دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش پروژه پایانی درس یادگیری عمیق

نگارش :

محمد امین رمضان دهنوی

سینا نبی گل

بهمن ماه ۱۴۰۰

فهرست مطالب

۱. شبکه تخمین عمق[1]

در سیستم مورد نظر از مدل AdaBins [1] که نسخه بهبود یافته مدل DenseDepth [2] می‌باشد، استفاده شده است. در ادامه به شرح ایده مطرح شده در آن خواهیم پرداخت.

۱.۱ مقدمه

هدف این شبکه ایجاد یک نقشه عمق (Depth Map) دقیق از روی یک تصویر RGB ورودی می‌باشد. شبکه‌های مختلفی برای این منظور طراحی شده‌اند که غالبا از ۲ روش برای تخمین عمق استفاده کرده‌اند:

۱- با استفاده از یک شبکه CNN مساله تخمین عمق را به صورت مساله رگرسیون حل کرده‌اند.

۲- با استفاده از معماری Encoder-Decoder نقشه عمق تصویر ورودی را محاسبه کرده‌اند.

شبکه AdaBins نیز یکی از شبکه‌های تخمین عمق می‌باشد که توانسته با کنار هم قرار دادن یک Transformer و یک Encoder-Decoder به دقت بالایی دست یابد. ایده اصلی این شبکه تعمیمی بر این امر است که چالش تخمین عمق از مساله رگرسیون به مساله طبقه‌بندی تبدیل شود و برای عمق به جای مقادیر پیوسته از مقادیر گسسته استفاده شود (در کارهای قبلی [3] نشان داده‌شده است که این ایده می‌تواند دقت را افزایش دهد). برای بهبود این روش دو کار در نظر گرفته شده‌است.

۱- مقادیر گسسته ثابت برای عمق باعث ایجاد خطا در خروجی شبکه می‌شود. به همین علت در این شبکه مقادیر گسسته عمق به صورت پویا و با توجه به تصویر ورودی تعیین گردد.

۲- در نظر گرفتن مقادیر گسسته باعث می‌شود تا رزولوشن نقشه عمق کاهش یابد و مقطع به نظر برسد. برای جبران این امر در این شبکه مقادیر نهایی عمق با استفاده از ترکیب خطی مقادیر گسسته عمق به دست می‌آید.

۱.۲ روش کار

۱.۲.۱ نحوه عملکرد شبکه

گام اول؛ باید بازه‌ای برای عمق در نظر گرفته‌شود $D = (d_{min}, d_{max})$ که سپس به N بخش تقسیم شود. این بازه باید با توجه به دیتاست و یا دلایل منطقی دیگری تعیین گردد. در سیستم ما از آنجا که دیتاست مورد استفاده NYU Depth v2 بود و حداکثر فاصله ۱۰ متر بود؛ بازه $(0, 10)$ در نظر گرفته شد. مساله مهم بعدی نحوه تقسیم بندی این بازه به مقادیر گسسته می‌باشد برای این امر ۴ حالت قابل تصور است:

[1] Depth Estimation Network

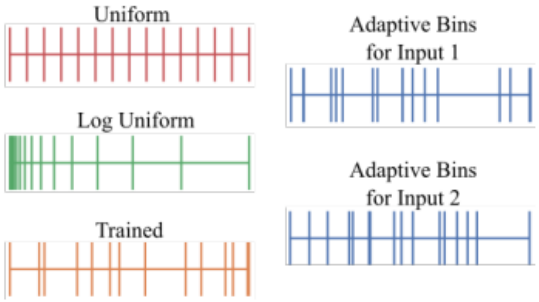
۱- تقسیم بندی یکسان

۲- تقسیم بندی با مقیاس لگاریتمی

۳- تقسیم بندی قابل یادگیری که باعث می‌شود برای هر دیتاست تقسیم بندی متفاوتی در نظر گرفته شود.

۴- تقسیم بندی بر اساس ویژگی‌های تصویر ورودی که تقسیم بندی به ازای هر تصویر ورودی به صورت مجزا انجام می‌شود.

در این شبکه پس از ارزیابی روش های بالا، از روش چهارم به عنوان بهترین روش استفاده گردیده است.

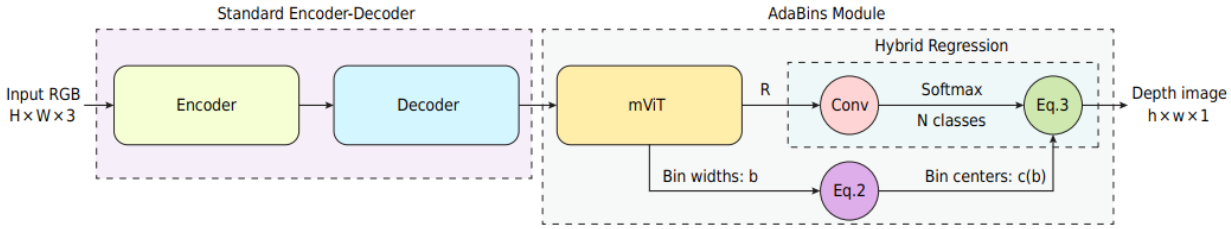


شکل ۱. نحوه تقسیم‌بندی بازه عمق

گام دوم؛ باید به هر پیکسل تصویر ورودی یکی از مقادیر گسسته به عنوان عمق تخصیص داده‌شود. ولی از آنجا که باعث ایجاد ناپیوستگی در نقشه عمق نهایی می‌شود؛ باید عمق نهایی به صورت ترکیب خطی از مقادیر گسسته محاسبه گردد.

۱.۲.۲ معماری شبکه

معماری کلی شبکه در شکل زیر نشان داده شده‌است.



شکل ۲. معماری شبکه AdaBins

همانگونه که در شکل ۲ نشان داده شده است، معماری کلی از دو قسمت تشکیل شده است:

۱- انکودر-دیکدر استاندارد (Standard Encoder-Decoder): این بخش در ورودی یک تصویر RGB به ابعاد $H \times W \times 3$ دریافت می کند و در خروجی (برخلاف کارهای قبلی انجام شده که در خروجی نقشه عمق را تولید می کردند)، یک نقشه ویژگی (Feature Map) به ابعاد $h \times w \times C_d$ تولید می کند که در مقاله به آن "Decoded Features" گفته شده است.

۲- Adaptive Bin-width Estimator Block (AdaBins Module): ورودی این بخش Decoded Features بدست آمده در بخش قبل و خروجی آن Depth Map به ابعاد $h \times w \times 1$ می‌باشد که به دلیل محدودیت های مربوط به حافظه GPU مورد

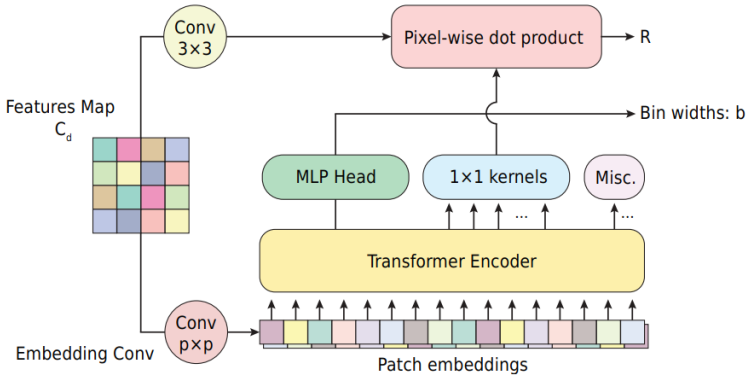
استفاده $h = \frac{H}{2}$ و $w = \frac{W}{2}$. در نهایت با استفاده از Bilinear Upsampling نقشه عمق خروجی با ابعاد $H \times W \times 1$ بدست می‌آید. هر کدام از ماژول‌های به کار رفته در این بخش در ادامه توضیح داده می‌شود.

- **Mini-ViT**

با توجه به اینکه در این مقاله هدف آن است تا عمل کوانتیزاسیون عمق برای هر تصویر ورودی به صورت مجزا و انحصاری انجام شود، لازم است تا همزمان به اطلاعات ساختاری محلی (Local Structural Information) و اطلاعات توزیعی عمومی (Global Distributional Information) بین پیکسل‌ها در هر تصویر ورودی دسترسی داشته باشیم. برای رسیدن به این هدف، از ایده Visual Transformer (ViT) [4] استفاده و یک نسخه ساده‌تر آن به نام Mini-ViT پیاده سازی شده است. معماری Mini-ViT و در شکل و جدول زیر آورده شده است.

جدول ۱. جزئیات معماری Mini-ViT

Patch size (p)	E	Layers	num heads	C	MLP Size	Params
16	128	4	4	128	1024	5.8 M



شکل۳. معماری بلوک Mini-ViT

ورودی این بخش Decoded Feature با ابعاد $h \times w \times C_d$ و خروجی آن بردار Bin Widths ($\boldsymbol{b}$) و Range Attention Maps می‌باشد. Bin Widths که از خروجی MLP Head بدست می‌آید، یک بردار به طول N است که درایه‌های آن بیانگر فواصل لازم برای کوانتیزه کردن عمق است. خروجی Encoder (Output Embeddings) دارای اطلاعات عمومی (Global Distributional Information) بین پیکسل‌ها و خروجی Decoded Features دارای اطلاعات محلی (Local Structural Information) بین پیکسل‌ها می‌باشد. برای ترکیب کردن این اطلاعات باهم، Decoded Features ها پس از عبور ها از یک کانولوشن 3x3، با خروجی Encoder ضرب داخلی می‌شوند. در این ضرب داخلی، خروجی Encoder همانند یک کرنل کانولوشنی عمل کرده و اجازه می‌دهد تا اطلاعات عمومی بدست آمده از Transformer و اطلاعات محلی موجود در Decoded Features با یکدیگر ترکیب شوند.

- **Hybrid Regression**

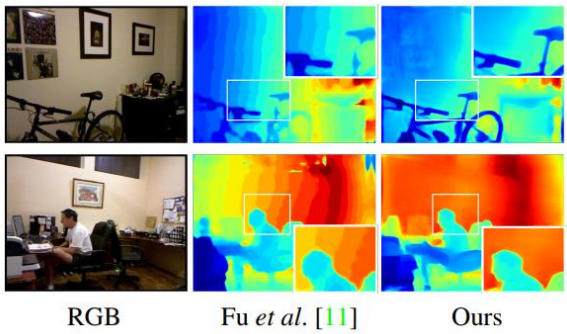
در این بخش Range Attention Maps بدست آمده از Mini-ViT از N عدد کرنل کانولوشنی 1x1 و یک لایه Softmax عبور داده می‌شوند تا یک نقشه ویژگی با بعد N بدست بیاید که به ازای هر پیکسل N عدد امتیاز Softmax (p_k) خواهیم داشت. در نهایت عمق هر پیکسل از رابطه زیر بدست می‌آید.

$$\tilde{d} = \sum_{k=1}^{N} c(b_k) p_k$$

که در آن Bin Center $c(b_i)$ مربوط به i امین بعد می‌باشد و بر اساس بردار Bin Widths از رابطه زیر بدست می‌آید.

$$c(b_i) = d_{min} + (d_{max} - d_{min})(b_i/2 + \sum_{j=1}^{i-1} b_j)$$

استفاده از این روش برای تخمین عمق هر پیکسل سبب می‌شود تا یک نقشه عمق با تغییرات عمق نرم و بدون گسستگی بدست آید. نمونه خروجی این شبکه در شکل زیر آورده شده است.



شکل٤. نمونه ای از خروجی های تولید شده توسط شبکه AdaBins

۱.۲.۳ تابع هزینه

تابع هزینه کلی متشکل از ۲ تابع هزینه است:

۱- تابع هزینه عمق برحسب پیکسل : برای این منظور از تابع هزینه Scale-Invariant(SI) استفاده شده‌است.

$$\mathcal{L}_{pixel} = \alpha \sqrt{\frac{1}{T}\sum_i g_i^2 - \frac{\lambda}{T^2}(\sum_i g_i)^2}$$

$$g_i = \log \tilde{d}_i - \log d_i$$

در تابع هزینه فوق d_i مقدار GroundTruth عمق و T نشان دهنده تعداد پیکسل‌هایی می‌باشد که مقادیر GroundTruth معتبر دارند. در رابطه بالا $\lambda = 0.85$ و $\alpha = 10$ در نظر گرفته شده‌است.

۲- تابع هزینه چگالی مراکز Bin ها: این تابع هزینه باعث می‌شـود تا توزیع مراکز Bin ها از توزیع مقادیر عمق در GroundTruth پیروی کند. در رابطه زیر C(b) مجموعه مراکز Bin ها و X مجموعه مقادیر عمق در GroundTruth می‌باشـد. به عنوان رگولایزر از تابع هزینه دو طرفه Chamfer استفاده شده‌است.

$$\mathcal{L}_{bins} = chamfer(X, c(\mathbf{b})) + chamfer(c(\mathbf{b}), X)$$

در نهایت تابع هزینه کلی به صورت زیر می‌باشد:

$$\mathcal{L}_{total} = \mathcal{L}_{pixel} + \beta \mathcal{L}_{bins}$$

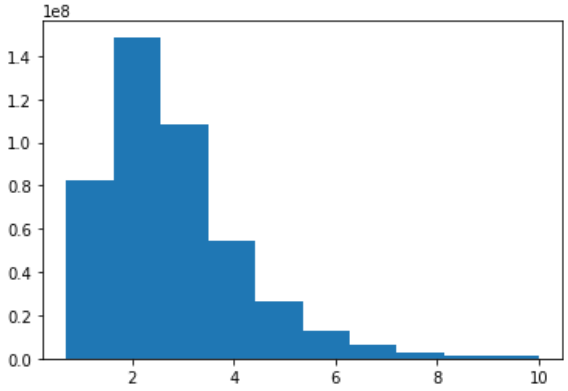
در رابطه بالا $\beta = 0.1$ در نظر گرفته شده‌است.

۱.۳ دیتاست آموزش

برای آموزش شبکه از دیتاست NYU Depth v2 استفاده شده‌است که شامل تصاویر داخلی و نقشه عمق مربوطه می‌باشد. این دیتاست شامل ۱۵۰.۰۰۰ داده آموزش و ۶۵۴ داده تست با ابعاد ۶۴۰×۴۸۰ می‌باشد. نهایت عمق این دیتاست ۱۰ متر می‌باشد. ابعاد خروجی شبکه ۳۲۰×۲۴۰ می‌باشد که برای آنکه هم اندازه تصویر ورودی شود، با استفاده از Bilinear Upsampling، ۲ برابر شده‌است.

۱.۳.۱. بررسی دیتاست NYU Depth v2

در شکل زیر هیستوگرام فواصل پیکسل‌های موجود در دیتاست NYU Depth v2 رسم شده است. همچنین مینیمم فاصله برابر با ۱.۲۳ متر و ماکزیمم آن برابر با ۹.۹۹ متر است. با توجه به هیستوگرام فاصله‌ها، اکثر اشیا در فواصل کمتر از ۵ متر قرار گرفته‌اند، بنابراین تغییر مقدار آستانه در تشخیص عمق در فواصل کمتر تاثیر بیشتری بر روی معیارهای ارزیابی شبکه AdaBins خواهند داشت. از نکته بیان شده در این بخش در قسمت تعریف معیار ارزیابی استفاده خواهیم نمود.



شکل۵. هیستوگرام فواصل پیکسل‌های موجود در دیتاست NYU

۱.٤ ارزیابی

۱.٤.۱ معیارهای ارزیابی

برای شبکه‌های تخمین عمق معیارهای ارزیابی به صورت زیر تعریف می‌گردد.

- average relative error (rel): $\frac{1}{n}\sum_p^n \frac{|y_p-\hat{y}_p|}{y}$;
- 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. $\max(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3$;

در روابط بالا y_p مقدار یک پیکسل در نقشه عمق y می‌باشد و $\hat{y}_p$ مقدار تخمین زده شده آن می‌باشد. n نیز تعداد کل پیکسل‌ها در نقشه عمق می‌باشد.

۱.٤.۲ ارزیابی شبکه

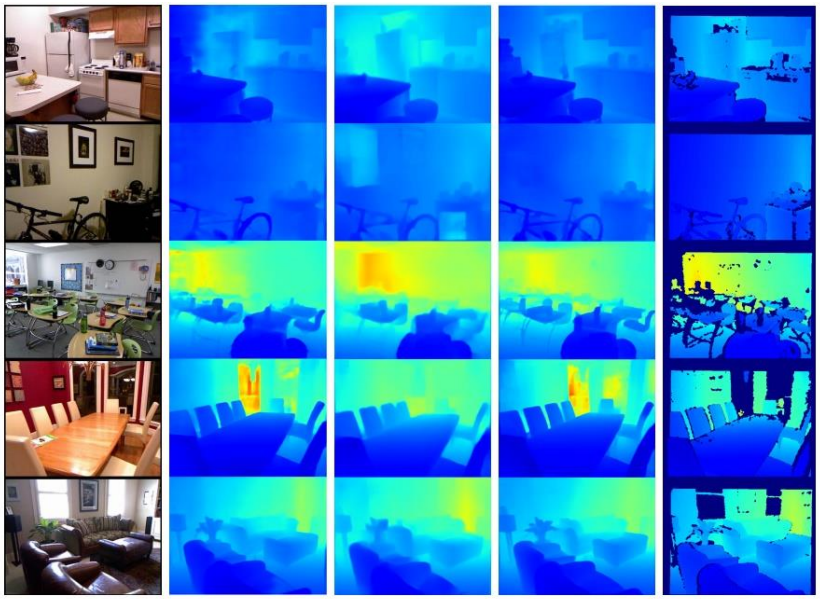
نتایج ارزیابی شبکه بر روی دیتاست NYU Depth v2 و مقایسه آن با دیگر شبکه‌های موجود در جدول زیر آورده شده است.

جدول۲. مقایسه عملکرد شبکه‌های مختلف در دیتاست NYU Depth v2

Method	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$	REL $\downarrow$	RMS $\downarrow$	$log_{10}\downarrow$
Eigen *et al.*	0.769	0.950	0.988	0.158	0.641	–
Laina *et al.*	0.811	0.953	0.988	0.127	0.573	0.055
Hao *et al.*	0.841	0.966	0.991	0.127	0.555	0.053
Lee *et al.*	0.837	0.971	0.994	0.131	0.538	–
Fu *et al.*	0.828	0.965	0.992	0.115	0.509	0.051
SharpNet	0.836	0.966	0.993	0.139	0.502	0.047
Hu *et al.*	0.866	0.975	0.993	0.115	0.530	0.050
Chen *et al.*	0.878	0.977	0.994	0.111	0.514	0.048
Yin *et al.*	0.875	0.976	0.994	0.108	0.416	0.048
BTS	0.885	0.978	0.994	0.110	0.392	0.047
DAV	0.882	0.980	0.996	0.108	0.412	–
AdaBins (Ours)	**0.903**	**0.984**	**0.997**	**0.103**	**0.364**	**0.044**

۱.٤.۳ نمونه‌هایی از خروجی شبکه

در تصاویر زیر خروجی شبکه AdaBins با خروجی برخی از شبکه‌های دیگر مقایسه شده‌است.



(a) RGB (b) BTS (c) DAV (d) Ours (e) GT

شکل٦. مقایسه کیفی عملکرد شبکه‌های مختلف در دیتاست NYU Depth v2

۲. شبکه تشخیص اشیا[۲]

برای شبکه تشخیص اشیا از معماری YOLOX [5] استفاده شده است. ایده اصلی این شبکه از YOLOv3 [6] گرفته شده است و با وام گرفتن از ایده های مطرح شده در نسخه های YOLOv4 [7] و YOLOv5 [8] عملکرد آن را بهبود داده‌اند.

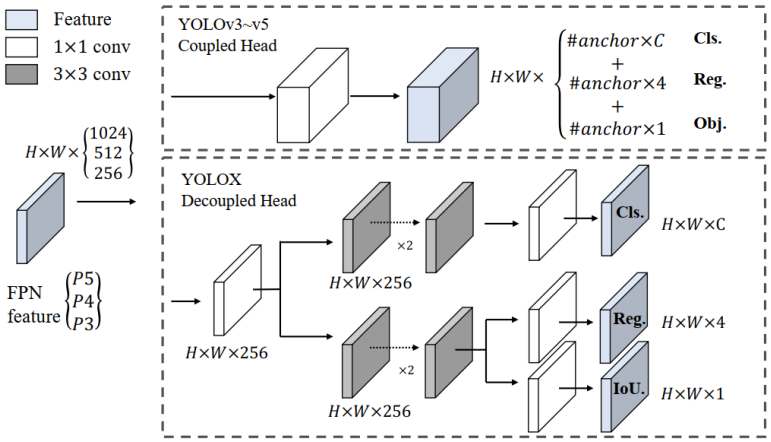
۲.۱. ساختار شبکه YOLOX

این شبکه برای Backbone از DarkNet53 که در مقاله YOLOv3 معرفی شده است به همراه لایه SPP استفاده می‌نماید. در ادامه بهبودهایی که شبکه YOLOX در مقایسه با نسخه های قبلی YOLO داده است، شرح داده می‌شوند.

- **Decoupled Head**

یکی از مشکلاتی که شبکه های پیشین YOLO دارند، استفاده از یک Head برای انجام تسک های Bbox Regression و Classification است. این کار سبب افت دقت در خروجی این شبکه ها می‌شود. استفاده از Head های متفاوت برای هر کدام از تسک‌ها سبب می‌شود تا سرعت همگرایی بیشتر شده و میزان افت دقت به هنگام ساخت حالت end-to-end شبکه کاهش یابد.

در شبکه YOLOX با استفاده از Head های مختلف برای هرکدام از تسک های ذکرشده توانسته‌اند دقت AP بر روی دیتاست COCO را نسبت به معماری YOLOv3 به اندازه 1.1% افزایش بدهند. ساختار معماری Decoupled Head و تفاوت آن با YOLOv3 Head در شکل زیر نشان داده شده است.



شکل ۷. ساختار معماری Decoupled Head و تفاوت آن با YOLOv3 Head

[2] Object Detection Network

- **Strong Data Augmentation**

برای آموزش شبکه از روش‌های داده‌سازی Mosaic [7] و MixUp [9] بهره گرفته شده است. استفاده از این کار سبب افزایش AP بر روی دیتاست COCO به میزان 2.4% شده است.

- **Anchor Free Detection**

تمامی معماری‌های پیشین عملیات Detection را به صورت Anchor Based انجام می‌دادند. استفاده از Anchor Box چندین مشکل جدی دارد:

- لازم است تا Anchor Box هایی متناسب با دیتاست مورد آموزش طراحی شوند که این کار از Generalization شبکه کم می‌کند.
- سبب افزایش پیچیدگی Prediction Head می‌شود.
- تعداد Bounding Box های پیش بینی شده برای هر تصویر را به طرز قابل توجهی افزایش داده و و نیاز به استفاده از Post Processing هایی نظیر NMS را ایجاد می‌کند.

در مقابل استفاده از شناساگرهای Anchor Free نظیر [10] علاوه بر رفع مشکلات قبلی سبب کاهش تعداد HyperParameter های لازم برای Tune کردن می‌شود. این شناساگرها با روش های مختلفی Bounding Boxهای مربوطه را تخمین می‌زنند. برای مثال در [11] تلاش می‌شود تا مکان گوشه‌های بالا سمت چپ و پایین سمت راست از Bounding Box های موجود در تصویر تخمین زده شده و سپس با استخراج ویژگی هایی از نواحی اطراف و انتقال بردار بدست آمده به یک فضای Embedding، گوشه های بالا سمت چپ و پایین سمت راست که متناظر با یک Object هستند به یکدیگر نزدیک شده و برعکس آنهایی که مربوط به یک شی نیستند از یکدیگر دور می‌شوند و بدین ترتیب Bounding Box موردنظر تخمین زده می‌شود.

معماری YOLOX با استفاده از حالت Anchor Free توانسته است دقت AP بر روی دیتاست COCO را نسبت به YOLOv3 به میزان 0.9% افزایش دهد.

- **Multi Positives**

استفاده از ایده Anchor Free سبب می‌شود تا برای هر شی تنها یک نمونه مثبت (Positive Sample) در نظر گرفته شود. این کار سبب می‌شود تا به هنگام آموزش با مشکل Imbalanced بودن نمونه های مثبت و منفی روبه‌رو شویم. برای حل این مشکل به جای یک نمونه مثبت، یک ناحیه 3x3 به عنوان نمونه‌های مثبت به ازای هر شی تعریف می‌شود. انجام این کار سبب افزایش AP به میزان 2.1% می‌شود. در جدول زیر تاثیر استفاده از هر کدام از موارد ذکر شده بر دقت و سرعت شبکه YOLOX در مقایسه با شبکه YOLOv3 آورده شده است.

جدول ۳. مقایسه تاثیر استفاده از متدهای مختلف بر روی شبکه YOLOv3

Methods	AP (%)	Parameters	GFLOPs	Latency	FPS
YOLOv3-ultralytics[2]	44.3	63.00 M	157.3	10.5 ms	95.2
YOLOv3 baseline	38.5	63.00 M	157.3	10.5 ms	95.2
+decoupled head	39.6 (+1.1)	63.86 M	186.0	11.6 ms	86.2
+strong augmentation	42.0 (+2.4)	63.86 M	186.0	11.6 ms	86.2
+anchor-free	42.9 (+0.9)	63.72 M	185.3	11.1 ms	90.1
+multi positives	45.0 (+2.1)	63.72 M	185.3	11.1 ms	90.1
+SimOTA	**47.3** (+2.3)	63.72 M	185.3	11.1 ms	90.1
+NMS free (optional)	46.5 (-0.8)	67.27 M	205.1	13.5 ms	74.1

۲.۲. دیتاست آموزش

برای آموزش شبکه از دیتاست COCO-2017[3] استفاده شده است. این دیتاست شامل تصاویری از اشیا گوناگون در ۸۰ دسته مختلف نظیر خودرو، انسان، میز، توپ و ... می‌باشد. در مجموع این دیتاست شامل ۱۱۸۰۰۰ داده آموزش، ۵۰۰۰ داده ارزیابی و ۴۱۰۰۰ داده تست است.

۲.۳. ارزیابی

۲.۳.۱. معیارهای ارزیابی

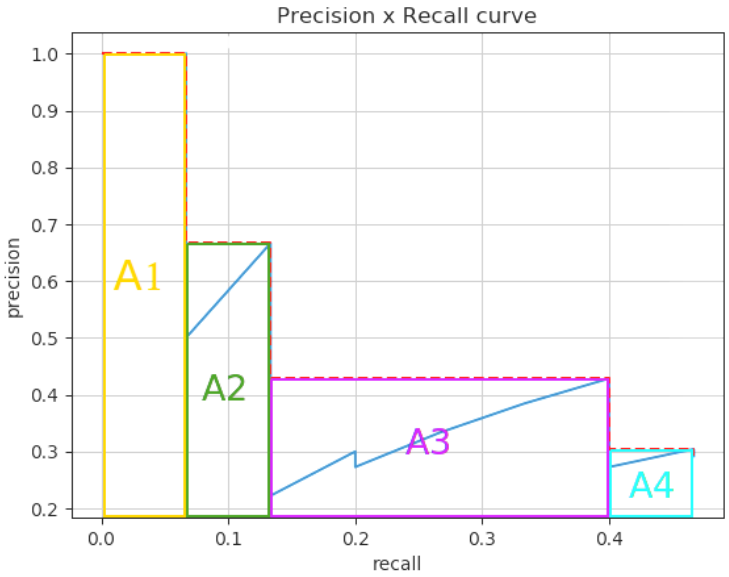
برای ارزیابی مدل از شاخص AP استفاده شده است. برای محاسبه AP، پس از رسم منحنی $Precision \times Recall$ ، برای هر مقدار Recall، بیشترین Precision به ازای همه مقادیر بزرگتر از آن را به عنوان Precision آن Recall در نظر می‌گیریم و سپس مساحت بدست آمده را محاسبه کرده و به عنوان شاخص AP اعلام می‌کنیم.

$$\sum_{n=0} \left(r_{n+1} - r_n\right) \rho_{\text{interp}}\left(r_{n+1}\right)$$

که در آن:

$$\rho_{\text{interp}}\left(r_{n+1}\right) = \max_{\tilde{r}:\tilde{r} \geq r_{n+1}} \rho\left(\tilde{r}\right)$$

[3] https://cocodataset.org/



شکل۸. منحنی Precision بر حسب Recall جهت محاسبه AP

۲.۳.۲. ارزیابی شبکه

نتایج ارزیابی شبکه بر روی دیتاست COCO و مقایسه آن با دیگر شبکه‌های موجود از لحاظ دقت و سرعت اجرا (Inference Time) در جدول زیر آورده شده است.

جدول ٤. ارزیابی و مقایسه شبکه های Object Detection بر روی دیتاست COCO از لحاظ دقت و سرعت اجرا

Method	Backbone	Size	FPS (V100)	AP (%)	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv3 + ASFF* [18]	Darknet-53	608	45.5	42.4	63.0	47.4	25.5	45.7	52.3
YOLOv3 + ASFF* [18]	Darknet-53	800	29.4	43.9	64.1	49.2	27.0	46.6	53.4
EfficientDet-D0 [28]	Efficient-B0	512	98.0	33.8	52.2	35.8	12.0	38.3	51.2
EfficientDet-D1 [28]	Efficient-B1	640	74.1	39.6	58.6	42.3	17.9	44.3	56.0
EfficientDet-D2 [28]	Efficient-B2	768	56.5	43.0	62.3	46.2	22.5	47.0	58.4
EfficientDet-D3 [28]	Efficient-B3	896	34.5	45.8	65.0	49.3	26.6	49.4	59.8
PP-YOLOv2 [11]	ResNet50-vd-dcn	640	68.9	49.5	68.2	54.4	30.7	52.9	61.2
PP-YOLOv2 [11]	ResNet101-vd-dcn	640	50.3	50.3	69.0	55.3	31.6	53.9	62.4
YOLOv4 [1]	CSPDarknet-53	608	62.0	43.5	65.7	47.3	26.7	46.7	53.3
YOLOv4-CSP [30]	Modified CSP	640	73.0	47.5	66.2	51.7	28.2	51.2	59.8
YOLOv3-ultralytics[2]	Darknet-53	640	95.2	44.3	64.6	-	-	-	-
YOLOv5-M [7]	Modified CSP v5	640	90.1	44.5	63.1	-	-	-	-
YOLOv5-L [7]	Modified CSP v5	640	73.0	48.2	66.9	-	-	-	-
YOLOv5-X [7]	Modified CSP v5	640	62.5	50.4	68.8	-	-	-	-
YOLOX-DarkNet53	Darknet-53	640	90.1	47.4	67.3	52.1	27.5	51.5	60.9
YOLOX-M	Modified CSP v5	640	81.3	46.4	65.4	50.6	26.3	51.0	59.9
YOLOX-L	Modified CSP v5	640	69.0	50.0	68.5	54.5	29.8	54.5	64.4
YOLOX-X	Modified CSP v5	640	57.8	**51.2**	69.6	55.7	31.2	56.1	66.1

اعداد جدول بالا در مقاله YOLOX گزارش شده‌اند. ما برای مطمئن شدن از صحت اعداد، یکبار دیگر شبکه YOLOX را بر روی دیتاست COCO – val2017 تست نمودیم و نتایج زیر بدست آمد.

جدول ۵. نتایج تست شبکه YOLOX بر روی دیتاست COCO-val2017

```
Average forward time: 44.77 ms, Average NMS time: 1.60 ms, Average inference time: 46.38 ms
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.469
 Average Precision  (AP) @[ IoU=0.50       | area=   all | maxDets=100 ] = 0.656
 Average Precision  (AP) @[ IoU=0.75       | area=   all | maxDets=100 ] = 0.512
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.289
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.521
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.623
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.362
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.587
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.628
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.439
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.688
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.780
```

per class AP:

class	AP	class	AP	class	AP
person	59.255	bicycle	37.264	car	47.387
motorcycle	49.486	airplane	72.458	bus	72.015
train	73.276	truck	40.695	boat	30.533
traffic light	28.812	fire hydrant	71.272	stop sign	68.840
parking meter	50.530	bench	29.527	bird	38.793
cat	73.188	dog	67.797	horse	65.713
sheep	58.958	cow	59.945	elephant	70.316
bear	75.404	zebra	72.199	giraffe	72.023
backpack	18.852	umbrella	46.802	handbag	19.803
tie	36.879	suitcase	47.620	frisbee	71.490
skis	29.038	snowboard	41.687	sports ball	45.708
kite	45.580	baseball bat	37.571	baseball glove	41.783
skateboard	58.878	surfboard	42.490	tennis racket	55.993
bottle	42.320	wine glass	38.689	cup	47.994
fork	45.377	knife	23.836	spoon	24.796
bowl	46.501	banana	30.679	apple	20.217
sandwich	41.242	orange	31.023	broccoli	25.915
carrot	27.147	hot dog	43.352	pizza	57.613
donut	52.773	cake	43.848	chair	37.202
couch	50.154	potted plant	31.355	bed	52.280
dining table	35.099	toilet	68.114	tv	62.789
laptop	66.167	mouse	63.524	remote	35.491
keyboard	55.142	cell phone	39.009	microwave	64.826
oven	40.579	toaster	38.931	sink	43.881
refrigerator	63.834	book	17.638	clock	53.239
vase	41.464	scissors	34.752	teddy bear	51.709
hair drier	7.864	toothbrush	27.818		

per class AR:

class	AR	class	AR	class	AR
person	67.274	bicycle	50.669	car	59.583
motorcycle	60.926	airplane	79.650	bus	80.000
train	80.053	truck	68.720	boat	48.632
traffic light	44.338	fire hydrant	76.238	stop sign	80.267
parking meter	60.833	bench	50.243	bird	46.370
cat	82.030	dog	77.706	horse	72.904
sheep	70.339	cow	68.952	elephant	80.516
bear	81.127	zebra	79.173	giraffe	79.483
backpack	44.852	umbrella	61.646	handbag	46.704
tie	50.556	suitcase	65.385	frisbee	77.217
skis	45.602	snowboard	54.058	sports ball	52.385
kite	60.122	baseball bat	55.517	baseball glove	53.851
skateboard	65.363	surfboard	55.019	tennis racket	66.444
bottle	60.790	wine glass	50.411	cup	64.749
fork	60.000	knife	43.354	spoon	51.581
bowl	68.812	banana	55.973	apple	52.797
sandwich	67.853	orange	58.667	broccoli	57.244
carrot	54.822	hot dog	58.720	pizza	72.782
donut	71.189	cake	63.065	chair	59.198
couch	74.521	potted plant	56.637	bed	72.025
dining table	63.094	toilet	80.447	tv	75.208
laptop	75.152	mouse	74.717	remote	56.961
keyboard	71.373	cell phone	56.069	microwave	75.455
oven	66.503	toaster	61.111	sink	61.467
refrigerator	78.413	book	39.920	clock	64.757
vase	62.774	scissors	49.722	teddy bear	67.158
hair drier	19.091	toothbrush	45.965		

۲.۳.٤. نمونه‌هایی از خروجی شبکه

در شکل‌های زیر نمونه هایی از خروجی شبکه در تشخیص اشیاء آورده شده است.

شکل۹. نمونه‌هایی از خروجی شبکه YOLOX

۳. اتصال دو شبکه به یکدیگر

۳.۱ نحوه اتصال

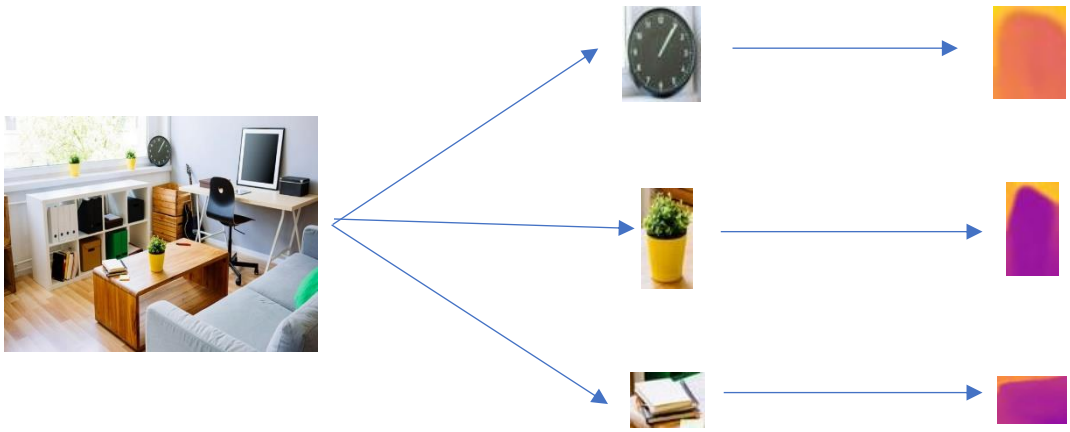
برای اتصال دو شبکه به یکدیگر ۳ حالت قابل تصور است.

حالت اول: در این حالت ابتدا شبکه AdaBins نقشه عمق را تخمین می‌زند و سپس این نقشه عمق با استفاده از یک colormap به یک تصویر RGB تبدیل می‌شود. سپس این تصویر RGB به دست آمده را به عنوان ورودی به شبکه YOLOX می‌دهیم و در خروجی اشیا تشخیص داده می‌شوند.

شکل ۱۰. حالت اول اتصال دو شبکه

در این حالت از آنجا که تصویر DepthMap وارد شبکه YOLOX می‌شود؛ خطای شبکه YOLOX زیاد می‌شود چون هم به علت خطای AdaBins و هم ذات ColorMap ها اشیا درون DepthMap به وضوع تصاویر RGB عادی مشخص نیستند و مرزهای آنها به راحتی قابل تشخیص نیست و هم رزولوشن افت می‌کند و هم رنگ اشیا در تصویر DepthMap مطابق واقعیت نیست و همه اینها خطای شبکه YOLOX را زیاد می‌کند.

حالت دوم: در این حالت ابتدا شبکه YOLOX اشیا را تشخیص می‌دهد و سپس این اشیا را از تصویر RGB اصلی Crop کرده و بعد از Resize کردن به ورودی شبکه AdaBins داده می‌شود و نقشه عمق برای هر شئ جداگانه بدست می‌آید.



شکل ۱۱. حالت دوم اتصال دو شبکه

در این حالت از آنجا که تصویر هر شئ جداگانه وارد شبکه AdaBins می‌شود خطای این شبکه خیلی زیاد می‌شود چون عمق نسبی است و در نتیجه شبکه AdaBins برای آنکه تخمین خوبی بزند؛ لازم دارد کل تصویر را در ورودی داشته باشد تا درک کافی از محیط داشته باشد و اگر تصویر هر شئ جداگانه به شبکه AdaBins داده شود درک کافی از محیط در این شبکه صورت نمی‌گیرد و دقت آن افت خواهد کرد.

حالت سوم: در این حالت تصویر RGB اصلی به هر دو شبکه AdaBins و YOLOX وارد می‌شود. سپس با استفاده از BoundingBox هر شئ نقشه عمق مربوط به آن را از نقشه عمق کلی بدست آمده Crop کرده و عمق آن تخمین زده می‌شود.

شکل ۱۲. حالت سوم اتصال دو شبکه

این حالت طبیعتا بهترین حالت ممکن است چون هم شبکه AdaBins و هم شبکه YOLOX تصویر RGB کامل را در ورودی دریافت می‌کنند و در نتیجه بهترین پیش‌بینی را می‌کنند و با ادغام خروجی ۲ شبکه می‌توان بهترین نتیجه نهایی را دریافت کرد. اما ایراد این روش این است که برای پردازش هر تصویر زمان بیشتری نسبت به دو روش قبلی لازم است و در نتیجه Throughput سیستم کاهش پیدا می‌کند.

با توصیفات صورت گرفته ما حالت سوم را برای پیاده‌سازی خود انتخاب کردیم.

۳.۲ تخمین عمق هر شئ

از آنجا که BoundingBox های بدست آمده برای اشیا شامل پس زمینه نیز می‌شوند، اگر برای تخمین عمق جسم میانگین عمق بر روی کل آن BoundingBox گرفته شود شبکه دچار خطا می‌گردد؛ از همین رو نیاز به یک الگوریتم Segmentation با عملکرد مناسب و حجم پردازش کم داریم که بتواند شئ را از پس زمینه جدا کند تا میانگین گیری فقط روی همان مقادیر پیکسل‌های مربوط به شئ صورت پذیرد.

الگوریتم مورد استفاده برای این منظور Otsu Binarization [12] نام دارد. این الگوریتم با توجه به هیستوگرام هر تصویر بهترین مقدار آستانه را برای باینری کردن آن بدست می‌آورد و سپس با استفاده از این مقدار آستانه تصویر را باینری می‌کند و به این ترتیب شئ از پس زمینه جدا می‌شود. نحوه عملکرد الگوریتم به این صورت است که به گونه‌ای مقدار آستانه را انتخاب می‌کند که واریانس درون هریک از ۲ کلاس کمینه شود.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

که در آن:

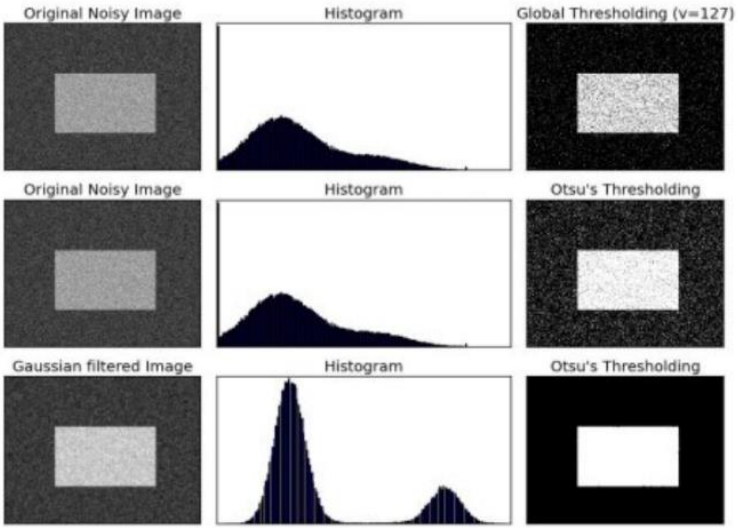
$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

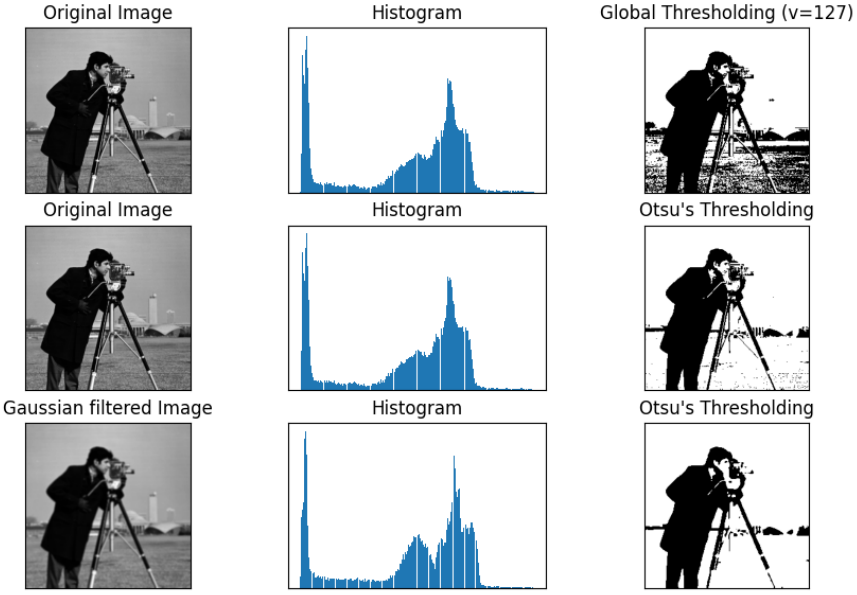
در روابط بالا t مقدار آستانه و P(i) تعداد پیکسل‌ها به ازای سطح روشنایی i می‌باشد.

این نکته مهم است که برای عملکرد بهتر هر الگوریتم Thresholding لازم است تا قبل از باینری کردن تصویر، اطلاعات فرکانس بالا و نویز تصویر حذف شوند که برای پیاده‌سازی این امر ما در سیستم خود تصویر را Blur می‌کنیم.



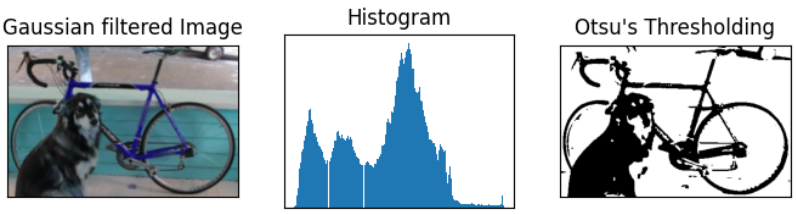
شکل۱۳. تاثیر الگوریتم معرفی شده

برای آنکه عملکرد این الگوریتم را ارزیابی کنیم تصویر CameraMan را تست کردیم.



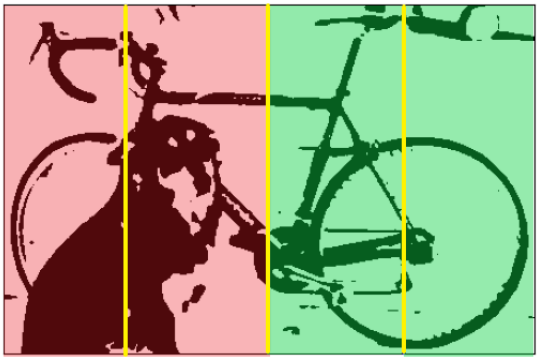
شکل۱٤. تاثیر الگوریتم بر تصویر CameraMan

گاهی اوقات در BoundingBox شئ موردنظر یک شئ دیگر در مقابل این شئ قرار دارد و بخشی از شئ مورد نظر را می‌پوشاند. با اعمال الگوریتم Otsu Binarization هم شئ مورد نظر و هم شئ مزاحم از پس زمینه جدا می‌شوند و وجود شئ مزاحم باعث می‌شود تا نتوانیم تخمین دقیقی از عمق شئ مورد نظر داشته باشیم.



شکل۱۵. وجود شئ مزاحم در BoundingBox

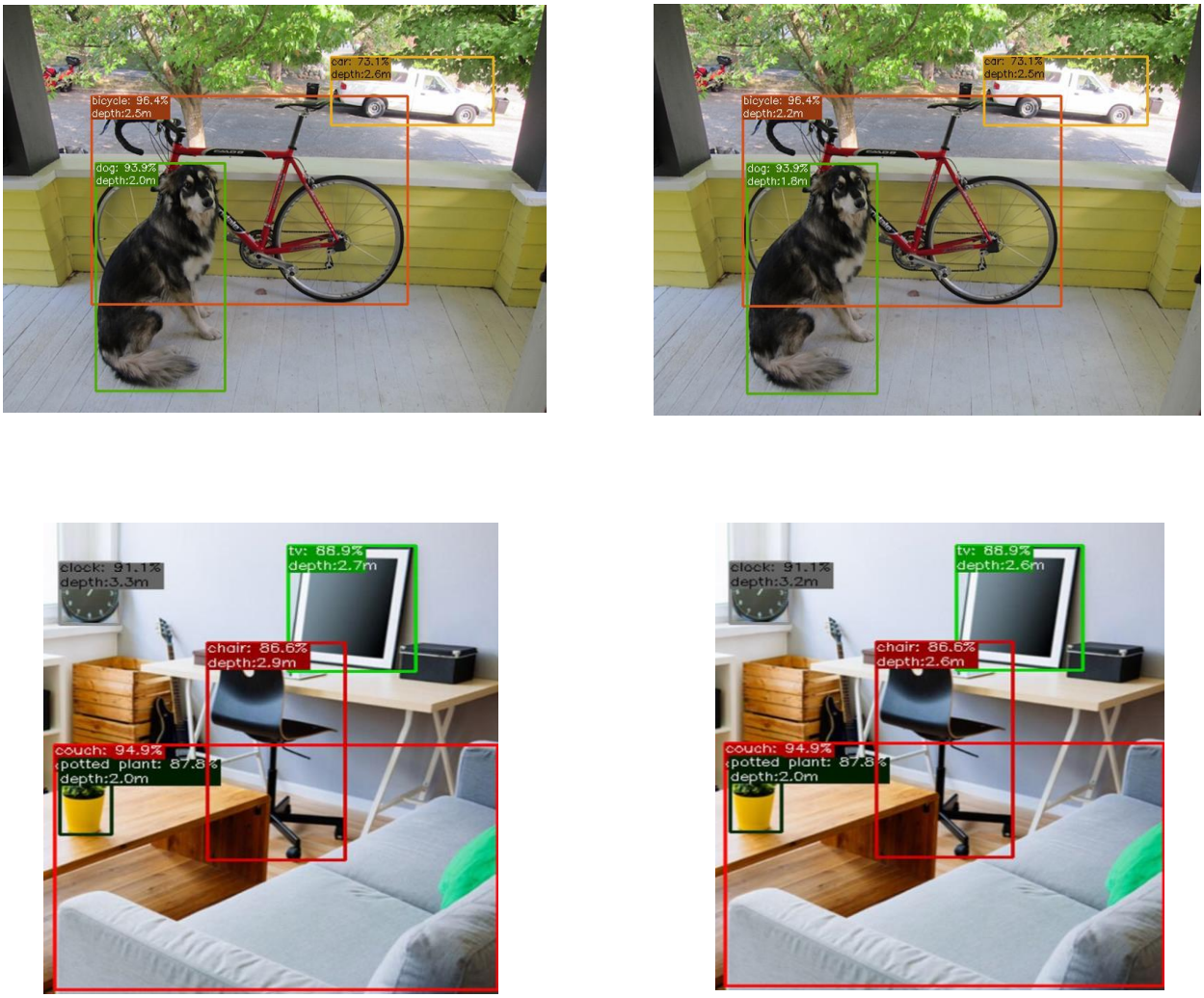
برای حل این مشـــکل پس اعمال الگوریتم Otsu Binarization ، BoundingBox موردنظر را با توجه به عمودی یا افقی بودن آن به ۴ قسمت تقسیم کرده و عمق شئ در هر یک از این ۴ قسمت جداگانه محاسبه می گردد و اگر عمق متوسط در یکی از قسمت‌ها از روند عمق متوســـط بقیه قســـمت‌ها پیروی نمی کرد احتمالا مشـــکلی دارد و حذف می گردد و میانگین عمق بقیه قســـمت‌ها گرفته می‌شود.(یعنی اگر سه قسمت باقی ماند، از سه مقدار عمق آنها میانگین گرفته می‌شود.)



شکل۱۶. حذف قسمت‌های نامناسب

عمده ایراد پدیده پوشانده شدن شئ زمانی است که شئ اصلی پیکسل‌های کمتری نسبت به شئ مزاحم داشته‌باشد ولی نکته مهم این است که در این الگوریتم حتی اگر قسمت مشکل‌دار BoundingBox هم حذف نشود باز هم اثر وجود شئ مزاحم کاهش می‌یابد زیرا سـهم آن شئ را در میانگین گیری نهایی کاهش می‌دهد (چون ۱ قسـمت از ۴ قسـمت مربوط به شـئ مزاحم خواهد بود و به اندازه ۰.۲۵ تاثیر دارد درحالی اگر روی پیکسل‌ها میانگین گرفته می‌شد اثر بیشتری روی محاسبه عمق نهایی می‌داشت).

در ادامه الگوریتم پیشـنهادی ارزیابی شـد. در شـکل۱۷ تصـاویر سـمت چپ خروجی شـبکه با اسـتفاده از میانگین گیری روی کل BoundingBox است و تصاویر سمت راست خروجی شبکه با استفاده از الگوریتم Segmentation می‌باشد. در دو تصویر بالا ، در حالت اسـتفاده از این الگوریتم عمق سـگ و دوچرخه کم‌تر تخمین زده شـده که ناشـی از حذف پس‌زمینه می‌باشـد. البته دقت تخمین عمق در این تصویر خیلی خوب نیست که علت آن تفاوت این تصویر با تصاویر دیتاست NYU Depth v2 است. در دو تصویر پایین، در حالت استفاده از این الگوریتم عمق صندلی به درستی کمتر تخمین زده شده‌است که ناشی از حذف پس‌زمینه بوده‌است.



شکل۱۷. تاثیر الگوریتم Segmentation در بهبود تخمین عمق اشیا

٤. معیار ارزیابی شبکه نهایی

در این بخش برای ارزیابی شبکه Joint سه معیار ارائه شده است. معیار اول که بر اساس Precision و Recall و محاسبه AP استوار است توسط خودمان طراحی، معیارهای دوم و سوم به ترتیب از ایده‌های مطرح شده در [13] و [2] الهام گرفته شده است.

٤.١ معیار اول – DepthPR

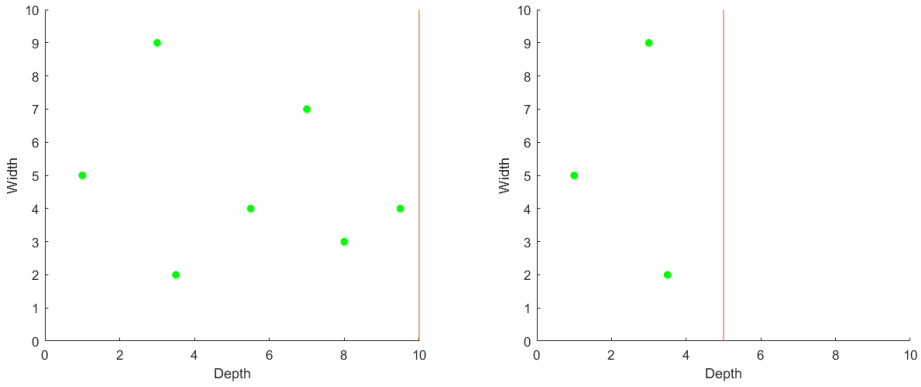
مسئله‌ای که شبکه نهایی به دنبال حل آن است، مسئله پیدا کردن اشیائی است که از یک آستانه‌ای به دوربین نزدیک‌تر باشند. به همین دلیل لازم است برای ارزیابی دقت شبکه در تشخیص اینگونه اشیا با در نظر گرفتن دقت خروجی شبکه‌های Detector و Estimator، معیاری بر اساس میزان آستانه (Threshold) تعریف نماییم.

معیار پیشنهادی ما عملکرد شبکه را در تشخیص اشیائی که از مقدار آستانه مشخصی نزدیکتر هستند می‌سنجد. بر این اساس پارامترهای TP,FP,TN,FN را صرفا بر اساس عمق اشیا تخمین زده شده به صورت زیر تعریف می‌نماییم.

- TP: اگر هم عمق پیشبینی شده و هم عمق GT برای شئ از مقدار آستانه کمتر بوده است.
- FP: اگر عمق پیشبینی شده کمتر از مقدار آستانه ولی عمق GT برای شئ از مقدار آستانه بیشتر بوده باشد.
- TN: اگر هم عمق پیشبینی شده و هم عمق GT برای شئ از مقدار آستانه بیشتر بوده است.
- FN: اگر عمق پیشبینی شده بیشتر از مقدار آستانه ولی عمق GT برای شئ از مقدار آستانه کمتر بوده باشد.

هدف آن است که با تعریف این پارامترها و با استفاده از معیار ارزیابی δ_2، معیارهای Precision و Recall برای ارزیابی عمق بر اساس مقدار آستانه بدست آیند. سپس می‌توانیم با ترکیب هر کدام از این دو معیار با معیارهای متناظر بدست آمده از Object Detector تاثیر دقت شبکه شناساگر در تشخیص اشیا را به معیار پیشنهادی اضافه نماییم. در نهایت یک Precision و Recall برای شبکه Joint خواهیم داشت و میتوان با محاسبه متریک AP برای آن به یک عدد برای ارزیابی و مقایسه شبکه های Joint دست پیدا کرد.

نکته ای که لازم است در نظر گرفته شود آن است که برای عملکرد صحیح معیاری که در ادامه ارائه خواهد شد، لازم است تا با در نظر گرفتن مقدار آستانه عمق، GT های تصاویر ورودی تعریف شوند. برای مثال اگر مقدار آستانه عمق برابر با ۵ متر در نظر گرفته می‌شود، تمامی GT های تصاویر ورودی دارای حداکثر عمق ۵ متر باشند. شکل زیر نمونه‌ای از این اصلاح GT ها را نشان می‌دهد.



شکل۱۸. با کاهش مقدار آستانه برخی از اشیا از GT حذف شده‌اند.

در واقع شبکه تشخیص اشیا با تشخیص اشتباه شئ و شبکه تخمین عمق با تخمین اشتباه عمق شئ به نحوی که باعث شود آن شی با توجه به مقدار آستانه، به اشتباه حذف یا اضافه گردد باعث کاهش دقت سیستم کلی می‌شوند. در نتیجه در معیار پیشنهادی ما عملکرد دو شبکه Estimator و Detector در نظر گرفته شده است.

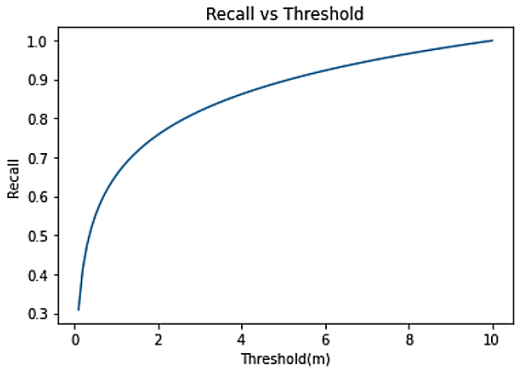
٤.١.١. تخمین معیار اول – DepthPR

برای آنکه بتوان معیار مورد نظر شبکه کلی را تخمین زد باید دو معیار Recall و Precision برای شبکه DepthEstimation تخمین زده شوند و سپس برای بدست آورد AP نهایی، حاصل عبارت زیر محاسبه شود.

$$AP_{Joint} = Precision_{Depth} \times Recall_{Depth} \times AP_{Detection}$$

در گام اول نیاز است معیار Recall را برای شبکه تخمین عمق برحسب مقدار آستانه تخمین بزنیم؛ به این منظور با استفاده از معیار δ_2 این شبکه معیار Recall را تخمین می‌زنیم. معیار δ_2 در شبکه AdaBins برابر۰.۹۸۴ می‌باشد، یعنی در بدترین حالت عمق تخمین زده شده برای حدود ۰.۹۸ درصد اشیا در بازه $\frac{2}{3}p < \hat{p} < \frac{3}{2}p$ قرار می‌گیرد ($\hat{p}$: عمق تخمین زده شده برای هر شئ، p: عمق واقعی و $1.25^2 \cong \frac{3}{2}$ در نظر گرفته شده است). با توجه به مطالب بیان شده در بخش ۱.۳.۱، با اینکه شبکه می‌تواند فواصل کمتر از ۱ را تشخیص دهد، اما به دلیل آنکه دیتاستNYU Depth v2 دارای فواصل بیشتر از ۱ متر است، ما کمترین مقدار فاصله را ۱ متر در نظر می‌گیریم. بنابراین به ازای مقدار آستانه ۱، مقدار عمق پیش بینی شده توسط شبکه به احتمال $\frac{2}{3}\delta_2 = 0.98 \times 0.66$ در بازه درست تخمین زده می‌شود و هرچه مقدار آستانه بزرگتری در نظر گرفته شود؛ این مقدار بزرگتر می‌شود تا نهایتا در آستانه ۱۰ متر (ماکزیمم فاصله‌ای که شبکه قادر به پیش‌بینی آن است) به ۱ می‌رسد. حال مسئله بعدی آن است که این روند افزایش به سمت ۱ چگونه خواهد بود. با توجه به اینکه شبکه Detection و Estimation اشیای نزدیک را بهتر از اشیای دور پیش‌بینی می‌کنند، بنابراین برای تخمین درست‌تر، این روند افزایشی را به صورت لگاریتمی در نظر می‌گیریم. به عبارت دیگر تغییر مقدار آستانه برای فواصل نزدیک تاثیر بیشتری بر روی Recall نهایی دارد تا برای فواصل دورتر. در نهایت Recall با فرض حداکثر عمق ۱۰ متر، بر حسب مقدار آستانه از رابطه زیر پیروی می‌نماید.

$$Recall(T) = 0.15\ln(T) + \frac{2}{3}\delta_2 \rightarrow Recall(T) = \begin{cases} \frac{2}{3}\delta_2, & T = 0m \\ 1, & T = 10m \end{cases}$$



شکل۱۹. منحنی Recall بر حسب آستانه عمق

سپس نیاز است معیار Precision را برای شبکه تخمین عمق برحسب مقدار آستانه تخمین بزنیم؛ به این منظور با استفاده از معیار δ_2 این شبکه معیار Precision را تخمین می‌زنیم. معیار δ_2 شبکه AdaBins برابر۰.۹۸۴ می‌باشد، یعنی در بدترین حالت عمق تخمین زده شده برای حدود ۰.۹۸ درصد اشیا در بازه $\frac{2}{3}p < \hat{p} < \frac{3}{2}p$ قرار می‌گیرد. مثال زیر را در نظر می‌گیریم:

شکل۲۰. مثالی از تقسیم بندی عمق بر اساس معیارهای کلاسبندی

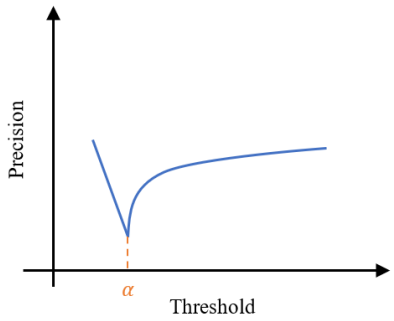
با تقریب می‌توان گفت قسمت سبز مربوط به TP ، قرمز FN ، آبی FP و زرد TN می‌باشد. همانطور که در شکل بالا قابل مشاهده است، با فرض مقدار آستانه در ۶ متری اگر عمق تخمین زده شده برای یک پیکسل در بازه صفر تا ۴ متر و یا ۹ تا ۱۰ متر باشد، میدانیم معیار δ_2 آن را با دقت ۹۸ درصد درست تخمین می‌زند. پس تنها بخشی که ممکن است دچار خطا شویم از بازه ۴ تا ۹ متر است. به عبارت دیگر Precision بر اساس آستانه از رابطه زیر بدست می‌آید:

$$Precision\ (T) = \begin{cases} \dfrac{10 - 2 \times \frac{1}{3} \times T}{10}, & T < \alpha \\ 0.15 \ln(T) + \dfrac{2}{3}\delta_2, & T \geq \alpha \end{cases}$$

نکته‌ای که وجود دارد آن است که با شروع از مقدار آستانه ۱، هرچه مقدار آستانه را افزایش دهیم، بخش سبز رنگ در شکل بالا بزرگتر می‌شود و با توجه به نکته بیان شده در بخش ۱.۳.۱، به دلیل آنکه اکثر اشیا در این فاصله قرار دارند، پس تاثیر افزایش مقدار آستانه در مقادیر عمق کمتر بیشتر از عمق‌های بزرگتر است.

از طرفی با افزایش مقدار آستانه بازه قرمز رنگ به حد نهایی خود یعنی ۱۰ متر نزدیک تر شده و به دلیل آنکه شبکه قادر به تخمین عمق‌های بیشتر از ۱۰ متر نیست، از ۱۰متر به بعد دیگر ناحیه قرمز رشد نکرده و در نتیجه از عرض آن کاسته می‌شود. بنابراین با افزایش مقدار آستانه، مقدار Precision افزایش می‌یابد.

با توجه به سه نتیجه گیری‌ای که در مورد Precision در عبارات بالا مطرح شد، می‌توان نتیجه گرفت که منحنی Precision باید دارای رفتاری به شکل زیر باشد.



شکل ۲۱. منحنی موردانتظار Precision بر حسب آستانه

پس از بررسی بر روی دیتاست NYU Depth v2، مقدار α کمتر از یک بدست آمد و به دلیل آنکه ما مبنای کار شبکه را از مقادیر بزرگتر از یک در نظر گرفته‌ایم، برای راحتی برای تخمین Precision تنها از بخش لگاریتمی استفاده می‌کنیم و مقدار $\alpha = 1$ در نظر می‌گیریم.

در نتیجه Precision به ازای مقدار آستانه ۱ در بدترین حالت قرار می‌گیرد و مقدار آن برابر با $\frac{2}{3}\delta_2 = 0.98 \times 0.66$ می‌شود اما هرچه این مقدار آستانه بزرگتر باشد شرایط بهتر می‌شود و به ازای مقدار آستانه ۱۰، Precision برابر ۱ می‌شود

نکته‌ای که در نمودارهای فوق وجود دارد این است که برخلاف معمول، به ازای مقادیر آستانه بزرگتر از α نمودارهای Precision و Recall هردو روند صعودی دارند و افزایش Recall باعث کاهش Precision نشده‌است. علت آن است که در شبکه AdaBins مقدار بیشینه‌ای برای عمق در نظر گرفته شده است و طبیعتا هرچه مقدار آستانه به این بیشینه مقدار نزدیکتر باشد، مقدار Recall بیشتر می‌شود زیرا تعیین ماکزیمم برای عمق باعث شده تا شبکه با احتمال کمتری مقادیر بزرگی را در خروجی بدهد و این باعث می‌شود که برای اشیایی که نزدیکتر از آستانه هستند با احتمال کمتری عمق بزرگتر از آستانه تخمین زده شود. همین امر در مورد Percision هم صادق است. یعنی هر چه مقدار آستانه بزرگتر باشد اشیای کمتری در عمق بزرگتر از آستانه قرار می‌گیرند و تاثیر گذاری این اشیا در مقدار Percision کم‌تر می‌شود و در نتیجه تقریبا روندی مثل Recall را طی می‌کند ولی به ازای مقادیر آستانه کم همان روند عادی دنبال می‌شود یعنی افزایش Recall باعث کاهش Percision شده است.

حال که مقادیر Recall و Precision برای شبکه تخمین عمق را داریم کافی است تا با ضرب این دو مقدار در Recall و Precision شبکه ObjectDetection مقادیر Recall و Precision شبکه کلی بدست می‌آید و با استفاده از این دو مقدار معیار AP برای هر مقدار آستانه جداگانه بدست می‌آید و با میانگین گیری از آنها می‌توان یک AP کلی بدست آورد. با توجه به اینکه برای محاسبه AP نیاز به اندازه‌گیری سطح زیر منحنی Precision×Recall داریم، در نتیجه استفاده از رابطه زیر معادل محاسبه AP برای Precision و Recall کلی سیستم است.

$$AP_{Joint} = Precision_{Depth} \times Recall_{Depth} \times AP_{Detection}$$

روابطی که در بالا برای Precision و Recall بر حسب T و δ_2 مطرح شد، در مینیمم‌ترین حالت ممکن بدست آمده است. اما شرایطی وجود دارد که ممکن است بازهم خروجی شبکه Depth Estimation درست بوده ولی معیار ما آن را در نظر نگرفته باشد. برای مثال ما در مورد معیار Precision فاصله $(T - \frac{1}{3}T, T + \frac{1}{3}T)$ را به عنوان فاصله ای که شبکه در آن اشتباه پیش‌بینی می‌کند در نظر گرفتیم، درصورتی که ممکن است برای مثال عمق واقعی یک پیکسل در بازه $(T - \frac{1}{3}T, T)$ بوده و پیش بینی شبکه از عمق

همچنان در همین فاصله باشد و از مقدار آستانه (T) بیشتر نشود که از TP به FN تغییر یابد. برای همین در روابط بالا لازم است که یک مقدار ϵ به $\frac{2}{3}\delta_2$ اضافه شود تا این حالت‌های خاص را پوشش دهد. پس از تست معیار بر روی تصاویر دیتاست NYU Depth v2، مقدار $\epsilon \approx 0.1$ بدست آمد.

٤.١.٢. پیاده‌سازی عملی معیار اول – DepthPR

برای پیاده سازی عملی معیارپیشنهاد شده در بخش قبل، تابع Depth_PR در فایل Eval.py پیاده سازی شده است. نکته‌ای که لازم است در پیاده‌سازی عملی در نظر گرفته شود آن است که ماکزیمم عمق برای هر تصویر باید به صورت مجزا در نظر گرفته شود. برای همین روابط بالا (که با فرض حداکثر عمق ۱۰ متر بدست آمده بود) در پیاده سازی عملی و برای حالت کلی به صورت زیر خواهد بود. (تاثیر مقدار $\epsilon = 0.1$ در روابط در نظر گرفته شده است)

$$Recall(T) = a_r \ln(T) + \frac{2}{3}\delta_2 + 0.1 \rightarrow Recall(T) = \begin{cases} \frac{2}{3}\delta_2, & T = 0m \\ 1, & T = \max(depth)\, m \end{cases},$$

$$where: \ a_r = \frac{0.9 - \frac{2}{3}\delta_2}{\ln(\max(depth))}$$

$$Precision\ (T) = \begin{cases} \frac{10 - 2 \times \frac{1}{3} \times T}{10}, & T < \alpha \\ a_p \ln(T) + \frac{2}{3}\delta_2 + 0.1, & T \geq \alpha \end{cases}$$

$$where: \ \ where: \ a_p = \frac{0.9 - \frac{2}{3}\delta_2}{\ln(\max(depth))}$$

٤.١.٣. محاسبه عددی مقدار تخمین زده شده معیار اول – DepthPR

با فرض $\max(depth) = 10$، $T = 5$، $\delta_2 = 0.98$ و براساس مقادیر AP گزارش شده در مقاله YOLOX، مقدار تخمینی AP کلی سیستم را محاسبه می‌نماییم.

$$a_p = a_r = \frac{0.9 - \frac{2}{3}\delta_2}{\ln(\max(depth))} = 0.107$$

$$Precision_{Depth} = a_p \ln(T) + \frac{2}{3}\delta_2 + 0.1 = 0.107 \ln(5) + \frac{2}{3}0.98 + 0.1 = 0.925$$

$$Recall_{Depth} = a_r \ln(T) + \frac{2}{3}\delta_2 + 0.1 = 0.107 \ln(5) + \frac{2}{3}0.98 + 0.1 = 0.925$$

$$AP_{Detection} = 0.464$$

بنابراین AP_{Joint} به صورت زیر بدست می‌آید.

$$\begin{aligned} AP_{Joint} &= Precision_{Depth} \times Recall_{Depth} \times AP_{Detection} \\ &= 0.925 \times 0.925 \times 0.464 \\ &= 0.397 \end{aligned}$$

٤.٢. معیار دوم – Object Depth Evaluation [13]

ایرادی که متریک‌های مطرح شده در بخش ۱.۴.۱ برای ارزیابی شبکه Joint دارند آن است که خطا را بر روی کل تصویر و نه بر روی Bounding Box های پیش‌بینی شده بدست می‌آورند. برای در نظر گرفتن نتایج Detection، مائوری و همکارانش در مقاله [13] روشی به نام Object Depth Evaluation ارائه داده‌اند که روند آن از ۴ قسمت تشکیل شده است.

۱- ابتدا نقشه عمق (Depth Map) بدست آمده با استفاده از Median Scaling، Scale می‌شود. یعنی میانه آن صفر و واریانس حول میانه آن یک می‌شود.

۲- ماسک‌های اشیا توسط یک شبکه Segmentation نظیر Mask-RCNN [14] بدست می‌آید. در کار ما این بخش از خروجی بخش Segmentation که بر روی هر Bbox زده می‌شود بدست می‌آید.

۳- ماسک‌های بدست آمده بر روی نقشه ویژگی قرار می‌گیرند و معیارهای خطا را برای پیکسل‌های آبجکت موردنظر محاسبه می‌کنند.

۴- خطاهای بدست آمده برای هر کلاس روی تعداد اشیا آن کلاس میانگین گرفته می‌شود.

نمونه‌ای از نتایج اعمال این معیار بر روی تصاویر دیتاست KITTI با استفاده از دو شبکه MonoDepth2 و BTS در جدول زیر آورده شده است.

جدول٦. نتیجه اعمال معیار ارائه شده بر روی دیتاست KITTI در کلاس‌های مختلف

TABLE II

OBJECT DISTANCE EVALUATION ON KITTI. THE ALGORITHMS EVALUATED ARE STATE-OF-THE-ART MONOCULAR DEPTH ESTIMATION METHODS: MONODEPTH2 (MD2) AND BTS. DEPTH ERRORS WERE COMPUTED FOR THE OBJECT CLASSES WITH ENOUGH INSTANCE IN THE TEST SPLIT. BOTH SRE AND RMSE ARE EXPRESSED IN METERS.

	RE		SRE		RMSE		logRMSE		a_1		a_2		a_3	
Object class	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS	MD2	BTS
Person	0.314	**0.166**	5.721	**1.786**	8.43	**5.892**	0.326	**0.253**	0.601	**0.772**	0.829	**0.894**	0.92	**0.947**
Bicycle	0.131	**0.116**	0.517	**0.467**	2.81	**2.669**	0.172	**0.163**	0.829	**0.839**	**0.964**	0.962	0.993	**0.994**
Car	0.206	**0.137**	3.132	**1.491**	7.924	**6.052**	0.271	**0.223**	0.773	**0.838**	0.883	**0.922**	0.938	**0.955**
Truck	0.215	**0.122**	2.769	**0.826**	6.978	**4.523**	0.259	**0.177**	0.694	**0.854**	0.903	**0.969**	0.964	**0.985**

٤.٣. معیار سوم

معیاری که در اینجا معرفی می‌شود؛ معیاری دیگر برای شبکه‌های تخمین عمق می‌باشد. استفاده از این معیار می‌تواند ارزیابی خوبی برای صحت و کیفیت(عدم وجود نویز) نقشه عمق خروجی باشد. رابطه کلی این معیار به صورت زیر است.

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

ترم اول در رابطه بالا میزان درستی مقادیر تخمین زده شده در مقایسه مقادیر واقعی عمق را بیان می‌کند.

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

ترم دوم شامل جمع اختلاف گرادیان‌های در راستاهای xوy بین نقشه عمق واقعی و نقشه عمق پیش‌بینی شده توسط شبکه می‌باشد. این ترم از رشد گرادیان در نقشه عمق پیش‌بینی شده توسط شبکه جلوگیری کرده و در نتیجه باعث نرم شدن خروجی نهایی می‌شود.

$$L_{grad}(y,\hat{y}) = \frac{1}{n}\sum_{p}^{n} |\boldsymbol{g}_{\mathbf{x}}(y_p,\hat{y}_p)| + |\boldsymbol{g}_{\mathbf{y}}(y_p,\hat{y}_p)|$$

ترم سوم بر حسب SSIM بدست می‌آید که غالبا در کاربردهای بازسازی تصویر استفاده می‌شود از آنجا که حد بالای SSIM برابر یک می‌باشد؛ این ترم به صورت زیر تعریف شده‌است.

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

٤. راهنمای عملی استفاده از شبکه

در این بخش شیوه استفاده از شبکه را شرح می‌دهیم.

٤.١ تنظیمات اولیه

برای دسترسی به کدها به لینک زیر مراجعه فرمایید.

https://github.com/amindehnavi/DL2021_FinalProject

برای دانلود وزن‌های شبکه AdaBins از لینک زیر استفاده فرمایید.

https://drive.google.com/drive/folders/1wUDM4fUDUV4LDhmVqts5p8-AIgnVlZsm?usp=sharing

وزن‌های شبکه AdaBins را در پوشه DL2021_FinalProject/AdaBins/pretrained قرار دهید.

برای دانلود وزن‌های شبکه YOLOX از لینک زیر استفاده فرمایید.

https://drive.google.com/drive/folders/1wUDM4fUDUV4LDhmVqts5p8-AIgnVlZsm?usp=sharing

وزن‌های شبکه YOLOX را در پوشه DL2021_FinalProject/YOLOX/pretrained قرار دهید.

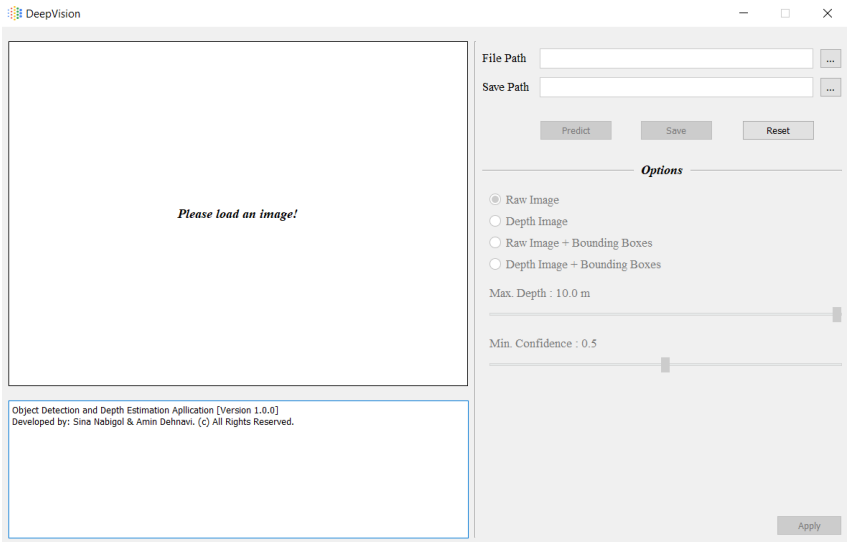
پکیج‌های موردنیاز در فایل requirements.txt قرار داده شده‌است؛ آنها با دستور زیر نصب بفرمایید.

```
pip3 install -r ./requirements.txt
```

پس از انجام مراحل فوق، با اجرای فایل GUI.py؛ رابط گرافیکی برنامه فعال شـــده و می‌توان با دادن تصـــویر در ورودی، خروجی‌های موردانتظار را مشاهده نمود.

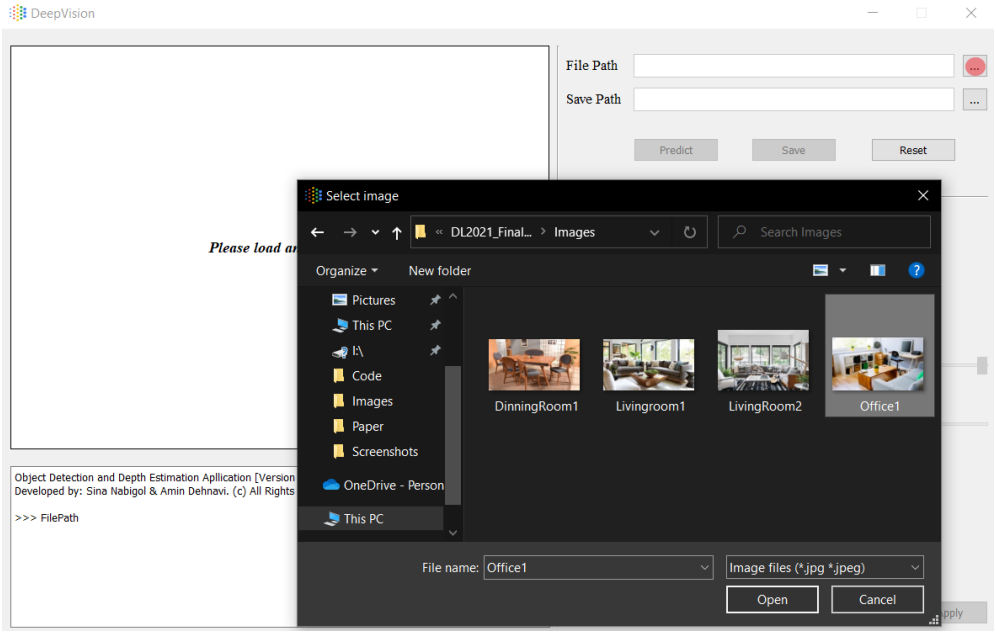
٤.٢ رابط گرافیکی (GUI)

نمای کلی GUI به صورت زیر می‌باشد.



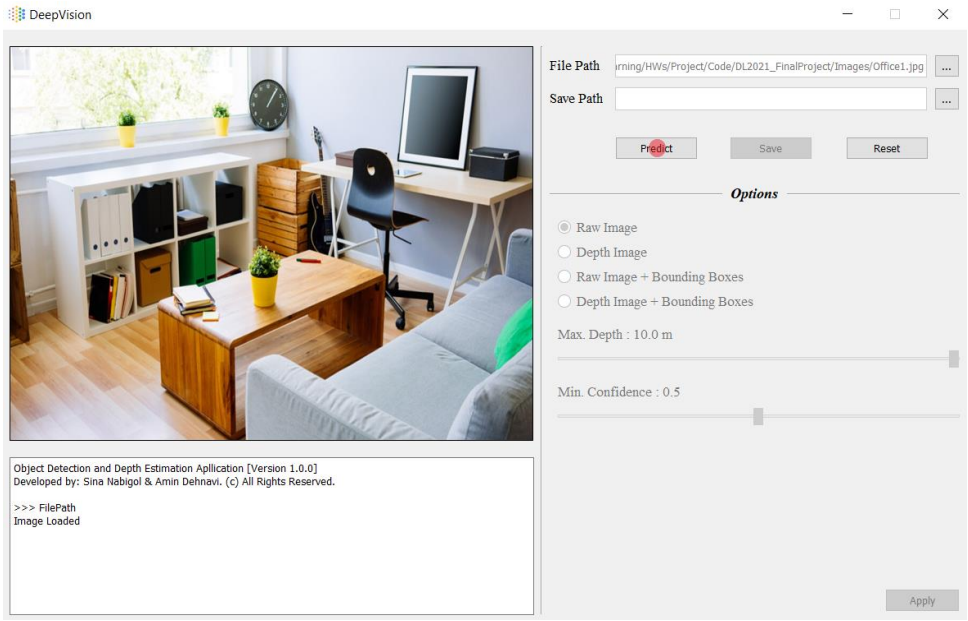
شکل۲۲. نمای رابط گرافیکی

ابتدا باید یک فایل تصویر به فرمت jpg. یا jpeg. انتخاب گردد. از آنجا که تصاویر ورودی به شبکه AdaBins باید ۶۴۰×۴۸۰ باشند؛ همه تصاویر ورودی به این ابعاد Resize می‌شوند.



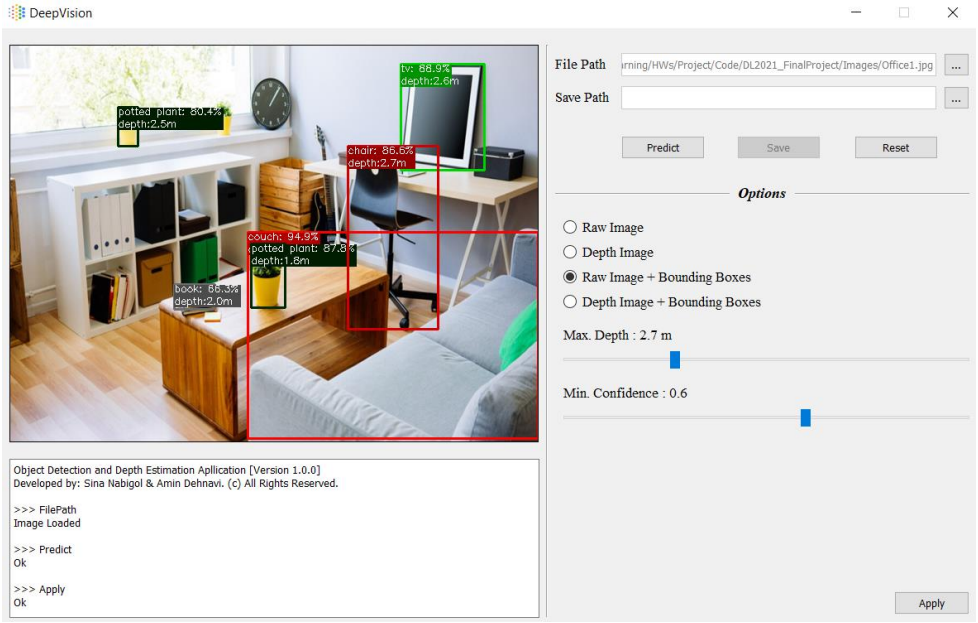
شکل۲۳. مراحل انتخاب تصویر ورودی

حال با کلیک بر روی Predict شبکه‌های AdaBins و YOLOX فعالیت‌های مربوط به خود را انجام می‌دهند.



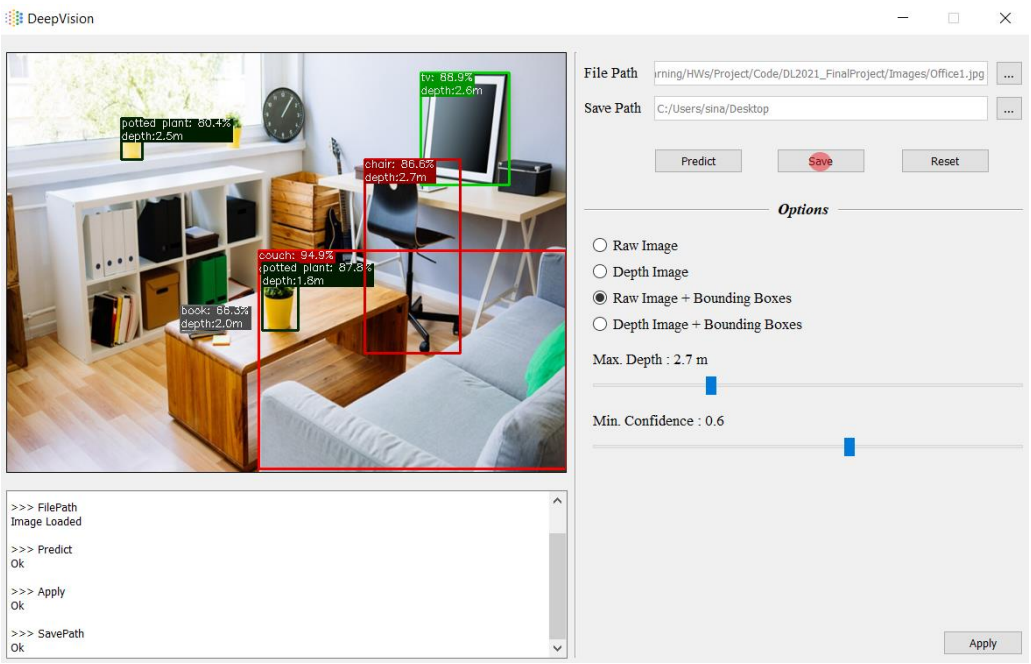
شکل۲٤. نحوه پیش‌بینی کردن شبکه

پس از اتمام کار Prediction منوی Options فعال می‌شـــود. در این منو با انجام تنظیمات موردنظر و کلیک بر روی Apply تصویر نمایش داده‌شده مطابق با تنظیمات موجود تغییر می‌کند.



شکل۲۵. تغییر تنظیمات برای رسیدن به خروجی مورد نظر

در هر مرحله پس از تعیین پوشه ذخیره تصاویر با کلیک بر روی save تصویر نمایش داده شده، ذخیره می‌گردد.



شکل۲۶. ذخیره تصویر خروجی

۵. مراجع

[1] S. F. Bhat, I. Alhashim, and P. Wonka, “AdaBins: Depth Estimation Using Adaptive Bins,” 2021, doi: 10.1109/CVPR46437.2021.00400.

[2] I. Alhashim and P. Wonka, “High Quality Monocular Depth Estimation via Transfer Learning,” 2018, [Online]. Available: http://arxiv.org/abs/1812.11941.

[3] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep Ordinal Regression Network for Monocular Depth Estimation,” 2018, doi: 10.1109/CVPR.2018.00214.

[4] M. Is, R. For, and E. At, “An image is worth 16x16 words,” *Int. Conf. Learn. Represent.*, 2021.

[5] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, “YOLOX: Exceeding YOLO Series in 2021,” pp. 1–7, 2021, [Online]. Available: http://arxiv.org/abs/2107.08430.

[6] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018, [Online]. Available: http://arxiv.org/abs/1804.02767.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” Apr. 2020, Accessed: Aug. 30, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[8] glenn jocher et al., “Yolov5,” *ultralytics/yolov5: v6.0 - YOLOv5n “Nano” models, Roboflow integration, TensorFlow export, OpenCV DNN support (v6.0). Zenodo*, 2021. https://github.com/ultralytics/yolov5.

[9] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “MixUp: Beyond empirical risk minimization,” 2018.

[10] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as Points,” 2019, [Online]. Available: http://arxiv.org/abs/1904.07850.

[11] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “CenterNet: Keypoint triplets for object detection,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 6568–6577, 2019, doi: 10.1109/ICCV.2019.00667.

[12] Liu Jianzhuang, Li Wenqing, and Tian Yupeng, “Automatic thresholding of gray-level pictures using two-dimension Otsu method,” 2002, doi: 10.1109/ciccas.1991.184351.

[13] A. Mauri, R. Khemmar, R. Boutteau, B. Decoux, J. Y. Ertaud, and M. Haddad, “A new Evaluation Approach for Deep Learning-based Monocular Depth Estimation Methods,” *2020 IEEE 23rd Int. Conf. Intell. Transp. Syst. ITSC 2020*, 2020, doi: 10.1109/ITSC45102.2020.9294620.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020, doi: 10.1109/TPAMI.2018.2844175.