
بسم الله الرحمن الرحيم

دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان

تشخیص موانع و جاده در معادن روباز به کمک پردازش تصویر

استاد راهنمای اول: دکترایمان ایزدی

استاد راهنمای دوم : دکتر نادرکریمی

معین خراسانی فردوانی

زمستان ۱۴۰۲

فهرست

چکیده

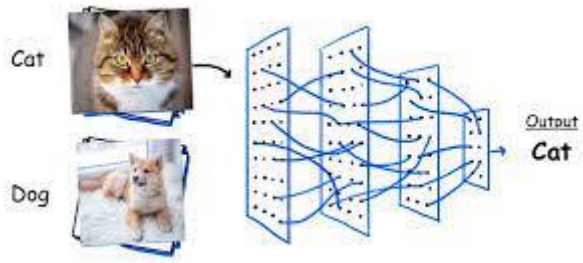
بهبود ایمنی، افزایش کارایی، کاهش هزینه انسانی و کاهش خطای انسانی از جمله مزیت های یک دامپ تراک خودران است.

برای حرکت یک دامپ تراک به صورت خودران به چندین مرحله نیاز داریم. در بالاترین سطح انتخاب بهترین مسیر برای حرکت از مبدا تا مقصد که با استفاده از تصاویر ماهواره ای و الگوریتم هایی مانند A* و غیره صورت می گیرد. این بالاترین سطح منطق برای یک تراک خودران است. اما این الگوریتم ها از جزئیات مسیر و موانع پویا (دینامیک) در مسیر آگاه نیستند، به همین علت ما نیازمند الگوریتم هایی هستیم تا پردازش را به صورت محلی انجام دهند و موانع و جاده را به صورت بلادرنگ تشخیص دهد.

تشخیص موانع و جاده را به ۲ مسئله مجزا در یادگیری ماشین تبدیل می کنیم. تشخیص موانع را به صورت *Object Detection* در نظر می گیریم و برای تشخیص جاده از *Semantic Segmentation* استفاده می کنیم. هر چند برای هر دوی این مسئله می توان راهکار های جایگزین نیز استفاده کرد. می توان تشخیص موانع را به صورت یک مسئله *Semantic Segmentation* در نظر گرفت و ماشین های مختلف را به عنوان کلاس های مختلف در نظر گرفت . و برای تشخیص جاده نیز می توان از روش های کلاسیک بینایی کامپیوتر و تبدیل هاف (برای تشخیص خطوط و منحنی در تصاویر باینری) استفاده کرد. برخی مقالات نیز از همین روش ها استفاده کردند.

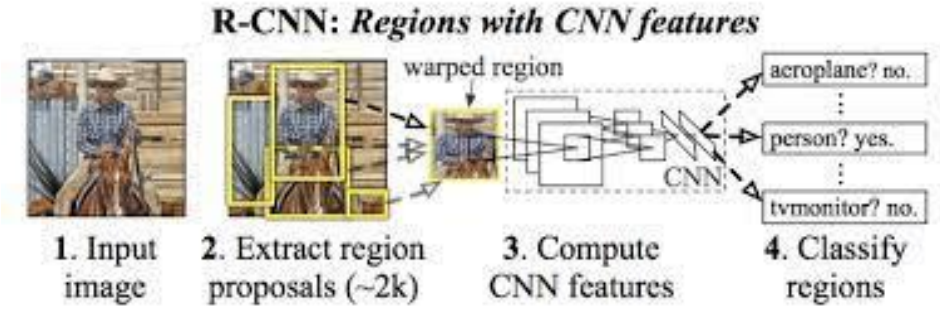
Object Detection

Object Detection بـــر اســـاس شـــبکه هـــای عصـــبی *RCNN(Regional Convolutional Network)* صـورت گرفتـه اسـت. تفـاوت ایـن شـبکه هـا بـا *CNN* هـا در ایـن اسـت کـه *CNN* هـا بـرای *Image Classification* بـه کـار مـی رونـد و نمـی تواننـد *Object Detection* را انجـام دهنـد. در *Image Classification* پـیش بینـی مـی شـود کـه ایـن تصـویر بـا چـه احتمـالی متعلـق بـه کـدام دسـته اسـت ، بـه عنـوان بـا چـه احتمـالی یـک تصـویر(کل تصـویر) ماشـین اسـت یـا یـک شـخص یـا جـاده . و در نهایـت کلاسـی کـه احتمال بیش تری دارد به عنوان خروجی تعیین نمی شود.

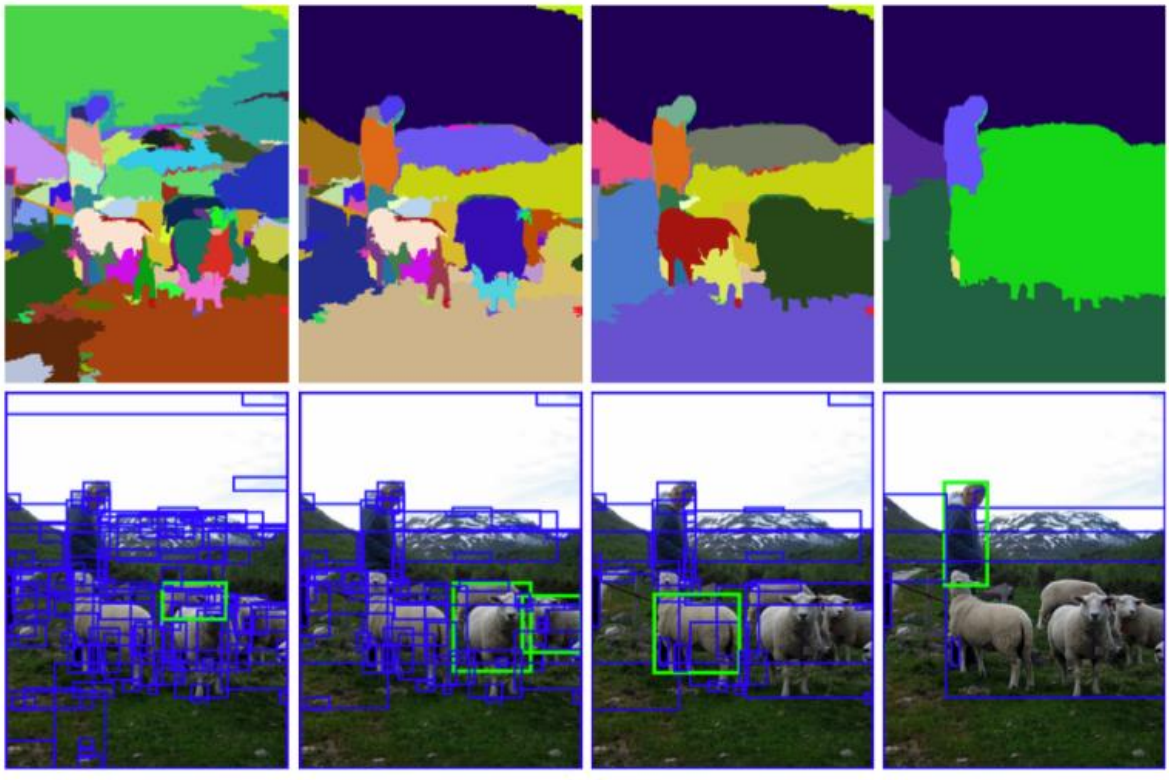


مسئله دسته بندی

امـا در دامـپ تـراک خـودران مـا نیـاز داریـم تـا در هـر تصـویر تشـخیص دهـیم چـه مـوانعی و در چـه مکـانی از تصـویر وجـود دارنـد. در واقـع ممکـن اسـت در یـک تصـویر چنـدین شـی متفـاوت در مکـان هـای مختلـف داشـته باشـیم و در تصـویری هـیچ شـی یـا مـانعی را تشـخیص نـدهیم. در واقـع مـی تـوان مسـئله *Object Detection* را بـه ۲ زیـر مسـئله تقسـیم کـرد اول *Localization* و دوم *Classification*. Image . *RCNN* هـا بـرای مـا همـین کـار را انجـام مـی دهنـد. درابتـدا نـواحی کـه تشـخیص مـی دهنـد ممکـن اسـت در آن هـا شـی وجـود داشـته باشـد را انتخاب مـی کنـد و هـر کـدام را بـه عنـوان یـک تصـویر مجـزا بـه عنـوان ورودی بـه *CNN* مـی دهـیم. در واقـع ممکـن است برای یک تصویر چندین *CNN* اجرا شود.



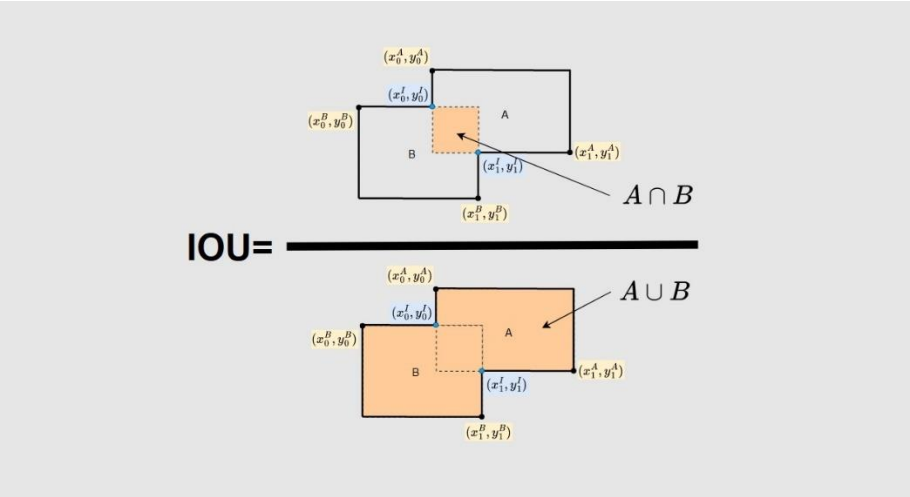
بـرای تشـخیص نـواحی پیشـنهادی (*Region Proposal*) از روش *Selective Search* اسـتفاده مـی شـود . *Selective Search* از *segmentation* هـای متـوالی اسـتفاده مـی کنـد. در هـر مرحلـه هـر ناحیـه بـه ناحیـه کوچک تر تقسیم می شود. در مراحل اولیه تعداد نواحی کم است و در مراحل آخر زیاد می شود.



همان طور که مشاهده می کنید در این جا مشکلاتی نیز دارد . مثل آنکه یک شی واحد چند شی تشخیص داده می شود.

برای رفع این مشکل از روش *NMS (Non Maximum Suppression)* استفاده می شود که در واقع روشی است که از بین چندین *Box* برای یک شی، ناحیه ای انتخاب می شود که اطمینان بیشتری دارد.

برای محاسبه خطا در این شبکه ها از *IOU(Intersection Over Union)* استفاده می شود.



مشکل دیگر اجرای *CNN* به صورت مجزا برای هر کدام از نواحی انتخابی است که می توان گفت برای یک تصویر ممکن در حدود چند صد یا چند هزار باشد. این کار هم زمان آموزش را به شدت افزایش می دهد و هم در زمان اجرای الگوریتم نیز زمان زیادی را از سیستم می گیرد که این با بلادرنگ(*Real Time*) بودن سیستم در تعارض است. برای حل این مشکل چندین راه حل وجود دارد.

۱. استفاده از جدید ترین شبکه های عصبی که تا حد امکان سریع باشند.

استفاده از *Fast RCNN* و *Faster RCNN* به جای *RCNN*

RCNN : در این الگوریتم نواحی بر اساس الگوریتم *Selective Repeat* انتخاب می شوند.

Fast RCNN: در این الگوریتم در ابتدا از *Feature Map* تصاویر استخراج می شود و از *ROI Pooling* برای یکسان سازی سایز نواحی انتخاب شده استفاده می شود.

Faster RCNN: این شبکه عصبی از *Selective Repeat* استفاده نمی کند. و در شبکه عصبی بلوک لایه ای به عنوان (*Region Proposal Network*)*RPN* به شبکه عصبی اضافه می شود تا خود نواحی محتمل را پیشنهاد دهد.

۲. استفاده از شبکه های عصبی سبک تر: کاهش لایه های (*Fully Connected*)*FC* یا کاهش ابعاد تصویر . با این کار ممکن است مدل ما *Underfit* شود. اما تا حدی که تصاویر شفاف و واضح باشند می توان حجم آن ها را کاهش داد و به تبع با کاهش ابعاد داده به تعداد لایه های کمتری از *Fully Connected* ها نیاز داریم.

۳. استفاده از پردازش موازی و سخت افزار مناسب: استفاده از *GPU* به جای *CPU*

Semantic Segmentation

مسئله بعدی *Semantic Segmentation*است . در Segmentationساده (که فقط تفکیک داده ها مهم است نه آنکه متعلق به چه دسته ای هستند) . تنها باید داده ها را تفکیک کنیم. می توان گفت Segmentationدر واقع یک مسئله *Clustering* در حوزه تصاویر است که می تواند با هر روشی مثل KNN() و ... پیاده سازی شود. اما در *Semantic Segmentation* علاوه بر آنکه بایستی پیکسل ها (داده ها) را از هم تفکیک کنیم ، باید تعیین کنیم که این پیکسل ها متعلق به کدام دسته می باشند. در دامپ تراک باید تعیین کنیم کدام پیکسل ها جاده هستند و کدام جاده نیستند.

ساختار شبکه عصبی ما از ۲ بخش تشکیل شده است :

۱.*Encoder*: در این مرحله ما از تصویر برخی ویژگی ها را استخراج می کنیم. در واقع نوعی *Down Sampling* داریم.

۲. Decoder: در این مرحله ما از کلاس های به دست آمده سعی می کنیم به فضای تصویر برویم و تعیین کنیم هر پیکسل متعلق به کدام دسته است . در این مرحله ما *UP Sampling* انجام می دهیم.

اما چگونه *Up Sampling* ما انجام می شود. در واقع چگونه از چند ویژگی به دست آمده از خروجی *CNN* متوجه می شویم که این ویژگی متعلق به کدام یک از پیکسل ها بوده است؟

این کار به کمک لایه های *Convolutional Transpose* و *Skip Connection* ها انجام می شود. *Convolutional Transpose* ها عکس لایه های کانولوشنی ساده هستند و *Skip Connection*ها نیز باعث اتصال لایه هایی از ابتدای شبکه به انتهای آن به صورت مستقیم خواهند شد. *Skip Connection* ها علاوه بر این که به دقت *Up Sampling* را بالا می برند، مشکل *Vanishing Gradient* را نیز حل می کنند و کمک می کنند تا ما به *Global Optima* برسیم. در واقع ساختار ما نباید یک ساختار خطی باشد و باید شبکه ما چیزی شبیه یک *DAG*(*Direct Acyclic Graph*) باشد.

انتخاب *FrameWork* مناسب برای کد زدن(متلب)

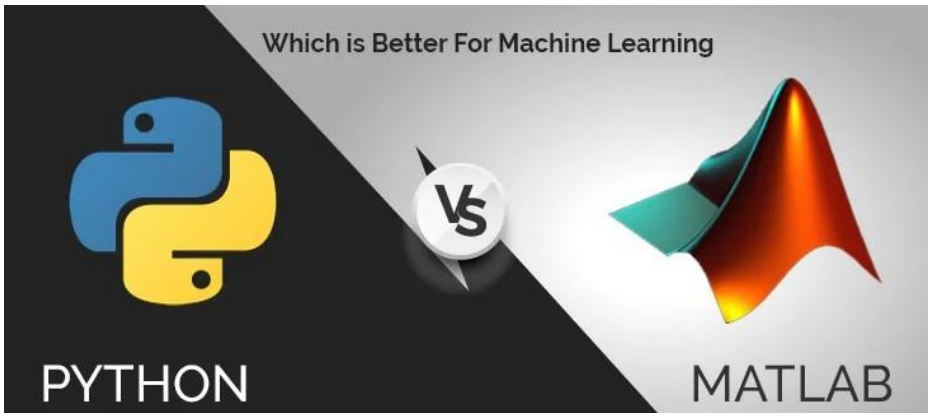
یکی از معیار های انتخاب یک فریمورک یکپارچگی آن با سایر بخش های نرم افزار است. از آن جا که متلب یک زبان برنامه نویسی جامع برای مهندسین است و ابزار هایی مثل *Simulink* و سایر ابزار های گرافیکی را برای مهندسین برق مهیا می کند، لذا بسیاری از پروژه هایی که در آن ها کار های صنعتی انجام شود از متلب استفاده می شود. برای آنکه این بخش از نرم افزار نیز بتواند با سایر بخش های نرم افزار ادغام شود، انتخاب زبان متلب یک گزینه مناسب بود.

عامل دیگر انتخاب، شهرت این زبان در پروژه های خودران است. شرکت های سرآمد در این حوزه مثل تسلا از همین زبان استفاده می کنند.

عامل دیگر اجرای بهینه آن روی *GPU* است. برای کار با *GPU* تنها نیاز به نصب کودا است و برای آموزش یک کارت گرافیک *NVIDIA GTX* با ۴ گیگ حافظه کافی است.

عامل دیگر ابزار های آماده برای برچسب زنی(*Labeling*) تصاویر است.

برای برچسب زنی تصاویر در زبان پایتون باید از سایر *GUI* ها استفاده کرد، که غالبا این ابزار ها تبدیل فرمت مناسب و استانداری برای شبکه های پایتون تهیه نمی کنند(به عبارتی فایل تولید شده توسط آن ها باید به یک فایل *COCO* یا *XML* مناسب تبدیل شود). برای داده های ویدیویی غالبا لازم است تا عکس ها را *capture* کرد و آن ها را به جداگانه به عنوان ورودی نرم افزار های استفاده شوند. شاید از دید اول این خیلی مطلب مهمی نباشد، اما ابزار های *Labeling* متلب از توالی فریم های یک ویدیو می توانند مکان یک شی را در فریم بعدی حدس بزنند.(توجه داشته باشید که این الگوریتم ها *Train* شده نیستند و تنها مجموعه ای از الگوریتم های کلاسیک هستند که برای برچسب زنی استفاده می شوند). این تفاوت به خصوص در برچسب زنی برای داده های *Object Detection* در متلب به شدت فرآیند برچسب زنی را تسریع می بخشد. در بخش Semantic Segmentationنیز استفاده از Super Pixel ها فرآیند برچسب زنی را تسریع می بخشد.



Transfer Learning

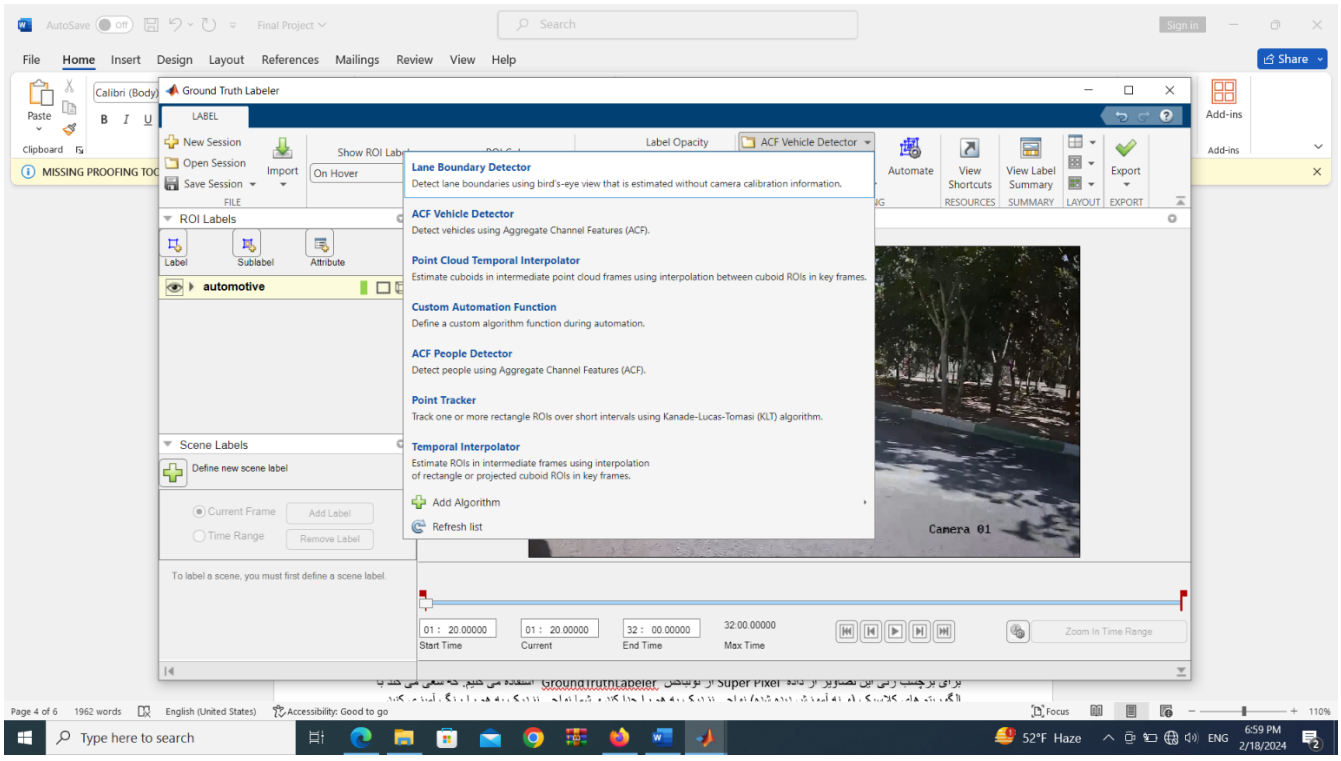
در هـر دو مسـئله *Object Detection , Semantic Segmentation* اگـر نخـواهیم از مـدل هـای آمـاده و دیتاسـت هـای آمـاده اسـتفاده کنـیم، تقریبـا حـل مسـئله نـاممکن مـی باشـد. تهیـه دیتاسـت خـود فرآینـد زمـان بـری است، کـه مـی تـوان بـه جـای آنکـه خودمـان ایـن کـار را انجـام دهـیم از سـایر دیتاسـت هـا اسـتفاده کنـیم و مـدل خـود را روی ایـن مـدل هـا آمـوزش دهـیم. ایـن مـدل آمـوزش داده شـده ممکـن اسـت روی دیتاسـت داده شـده خـوب کـار کنـد امـا بـرای محـیط معـدن مناسـب نیسـت. اکثـر دیتاسـت هـا در حـوزه خـودران بـرای خودروهـای شخصـی و خیابـان هـا مـی باشـد، نـه بـرای تـراک و مسـیر هـای معـدنی. بـه ویـژه در مبحـث *Segmentation* مسـیر هـای معـدن با خیابـان بـه شـدت متفـاوت اسـت و خـط مشخصـی وجـود نـدارد کـه ایـن دو را از هـم تفکیـک کنـد. پـس نیـاز اسـت تـا مـدل بـه دسـت آمـده یکبـار دیگـر بـا داده هـای معـدن مـس سـر چشـمه آمـوزش داده شـود. دیتاسـت بایـد از تصـاویری کـه از توسـط دوربـین هـا گرفتـه شـده و توسـط *NVR* ذخیـره شـده فـراهم شـود. پـس از آمـوزش روی داده های معدن مدل جدید می تواند برای معدن مناسب باشد.

برچسب زنی داده ها

برای برچسب زنی داده های متلب از ابزار *Ground Truth Labeler* که در قسمت *Automotive* قرار دارد استفاده می کنیم.

Object Detection:

ابتدا را در نرم افزار ویدیوی یکی از دوربین ها را *import* کرده و بعد از آن کلاس های مورد نیاز را تعریف می کنیم. نوع داده تعریف شده باید از نوع مستطیل باشد. این ابزار به شما امکان برچسب زنی به صورت چند ضلعی های مختلف را می دهد اما شبکه های عصبی معروف غالبا همان ورودی مستطیل را از شما قبول می کنند. متناظر با هر فریم . یک آرایه از این مستطیل ها ب همراه نوع آن ها ذخیره می شود. فریم هایی که برای آن ها هیچ مانعی تشخیص داده نشده در کد قبل از آموزش شبکه از داده ها حذف می کنیم. مزیت این تولباکس در قسمت *Automotive* است که با انتخاب *label* در یک فریم آن *label* را در فریم های متوالی دنبال می کند و نیاز به دوباره کشیدن boxهای برچسب نیست. برای این کار از الگوریتم های *ROI Point Tracker* و *ACF*(*Aggregative Channel Feature*) استفاده می شود. در نهایت از نرم افزار *Export* گرفته و داده ها را در یک فایل *.mat* ذخیره می کنیم.

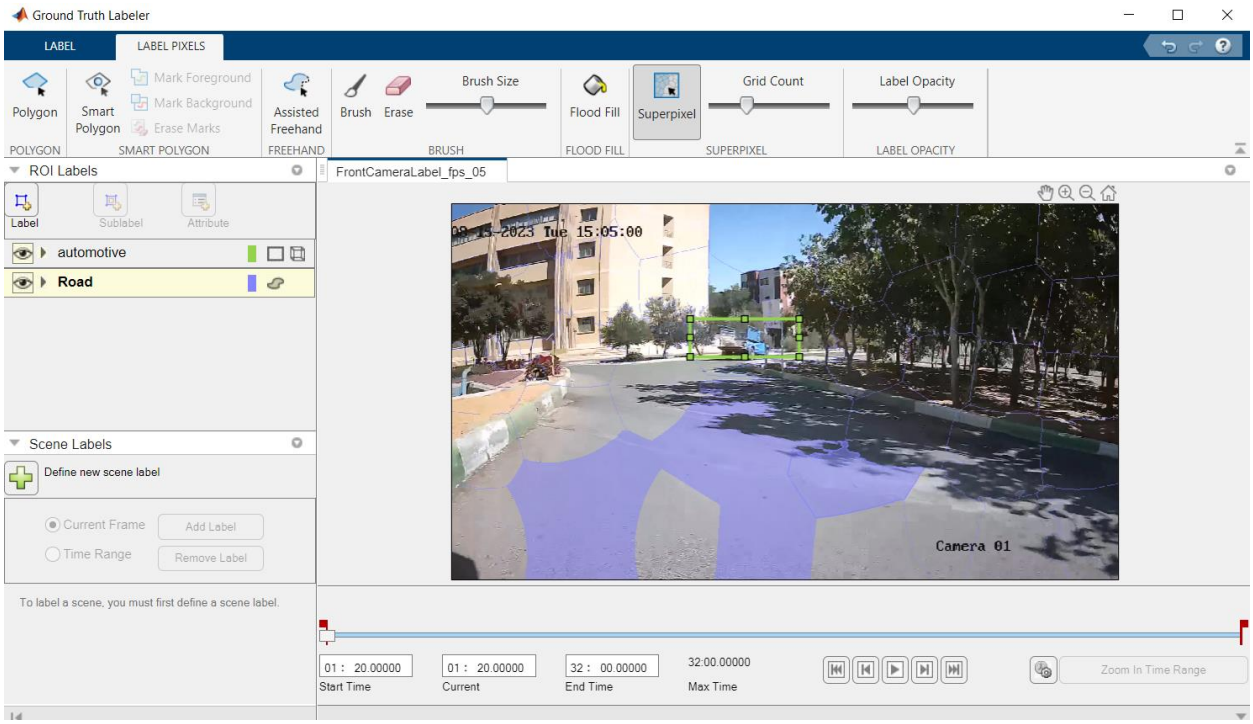


نکته مهم : (*fps*(*frame per second* ویدیو ها زیاد است و لازم است قبل از آموزش آن را تا حد لازم کم کنیم. چرا که فریم های متوالی با فاصله زمانی پایین خیلی تفاوتی با هم ندارند و مانند داده تکرای برای مدل تلقی می شوند. برای *Object Detection* باید *fps*بیشتر از *segmentation Semantic* در نظر گرفته شود، چرا که سایر تراک ها نیز در حرکت هستند و در فریم های متوالی، تفاوت بیشتری است و نکته بعدی آنکه همیشه مانع وجود ندارد پس بهتر است در زمان های اندکی که مشاهده می کنیم فریم های بیشتری را ذخیره کنیم.

Semantic Segmentation :

برای برچسب زنی این تصاویر از داده *Super Pixel* از تولباکس *Ground Truth Labeler* استفاده می کنیم. این ابزار سعی می کند با الگوریتم های کلاسیک (و نه آموزش دیده شده) نواحی نزدیک به هم را جدا کند و شما نواحی نزدیک به هم را رنگ آمیزی کنید.

در نهایت داده ها را در یک فایل *mat*. ذخیره می کنیم. و پوشه تصاویر برچسب داده شده را نیز *export* می کنیم.


Ground Truth Labeler
LABEL
LABEL PIXELS
Polygon
Smart Polygon
Mark Foreground
Mark Background
Erase Marks
Assisted Freehand
Brush
Erase
Brush Size
Flood Fill
Superpixel
Grid Count
Label Opacity
POLYGON
SMART POLYGON
FREEHAND
BRUSH
FLOOD FILL
SUPERPIXEL
LABEL OPACITY
ROI Labels
Label
Sublabel
Attribute
automotive
Road
Scene Labels
Define new scene label
Current Frame
Time Range
Add Label
Remove Label
To label a scene, you must first define a scene label.
FrontCameraLabel_fps_05
03-15-2023 Tue 15:05:00
Camera 01
01 : 20.00000
Start Time
01 : 20.00000
Current
32 : 00.00000
End Time
32:00.00000
Max Time
Zoom In Time Range

پیش پردازش روی داده ها

برای استفاده از داده های *Ground Truth Labeler* به عنوان ورودی شبکه عصبی باید تغییری را اعمال کنیم. مشکل در زمانی هست که شبکه عصبی برای آموزش یک پوشه تصاویر و برچسب ها را می خواهد. اما ابزار برچسب ها را بر روی ویدیو اعمال کرده ، برای همین باید با توجه به *fps* ویدیو تصاویر را *Capture* کرده و یک تناظر میان زمان های مختلف ویدیو و شماره عکس ها برقرار کنیم. و سپس نام قسمت *Time* رو به *ImageFileName* تغییر نام داه و آدرس عکس ها را در آن قرار دهیم . پس از برقراری این تناظر نوبت به تغییر سایز تصاویر می رسد .

یکی از کار هایی که باعث بهبود دقت مدل شد این بود که برچسب خودرو های کوچک یا دامپ تراک های کوچک که به معنی دور بودن آن ها از تراک است را حذف کنیم. چرا که در نظر گرفتن آن ها *FP* مدل را به شدت بالا می برد و علاوه بر آن برای ما موانعی قابل توجه است که در فواصل نزدیک به ما باشد نه فواصل بسیار دور.

توجه داشته باشید که اگر یک تصویر ورودی با نسبتی کوچک شد ، مختصات Box مشخص شده برای اشیا نیز باید با همین نسبت تغییر کنند.

برای تصاویر *Segmentation* باید کار دیگری نیز صورت بگیرد. شبکه عصبی استفاده شده به ازای هر کلاس (مثلا جاده) یک کد رنگ در نظر گرفته است . به عنوان مثال اگر یک پیکسل در *Label* (نه در تصویر اصلی) کد *#ff9988* را داشت این پیکسل به عنوان جاده تلقی می شود . اما خروجی *Ground Truth Labeler* یک ماتریس است که در هر درایه آن شماره کلاس ذخیره شده است. پس باید یک تناظر میان شماره کلاس ها کد رنگ خواسته شده برقرار شود. در کد این کار با تابع *applyRoadColor* انجام شده است.

دیتاست

در بخش *Object Detection* ابتدا مدل بر روی *CIFAR 10* آموزش می بینند . این دیتاست بر روی یک شبکه *CNN* آموزش داده می شود و برای داده های *Classification* استفاده می شود. این آموزش یک *Backbone*خوب برای استخراج ویژگی ها را تهیه می کند. و سپس با داده های جمع آوری شده از معدن آموزش می دهیم.

در بخش *Semantic Segmentation* از دیتاست *Camvid* استفاده شده است. پس از آموزش این مدل ، مدل را با داده های معدن آموزش می دهیم .

ساختار شبکه عصبی

بـرای قسـمت *Object Detection* از شـبکه عصـبی بـا سـاختار زیـر اسـتفاده مـی کنـیم. ایـن سـاختار از سـایت متلب برداشته شده است.

ساختار این شبکه عصبی به سه قسمت تقسیم می شود.

Input Block:

```
imageInputLayer([33,33,3])
```

Middle Block: (استخراج ویژگی های مناسب به کمک *CNN* و *Pooling Layer*)

```
filterSize = [5 5];
numFilters = 32;
```

```
middleLayers = [

% The first convolutional layer has a bank of 32 5x5x3
filters. A
% symmetric padding of 2 pixels is added to ensure that
image borders
% are included in the processing. This is important to
avoid
% information at the borders being washed away too early
in the
% network.
convolution2dLayer(filterSize,numFilters,'Padding',2)

% Note that the third dimension of the filter can be
omitted because it
% is automatically deduced based on the connectivity of
the network. In
```

```
% this case because this layer follows the image layer,
the third
% dimension must be 3 to match the number of channels in
the input
% image.

% Next add the ReLU layer:
reluLayer()

% Follow it with a max pooling layer that has a 3x3
spatial pooling area
% and a stride of 2 pixels. This down-samples the data
dimensions from
% 32x32 to 15x15.
maxPooling2dLayer(3,'Stride',2)

% Repeat the 3 core layers to complete the middle of the
network.
convolution2dLayer(filterSize,numFilters,'Padding',2)
reluLayer()
maxPooling2dLayer(3, 'Stride',2)

convolution2dLayer(filterSize,2 * numFilters,'Padding',2)
reluLayer()
maxPooling2dLayer(3,'Stride',2)

];
```

Final Block (کلاس بندی تصویر با توجه به ویژگی های استخراج شده در لایه قبل)

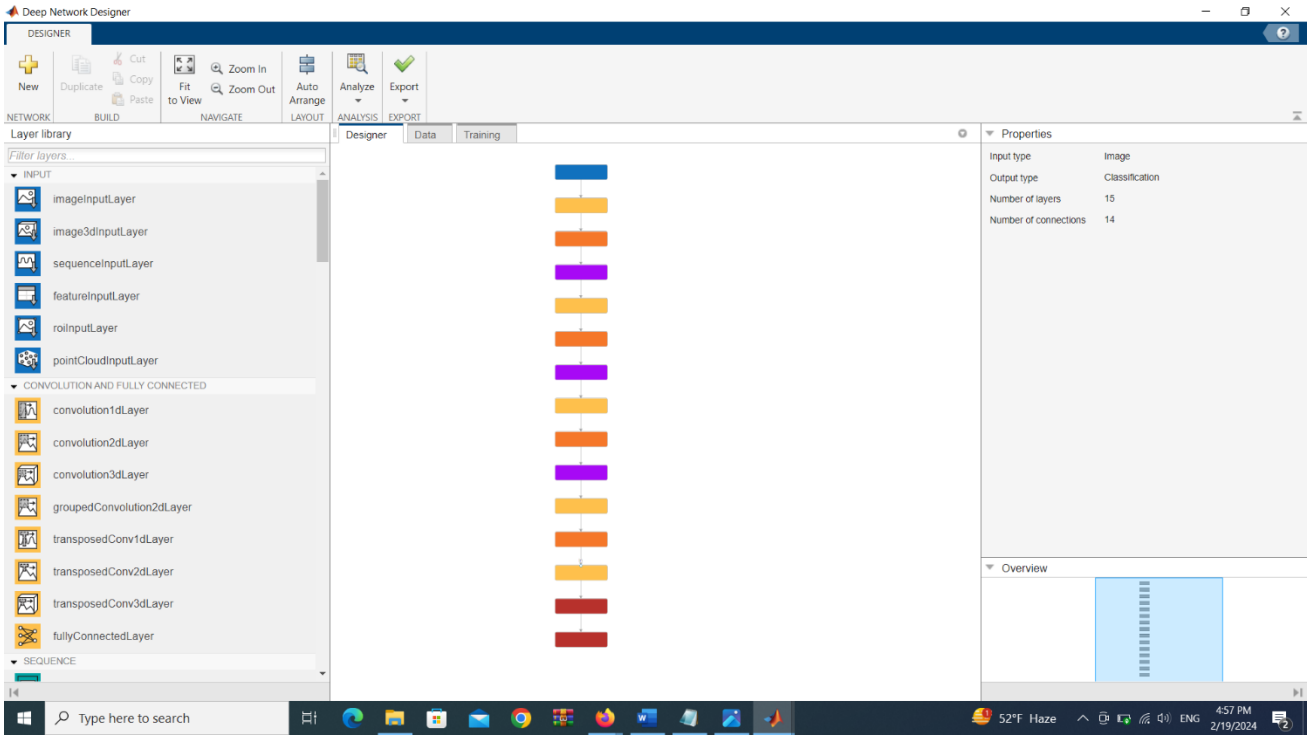
```
finalLayers = [

% Add a fully connected layer with 64 output neurons. The
output size of
% this layer will be an array with a length of 64.
fullyConnectedLayer(64)

% Add an ReLU non-linearity.
reluLayer

% Add the last fully connected layer.
fullyConnectedLayer(numImageCategories)

softmaxLayer
classificationLayer
];
```



یـک شـبکه ای هماننـد *UNet* و شـکل *DAG* داریـم کـه بـرای مـا کـار *Semantic Segmentation* را انجام میدهد و در نهایت به ۱۱ کلاس طبقه بندی می کند.

```
classes = [
    "Sky"
    "Building"
    "Pole"
    "Road"
    "Pavement"
    "Tree"
    "SignSymbol"
    "Fence"
    "Car"
    "Pedestrian"
    "Bicyclist"

    ];
```

تابع *CamvidLabelID* این برای هر کدام از این کلاس ها کد رنگ مربوطه را معین می کند

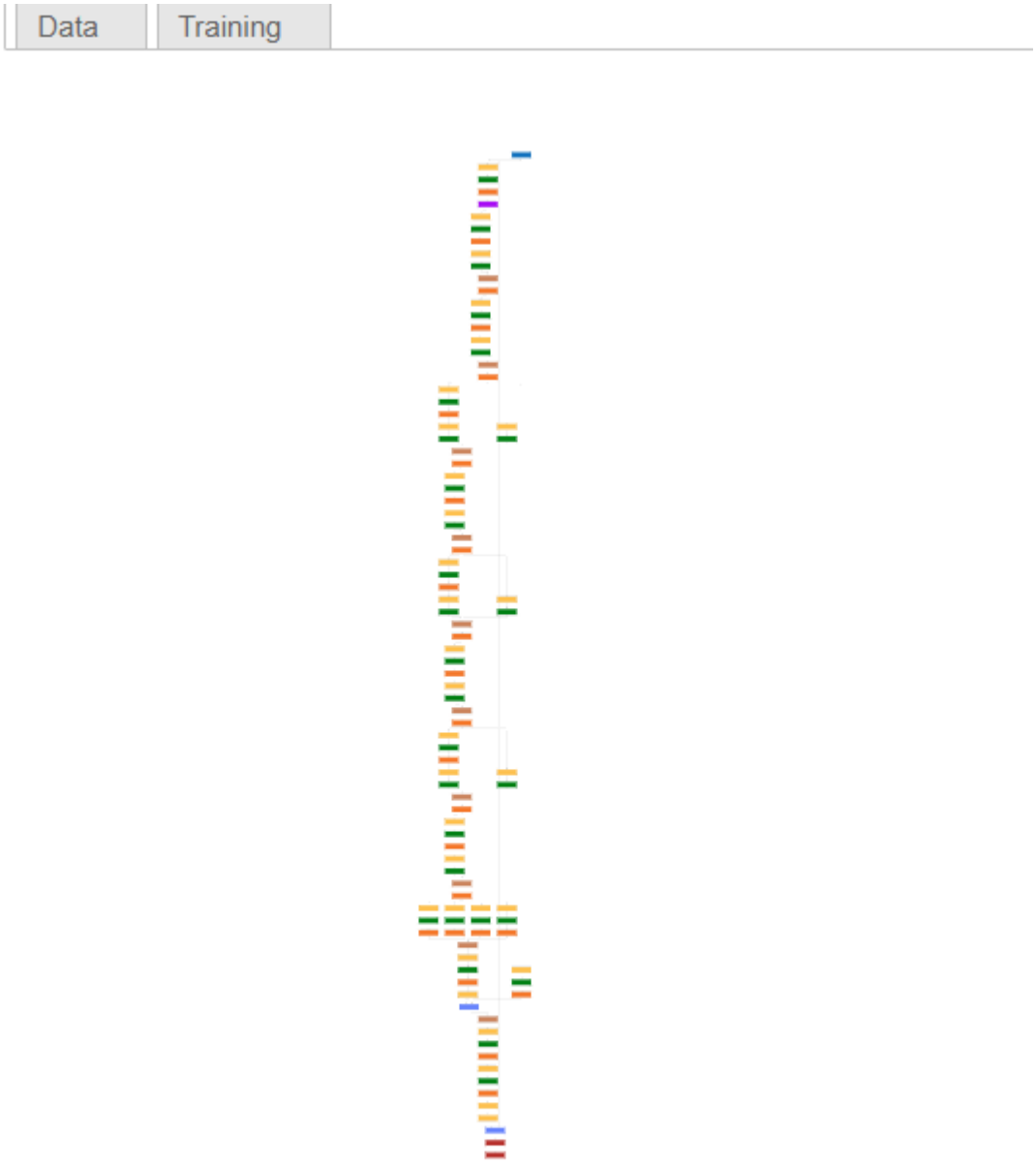
```
numFilters = 256;
filterSize = 3;
numClasses = 11;

layers = [
    imageInputLayer([64 86 3])
    convolution2dLayer(filterSize,numFilters,'Padding',1)
    reluLayer()
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(filterSize,numFilters,'Padding',1)
    reluLayer()

transposedConv2dLayer(4,numFilters,'Stride',2,'Cropping',
1);
    convolution2dLayer(1,numClasses);
    softmaxLayer()
    pixelClassificationLayer()
```

]

اما ساختار گرافیکی آن به صورت زیر است.



آموزش

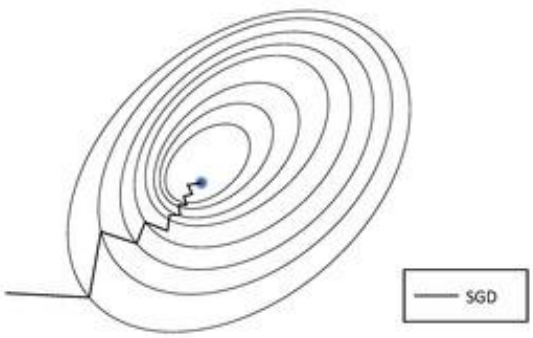
Object Detection: ابتـدا مـدل مـا بـا داده هـای *CIFAR10* توسـط تـابع *TrainNetwork* آمـوزش داده می شوند(فایل *RCNN*). و پس از آن داده های

معـدن آمـوزش داده مـی شـوند. بـرای ایـن کـار از تـابع *TrainRCNN* اسـتفاده شـده اسـت (فایـل *vehicle_training*)

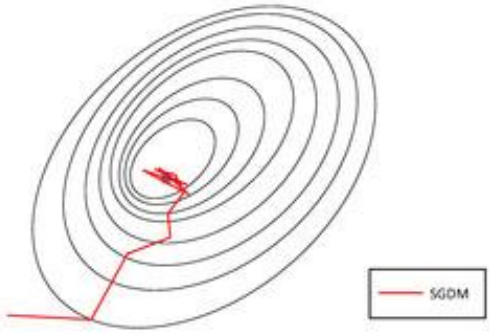
```
options = trainingOptions('sgdm', ...
        'MiniBatchSize', 128, ...
        'L2Regularization', 0.3,...
        'InitialLearnRate', 1e-3, ...
        'LearnRateSchedule', 'piecewise', ...
        'LearnRateDropFactor', 0.1, ...
        'LearnRateDropPeriod', 100, ...
        'MaxEpochs', 6, ...
        'Plots','training-progress',...
        'Verbose', true);

    % Train an R-CNN object detector. This will take
several minutes.
    madan_19_back_net_dataset                                =
trainRCNNObjectDetector(non_empty_data,             cifar10Net,
options, ...
    'NegativeOverlapRange',              [0              0.3],
'PositiveOverlapRange',[0.6 1]);
```

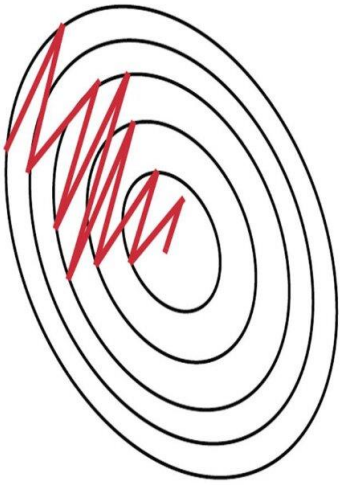
Optimizer اســـــــــتفاده شـــــــــده *SGDM(Stochastic Gradient Descent With Momentum)* اســت . تفـاوت ایـن روش بـا *SGD* سـاده در حافظـه دار بـودن ایـن روش اسـت .یعنـی جهـت حرکـت قبلـی خـود را می داند و با توجه به آن و گرادیان فعلی حرکت خواهد کرد.



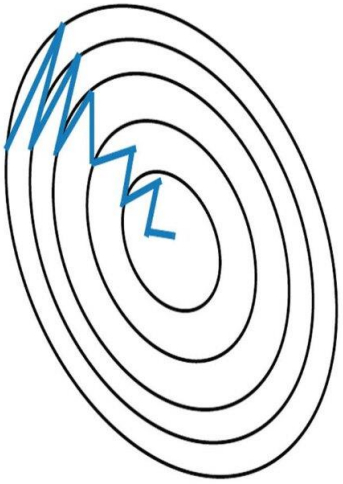
(a) SGD



(b) SGDM



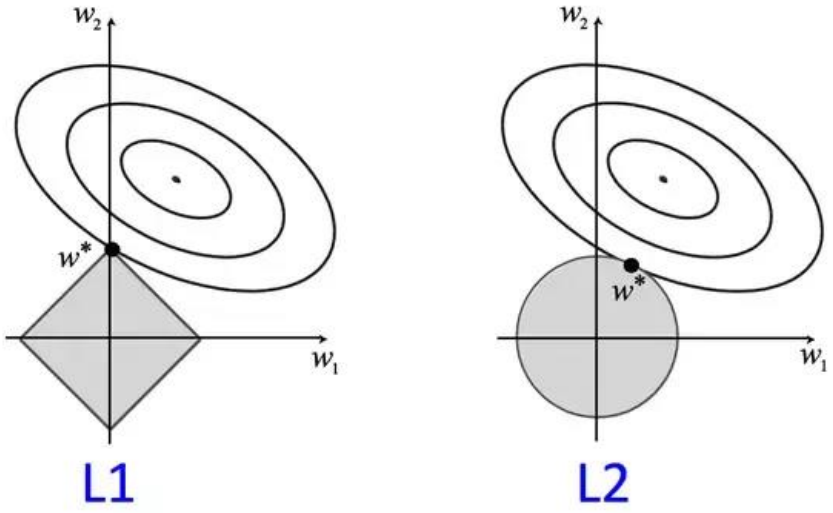
Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

$$V_t = \beta V_{t-1} + (1-\beta) S_t$$
$$\beta \in [0,1]$$

بـــرای *Regularization* از روش *L2* اســـتفاده شـــده اســـت کـــه در واقـــع همـــان *Ridge Regularization* اسـت. تفـاوت ایـن روش بـا روش *L1* ایـن اسـت کـه ویژگـی هـا را حـذف نمـی کنـد، بلکـه اثـر آن هـا را کـم مـی کند.

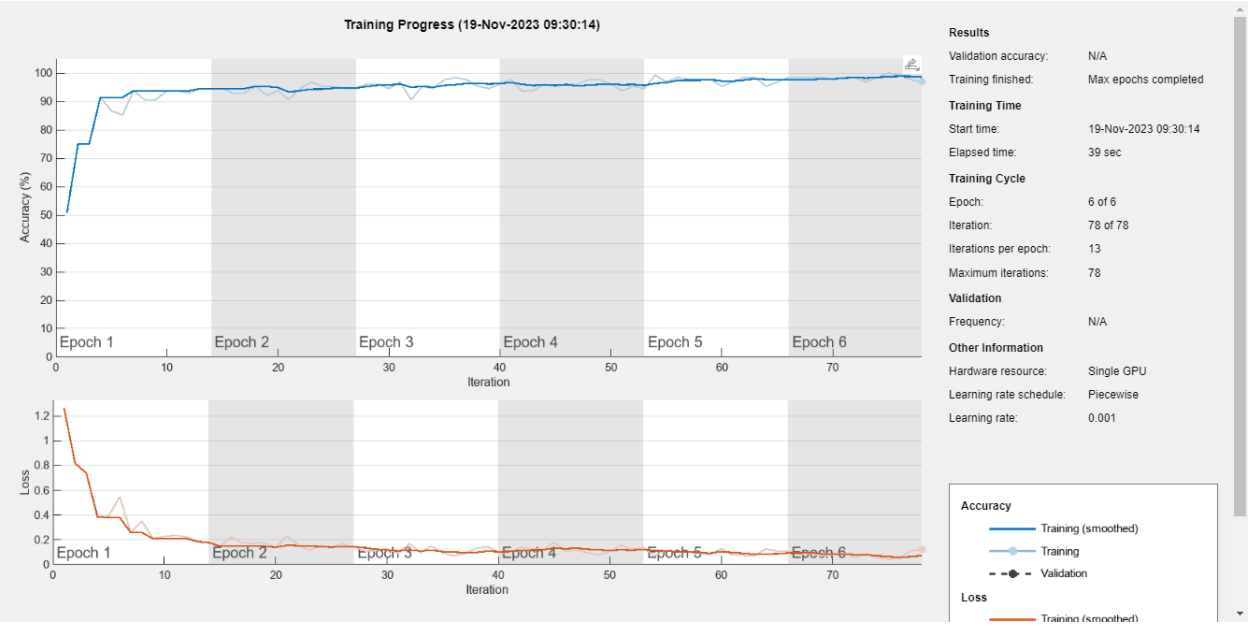


L1 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} |W_i|$$

L2 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} W_i^2$$



آموزش *Segmentation* در دو مرحله انجام می شود یک مرحله برای داده ها *Camvid* و یکی برای داده های معدن

```
opts = trainingOptions('sgdm', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',400, ...
    'MiniBatchSize',64);
```

و مرحله دوم بر روی داده های معدن با این مشخصات:

```
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',10,...
    'LearnRateDropFactor',0.3,...
    'Momentum',0.9, ...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',0.005, ...
    'MaxEpochs',1, ...
    'MiniBatchSize',8, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
```

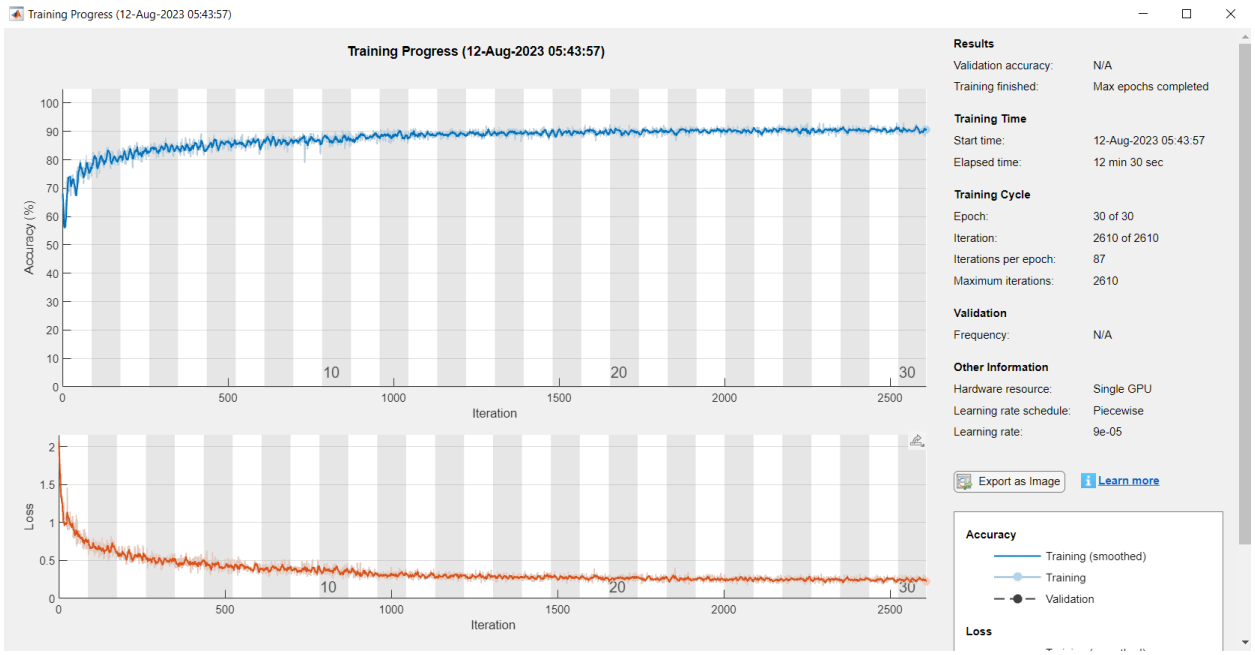
```
    'Plots','training-progress',...
    'ValidationPatience', 4);



doTraining = true;
if doTraining
    [net_madan_19_left_with_pretrained,          info]          =
trainNetwork(combine(imds,pxds),lgraph_512_pretrained,opt
ions);
end
```

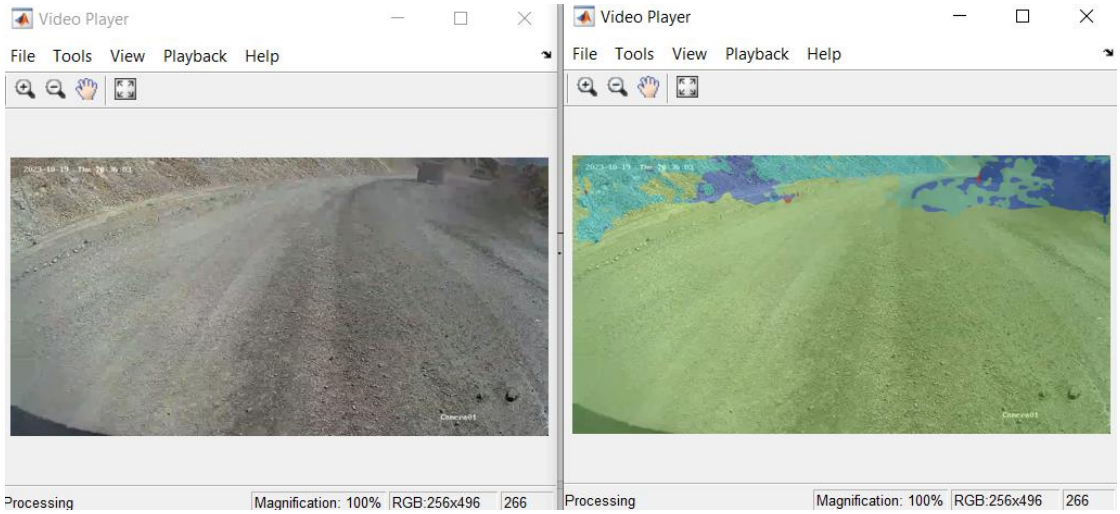
توجه داشته باشید که اگر بخواهیم سایز تصاویر آموزشی را تغییر دهیم کافی است مدل آموزش دیده اولیه را در ابزار *Deep Network Designer* بارگذاری کرده و لایه اول آن را تغییر دهیم و سپس این لایه را به عنوان *Backbone* به مدل بدهیم.

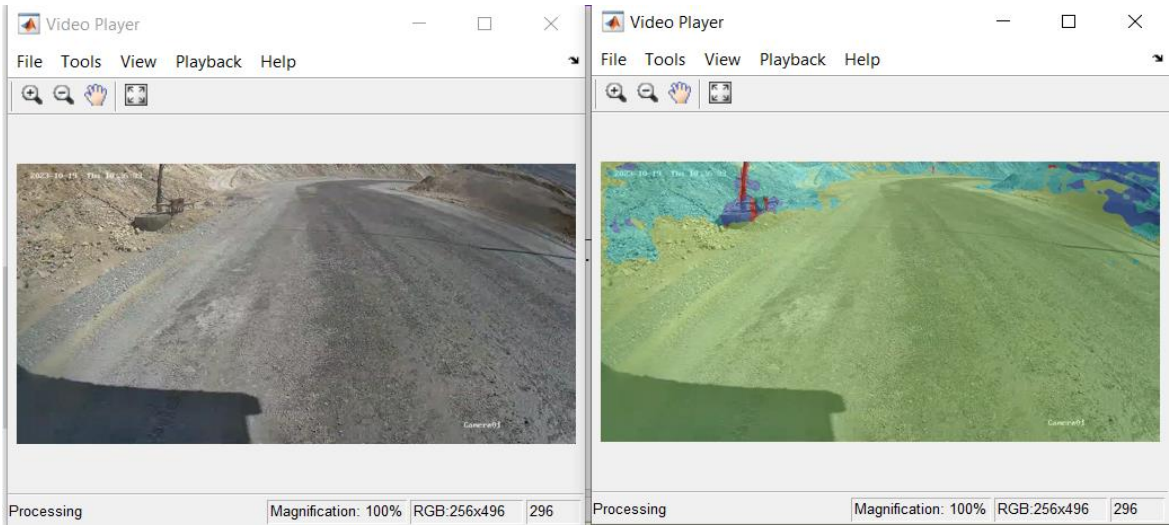
سایر تصاویر معدن 512x916 در نظر گرفته شده است. شاید این سایز تصاویر بزرگ به نظر برسد اما این کار باعث افزایش دقت مدل می شود چرا که لبه مشخصی برای تفکیک جاده وجود ندارد و همه چیز از یک نوع است تنها ارتفاع تغییر می کند. با کوچک کردن تصاویر این جزئیات از بین می رود. البته راهکار های دیگری نیز وجود دارد مثل استفاده از داده های لایدار برای تشخیص ارتفاع که بعدا بحث خواهد شد.

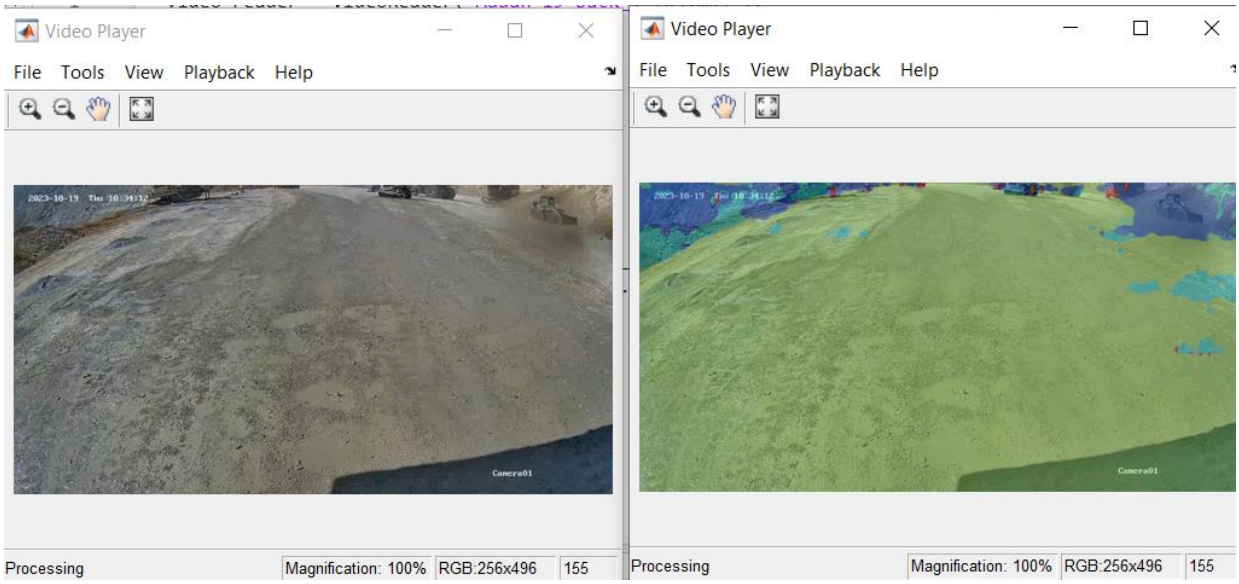

Training Progress (12-Aug-2023 05:43:57)
Training Progress (12-Aug-2023 05:43:57)
Accuracy (%)
Iteration
Loss
Iteration
10
20
30
Results
Validation accuracy: N/A
Training finished: Max epochs completed
Training Time
Start time: 12-Aug-2023 05:43:57
Elapsed time: 12 min 30 sec
Training Cycle
Epoch: 30 of 30
Iteration: 2610 of 2610
Iterations per epoch: 87
Maximum iterations: 2610
Validation
Frequency: N/A
Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Learning rate: 9e-05
Export as Image
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss

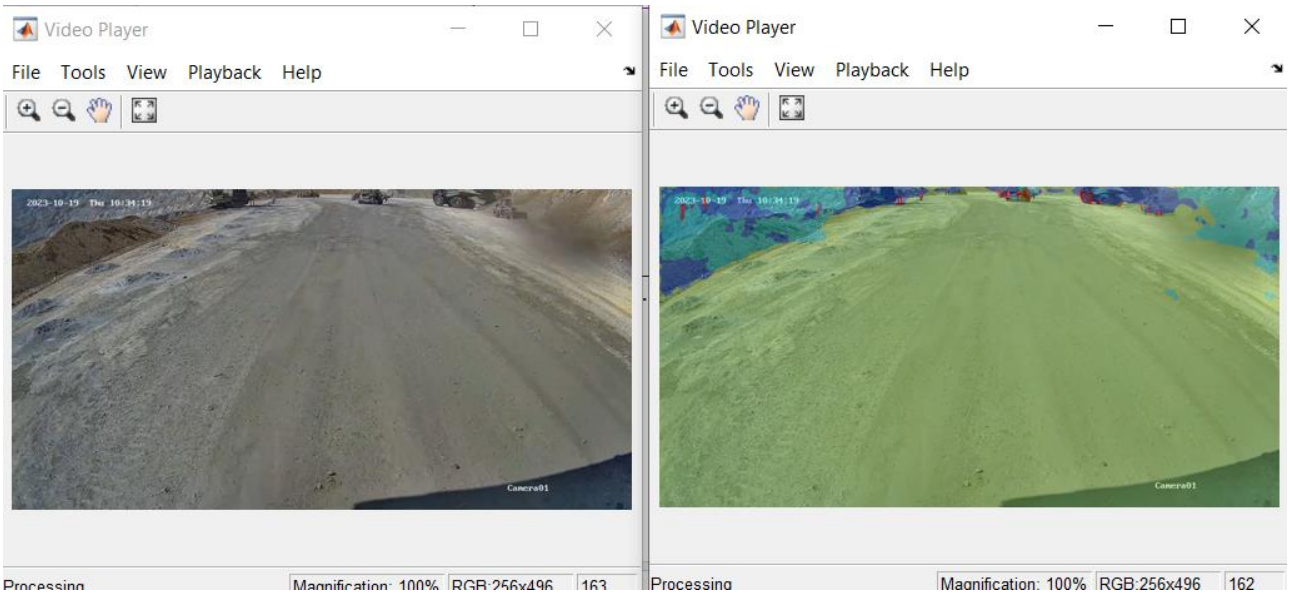
خروجی مدل بر روی تصاویر

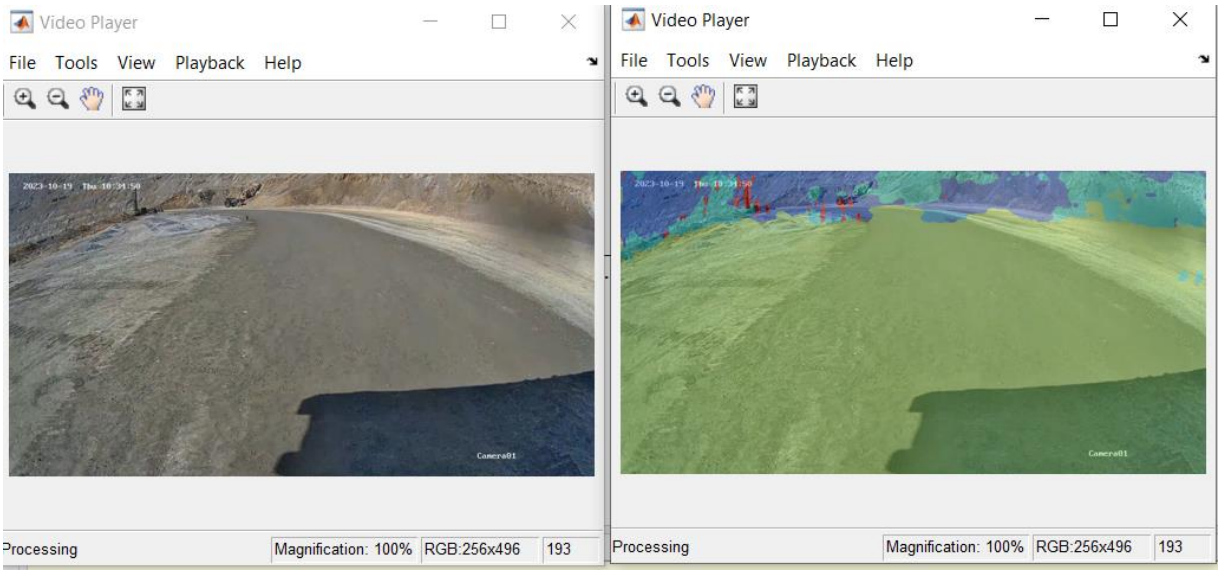
Semantic Segmentation

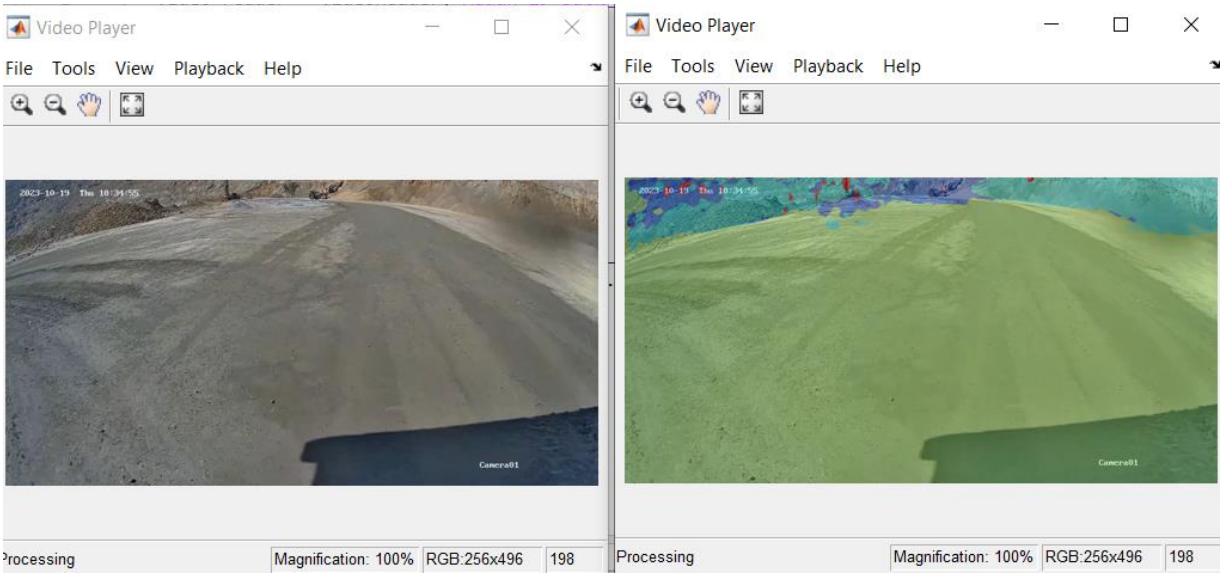


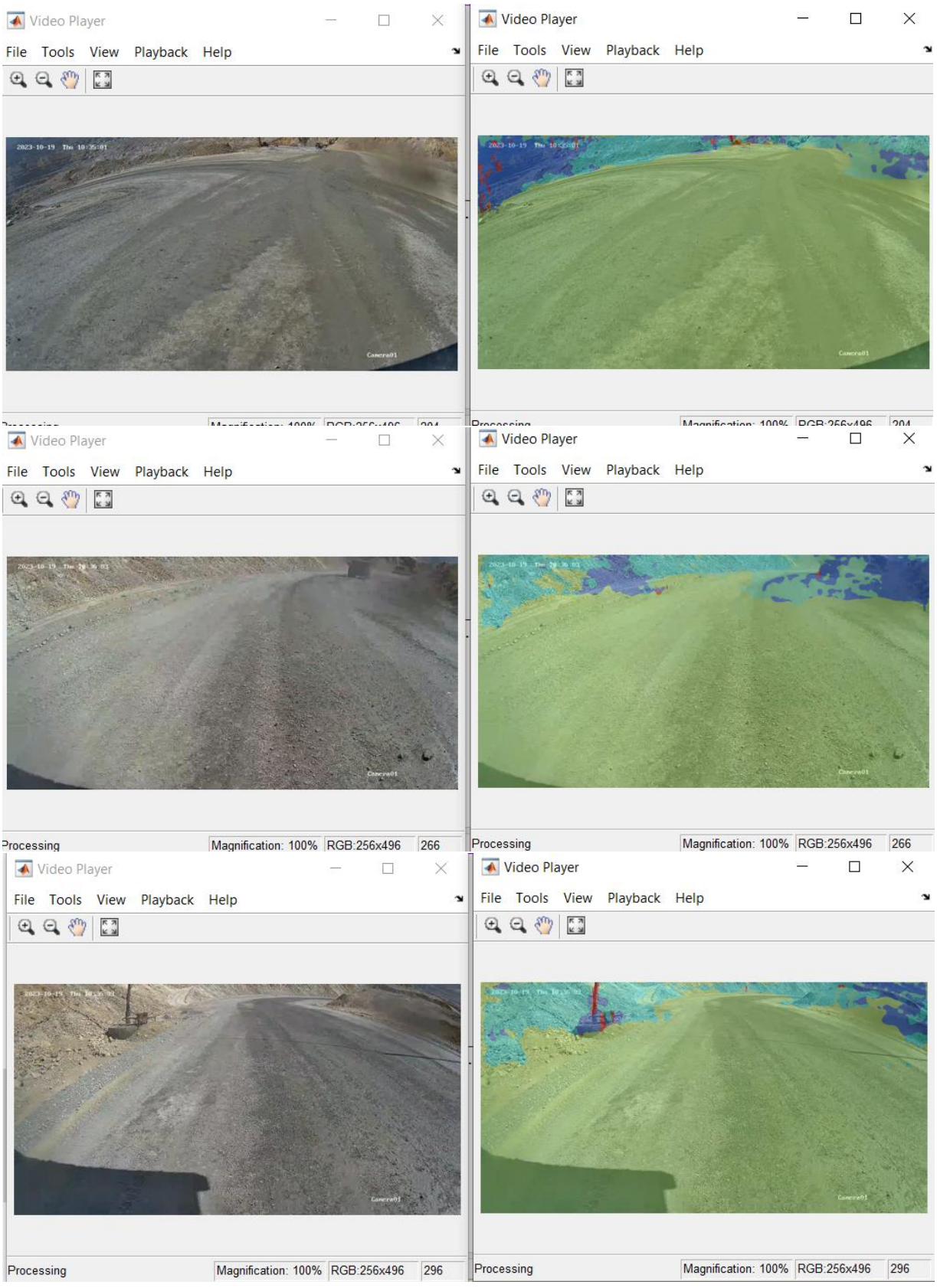



Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
155
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
155

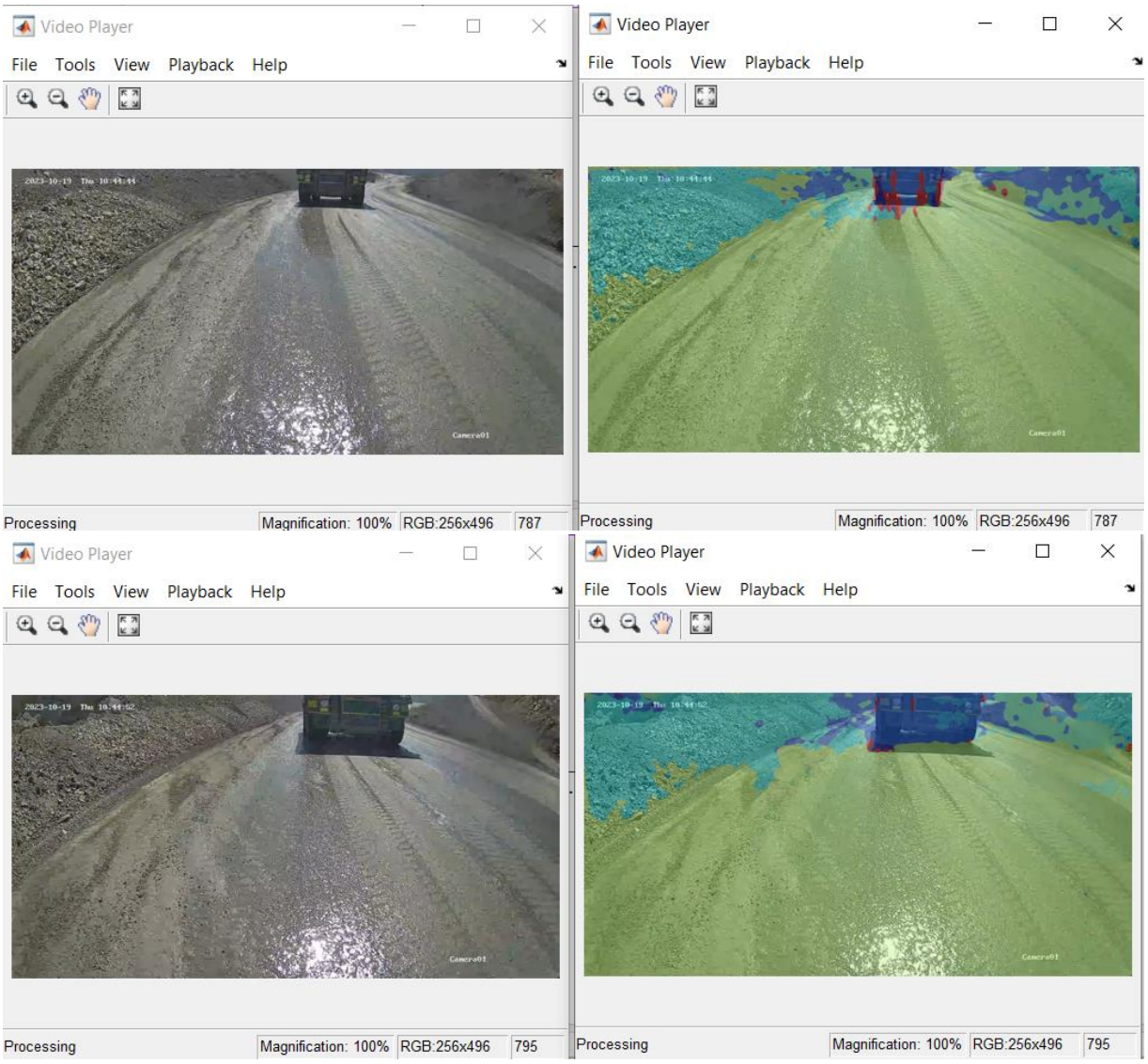

Video Player
File Tools View Playback Help
Video Player
File Tools View Playback Help


Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
193
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
193

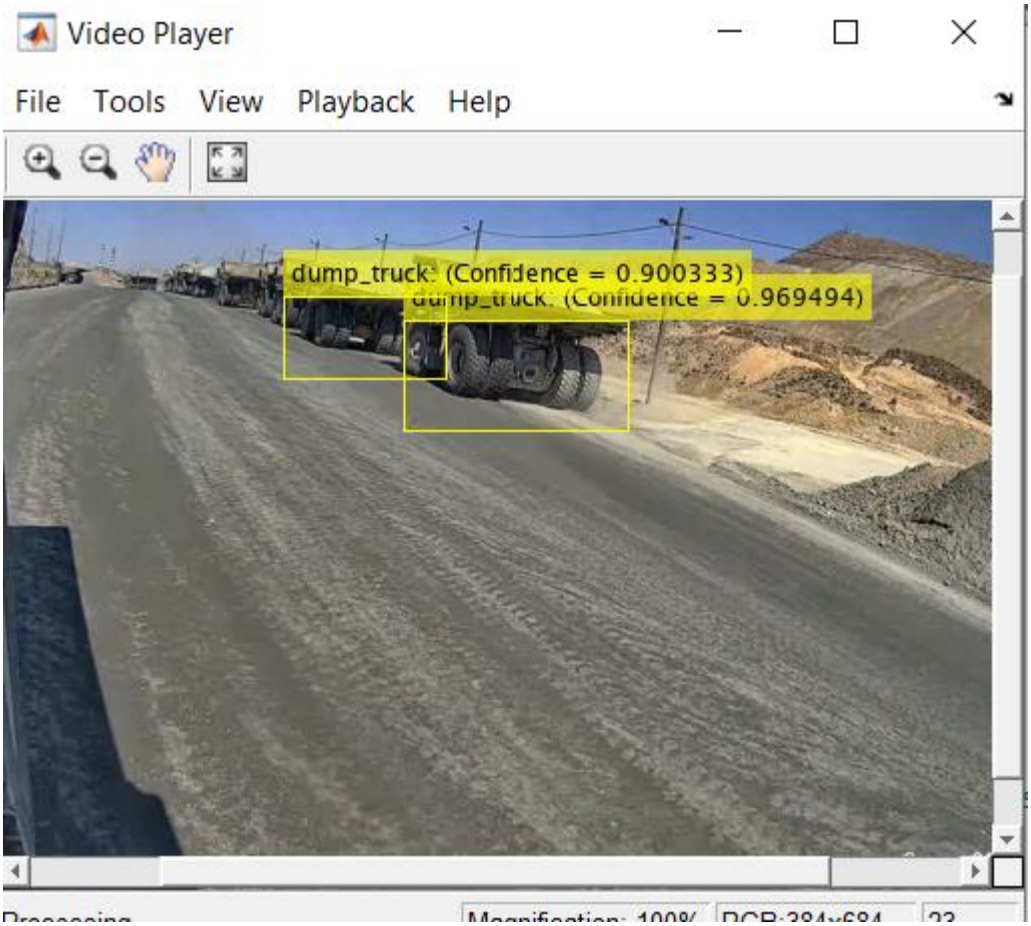

Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
198
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
198

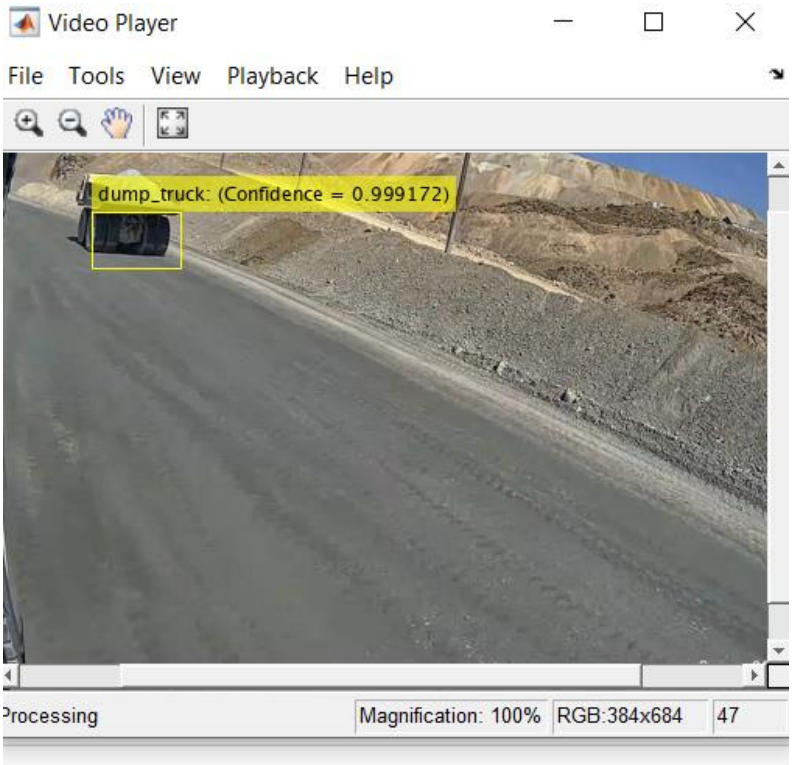

Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
266
296

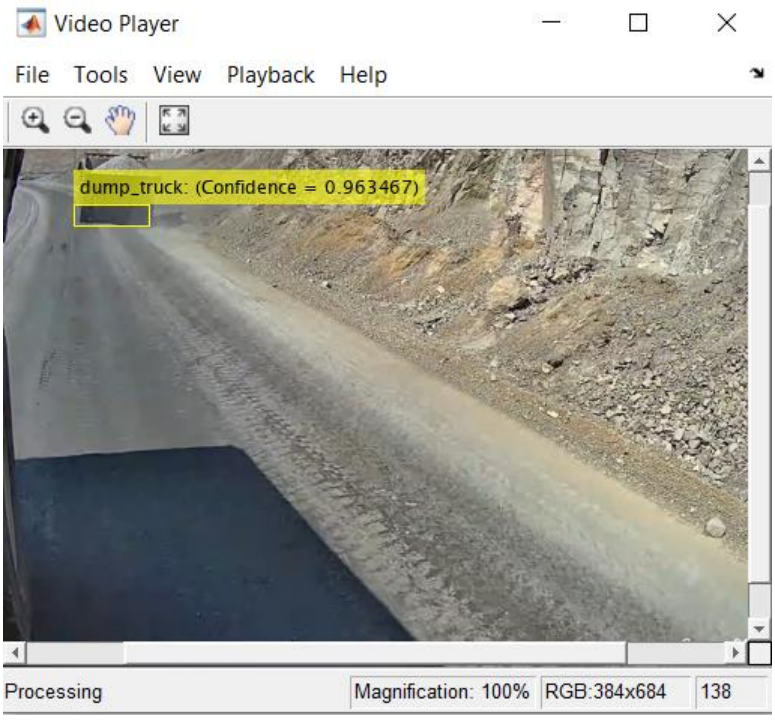

Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
482
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
482
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
553
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
553
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
719
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
719


Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
787
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
787
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
795
Video Player
File Tools View Playback Help
Processing
Magnification: 100%
RGB:256x496
795

Object Detection




Video Player
File Tools View Playback Help
dump_truck: (Confidence = 0.999172)
Processing
Magnification: 100%
RGB:384x684
47


Video Player
File Tools View Playback Help
dump_truck: (Confidence = 0.963467)
Processing
Magnification: 100%
RGB:384x684
138

راهکارهای افزایش دقت تشخیص جاده و مانع

در شبکه های عصبی دسترسی به دقت صد درصد بسیار دشوار و تقریبا ناممکن است اما تراک خودران نمی تواند خطا داشته باشد. چه گونه باید آن را طراحی کرد؟ مسئله این است که شاید ما نتوانیم *Accuracy* از حدی به بعد افزایش دهیم اما می توانیم *FN (False Negative)* خود را کاهش دهیم و بر تعداد *FP(False positive)*ها بیفزاییم. چرا اگر مانعی را تشخیص ندهیم ممکن است منجر به تصادف شود و خسارت جبران ناپذیری را در پی داشته باشد اما در صورتی که برعکس آن اتفاق بیفتد منجر به ایست های اضافی خواهد شد. در قسمت تشخیص جاده نیز باید عکس این عمل شود، یعنی اگر مدل می گوید که این قسمت جاده است حتما باید جاده باشد، و اگر مدل اشتباه کرده باشد منجر به سقوط تراک در دره خواهد شد . برای آنکه *FN* خود را کاهش دهیم باید *Threshold*قبولی موانع را افزایش دهیم. به علاوه برای جاده می توان از عملیات های *Morphology* فرسایشی استفاده کرد.

توجه داشته باشید که از طرفی *FP* ها بسیار زیاد شود، ایست های متوالی تراک مانع حرکت آن خواهد شد. برای حل این مشکل چندین راهکار وجود دارد .

۱. *Temporal Consistency:* در این روش از قواعد معناداری که در محیط است در کنار نتایج مدل استفاده می شود تا تصمیم گیری بهتری انجام شود. مثال های متنوعی برای شرایط وجود دارد :

بررسی مانع یا جاده در فریم های مختلف با توجه به حرکت خودرو: در واقع با بافر کردن نتایج قبلی و خروجی مدل تصمیم می گیریم که آیا یک شی وجود دارد یا خیر. به عنوان مثال اگر تنها در یک فریم یک شی با درصد کمی ظاهر شد آن را بافر کرده ولی در نظر گرفته نشود، یا بالعکس اگر در فریم های

قبلی مانعی با درصد اطمینان بالا بوده ولی در این فریم درصد اطمینان آن کم شده، بایستی این مانع در نظر گرفته شود.

نبودن مانع یا جاده در مختصات های خاص به توجه به دوربین: از آن جا که اتصالات دوربین ها ثابت می باشد می توان برای هر دوربین نواحی به خصوصی داشت که آن نواحی نباید مانعی وجود داشته باشد. به عنوان مثال پیکسل هایی که خود تراک را به نمایش می گذارند.

هماهنگی سنسور ها و دوربین ها با یکدیگر: کالیبراسیون دوربین ها ثابت است و در نتیجه باید ارتباطی میان خروجی مدل ها برای دوربین های مختلف باشد. برای مثال با استفاده از تکنیک هایی مثل *Stereo Matching* و دانستن عمق و یا استفاده از دیگر سنسور ها مثل لایدار برای تشخیص عمق می توان تناظری میان پیکسل های دو دوربین چپ و راست جلو برقرار کرد. پس از برقراری این تناظر می توان این طور فرض کرد که اگر در مختصات خاصی از دوربین چپ مانعی پیدا شد باید در ناحیه متناظر آن نیز در دوربین راست این مانع تشخیص داده شده باشد.

۲.ترکیب اطلاعات مدل های مختلف:

ما دو مدل Semantic *Segmentation* و *Object Detection* داریم . مدل *Semantic Segmentation* بر مبنای یازده کلاس از قبل آموزش داده شده است که یکی از آن ها خودرو است. در ساده ترین حالت می توان یک میانگین خطی میان این دو گرفت که درصد اطمینان نهایی را تعیین کند. با اینکه این کار تا حدودی ممکن است بهبود ببخشد اما راهکار ساخت یافته ای نیست و راهکار های ساختار یافته تری برای ترکیب دو مدل ضعیف وجود دارد تا از *Overfit* شدن مدل ها جلوگیری کند.

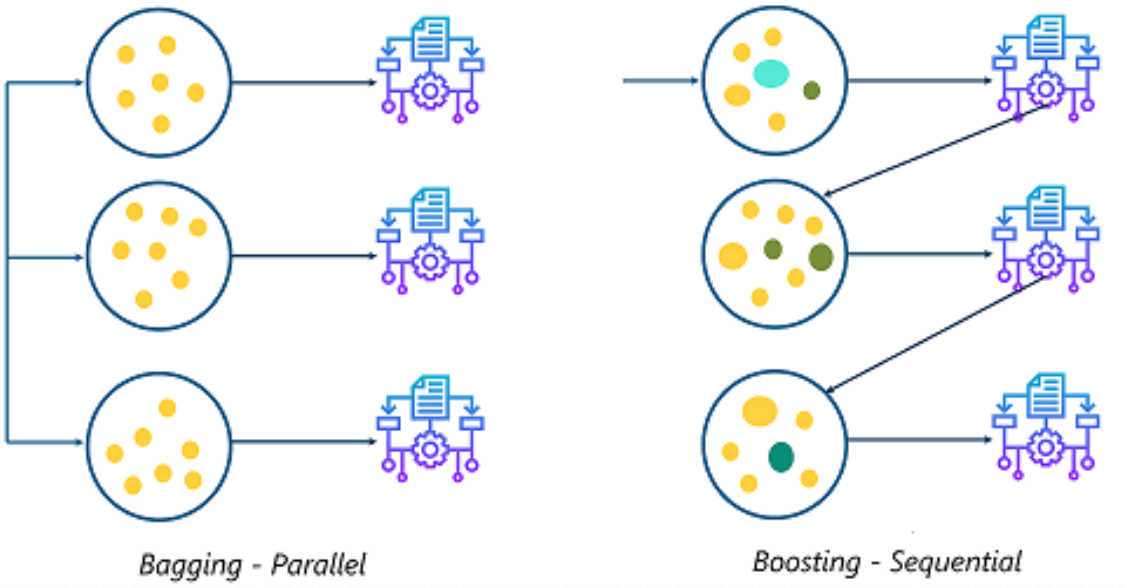
راهکار های ساخت یافته:

ترکیب نتایج دو مدل:

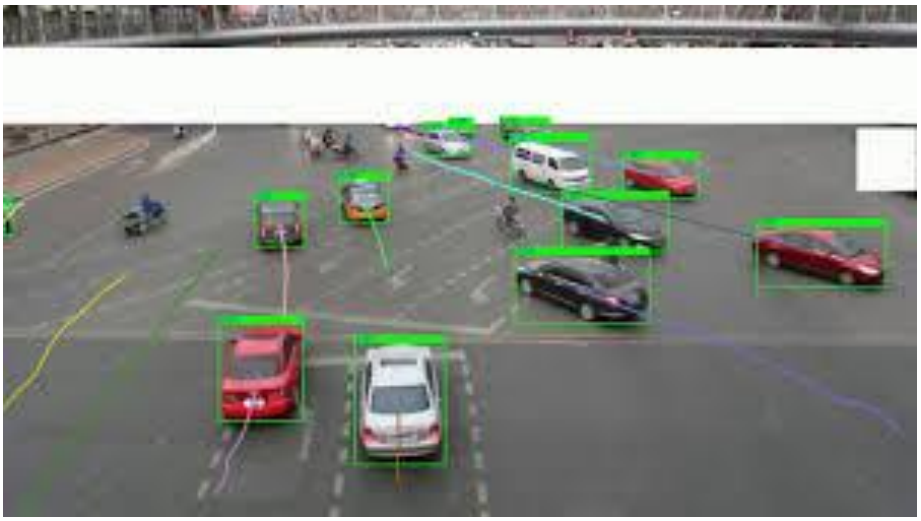
Bagging: که دو مدل را به صورت موازی آموزش می دهد و میان نتیجه آن ها یک میانگین خطی می گیرد.

Boosting: مدل ها را پشت سر هم آموزش می دهد.

Stacking: چندین مدل همگن را به صورت موازی آموزش می دهد و با استفاده از یک مدل بالاتر از نتیجه این دو مدل نتیجه نهایی را استخراج می کند.



فیلتر کالمن: *(Kalman Filter)* یک تخمین گر است که از تخمین حالت قبل و مشاهده فعلی برای محاسبه تخمین حالت فعلی استفاده می کند و یک ابزار بسیار قوی برای ترکیب اطلاعات در حضور نامعینی‌ها است. برای تبدیل های غیر خطی می توان از *EKF(Extended Kalman Filter)* یا *UKF* استفاده کرد.



سطوح *Data Fusion*

همان طور که می دانید ما در تراک خودران چندین ورودی تصویر داریم. دوربین ها و لایدار هر کدام داده جداگانه ای دارند که اجرای مدل بر روی آن ها خروجی متفاوتی را به همراه دارد. اما در نهایت باید چه گونه تصمیم بگیریم و موانع را تشخیص دهیم.

برای این کار باید ۲ مدل را با هم ترکیب کرد، یا به اصطلاح *Data Fusion* انجام دهیم. طبقه بندی های مختلفی برای سطوح *Data Fusion* وجود دارد مثل *(JDB)* اما به صورت ساده می توان بررسی کرد آن ها را به سه دسته تقسیم بندی کرد.

۱. *Low Level:* در این سطح داده های تصاویر با هم ترکیب می شوند و سپس به مدل داده می شود. هدف از ترکیب داده ها در این سطح افزایش *SNR(Signal Noise Ratio)* است . چرا که سنسور ها و تصاویر داری نویز می باشند و در این مرحله می توان این نویز را کاهش داد. در واقع در این *Level Image Fusion* داریم.

۲. *Middle Level:* در این از ویژگی های استخراج شده از هر تصویر استفاده شده و نتایج آن ها با هم ترکیب می شود.

۳. *High Level*: در این سطح تصمیم گیری بر اساس نتایج به دست آمده از مدل های مختلف انجام می شود.

چالش ها

یکی از بزرگترین چالش ها تهیه دیتا های بزرگ برچسب دار است. *Ground Truth Labeler* به شدت در برچسب زنی برای *Object Detection* کمک کرد اما برچسب زنی داده های *Semantic Segmentation* به شدت زمانبر است و ممکن است با خطای انسانی در حجم عظیم داده مواجه شویم. هر چه قدر که مدل ما قوی باشد اگر داده ها به اندازه کافی نباشند و خوب برچسب زنی نشوند ، نتیجه مدل مطلوب نخواهد بود. برای حل این مشکلات راه حل هایی پیشنهاد می شود:

یادگیری فعال : در یادگیری فعال یک مدل برچسب زن از نمونه برچسب های انسانی یاد می گیرد که چه گونه بقیه داده ها را *Label* گذاری کند . هر گاه موردی متفاوت دریافت کرد که درصد اطمینان آن پایین است آن را ارجاع داده و برچسب آن را می پرسد تا توسط نیروی انسانی برچسب زده شود.

یادگیری نیمه نظارتی: در این یادگیری ماشین با مجموعه از داده های برچسب دار در کنار داده های بدون برچسب آموزش داده می شود.(غالبا تعداد داده های برچسب دار از داده های بدون بر چسب بسیار کمتر است). سپس توسط الگوریتمی (مثل الگوریتم های خوشه بندی (*Clustering*)) داده های بدون برچسب، برچسب گذاری می شوند.

چالش بعدی اجرای تشخیص جاده در معادن است که بسیار با خیابان ها متفاوت است، چرا که خیلی مرز مشخصی میان جاده و حاشیه آن وجود ندارد. و از آن جا که مدل با توجه تصاویر رنگی آموزش داده می شود ، می تواند تغییرات کوچک آن را به اشتباه بیندازد. به عنوان مثال تشخیص روز و شب ۲ مدل مجزا می خواهد و به شرایط نوری حساس می شود. که باید از سنسور های دیگری به کمک دوربین استفاده شود.

نتیجه‌گیری و پیشنهادات

مورد اول استفاده از داده سنسور های دیگر در کنار دوربین است . لایدار می تواند برای ما عمق را تشخیص دهد و اگر شبکه عصبی با آن آموزش داده شود دیگر به شرایط نوری (مثل شب و روز و یا سایه ها) حساس نمی باشد و برای تشخیص جاده بهتر است.

برای تشخیص اشیا به صورت دقیق تر، راهکار هایی وجود دارد که نیاز به داده های زیاد و برچسب زنی دقیق تر دارد. به عنوان مثال اینکه تراک به عنوان یک شی آموزش ندهیم و آن را به کلاس های جداگانه تفکیک کنیم. می توان زوایای مختلف تراک را کلاس مجزا دانست (*Truck From Back, Truck From Front*) . می توان اجزای یک مانع را نیز جداگانه آموزش داده های دقیق تری داشته باشیم. البته ممکن است با این روش به عملکرد مطلوب نرسیم چرا که این روش به داده زیاد نیاز دارد.

و در نهایت باید از *Data Fusion* مناسب استفاده کرد تا به دقت مطلوب برای تصمیم گیری برسیم و خطای مدل را کاهش دهیم.

منابع:

https://medium.com

https://towardsdatascience.com

https://www.mathworks.com