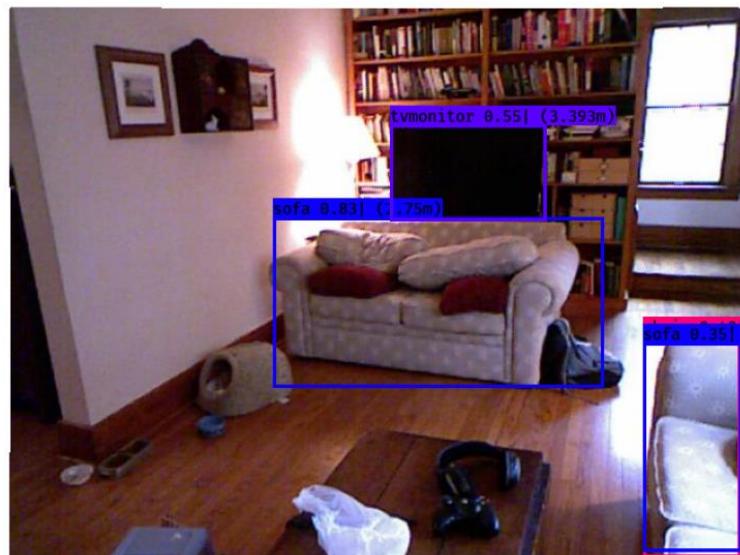
۱۱۵۹ تا از داده ها به داده های آموزش ۲۹۰ تا به تست اختصاص یافت.

بعد از آموزش، نتایج را روی برخی از دادگان **تست** مشاهده می کنیم:

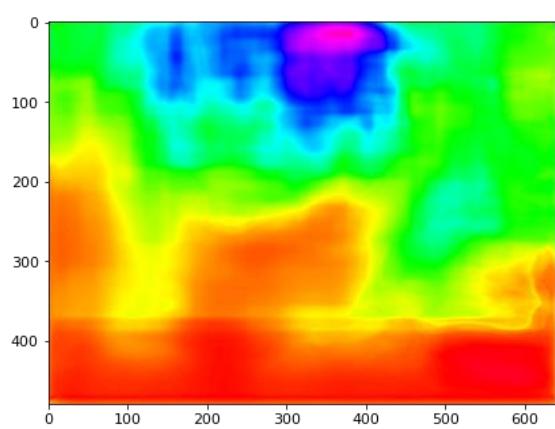
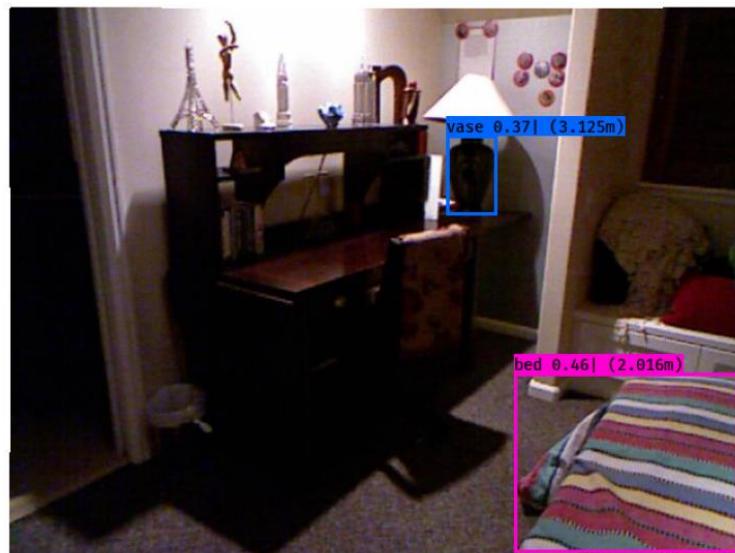
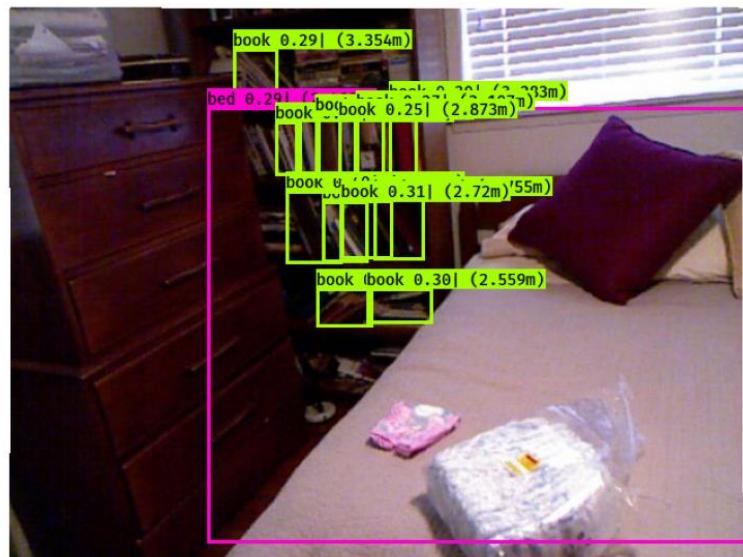


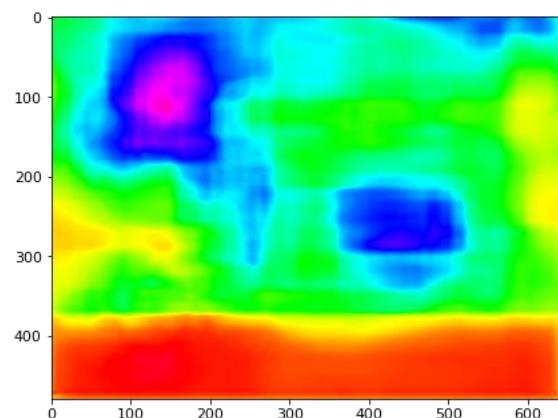
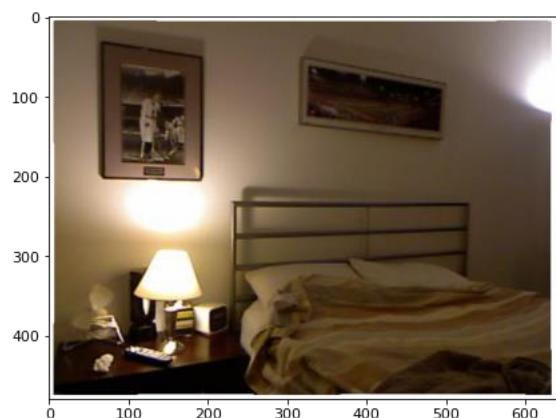
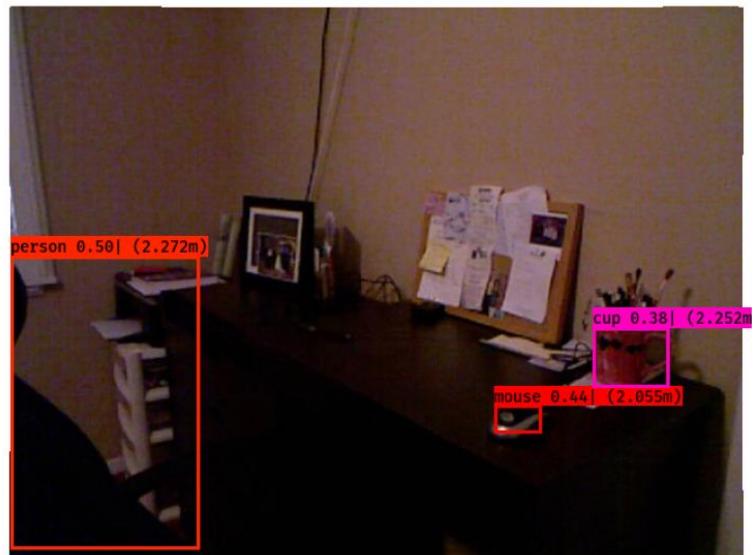
	xmin	ymin	xmax	ymax	confidence	class	name
0	230.859543	184.375519	512.695862	329.444305	0.833860	57	couch
1	330.894012	105.428986	464.425995	186.863129	0.551312	62	tv
2	547.402710	284.841370	630.286133	474.212891	0.482500	56	chair
3	547.300781	291.299438	631.155762	469.366760	0.353693	57	couch

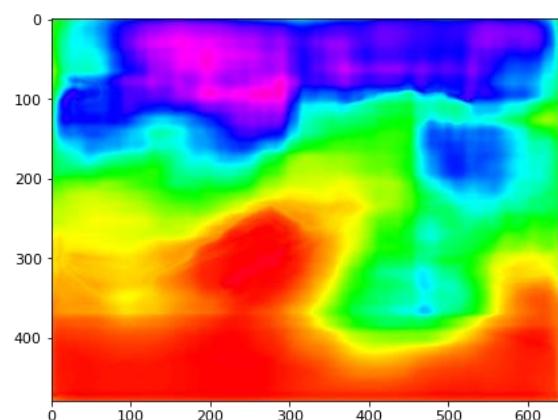
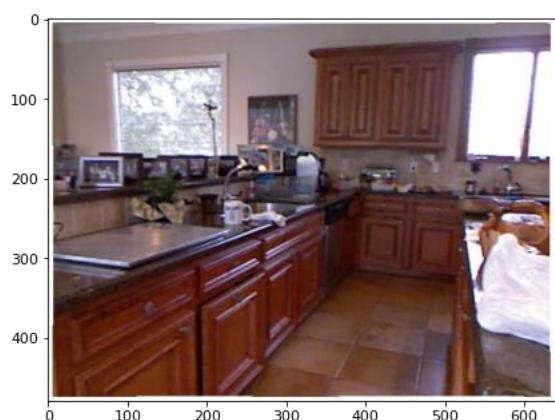
	xmin	ymin	xmax	ymax	confidence	class	name	depth
0	230.859543	184.375519	512.695862	329.444305	0.833860	57	couch	2.750
1	330.894012	105.428986	464.425995	186.863129	0.551312	62	tv	3.393
2	547.402710	284.841370	630.286133	474.212891	0.482500	56	chair	2.430
3	547.300781	291.299438	631.155762	469.366760	0.353693	57	couch	2.440

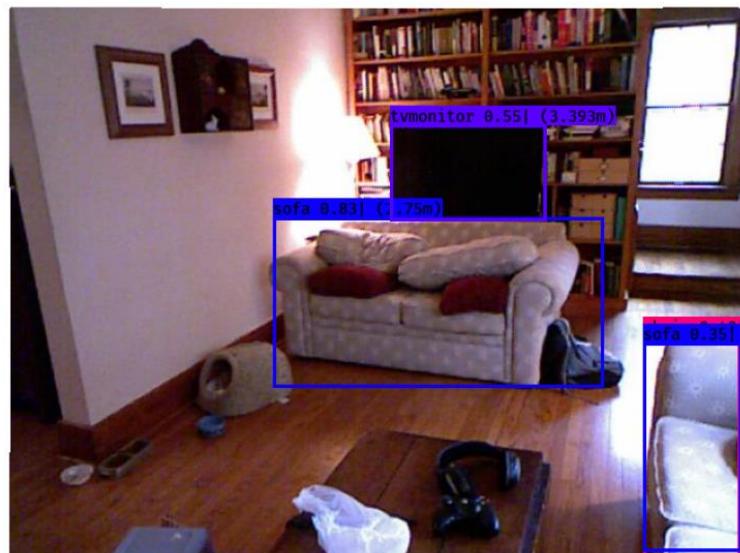
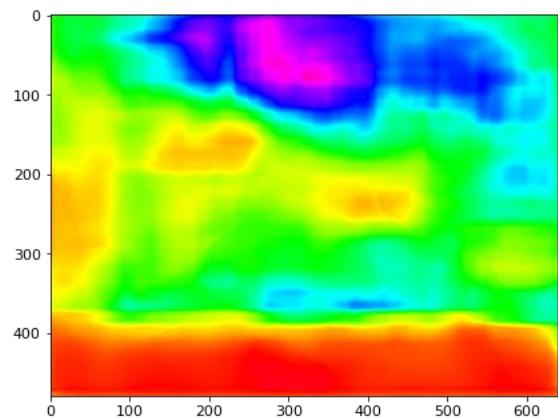
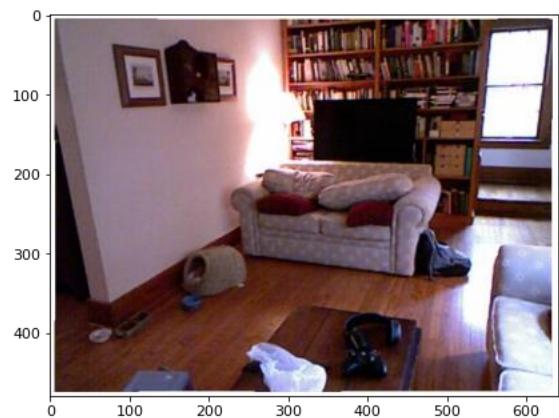


	xmin	ymin	xmax	ymax	confidence	class	name	depth
0	273.208313	169.881424	311.902588	221.692490	0.517434	73	book	2.695
1	242.077393	160.822937	290.000610	223.135406	0.400619	73	book	2.736
2	267.898499	243.662567	315.698120	277.414581	0.378825	73	book	2.453
3	250.095032	99.894241	270.832764	148.822372	0.362942	73	book	2.855
4	317.255310	164.551682	359.911499	220.342300	0.326428	73	book	2.755
5	288.722198	169.333298	333.362396	219.378860	0.309066	73	book	2.720
6	329.895752	84.483261	354.349854	147.020737	0.304980	73	book	3.283
7	311.455902	243.618698	366.893585	274.463348	0.296485	73	book	2.559
8	197.238113	40.284401	234.953873	83.659966	0.294685	73	book	3.354
9	174.838608	89.571457	637.029785	460.588806	0.288448	59	bed	2.468
10	232.845612	101.688004	255.387756	147.995499	0.280421	73	book	2.895
11	267.175476	95.272980	289.398926	149.929260	0.280339	73	book	2.823
12	302.108673	92.066177	326.221405	146.814880	0.272504	73	book	3.107
13	287.127716	98.534454	301.510101	148.659622	0.251443	73	book	2.873









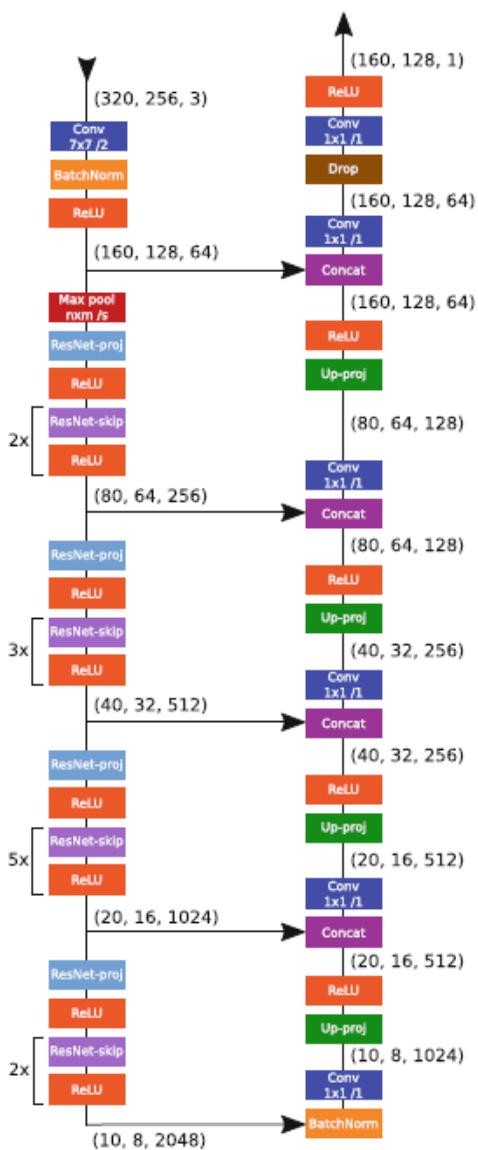
4-2-4 روش دوم: استفاده از شبکه تخمین عمق از پیش آموزش داده شده به همراه تشخیص

YOLOv5 اشیاء با

علاوه بر آموزش دادن یک مدل تعیین عمق از ابتدا، سعی کردیم در این قسمت همین مراحل را با

مدل تعیین عمق pretrained و yolo_v5 انجام دهیم. مدل تعیین عمقی که به این منظور انتخاب شد

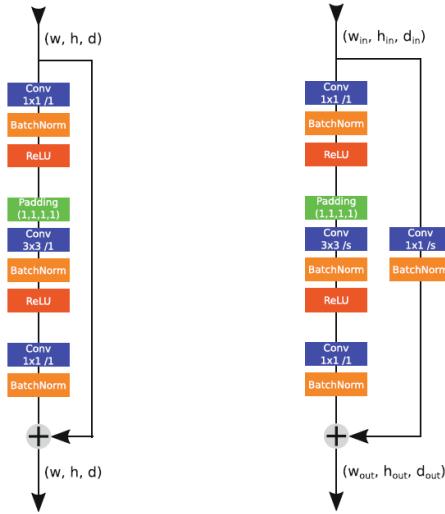
برگرفته از مقاله‌ی [13] است



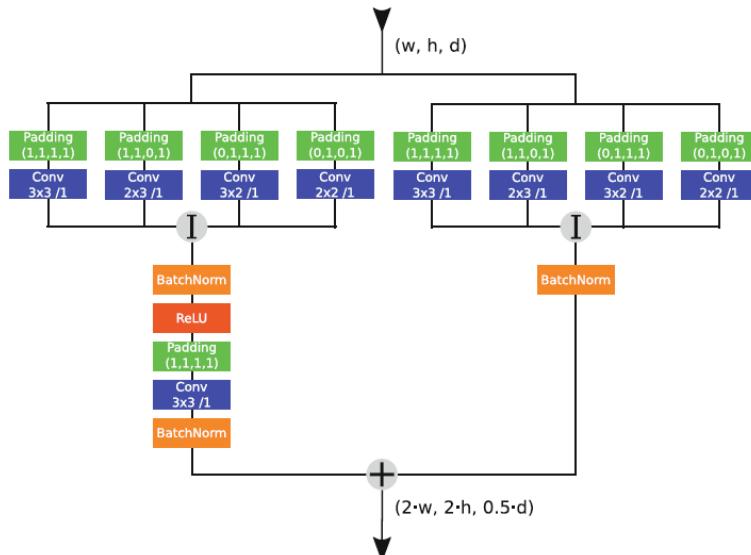
در این مقاله هدف این بود که با استفاده از دوربین‌های کمتر فرایند تشخیص عمق تا حد ممکن راحت‌تر شود.

ساختاری که در این جا استفاده شد مشابه قبل یک ساختار انکودر دیکودری به همراه skip connection هاست. و به همین دلیل اسمش گذاشته شده که هم از ساختار Hybrid U-net استفاده کرده و هم از ایده‌ی skip connection در بهره برده. همچنین قسمت انکودر این مدل بر اساس ResNet50 پیاده سازی شده است. ساختار شبکه در سمت چپ آورده شده است.

Fig. 4. The illustration of our network architecture.



۱. (a) Residual-skip building block, (b) Residual-projection building block



یک بلوک up projection

ساختار مدل و وزن های آن در لینک^۱ زیر موجود هستند.

در این مقاله به جای loss mse از BerHu loss function استفاده شده است. این تابع

هزینه وزن بیشتری را به پیکسل هایی که دارای مانده‌ی بیشتری هستند اختصاص می‌دهد چرا که به

این پیکسل ها L2 اعمال می کند و پیکسل هایی که مانده ای کمتری دارند در یادگیری اثر بیشتری دارند چون که روی آن ها از $\text{loss} = \text{L1}$ استفاده می شود. این تابع در ادامه آورده شده است:

$$\mathcal{B}(y, \tilde{y}) = \begin{cases} |y - \tilde{y}|, & \text{where } |y - \tilde{y}| \leq c. \\ \frac{(y - \tilde{y})^2 + c^2}{2c}, & \text{otherwise.} \end{cases}$$

نتایج ارزیابی معیار های رایج تخمین عمق در این مقاله در ادامه آورده شده است.

Table 1. Quantitative comparison with state-of-the-art CNN based methods on the NYU Depth v2 dataset. In the case of RMSE, REL, and Log_{10} , lower is better. For the δ_i accuracies, higher is better.

	RMSE	REL	Log_{10}	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. [3]	0.907	0.215	–	0.611	0.887	0.971
Wang et al. [28]	0.745	0.220	–	0.605	0.890	0.970
Eigen and Fergus [2]	0.641	0.158	–	0.769	0.950	0.988
Laina et al. [9]	0.597	0.137	0.059	0.818	0.955	0.988
Proposed	0.593	0.130	0.057	0.833	0.960	0.989

یک خروجی نمونه از عملکرد شبکه را می توانیم در تصویر زیر بینیم:



Detected bounding boxes



Calculated depths

همانطور که مشاهده می‌کنیم شبکه ما به خوبی توانسته که اشیاء موجود در تصویر را شناسایی و عمق آنها را تشخیص دهد.

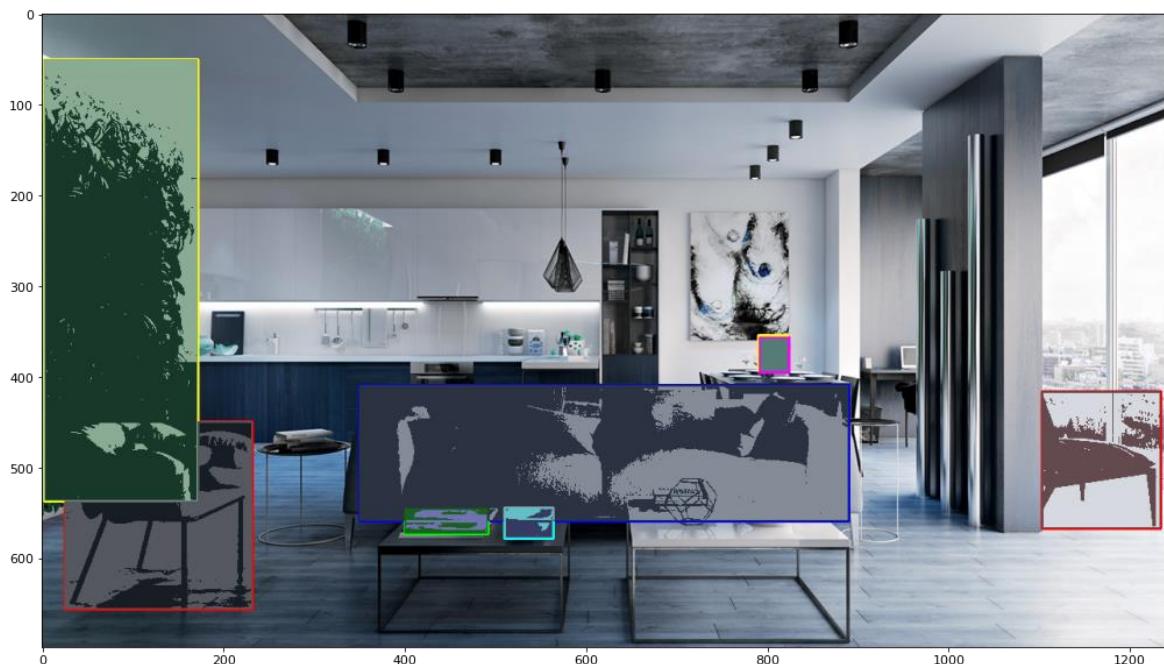
ایده‌ی دیگری که پیاده سازی کردیم:

برای اینکه بتوانیم نتیجه را دقیق‌تر کنیم، باید توجه خود را به سمت عوامل ایجاد خطأ معطوف کنیم. یکی از دلایلی که موجب ایجاد خطأ در گزارش عمق هر شئ می‌شود، این است که ما بر روی عمق تمام پیکسل‌های موجود در bounding-box هر شئ میانگین گرفته‌ایم که طبعاً این کار برای بعضی از اشیائی که تنها بخشی از باکس خود را پر کرده‌اند ایجاد خطأ کرده می‌کند.

برای رفع این مشکل در این قسمت سعی کردہ‌ایم که محدوده‌ی به دست آمده برای هر Object را با k-means خوش‌بندی کنیم و صرفاً مختصات مربوط به خوش‌بندی شئ را برای تعیین عمق در نظر بگیریم؛ به عبارتی، برای این قسمت فرض کردیم که شئ تشخیص داده شده بخش اعظم bounding

box را می پوشاند. و شی را تقریبا همگن فرض کردیم و گفتیم اگر تصویر را به دو کلاستر تقسیم کنیم احتمالا یکی foreground و دیگری background خواهد بود.

نتیجه اعمال خوشبندی و تعیین پیکسل های غالب (پیکسل هایی که به احتمال بیشتر مربوط به شی هستند و جزئی از پس زمینه نیستند) و سپس تخمین عمق از روی این پیکسل ه را می توانیم در شکل های زیر ببینیم:



Masked image after clustering on the bounding boxes



Producing more accurate results using masked depth calculation

4-3-4 روش سوم: استفاده از مدل از پیشآموزش داده شدهی عمق و مدل تشخیص اشیاء

YOLOV2

این بار باز هم از شبکه pre-trained برای تشخیص عمق استفاده می‌کنیم؛ با این تفاوت که این‌بار، از نسخه دوم شبکه YOLOv2 یعنی YOLOv2 بهره می‌گیریم. ساختار این شبکه را به طور کامل در فصل سوم بررسی کردیم. در کد، با پیاده‌سازی شبکه DarkNet و وزن‌های این مدل، شبکه YOLOv2 را بر اساس مقاله آن به کار می‌گیریم. به منظور پیاده‌سازی مفاهیم مطرح شده در مقاله، تعداد تابع کمکی برای محاسبه IoU و Non-Max Suppression و در نهایت رسم Bounding-Box ها تعریف شده که در کد موجود می‌باشد.

پس از تشخیص اشیاء توسط شبکه YOLOv2 و دادن نتایج آن به شبکه تخمین عمق (که در نهایت با میانگین‌گیری بین عمق تمامی پیکسل‌های باکس هر شیء و با بهره‌گیری از تابع DEPTH_Calc به

محاسبه عمق هر شیء میپردازد)، نتایج رو روی تصویر اولیه رسم میکنیم. یک خروجی نمونه به صورت

زیر است:



با مقایسه تصویر فوق با تصویری که از خروجی شبکه متصل شده شبکه های YOLOv5 و تخمین عمق در مرحله قبل به دست آمد، مشاهده میکنیم که این دو ساختار در تشخیص چند شیء محدود با یکدیگر تفاوت دارند و گرنه به طور تقریبی میتوان گفت که عملکرد مشابهی دارند.

4-3 جمع‌بندی

در این فصل به پیاده‌سازی شبکه کلی (شبکه تشخیص همزمان شیء و عمق) با بهره‌گیری از دو زیرشبکه تخمین عمق و تشخیص اشیاء با استفاده از سه حالت آموزش شبکه تشخیص عمق و یا استفاده از یک ساختار از پیشآموزش داده شده و ساختارهای YOLO برای تشخیص اشیاء پرداختیم و نتایج مدل در هر سه روش را با یکدیگر دیدیم.

۵ فصل پنجم: روش‌های ارزیابی عملکرد شبکه

5-1 مقدمه

حال که شبکه را پیاده سازی کردیم و نتایج آن را دیدیم، خوب است که یک بررسی روی معیارهای ارزیابی شبکه داشته باشیم. شبکه ما از اتصال دو شبکه تخمین عمق و تشخیص شیء بدست آمد. حال سوال این است که معیارهای ارزیابی حداگانه هر یک از این شبکه‌ها چه هستند؟ معیار ارزیابی شبکه کلی، که از اتصال دو شبکه فوق به دست آمده را چگونه می‌توان تعریف کرد؟ در این فصل می‌خواهیم به این سوالات پاسخ دهیم.

5-2 ارزیابی شبکه تشخیص اشیاء

Object Detection Accuracy برای این کار استفاده کرد. برای مسلمانی توان از معیاری مثل

معیارهای ارزیابی دیگر وجود دارد.

در لینک های زیر دقیق yoloV5 روی دیتابست COCO محاسبه شده است.

<https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb#scrollTo=zR9ZbuQCH7FX>

<https://github.com/ultralytics/yolov5>

که نتایج آن در عکس زیر آورده شده است.

```
Class      Images     Labels      P      R      mAP@.5  mAP@.5:.95: 100%
      all       5000    36335    0.729    0.63    0.683    0.496
Speed: 0.1ms pre-process, 4.9ms inference, 1.9ms NMS per image at shape (32, 3, 640, 640)

Evaluating pycocotools mAP... saving runs/val/exp/yolov5x_predictions.json...
loading annotations into memory...
Done (t=0.46s)
creating index...
index created!
Loading and preparing results...
DONE (t=5.15s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=90.39s).
Accumulating evaluation results...
DONE (t=14.54s).

Average Precision (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.507
Average Precision (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.689
Average Precision (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.552
Average Precision (AP) @[ IoU=0.50:0.95 | area= small  | maxDets=100 ] = 0.345
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.559
Average Precision (AP) @[ IoU=0.50:0.95 | area= large  | maxDets=100 ] = 0.652
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 1 ] = 0.381
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.630
Average Recall    (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.682
Average Recall    (AR) @[ IoU=0.50:0.95 | area= small  | maxDets=100 ] = 0.526
Average Recall    (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.732
Average Recall    (AR) @[ IoU=0.50:0.95 | area= large  | maxDets=100 ] = 0.829
```

Pretrained Checkpoints

Model	size (pixels)	mAP ^{val} 0.5:0.95	mAP ^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.4	46.0	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.2	56.0	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.2	63.9	224	8.2	1.7	21.2	49.0
YOLOv5l	640	48.8	67.2	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	34.0	50.7	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.5	63.0	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.0	69.0	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.6	71.6	1784	15.8	10.5	76.7	111.4
YOLOv5x6 + TTA	1280 1536	54.7 55.4	72.4 72.3	3136	26.2	19.4	140.7	209.8

در مقاله اصلی معیار ap ذکر شده که در اینجا آورده شده است.

Mean Average Precision 5-2-1

mAP . یکی از رایج‌ترین معیارهای ارزیابی، mean Average Precision یا به اختصار mAP هست.

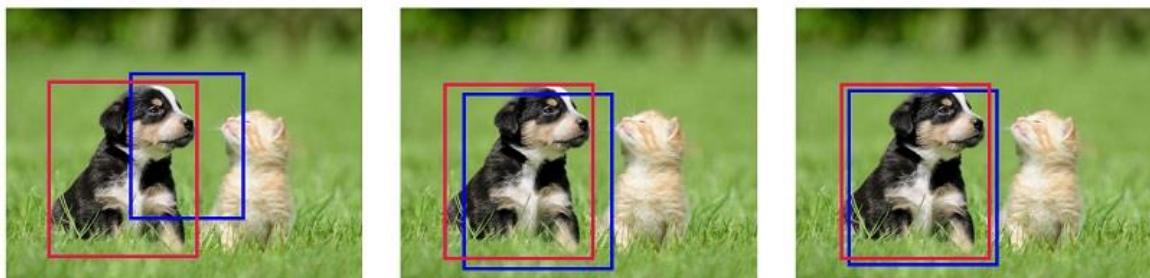
برمبنای مقایسه چارچوب پیش‌بینی و چارچوب هدف محاسبه می‌شود. منطقی هست که دو چارچوب

پیش‌بینی و هدف را باهم مقایسه کنیم تا بینیم چقدر به هم نزدیک هستند.

به تصاویر نمونه زیر نگاه کنید؛ چارچوب قرمز، هدف و آبی معادل پیش‌بینی برای سگ است. می‌توانید

بینید که بعضی‌ها خیلی خوب روی هم منطبق شده‌اند. اما تعدادی هم خیلی از هم فاصله دارند.

■ Target
■ Prediction



برای این که بتوانیم مقایسه‌ی بالا را به صورت عددی انجام دهیم از IoU استفاده می‌کنیم که در درس به آن اشاره شد.

$$IoU = \frac{\text{Intersection}}{\text{Union}} = \frac{I}{U}$$

$$0 \leq IoU \leq 1$$

این پارامتر بیان می‌کند که نسبت اشتراک به اجتماع دو چارچوب چقدر است. معیار mAP بر اساس این پارامتر به دست می‌آید. برای محاسبه mAP ، ابتدا نمودار Recall-Precision براساس IoU رسم می‌شود. مساحت زیر سطح این نمودار برابر با mAP خواهد بود. ین معیار، متوسط بیشینه مقادیر دقت (Precision) به ازاء مقادیر صحت (Recall) متفاوت را اندازه‌گیری می‌کند (برای تمامی مقادیر حد آستانه IoU).

در این پروژه برای این که معیار mAP را بسنجیم از کتابخانه‌ی COCO API استفاده می‌کنیم.

5-3 ارزیابی شبکه تخمین عمق

در DenseDepth که لینک^۱ مربوط به ریپازیتوری آن آورده شده است [14]

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	$rel \downarrow$	$rms \downarrow$	$log_{10} \downarrow$
Eigen et al. [7]	0.769	0.950	0.988	0.158	0.641	-
Laina et al. [23]	0.811	0.953	0.988	0.127	0.573	0.055
MS-CRF [37]	0.811	0.954	0.987	0.121	0.586	0.052
Hao et al. [14]	0.841	0.966	0.991	0.127	0.555	0.053
Fu et al. [9]	0.828	0.965	0.992	0.115	0.509	0.051
Ours	0.846	0.974	0.994	0.123	0.465	0.053
Ours (scaled)	0.895	0.980	0.996	0.103	0.390	0.043

در منبعی که ما برای تشخیص عمق از آن استفاده کردیم معیار های زیر مورد استفاده قرار گرفته بود.

Evaluation

- Quantitative results:

REL	RMSE	Log10	δ_1	δ_2	δ_3
0.130	0.593	0.057	0.833	0.960	0.989

در پژوهش های دیگر نیز از معیار های مشابهی استفاده شده است که در ادامه به آن ها اشاره می شود:

Root Mean Squared Error: $\sqrt{\frac{1}{n} \sum_{i=0}^n (y_i - \tilde{y}_i)^2}$

Relative Absolute Error: $\frac{1}{n} \sum_{i=0}^n \frac{|y_i - \tilde{y}_i|}{\tilde{y}_i}$

Mean Log₁₀ Error: $\frac{1}{n} \sum_{i=0}^n |\log_{10}(y_i) - \log_{10}(\tilde{y}_i)|$

% of pixels where $\max(\frac{y_i}{\tilde{y}_i}, \frac{\tilde{y}_i}{y_i}) = \delta < \text{threshold}$

[https://github.com/ialhashim/DenseDepth'](https://github.com/ialhashim/DenseDepth)

- **Threshold:** % of y such that $\max\left(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}\right) = \sigma < thr$
- **Absolute rel. diff.:** $rel = \frac{1}{T} \sum_{i,j} |y_{i,j} - y_{i,j}^*| / y_{i,j}^*$
- **Squared rel. diff.:** $srel = \frac{1}{T} \sum_{i,j} |y_{i,j} - y_{i,j}^*|^2 / y_{i,j}^*$
 - **RMS (linear):** $RMS = \sqrt{\frac{1}{T} \sum_{i,j} |y_{i,j} - y_{i,j}^*|^2}$
 - **RMS (log):** $\log_{10} = \sqrt{\frac{1}{T} \sum_{i,j} |\log y_{i,j} - \log y_{i,j}^*|^2}$

- average relative error (rel): $\frac{1}{n} \sum_p^n \frac{|y_p - \hat{y}_p|}{y_p};$
- root mean squared error (rms): $\sqrt{\frac{1}{n} \sum_p^n (y_p - \hat{y}_p)^2};$
- average (\log_{10}) error: $\frac{1}{n} \sum_p^n |\log_{10}(y_p) - \log_{10}(\hat{y}_p)|;$
- threshold accuracy (δ_i): % of y_p s.t. $\max\left(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}\right) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3;$

5-4 ترکیب شبکه‌ها

یک راه ترکیب دو شبکه این است که میانگین عمق های تشخیص داده شده در چارچوب

ها رو پیدا کنیم.

راه دیگر این است که از رابطه‌ی زیر استفاده کنیم. که در مقاله‌ی ۱ در بخش بعدی آمد

است

$$disp_{mean}^{object} = \frac{1}{0.4w \times 0.4h} \times \sum_{cx=0.2w}^{cx+0.2w} \sum_{cy=0.2h}^{cy+0.2h} disp(i, j) \quad (3)$$

and

$$disp_{median}^{object} = median\{disp(i, j) | (i, j) \in BB\} \quad (4)$$

طبق این فرمول می توانیم از کل عمق های داخل چارچوب median بگیریم و آن را به عنوان عمق نهایی گزارش کنیم. و یا می توانیم با فرض این که شیء در میان باندینگ باکس قرار دارد از پیکسل های مرکزی میانگین بگیریم.

5-5 سایر مقالات بررسی شده

در مقاله‌ی [16] به کمک نماهای مختلف از یک تصویر توانستند عمق و پوزیشن اشیاء موجود در تصویر را تخمین بزنند. در این مقاله برای ارزیابی عملکرد از معیار Average Precision استفاده شده است. هم‌معیاری است که در stereo R-CNN به کار رفته که برای این که بتوانند کارایی را از نظر association حساب کنند مورد استفاده قرار گرفته بود.

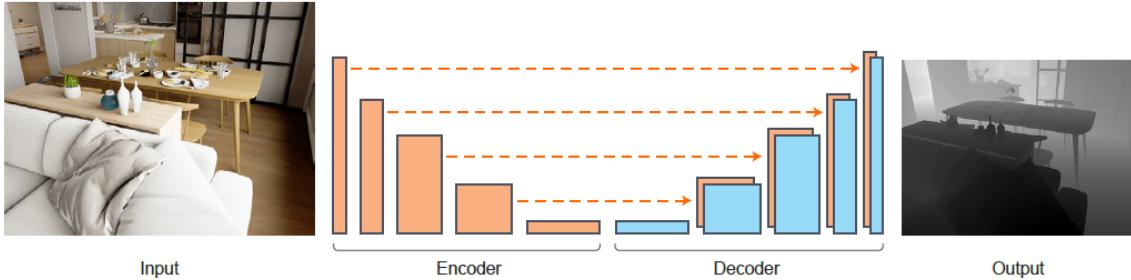
Average precision (in %) of detection, evaluated on the KITTI evaluation set

Method	Setting	AP _{left}			AP _{stereo}			Runtime(s)
		Easy	Mode	Hard	Easy	Mode	Hard	
Faster R-CNN [30]	-	99.23	98.40	90.88	-	-	-	0.082
MFFD [44]	-	91.16	84.01	72.43	-	-	-	0.005
YOLOv3 [37]	-	95.96	95.51	88.26	-	-	-	0.031
SSD [5]	-	98.78	96.06	88.49	-	-	-	0.027
Disparity Detector	strategy stereo [19] anchor pair (ours)	98.37 98.55	95.54 96.00	85.93 88.62	93.17 93.69	90.37 91.87	81.15 83.14	0.027 0.027

که جزئیات مربوط به روش محاسبه‌ی عمق چارچوب‌ها در این مقاله به عنوان ایده مورد استفاده قرار گرفته است.

مقاله‌ی بعدی مقاله‌ی [14] است. در این مقاله از transfer learning برای آموزش شبکه استفاده شده است. و همچنین ساختار شبکه encoder-decoder بوده. در واقع از شبکه‌های از پیش آموزش

دیده برای استخراج ویژگی استفاده کردند. مزیت این شبکه پارامترهای کمتر و نیاز به iteration های کمتر برای آموزش است.



ساختاری که این مقاله ازش استفاده کرده مشابه ساختاری هست که ما آموزش دادیم. یعنی یک ساختار انکودر دیکودری با pretrained denseNet-169 از 169 skip connection. که قسمت انکودر آن از imageNet ترین شده است. سپس در قسمت decoder با concatenate کردن خروجی بلوک های انکودر (skip connection) و خروجی upsampling ها عمل دیکودینگ انجام می شود. در قسمت دیکودر با یک لایه ی کانولوشنی با کرنل 1×1 شروع می کنیم و متوالیا با کرنل های 3×3 upsample می کنیم تا در نهایت به سایز نصف تصویر اصلی برسیم و هنگام تست ان را یک بار دیگر upsample می کنیم که سایز تصویر نهایی با ورودی یکی شود.

در این مقاله معیاری که به عنوان loss انتخاب شده کمینه کردن اختلاف عمق های پیش بینی شده و عمق های اصلی است. در حالی که دارد اعوجاج هایی که در اثر جزئیات تصویر و لبه ها و مولفه های فرکанс بالا ایجاد می شود را هم برایشان پنالتی در نظر می گیرد.

$$L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y}).$$

این مشابه همان معیاری است که ما برای آموزش شبکه ی تشخیص عمق استفاده کردیم. چرا که ترم اول آن یک تجمعی از نرم L1 روی تک تک پیکسل هاست که فورس می کند که پیکسل های target و prediction مقادیر مشابه داشته باشند

$$L_{depth}(y, \hat{y}) = \frac{1}{n} \sum_p^n |y_p - \hat{y}_p|.$$

هم چنین ترم دوم loss همان قیدی را اعمال می کند که روی گرادیان تصاویر پیش تر بیان کردیم.

در اینجا سعی می کنیم لبه ها را مشابه کنیم به این ترتیب که اختلاف گرادیان ها را می نیمم می

کنیم.

ترم سوم SSIM است که پیش تر در مورد آن توضیحات لازمه داده شد. و غالبا برای سنجش شباهت ساختاری تصاویر بازسازی شده نسبت به تصویر اصلی مورد استفاده فرار می گیرد. از آن جایی که کران بالای این پارامتر ۱ است و ما می خواهیم آن را به فرم loss در بیاوریم آن را حول ۰.۵ قرینه می کنیم که هر جا دو تصویر کاملاً شبیه بودند loss مربوط به آن صفر شود.

$$L_{SSIM}(y, \hat{y}) = \frac{1 - SSIM(y, \hat{y})}{2}.$$

مشکل ذاتی این معیار ها این است که وقتی که عمق پیکسل های GT خیلی زیاد باشد این پارامتر ها هم زیاد می شوند. به همین منظور برای این که این مشکل حل شود از معکوس عمق استفاده می کنیم.

و ماکزیمم عمق داده ها را به عمق هر پیکسل تقسیم می کنیم. در دیتابست NYU این عمق برابر با

۱۰ می باشد و با اعمال این تبدیل به پیکسل ها داریم :

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	$rel \downarrow$	$rms \downarrow$	$log_{10} \downarrow$
Eigen et al. [6]	0.769	0.950	0.988	0.158	0.641	-
Laina et al. [23]	0.811	0.953	0.988	0.127	0.573	0.055
MS-CRF [37]	0.811	0.954	0.987	0.121	0.586	0.052
Hao et al. [14]	0.841	0.966	0.991	0.127	0.555	0.053
Fu et al. [9]	0.828	0.965	0.992	0.115	0.509	0.051
Ours	0.846	0.974	0.994	0.123	0.465	0.053
Ours (scaled)	0.895	0.980	0.996	0.103	0.390	0.043

5- معیار ارزیابی شبکه‌ی joint

همان طور که گفتیم دقیقی که برای شبکه تشخیص اشیا وجود داره از جنس AP هست. خطاب برای تخمین عمق هم از جنس MSE یا MAE و معادلهای دیگر آنها هست. می توانیم از معکوس AP به عنوان معیار Loss یا خطاب استفاده کنیم. با توجه به این که AP سطح زیر نمودار است مقدار صفر ندارد پس معکوس کردنش مانعی ندارد.

در مقالات دیگر joint برای تشخیص اشیا و تخمین عمق هم، یک دقت مربوط به شبکه تشخیص اشیا گزارش شده و خطابی که مربوط به تخمین عمق هست و تنها برای bounding box هایی که پیدا شدن محاسبه شده و میانگین گرفته می شود.

نکته‌ای که باید به آن توجه کرد این است که ما در پیاده‌سازی شبکه کلی، هر یک از دو شبکه تخمین عمق و تشخیص شی را جداگانه آموزش دادیم و در نهایت با پشت سر هم قرار دادن نتایج این دو شبکه، توانستیم خواسته مسئله را برآورده کنیم. حال حالتی را در نظر بگیریم که بتوانیم شبکه کلی را از آموزش همزمان این دو شبکه به دست آوریم. در این حالت اصطلاحاً با مسئله‌ی MultiTask و MultiModal

مواجه می شویم که به حل مسئله بهینه سازی Multi-objective منجر می شود که در آن باید به طور همزمان،

بیش از یک تابع هدف بهینه شود. در مقاله [15] الگوریتم زیر برای حل مسئله بهینه سازی ارائه شده:

Algorithm 2 Update Equations for MTL

```

1: for  $t = 1$  to  $T$  do
2:    $\theta^t = \theta^t - \eta \nabla_{\theta^t} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t)$             $\triangleright$  Gradient descent on task-specific parameters
3: end for
4:  $\alpha^1, \dots, \alpha^T = \text{FRANKWOLFESOLVER}(\theta)$             $\triangleright$  Solve (3) to find a common descent direction
5:  $\theta^{sh} = \theta^{sh} - \eta \sum_{t=1}^T \alpha^t \nabla_{\theta^{sh}} \hat{\mathcal{L}}^t(\theta^{sh}, \theta^t)$             $\triangleright$  Gradient descent on shared parameters

6: procedure FRANKWOLFESOLVER( $\theta$ )
7:   Initialize  $\alpha = (\alpha^1, \dots, \alpha^T) = (\frac{1}{T}, \dots, \frac{1}{T})$ 
8:   Precompute  $M$  st.  $M_{i,j} = (\nabla_{\theta^{sh}} \hat{\mathcal{L}}^i(\theta^{sh}, \theta^j))^T (\nabla_{\theta^{sh}} \hat{\mathcal{L}}^j(\theta^{sh}, \theta^j))$ 
9:   repeat
10:     $\hat{t} = \arg \min_r \sum_t \alpha^t M_{rt}$ 
11:     $\hat{\gamma} = \arg \min_\gamma ((1-\gamma)\alpha + \gamma e_{\hat{t}})^T M ((1-\gamma)\alpha + \gamma e_{\hat{t}})$             $\triangleright$  Using Algorithm 1
12:     $\alpha = (1-\hat{\gamma})\alpha + \hat{\gamma} e_{\hat{t}}$ 
13:   until  $\hat{\gamma} \sim 0$  or Number of Iterations Limit
14:   return  $\alpha^1, \dots, \alpha^T$ 
15: end procedure
  
```

در این مقاله ادعا شده است که در صورت برآورده شدن یک سری شروط، الگوریتم فوق قادر به حل

مسئله بهینه سازی برای هر ساختار عمیق مبتنی بر مدل‌های انکودر-دیکودر است.

برای ارزیابی عملکرد شبکه‌ی joint می‌توانیم معیار‌های ارزیابی شبکه‌ها را با هم ترکیب کنیم. به

طول مثال می‌توانیم ضریبی از معکوس AP در شبکه تشخیص اشیاء را با هر یک از سه معیار

relative error, RMSE, average log10 error

ارزیابی نهایی در نظر بگیریم.

یا میتوانیم AP را به همراه میانگین threshold accuracy به عنوان معیار ارزیابی عملکرد در نظر

بگیریم.

در نهایت با استفاده از Hybrid CNN for Single Image Depth Estimation و yolo_v5 می‌توان

معیار نهایی را به صورت زیر تعریف کرد.

$$\frac{\alpha}{AP} + \frac{\beta}{(\delta_1 + \delta_2 + \delta_3)} + \gamma(REL + RMSE + Log10)$$

با در نظر گرفتن ۶۸٪ به عنوان mAP از تست کردن شبکه yolo_v5 روی دیتابست coco و مقادیر به دست آمده برای شبکه‌ی تعیین عمق این پارامتر به صورت زیر محاسبه می‌شود.

$$\frac{\alpha}{0.68} + \frac{\beta}{(2.77)} + \gamma(0.13 + 0.593 + 0.057)$$

5-7 پیشنهادها

- پیشنهاد می‌شود در روش اول ضرایب مربوط به هر کدام از loss‌ها بهینه سازی شود. tune کردن وزن‌های هر کدام از loss‌ها می‌توان نقش موثری در نتیجه‌ی خروجی داشته باشد.

- در مدلی که آموزش دادیم می‌توانیم به جای این که در قسمت انکودر از ResNet استفاده کیم از DenseNet استفاده کیم.

- همان طور که پیاده سازی شد دیدیم که cluster کردن باندینگ باکس به دو ناحیه‌ی foreground و background متنه‌ی به نتایج بهتری شد. از دیگر راهکار‌های مشابه برای محاسبه‌ی دقیق‌تر عمق یک شیء میتوان به روش‌های زیر اشاره کرد:

- می‌توان تنها پیکسل‌های مرکزی و یا یک ناحیه‌ی کوچک در یک چارچوب را برای محاسبه‌ی عمق در نظر گرفت.

- می‌توان به جای میانگین از شاخص‌های دیگر از جمله میانه و یا مد استفاده کرد.

5-8 جمع بندی

در این فصل به بررسی معیارهای ارزیابی جداگانه شبکه‌های تشخیص شیء و تخمین عمق پرداختیم و پس از آن، نحوه اتصال دو شبکه به یکدیگر برای تشکیل شبکه کلی و همچنین معیارهای ارزیابی شبکه کلی یعنی تشخیص همزمان اشیاء و تخمین عمق را توضیح دادیم. در انتهای فصل نیز چند پیشنهاد برای بهبود عملکرد شبکه را عنوان کردیم.

۶ فصل ششم: طراحی رابطهای کاربری

6-1 مقدمه

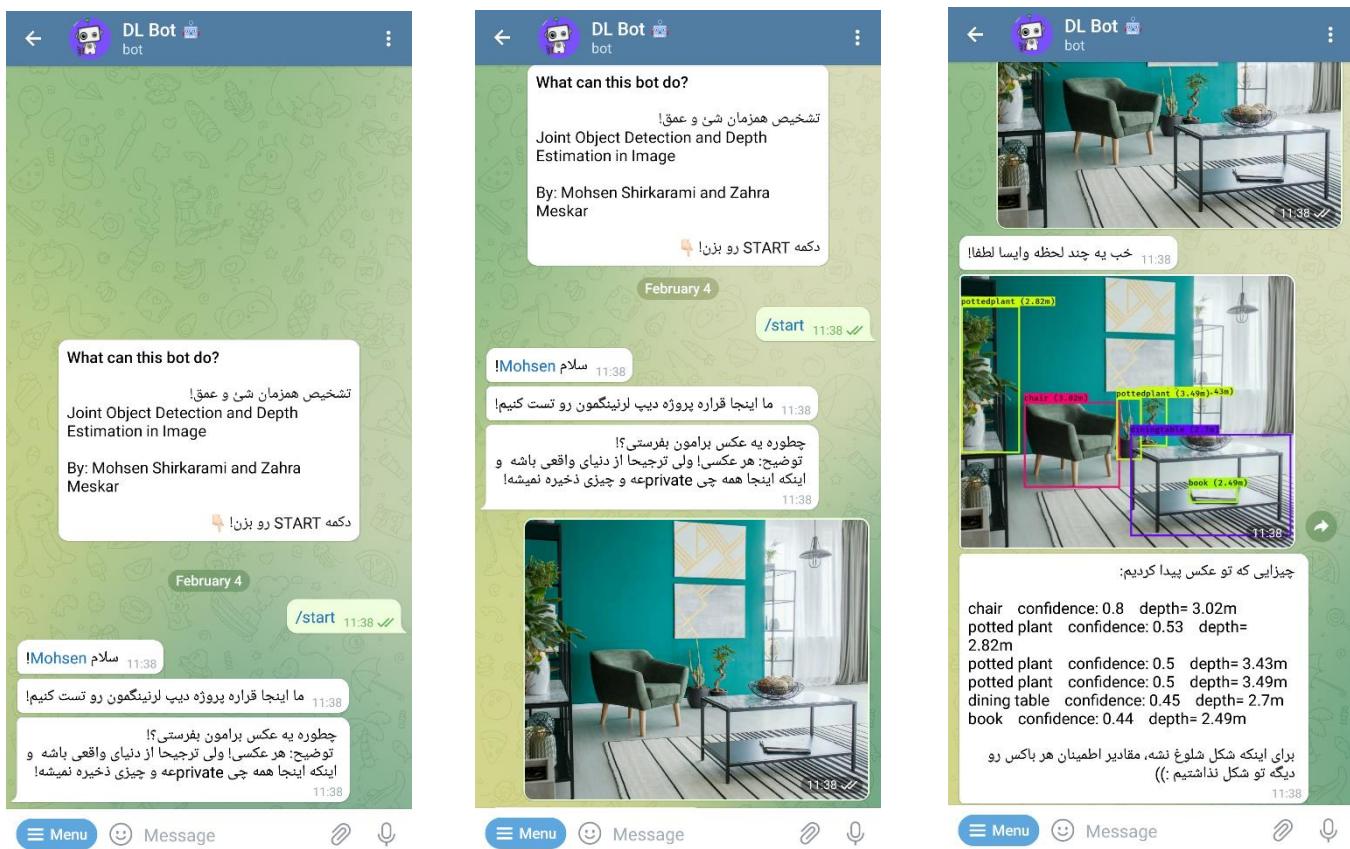
یکی از مهم‌ترین بخش‌ها در هر پروژه‌ای، نحوه‌ی عرضه آن به کاربر برای بهره‌گیری از خواص آن است. رابط کاربری‌ای که قرار است نقش واسط بین کاربر و شبکه را ایفا کند، باید ویژگی‌هایی از قبیل سهولت دسترسی، سرعت در پردازش، نیاز به حداقل سخت‌افزار و نرم‌افزار ممکن و... را داشته باشد. در این فصل به بررسی دو رابط کاربری پیاده‌سازی شده برای پروژه خواهیم پرداخت.

6-2 ربات تلگرام

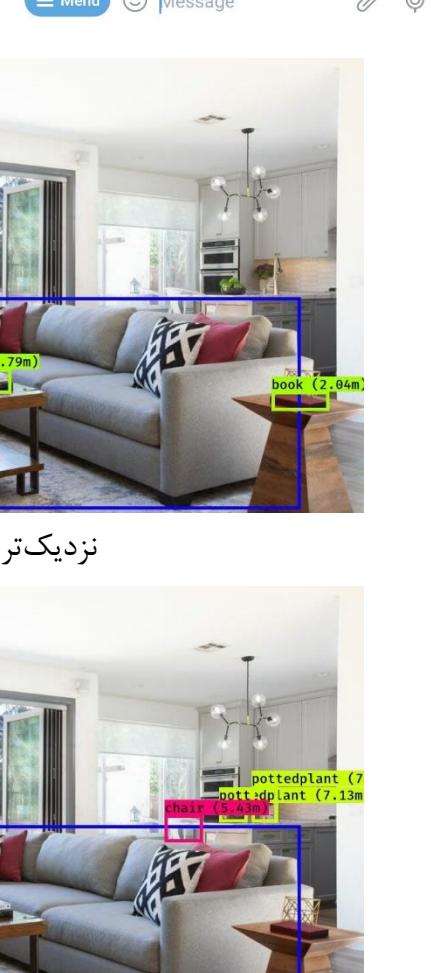
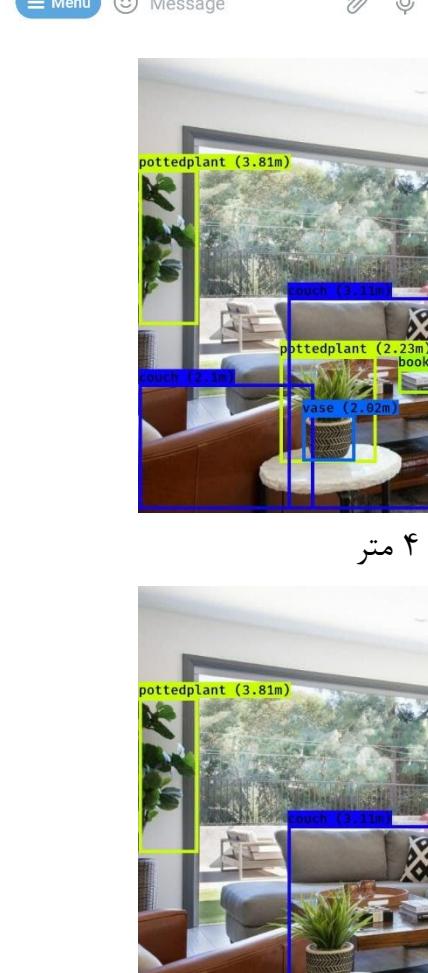
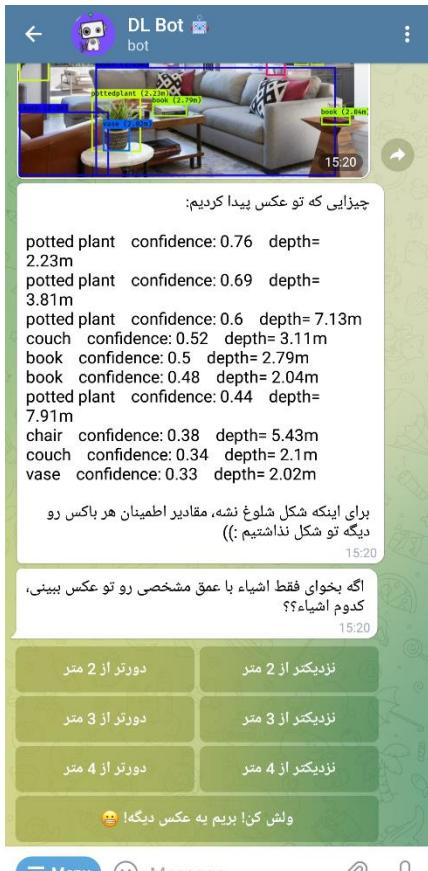
برای بکارگیری عملی این پروژه و با تاکید بر ویژگی سهولت دسترسی، تصمیم گرفتیم یک ربات تلگرام طراحی کنیم که با استفاده از آن، کاربر بتواند به سادگی و در کمترین زمان از این پروژه استفاده کند (با آدرس [@DL_Shafif_Project_bot](https://t.me/DL_Shafif_Project_bot) در تلگرام).

در طراحی این ربات، پس از مقایسه و کار با چند کتابخانه موجود در طراحی ربات تلگرام، در نهایت، از کتابخانه python-telegram-bot استفاده کردیم. نحوه عملکرد این ربات به این صورت است که یک عکس از کاربر دریافت کرده و پس از پردازش سریع، یک عکس شامل اشیاء مشخص شده به صورت عکس از کاربر دریافت کرده و پس از مفایضه و همچنین یک متن شامل لیست اشیاء به همراه میزان اطمینان bounding-box با عمق هر کدام و همچنین یک متن شامل لیست اشیاء به همراه میزان اطمینان (confidence) و عمق هر کدام را برای کاربر ارسال می‌کند.

یک نمونه از عملکرد این ربات را می‌توانیم در شکل زیر بینیم:



قابلیت دیگر ربات، فیلتر اشیاء شناسایی شده بر اساس عمق آنها است. پس از تشخیص برای هر تصویر، به کاربر چند انتخاب از بین عمق‌های مختلف داده می‌شود که بر اساس اینکه کاربر فیلتر دورتر یا نزدیکتر از یک عمق مشخص را انتخاب کند، در پاسخ، ربات برای او تصویر خروجی تنها شامل اشیاء دارای شرایط عمیقی تعیین شده توسط او ارسال می‌شود. نمونه‌ای از این ویژگی را می‌توانیم در تصاویر زیر بینیم:

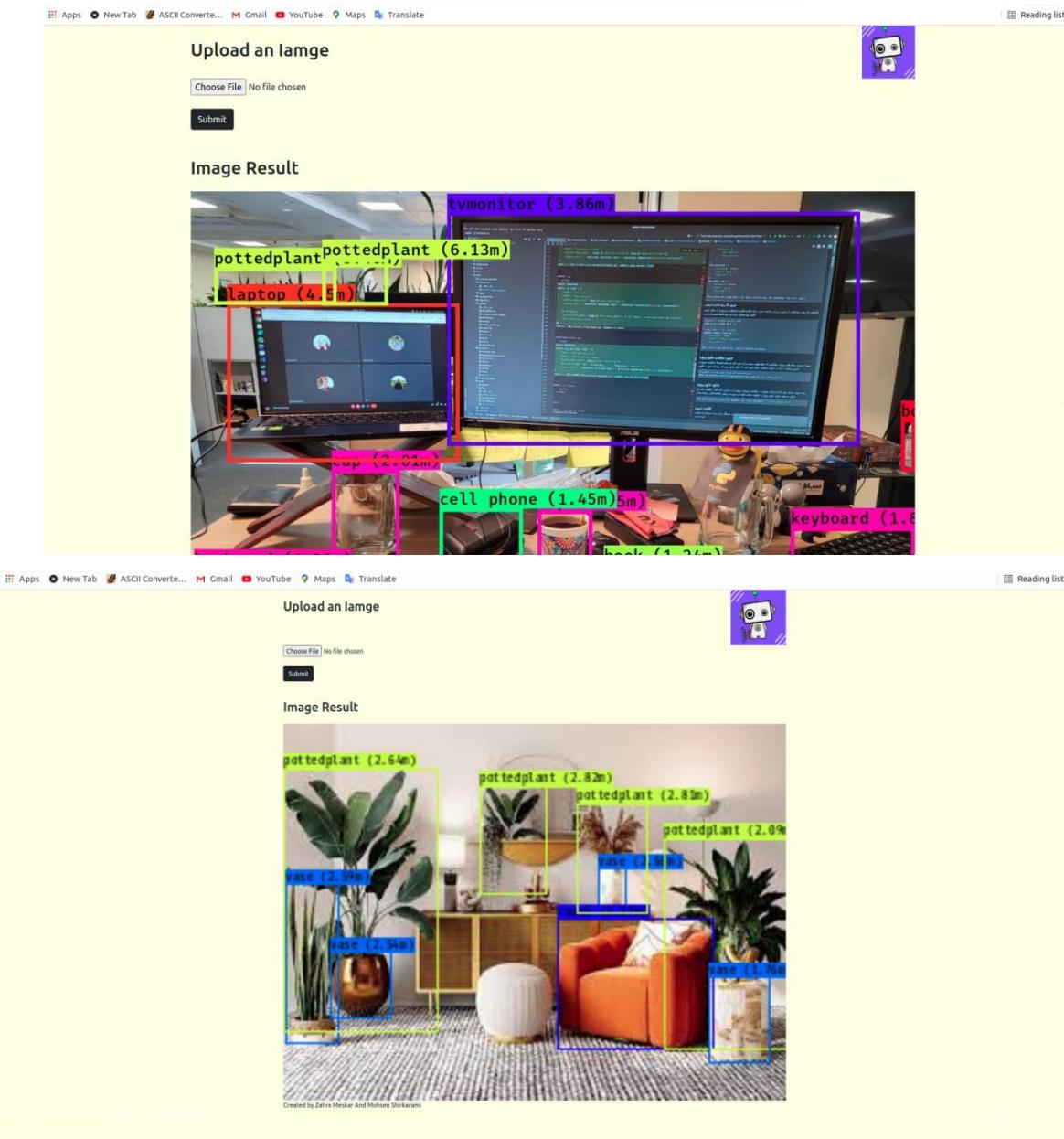


در صورتی که کاربر، محتوایی جز عکس برای ربات ارسال کند، ربات به شکل دوستانه‌ای به او یادآور می‌شود که برای استفاده از عملکرد ربات تنها باید عکس ارسال کند. همچنین اگر احیاناً شبکه قادر به تشخیص object‌ای در عکس نباشد (از لیست اشیاء قابل تشخیص توسط شبکه YOLO نباشد و یا اینکه به هر دلیل دیگری نتواند شئی‌ای را در تصویر ارسالی شناسایی کند) باز هم این مورد به اطلاع کاربر می‌رسد و از او درخواست می‌شود که تصویر دیگری را امتحان کند. در کل تلاش شده که کاربر با استفاده از این ربات، حس سهولت و سرعت در بهره‌مندی از خواص این پروژه را تجربه کند.

یکی از چالش‌های پیاده‌سازی ربات، قرار گرفتن آن روی سرور است. می‌دانیم که سرور تلگرام با مکانیزم polling کار می‌کند و ما قادر به بالا آوردن ربات روی سیستم خودمان نیز هستیم. با دستورات update و API موجود در تلگرام، ارتباط بین سیستم ما و کاربر با واسطه‌گری سرور تلگرام انجام می‌شود. اگر بخواهیم ربات را بر روی سرور قرار بدهیم، باید تنظیمات و کانفیگ مربوط به Webhook را اعمال کنیم که با توجه به فیلتربودن تلگرام با آی‌پی ایران و مشکلات متعددی که در زمینه پایدار نگهداشتن عملکرد ربات به وجود می‌آید این کار را انجام ندادیم و به همین هست بودن دسکتاپ خودمان بسته کرده ایم. (عملکرد ربات ما توسط تی‌ای بررسی و مورد تایید قرار گرفت)

6-3 وب اپلیکیشن

در یک پیاده‌سازی دیگر، یک وب اپلیکیشن برای بکارگیری شبکه طراحی کردیم که در آن، کاربر با آپلود عکس خود، یک عکس در خروجی دریافت می‌کند که شامل اشیاء موجود در تصویر به همراه عمق آن‌هاست. یک نمونه از خروجی این وب اپلیکیشن را در تصویر زیر می‌بینیم:



برای بهره‌گیری از خاصیت end-to-end API که از کاربر یک تصویر گرفته و در نهایت پس از انجام تمامی پردازش‌ها به او یک تصویر خروجی تحویل دهد، کد را refactor کردہ‌ایم. برای این که را توسعه دهیم از چارچوب Django استفاده کردہ‌ایم. برای این که یک وب اپلیکیشنی تولید کنیم که شامل یک فرم ساده است که در آن یک عکس توسط کاربر آپلود شده و بعد از اجرای پردازش‌ها عکس خروجی به کاربر در browser نمایش داده می‌شود. برای انجام این کار، نیازمند این بودیم که Django و

مدت فراغیری این مباحث در حدی که این پیاده‌سازی انجام شود صورت گرفت.

6-4 جمع بندی

در این فصل، دو رابط کاربری پیاده‌سازی شده از شبکه (ربات تلگرام و وب اپلیکشین) و جزئیات مرتبط به هر یک را مشاهده کردیم.

مراجع ٧

- [1] Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision Transformers for Dense Prediction. arXiv preprint arXiv:2103.13413, 2021. 3, 24
- [2] Katrin Lasinger, Rene Ranftl, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. arXiv preprint arXiv:1907.01341, 2019. 2, 3
- [3] Wang, Yan & Lai, Zihang & Huang, Gao & Wang, Brian & van der Maaten, Laurens & Campbell, Mark & Weinberger, Kilian. (2018). Anytime Stereo Image Depth Estimation on Mobile Devices. <https://arxiv.org/abs/1810.11408>
- [4] Redmon, Joseph & Divvala, Santosh & Girshick, Ross & Farhadi, Ali. (2016). You Only Look Once: Unified, Real-Time Object Detection. 779-788. 10.1109/CVPR.2016.91.
- [5] Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. 9905. 21-37. 10.1007/978-3-319-46448-0_2.
- [6] J. Xiang and G. Zhu, "Joint Face Detection and Facial Expression Recognition with MTCNN," 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 2017, pp. 424-427, doi: 10.1109/ICISCE.2017.95.
- [7] Chang, J.-R. and Chen, Y.-S., "Pyramid Stereo Matching Network" <https://arxiv.org/abs/1803.08669>
- [8] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox, titled: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation
- [9] Godard, C., Mac Aodha, O., and Brostow, G. J., "Unsupervised Monocular Depth Estimation with Left-Right Consistency", <https://arxiv.org/abs/1609.03677>
- [10] Liu, Pengfei & Zhang, Ji & Leung, Cane & He, Chao & Griffiths, Thomas. (2018). Exploiting Effective Representations for Chinese Sentiment Analysis Using a Multi-Channel Convolutional Neural Network.
- [11] https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/depth_estimation.ipynb#scrollTo=xzuXJtZdr24C
- [12] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted*

Intervention – MICCAI 2015, Cham, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., 2015// 2015: Springer International Publishing, pp. 234-241.

[13] Harsányi, Károly & Kiss, Attila & Majdik, András & Szirányi, Tamás. (2019). A Hybrid CNN Approach for Single Image Depth Estimation: A Case Study: Proceedings of the 11th International Conference MISSI 2018. 10.1007/978-3-319-98678-4_38.

[14] Alhashim, Ibraheem and Peter Wonka. “High Quality Monocular Depth Estimation via Transfer Learning.” ArXiv abs/1812.11941 (2018): n. pag.

[15] Sener, Ozan, and Vladlen Koltun. "Multi-task learning as multi-objective optimization." Advances in neural information processing systems 31 (2018).

[16] Zhou, Changxin & Liu, Yazhou & Sun, Quansen & Lasang, Pongsak. (2019). Joint Object Detection and Depth Estimation in Multiplexed Image. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2936126.