



دانشگاه صنعتی شریف  
دانشکده مهندسی برق

## گزارش پروژه یادگیری عمیق

نگارش

پوریا اشرفیان (۹۶۱۰۱۲۲۷)

میلااد صمیمی فر (۴۰۰۲۰۵۵۷۷)

بهمن ۱۴۰۰

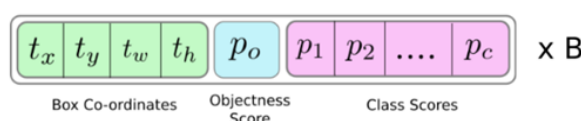


## فهرست مطالب

ج	فهرست مطالب
۱	۱ تشخیص اشیاء
۵	۲ تشخیص عمق
۵	۱.۲ ساختار CycleGAN
۶	۲.۲ ساختار Unet
۹	۳ آموزش مشترک
۱۲	۴ وب اپلیکیشن
۱۲	۱.۴ Streamlit
۱۲	۲.۴ Django
۱۲	۳.۴ چالش ها
۱۵	مراجع

## فصل ۱ تشخیص اشیاء

در این قسمت از مدل آماده YOLO-v3 استفاده کردیم. این مدل با توجه به ساختاری که دارد در پردازش های real-time استفاده می شود و سرعت بالایی در یافتن اشیاء دارد.



شکل ۱.۱: خروجی های شبکه YOLO-v3

همان طور که از تصویر ۱.۱ مشخص است خروجی این مدل حاوی مختصات box bounding های اشیاء تشخیص داده شده است که در قسمت اتصال دو مدل تشخیص عمق و تشخیص اشیاء از این خاصیت استفاده می کنیم. در ادامه روش اصلی و مزیت های YOLO-v3 نسبت به ورژن های قبلی YOLO را بیان می کنیم. در این روش برای تشخیص صحیح تر داده هایی که دارای اشیاء با پراکندگی اندازه های بالا<sup>۱</sup> هستند، چند روش استفاده شده که یکی از این روش ها Predictions Across Scales نام دارد. برای این کار از مفهوم Feature Pyramid Networks بهره گرفته شده است. در این شبکه برای هر scale سه باکس پیش بینی می شود.

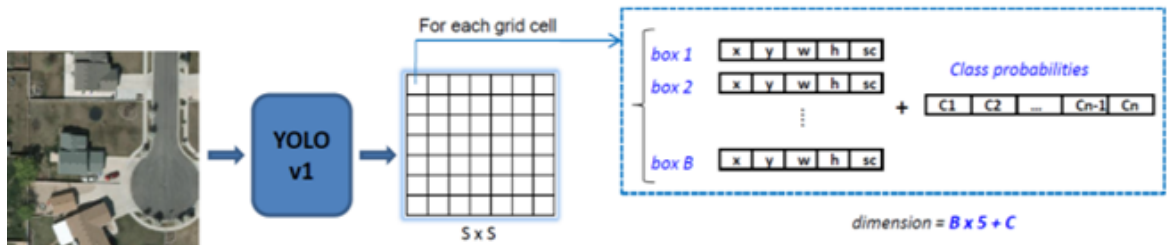
در مرحله بعد feature map دو لایه پیشین را با ضریب ۲ upsample کرده و با feature map خود ترکیب می کنیم (concatenate). با همین روش scale سوم را هم می سازیم. با توجه به این که در مرحله آخر از مراحل قبلی استفاده می کنیم، از محاسبات قبلی به نحو احسن استفاده می شود و مشکل پراکندگی اندازه های بالا کمتر پیش خواهد آمد. در شکل های ۲.۱ و ۳.۱ به ترتیب نمای کلی و ساختار شبکه YOLO-v3 آمده است.

معیار ارزیابی مدلی که ما استفاده کردیم mAP<sup>۲</sup> است و مقدار آن برای این مدل ۵۷/۹ درصد است. این مدل از [pjreddie.com/darknet/yolo](https://pjreddie.com/darknet/yolo) قابل دانلود است.

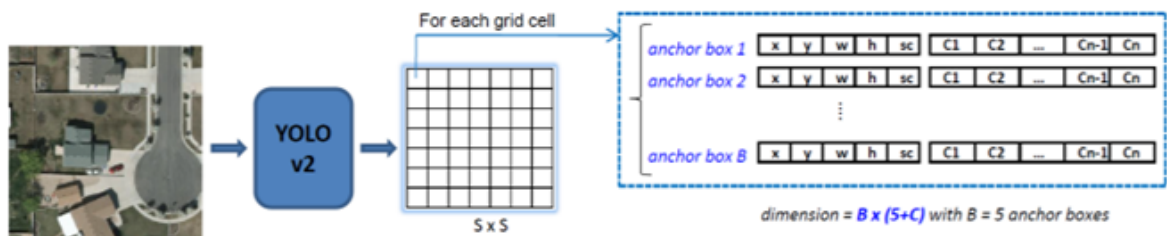
در شکل ۴.۱ تابع هزینه شبکه YOLO-v2 آمده است. در شبکه ما به جای مربعات از آنتروپی متقابل استفاده می شود. بنابراین به نوعی در این ورژن از logistic regression برای به محاسبه confidence و predictions استفاده می کنیم. مفهوم کلی این تابع هزینه به این معنی است که برای هر ground truth box باکس خود را تولید می کنیم به

<sup>۱</sup>High Size Variation

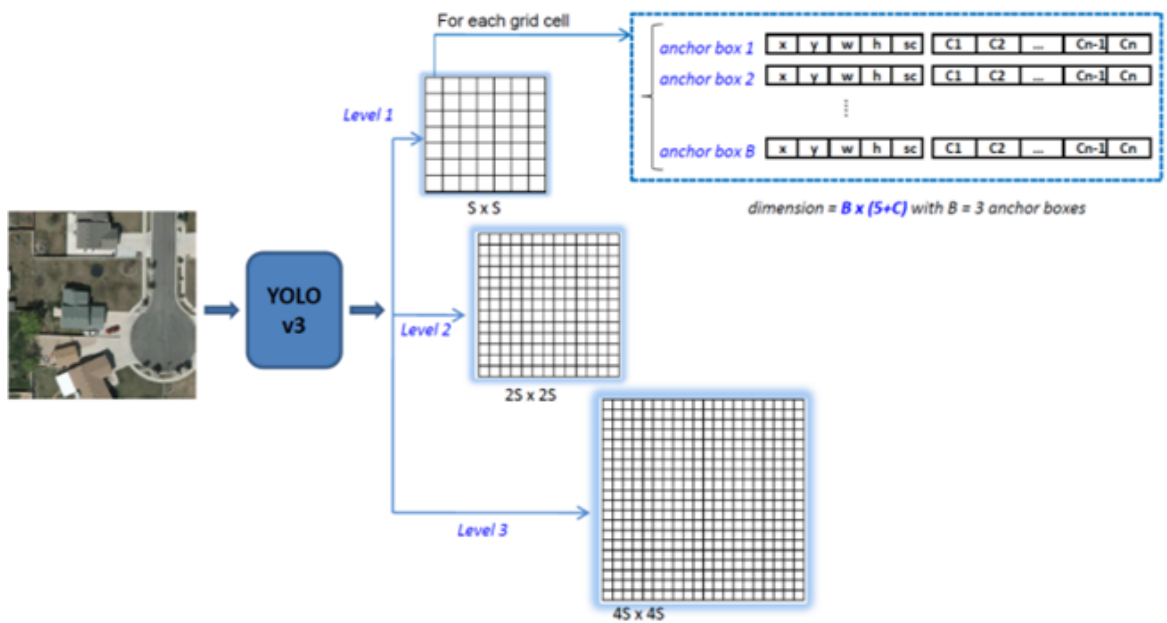
<sup>۲</sup>Mean Average Precision



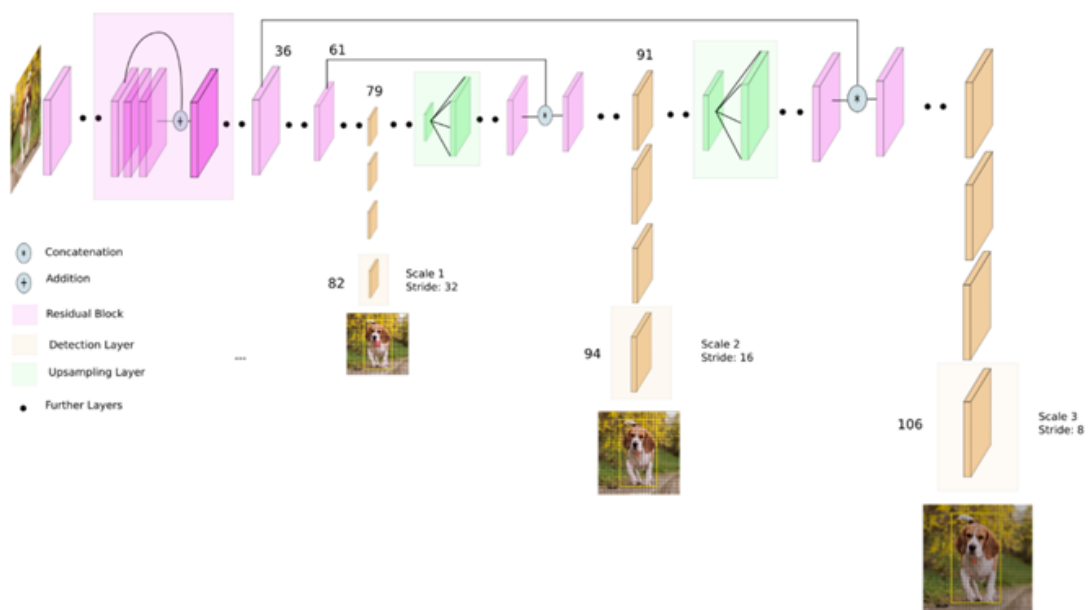
(a)



(b)



شکل ۲.۱: نمای کلی YOLO-v3



شکل ۳.۱: ساختار شبکه YOLO-v3

طوری که بیشترین همپوشانی را با باکس اصلی داشته باشد.

Regression  
loss

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

Confidence  
loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\ + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

Classification  
loss

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

شکل ۴.۱: تابع هزینه YOLO-v2

## فصل ۲ تشخیص عمق

### ۱.۲ ساختار CycleGAN

این روش اولین بار در [۱] ارائه شد. در این روش به دنبال یافتن دو تبدیل  $G$  و  $F$  هستیم که از دامنه تصاویر RGB به تصاویر عمق و برعکس کار می‌کنند. همچنین به دنبال این هستیم که این دو تبدیل معکوس یکدیگر باشند. تابع هزینه مورد استفاده در این روش در شکل ۱.۲ آمده است. علاوه بر شبکه‌های  $G$  و  $F$ ، دو شبکه تمایزدهنده برای  $D_X$  و  $D_Y$  برای دامنه‌های تصاویر اصلی و تصاویر عمق وجود دارند که برخلاف شبکه‌های مولد عمل می‌کنند. ترم اول و دوم مربوط به این شبکه‌هاست و به صورت میانگین مربعات تفاضل پیکسل‌های لابه آخر با ماتریس یکانی محاسبه می‌شوند. ترم سوم برای اجبار به معکوس بودن تبدیل‌های مولد است و در شکل ۲.۲ آمده است.

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

شکل ۱.۲: تابع هزینه مورد استفاده در CycleGAN

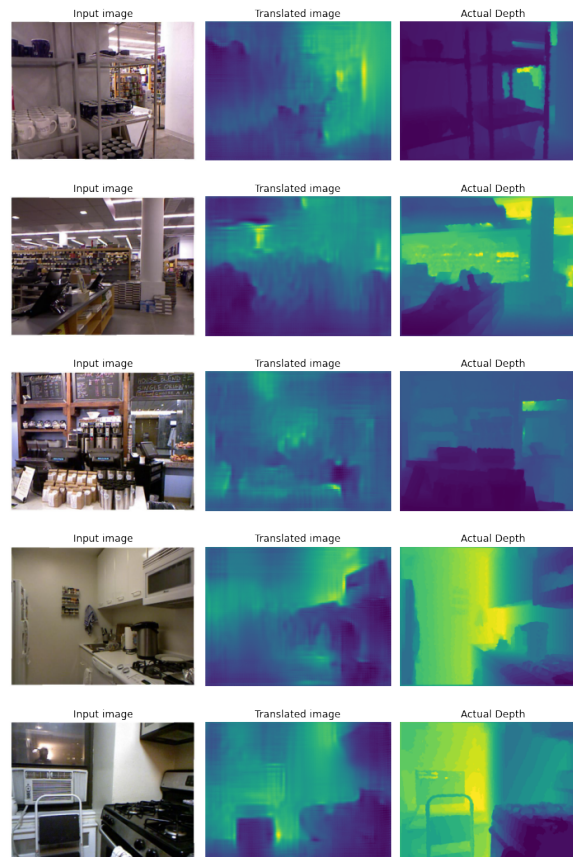
$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = & \mathbb{E}_{x \sim p_{\text{data}}(x)} [\|F(G(x)) - x\|_1] \\ & + \mathbb{E}_{y \sim p_{\text{data}}(y)} [\|G(F(y)) - y\|_1].\end{aligned}$$

شکل ۲.۲: ترم هزینه مربوط به معکوس بودن

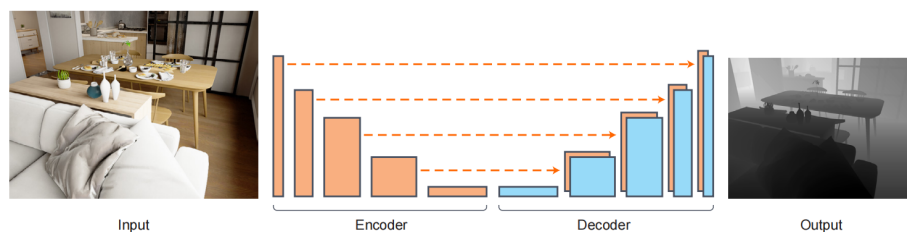
شبکه‌های  $G$  و  $F$  هر کدام شامل ۲ بلوک downsampling، پنج resblock و دو بلوک upsampling هستند. شبکه‌های  $D$  هم شامل سه بلوک downsampling هستند. جزئیات لایه‌های این شبکه‌ها در نوت بوک مربوطه وجود دارد. بهترین شبیه‌سازی انجام شده روی این مدل با بهینه‌ساز آدام، نرخ یادگیری  $2 \times 10^{-4}$  و مقدار  $\beta_1$  برابر با  $0.5$  انجام شد. معیار ارزیابی RMS<sup>۱</sup> بود که پس از ۲۰ مرحله آموزش با سایز بچ ۱ به  $0.3$  روی داده‌های تست رسید. در طول آموزش ۱۱۰۰ تصویر برای آموزش و ۳۴۹ تصویر برای تست استفاده شدند. در شکل ۳.۲ نتیجه تست شبکه روی تعدادی

<sup>۱</sup>Root Mean Squared





شکل ۳.۲: نتایج تست شبکه CycleGAN آموزش دیده



شکل ۴.۲: نمای کلی شبکه

از داده های تست آورده شده است. چون با شبیه سازی های متوالی نتایج به حد مطلوبمان نرسید به سراغ شبکه Unet رفتیم.

## ۲.۲ ساختار Unet

این ساختار را از [۲] برداشتیم. همان طور از در شکل ۴.۲ دیده می شود، شبکه شامل یک انکودر و دکودر است. انکودر شبکه شامل یک تقطیع از شبکه DenseNet-169 است. انکودر ورودی را به یک بردار ویژگی تبدیل می کند، سپس دکودر با upsample های متوالی آن را به تصویر عمق خروجی تبدیل می کند. جزئیات لایه ها در جدول ۱.۲ و در نوت بوک مربوطه وجود دارند.

جدول ۱.۲: لایه های شبکه؛ تا قبل از لایه UP1 تمام لایه ها، لایه های DenseNet-169 هستند.

LAYER	OUTPUT	FUNCTION
INPUT	$480 \times 640 \times 3$	
CONV1	$240 \times 320 \times 64$	DenseNet CONV1
POOL1	$120 \times 160 \times 64$	DenseNet POOL1
POOL2	$60 \times 80 \times 128$	DenseNet POOL2
POOL3	$30 \times 40 \times 256$	DenseNet POOL3
...	...	...
CONV2	$15 \times 20 \times 1664$	Convolution $1 \times 1$ of DenseNet BLOCK4
UP1	$30 \times 40 \times 1664$	Upsample $2 \times 2$
CONCAT1	$30 \times 40 \times 1920$	Concatenate POOL3
UP1-CONVA	$30 \times 40 \times 832$	Convolution $3 \times 3$
UP1-CONVB	$30 \times 40 \times 832$	Convolution $3 \times 3$
UP2	$60 \times 80 \times 832$	Upsample $2 \times 2$
CONCAT2	$60 \times 80 \times 960$	Concatenate POOL2
UP2-CONVA	$60 \times 80 \times 416$	Convolution $3 \times 3$
UP2-CONVB	$60 \times 80 \times 416$	Convolution $3 \times 3$
UP3	$120 \times 160 \times 416$	Upsample $2 \times 2$
CONCAT3	$120 \times 160 \times 480$	Concatenate POOL1
UP3-CONVA	$120 \times 160 \times 208$	Convolution $3 \times 3$
UP3-CONVB	$120 \times 160 \times 208$	Convolution $3 \times 3$
UP4	$240 \times 320 \times 208$	Upsample $2 \times 2$
CONCAT3	$240 \times 320 \times 272$	Concatenate CONV1
UP2-CONVA	$240 \times 320 \times 104$	Convolution $3 \times 3$
UP2-CONVB	$240 \times 320 \times 104$	Convolution $3 \times 3$
CONV3	$240 \times 320 \times 1$	Convolution $3 \times 3$

۸۰ درصد داده ها برای آموزش استفاده شدند، ۱۰ درصد برای اعتبارسنجی و ۱۰ درصد برای تست. معیار ارزیابی روش ما با مقاله فرق دارد. صحت ما به این صورت تعریف می شود که خروجی پیکسل های خروجی شبکه (که خروجی یک تابع سیگنوید هستند) به نزدیک ترین عدد صحیح (۰ تا ۱) گرد می شوند. سپس همین اتفاق برای پیکسل های خروجی واقعی (که مقادیرشان بین صفر و ۱ نرمالیزه شده اند) انجام می شود. مقادیر حاصل در پیکسل ها تک به تک مقایسه می شوند و درصد درستی به عنوان صحت گزارش می شود.

تابع هزینه شبکه همراه با تعریف هر ترم آن در تصویر زیر ۵.۲ آمده است. ترم اول هزینه L1 نقطه به نقطه است. ترم دوم تابع هزینه L1 است که برای گرادیان های تصویر به صورت جداگانه روی مولفه های x و y محاسبه شده است. ترم سوم هم شباهت ساختاری است که یک تابع هزینه رایج در بازسازی تصاویر است. مقدار  $\lambda$  هم برابر با ۰/۱ در نظر گرفتیم.

شبکه را با نرخ یادگیری آموزش ۰/۰۰۰۱، سایز بچ ۸ و بهینه ساز آدام آموزش دادیم. بعد از سه مرحله آموزش حدود ۸۴ درصد صحت روی داده های تست بدست آمد که درصد معقولی است. در شکل ۶.۲ دو نمونه از تست های شبکه آورده شده است.

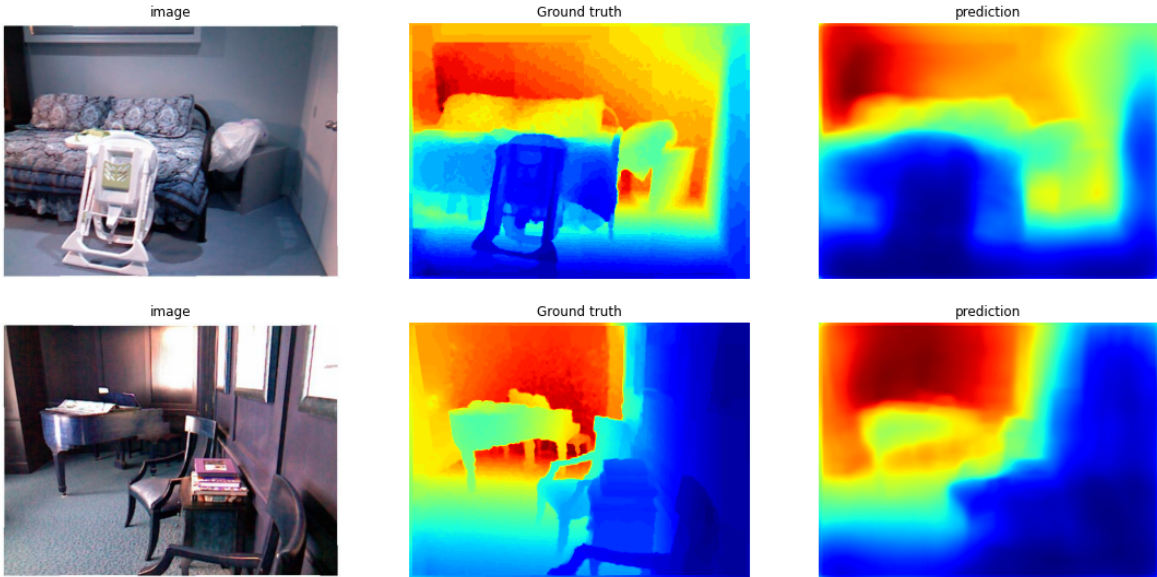
$$L(y, \hat{y}) = \lambda L_{depth}(y, \hat{y}) + L_{grad}(y, \hat{y}) + L_{SSIM}(y, \hat{y}).$$

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

$$L_{grad}(y, \hat{y}) = \frac{1}{n} \sum_p^n |g_x(y_p, \hat{y}_p)| + |g_y(y_p, \hat{y}_p)|$$

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

شکل ۵.۲: تابع هزینه شبکه Unet



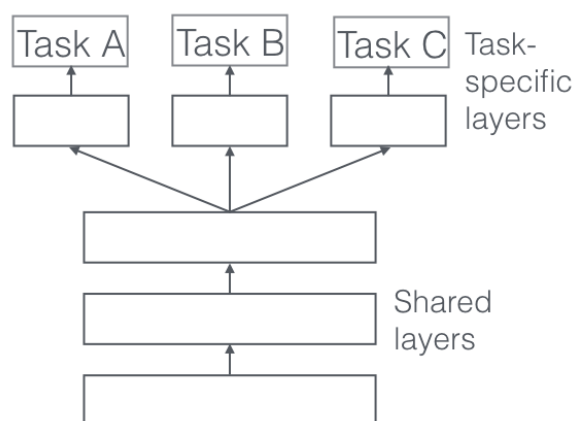
شکل ۶.۲: ۲ نمونه از تست های شبکه آموزش دیده

## فصل ۳ آموزش مشترک

در روش های مورد استفاده ما، شبکه های تشخیص اشیا و تشخیص عمق به صورت جداگانه از هم آموزش داده شده اند. در خصوص اینکه آیا می توان شبکه ها را به صورت همزمان با هم آموزش داد و به دقت مطلوب رسید، ما تحقیقاتی انجام دادیم. در این رابطه ما با یک مسئله یادگیری چندوظیفه ای<sup>۱</sup> طرف هستیم. جواب سوال احتمالاً این است که می توانیم به همان دقت مطلوب یا حتی بیشتر برسیم.

در مورد یادگیری چندوظیفه ای مقالات مروری [۳] و [۴] را مطالعه کردیم. طبق [۳] دو روش کلی برای آموزش مشترک شبکه های عصبی عمیق برای چند وظیفه وجود دارد.

اولی که اشتراک وزن سخت نام دارد، به این صورت است که تا تعدادی لایه، داده های موجود در تمام وظایف از لایه های مشابه می گذرند و از جایی به بعد لایه های مخصوص وظایف وارد کار می شوند. در حالت دوم که اشتراک وزن نرم نام دارد، هر وظیفه مدل خود را دارد، با وزن های و پارامترهای مخصوص خود. در این روش قیدی روی وزن های مدل های مختلف گذاشته می شود تا وزن ها از هم فاصله زیادی نگیرند. در شکل های ۱.۳ و ۲.۳ این حالت ها نمایش داده شده اند.

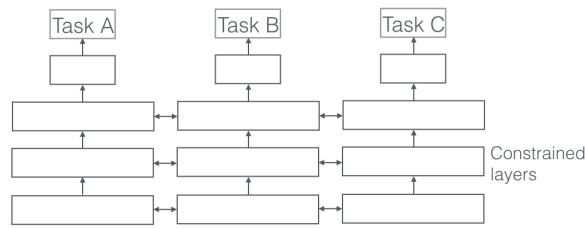


شکل ۱.۳: اشتراک وزن سخت

یادگیری چندوظیفه ای چند حسن دارد از جمله:

- افزایش تعداد داده های آموزشی: در این روش ها به دلیل استفاده از داده های موجود در چند وظیفه مختلف، در

<sup>1</sup>Multi-Task Learning

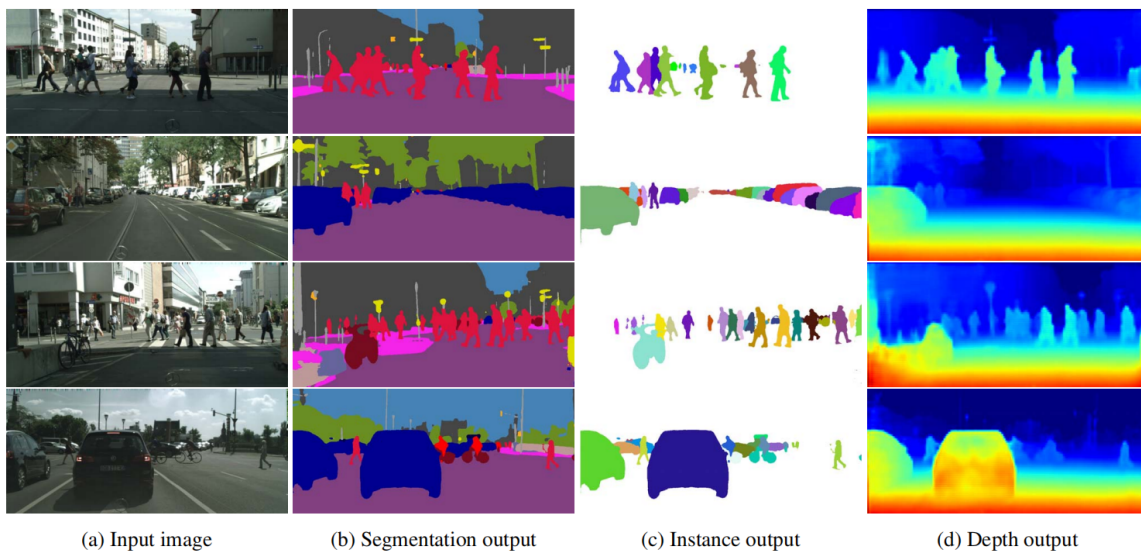


شکل ۲.۳: اشتراک وزن نرم

صورت استفاده از یک ساختار مشترک تعداد داده‌های آموزشی ما خیلی بیشتر می‌شوند.

- تمرکز بیشتر روی ویژگی‌های اساسی: این مدل‌ها معمولاً نسبتاً نويز مقاومت خیلی بیشتری نشان می‌دهند.
- یادگیری ویژگی‌های راحت‌تر از دیگر وظایف: در این مدل‌ها معمولاً ویژگی‌های مشترک بین وظایف به آموزش بهتر مدل کمک می‌کنند.
- رگولاریزاسیون: در این مسائل با اضافه شدن یک بایاس به مدل به دلیل وجود وظایف مختلف، احتمال بیش‌برازش و فیت شدن به نويز کمتر می‌شود.

در مطالعات ما، مقاله‌ای که از معیار ارزیابی مشترک برای مدل آموزش دیده استفاده کند یافت نشد. روال کار همه مقالات به این صورت بود شبکه‌ها به صورت مشترک و جداگانه آموزش می‌دیدند، و نهایتاً با معیار ارزیابی هر وظیفه به صورت جداگانه ارزیابی می‌شدند. با این تمام مقالات در حالت آموزش مشترک از یک تابع هزینه مشترک برای آموزش استفاده می‌کردند. برای مثال در [۵] از جمع وزن دار سه تابع هزینه جداگانه برای وظایف semantic segmentation، instance segmentation و تشخیص عمق استفاده می‌شود. برای وظیفه اول از آنتروپی متقابل پیکسل به پیکسل، برای وظیفه دوم از نرم  $L_1$  تفاضل instance های یافت شده و برای وظیفه سوم از نرم  $L_1$  تفاضل پیکسل به پیکسل استفاده شده است. در شکل ۳.۳ یک نمونه‌ای از نتایج این مقاله آورده شده است.



شکل ۳.۳: مثالی از نتایج ساختار چندوظیفه ای مقاله [۵]

## فصل ۴ وب اپلیکیشن

از دوروش برای ساختن وب اپلیکیشن پروژه استفاده کردیم که در ادامه معرفی می‌شوند.

### ۱.۴ Streamlit

برای اجرای این قسمت پس از دانلود وزن های مدل YOLO-v3 و نیز مدل و وزن های مدل آموزش داده شده Unet و قرار دادن آن‌ها در پوشه های مورد نظر (برای این کار تنها کافیسیت کدهای نوشته شده در README.md را به ترتیب اجرا کنید، این وزن‌ها به دلیل اینکه حجمشان بیشتر از ۱۰۰ مگابایت بود امکان آپلود در گیت‌هاب نداشتند.) کد streamlit ./src/app.py run را اجرا کنید تا برنامه روی لوکال هاست شما اجرا شود. تعدادی از تصاویر داده شده به این وب اپلیکیشن را می‌توانید در شکل ۱.۴ ببینید:

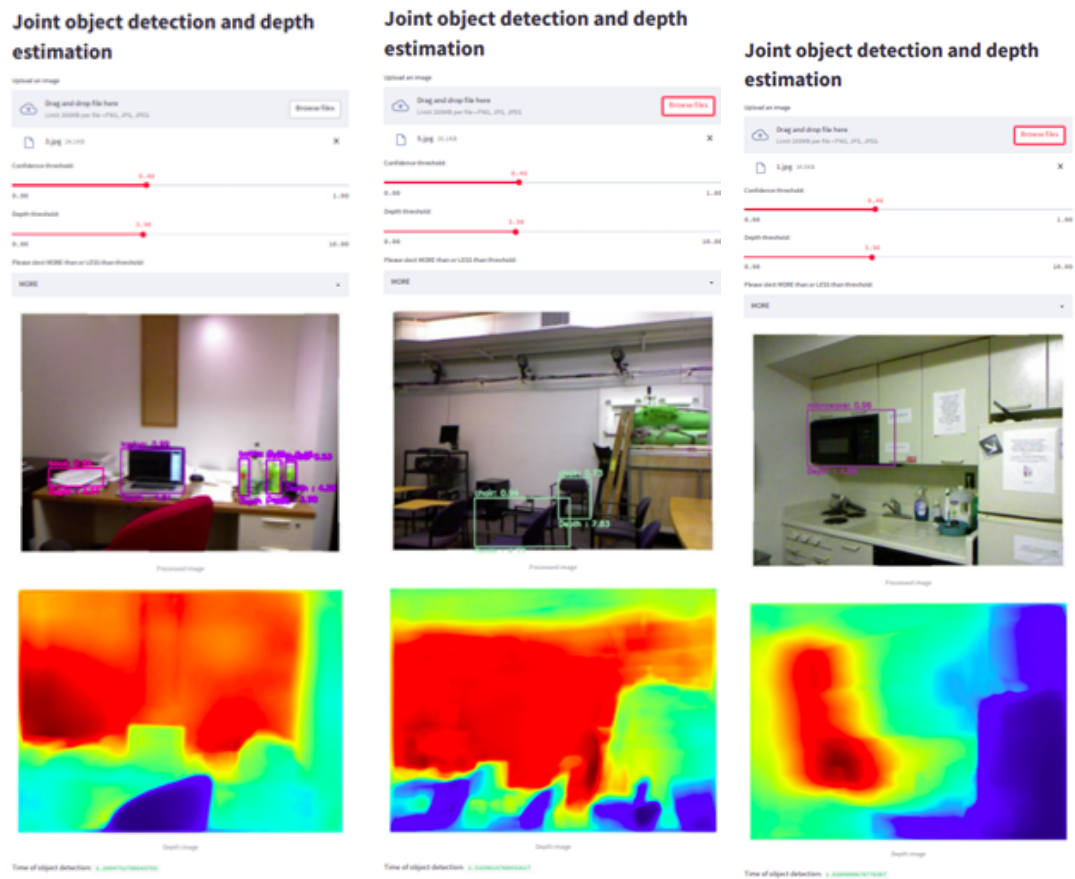
### ۲.۴ Django

برای اجرای این قسمت پس از دانلود وزن های مدل YOLO-v3 و قرار دادن آن‌ها در پوشه مورد نظر (برای این کار تنها کافیسیت کدهای نوشته شده در README.md را به ترتیب اجرا کنید، این وزن‌ها به دلیل اینکه حجمشان بیشتر از ۱۰۰ مگابایت بود امکان آپلود در گیت‌هاب نداشتند.) کدهای ذکر شده در README.md را اجرا کنید تا برنامه روی لوکال هاست شما اجرا شود. تعدادی از تصاویر داده شده به این وب اپلیکیشن را می‌توانید در شکل ۲.۴ ببینید:

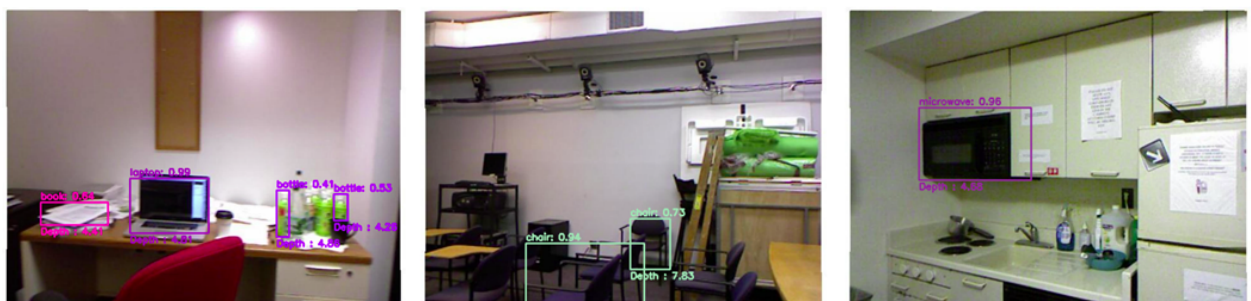
### ۳.۴ چالش‌ها

با توجه به این که باکس تشخیص داده شده توسط مدل تشخیص اشیاء لزوماً شامل تصویر ما نیست (یک مستطیل تماماً انسان را تصور کنید!) برای اندازه‌گیری عمق، مستطیل داخلی باکس (هر ضلع سی درصد مستطیل تشخیص داده شده ی اصلی) را در نظر گرفتیم. برای پیشرفت کد و ادامه ی آن می‌توان مناطق مختلف تصویر را با توجه به کلاسی که تشخیص می‌دهیم در نظر گرفت یا خیر. برای مثال در تشخیص انسان با توجه به این که عمق انسان در تصویر تنها یک عدد است الگوریتم را تغییر نمیدهیم، اما در تشخیص میز با توجه به تغییر عمق آن می‌توان یک عمق ابتدایی و انتهایی برای در نظر





شکل ۱.۴: نمونه های اجرای وب اپلیکیشن در حالت streamlit



شکل ۲.۴: نمونه های اجرای وب اپلیکیشن در حالت Django



گرفت تا تشخیص آن دقیق تر باشد.

الگوریتم پیاده شده ما بر اساس مقاله ذکر شده در صورت پروژه است که مقدار را به روش زیر محاسبه کرده است:

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

ریت مستطیل داخلی یک هایپرپارامتر است که ما هر ضلع را برابر سی درصد مستطیل تشخیص داده شده در نظر گرفتیم. روش دیگر ارایه شده توسط مقاله به دست آوردن median مقادیر بود:

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

- [1] Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- [2] Alhashim, Ibraheem and Wonka, Peter. High quality monocular depth estimation via transfer learning, 2019.
- [3] Ruder, Sebastian. An overview of multi-task learning in deep neural networks, 2017.
- [4] Zhang, Yu and Yang, Qiang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.
- [5] Kendall, Alex, Gal, Yarin, and Cipolla, Roberto. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018.