HybridNets: End-to-End Perception Network

Thanh Dat Vu Viet Hoai Bao Ngo

Thesis Defense

Advisor
Assoc. Prof. Duy Hung Phan

Outline

Outline

Outline

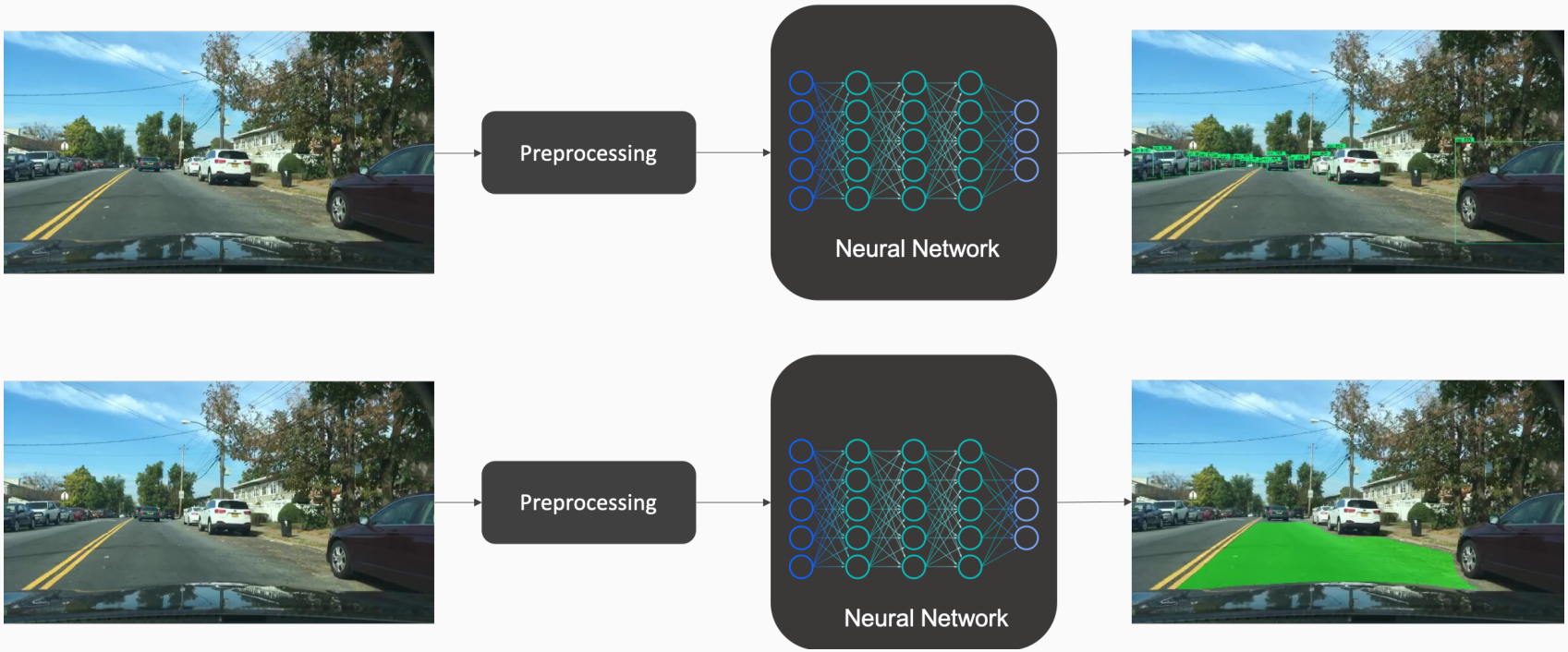
Outline

Outline

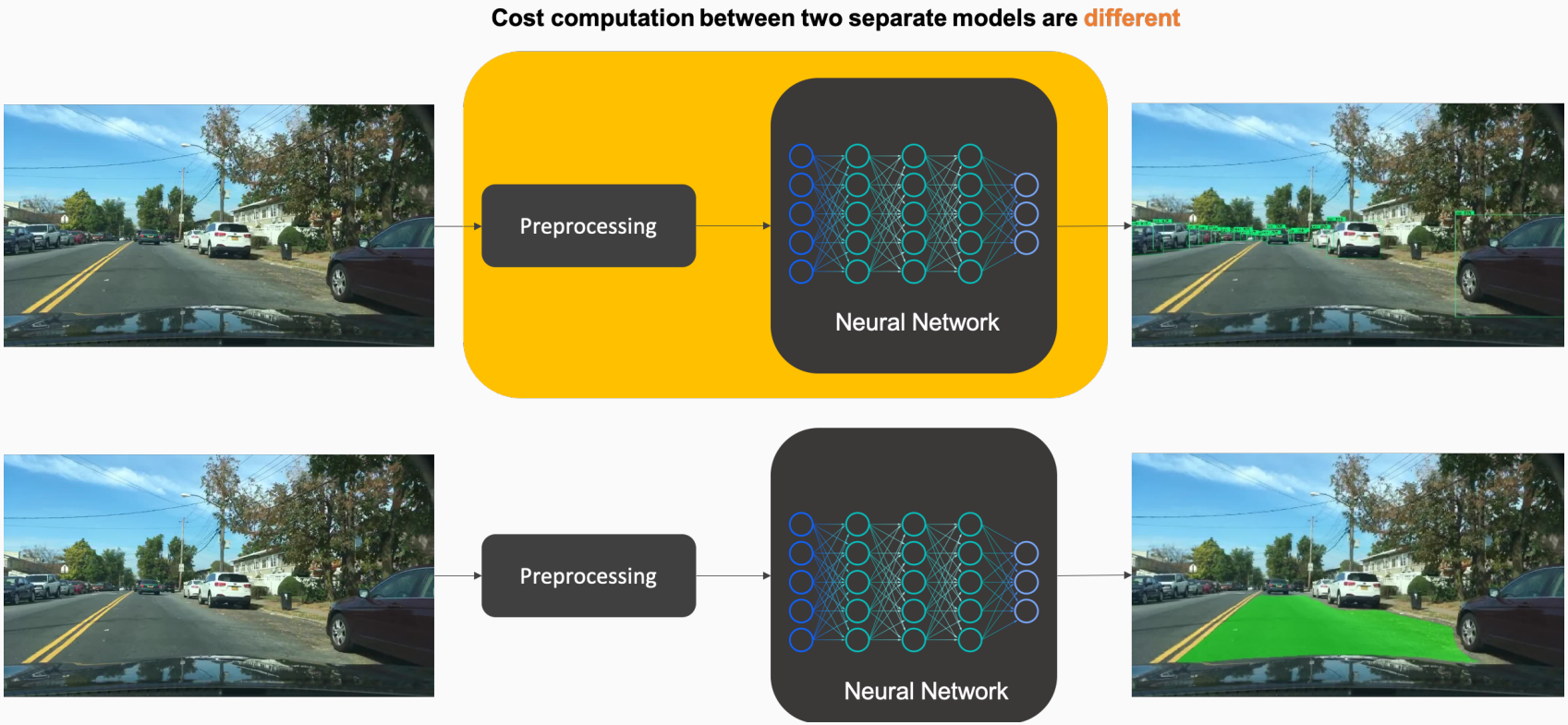
Introduction

Main Problem

Two separate models for object detection and semantic segmentation



Cost Computation



Synchronous Problem

Assume that Segmentation Model is faster than Object Detection Model

Decision Making

Assume that Segmentation Model is faster than Object Detection Model

Detect obstacle

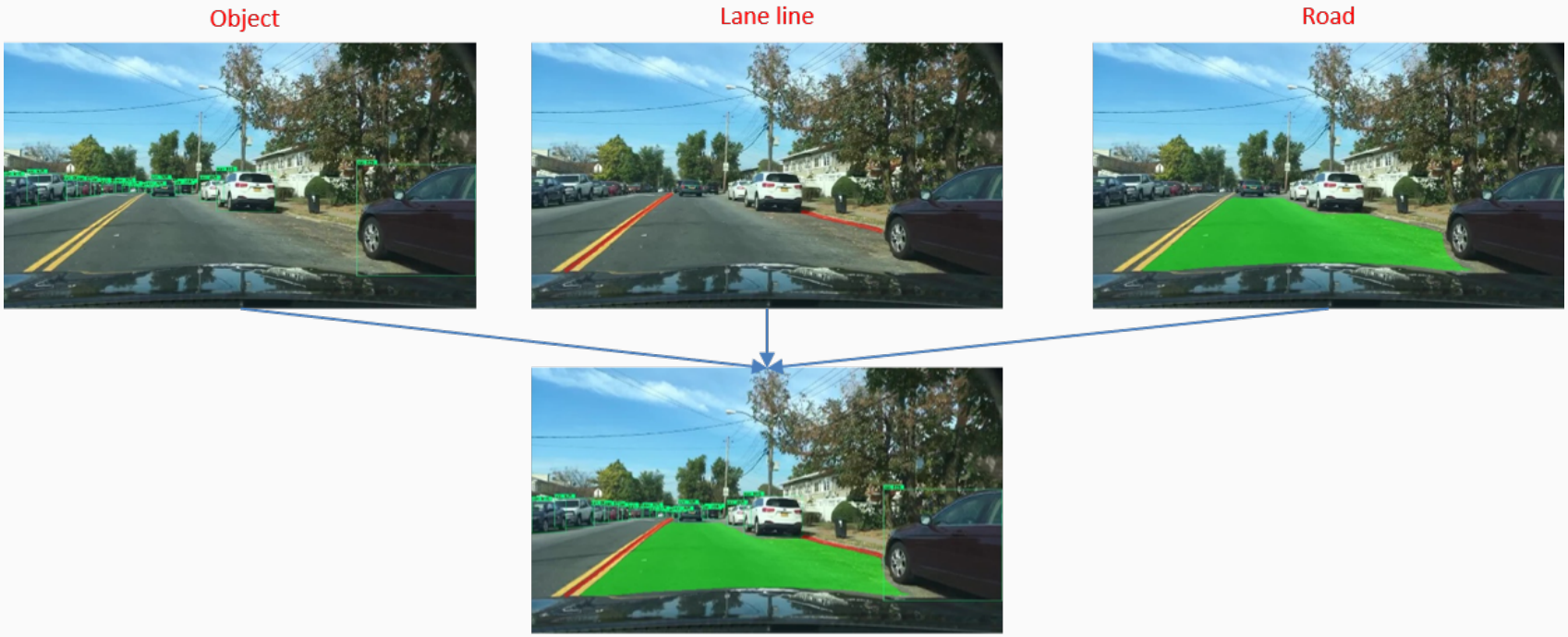
Predict angle

Go straight while the obstacle is in front of you.

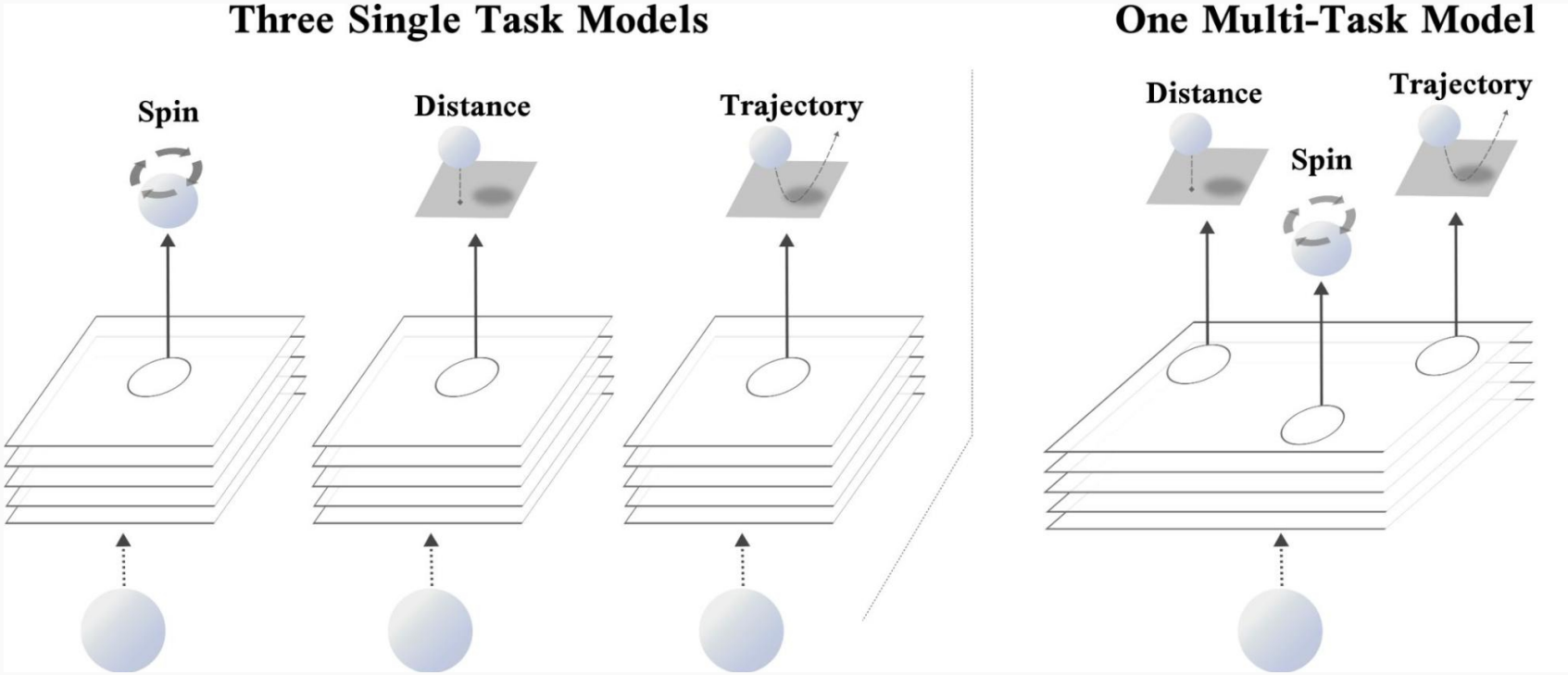
Very Dangerous!!!

Our purpose

- Build End-to-End Perception Network for Multi-tasks.
- Our work focus on Autonomous Driving Perception.

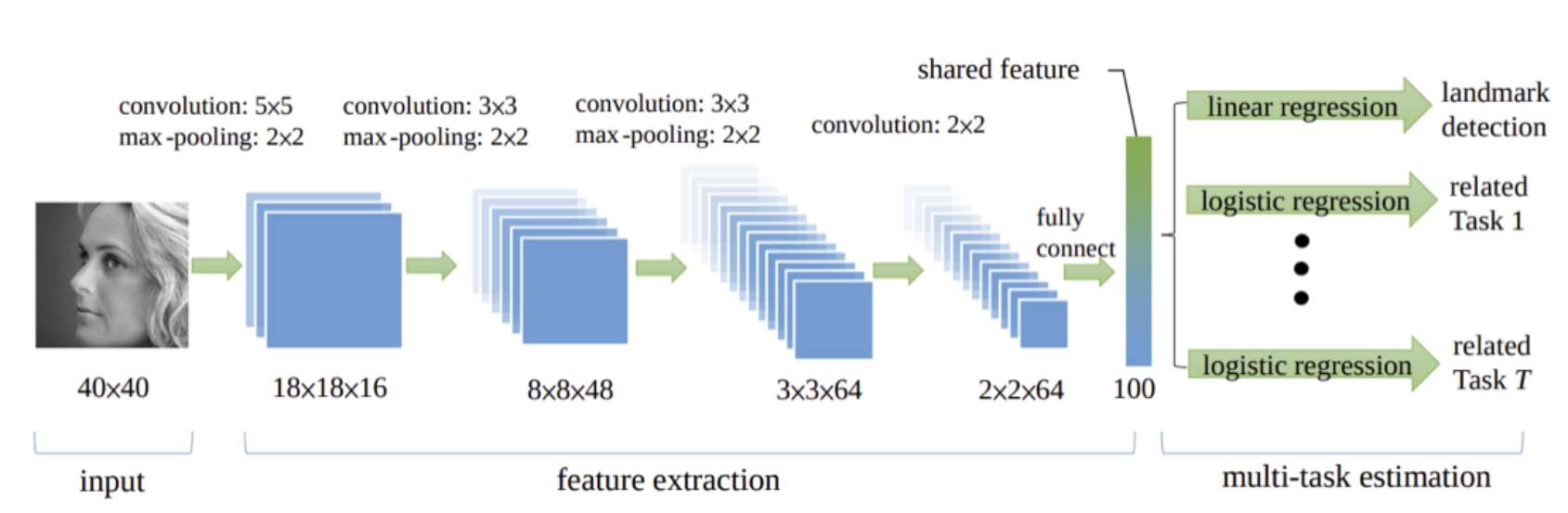


Literature Review



The Survey

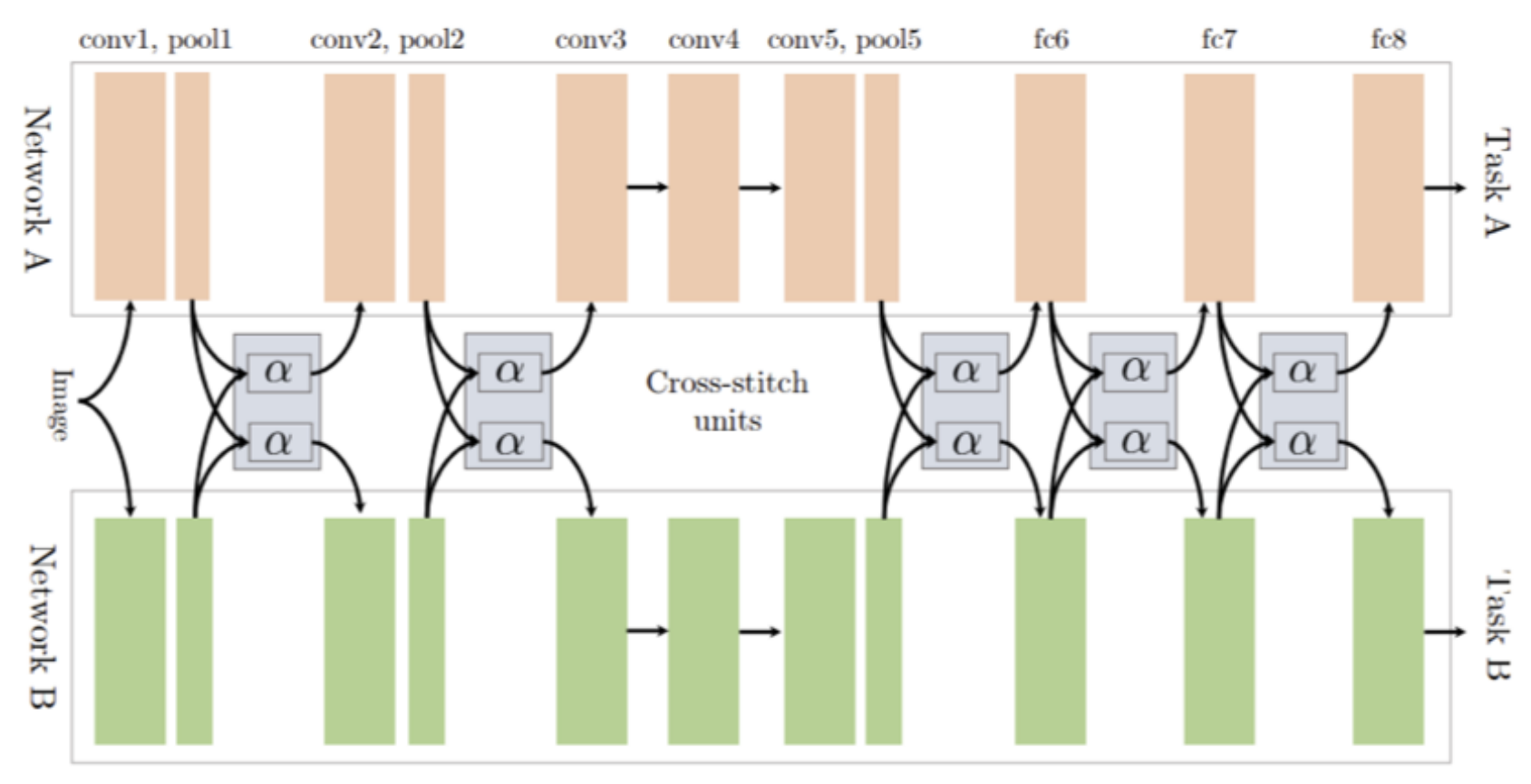
- The base feature extractor is made of a series of convolutional layers which are shared between all tasks, and the extracted features are used as input to task-specific output heads.



[1] Zhanpeng Zhang et al., Facial landmark detection by deep multi-task learning, ECCV 2014

The Survey

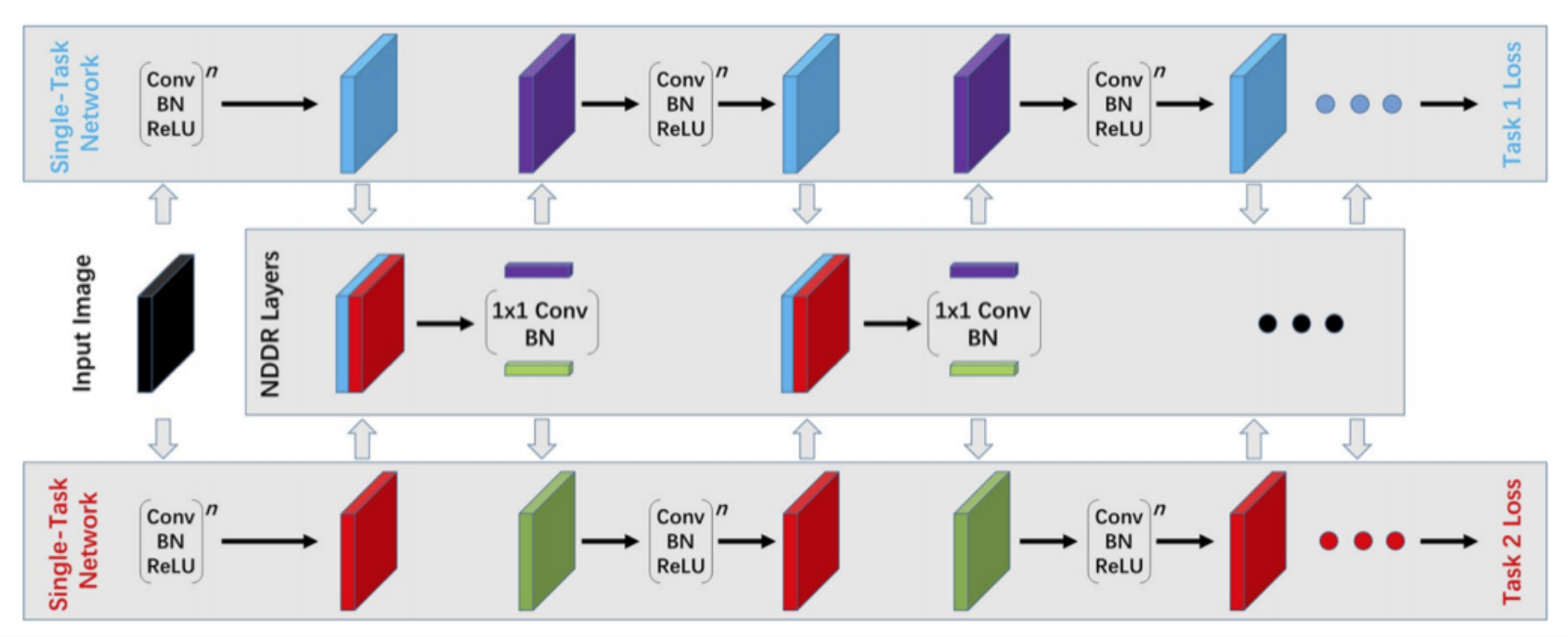
- Each task has a separate network, but cross-stitch units combine information from parallel layers of the different task networks with a linear combination is called Share Tunk problem.



[1]Ishan Misra et al., FCross-stitch networks for multi-task learning, IEEE Conference on Computer Vision and Pattern Recognition 2016

The Survey

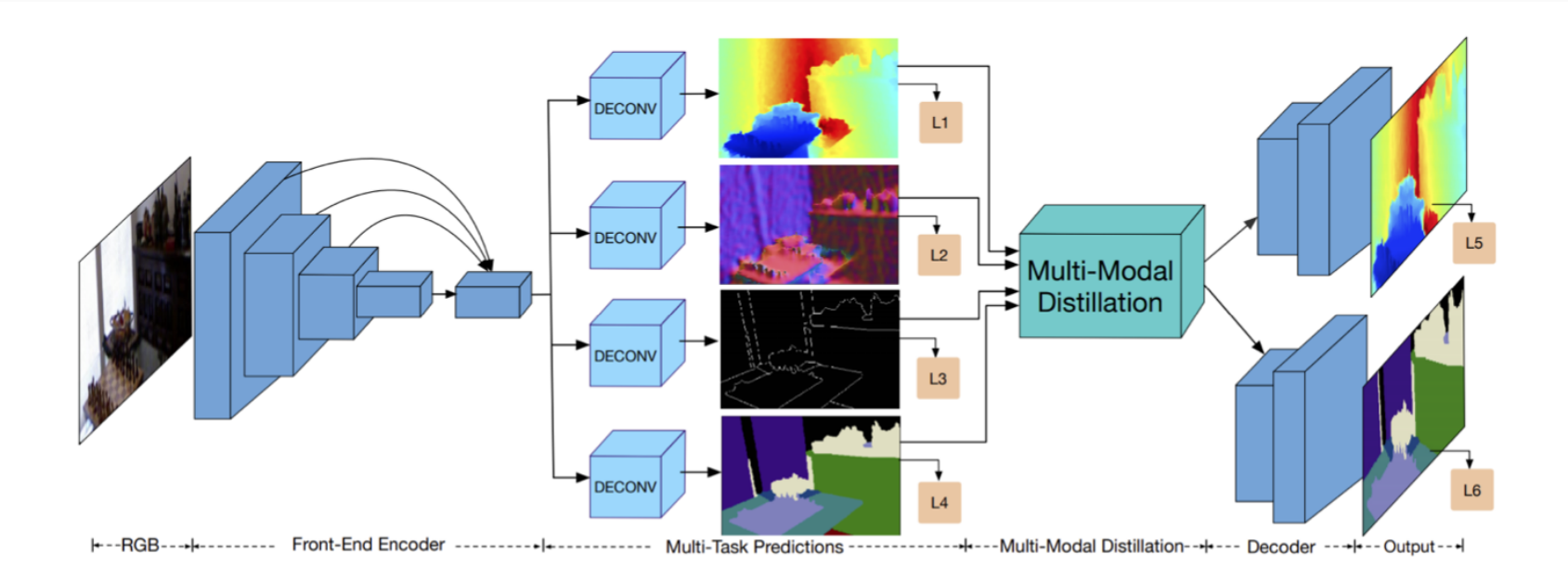
- Instead of combining information from different task networks with a linear combination of parallel features (as in Cross-Stitch networks (Misra et al., 2016)), NDDR-CNN uses concatenation and a 1x1 convolution to fuse features from separate task networks.



[1] Gao et al., Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction, IEEE Conference on Computer Vision and Pattern Recognition 2019

The Survey

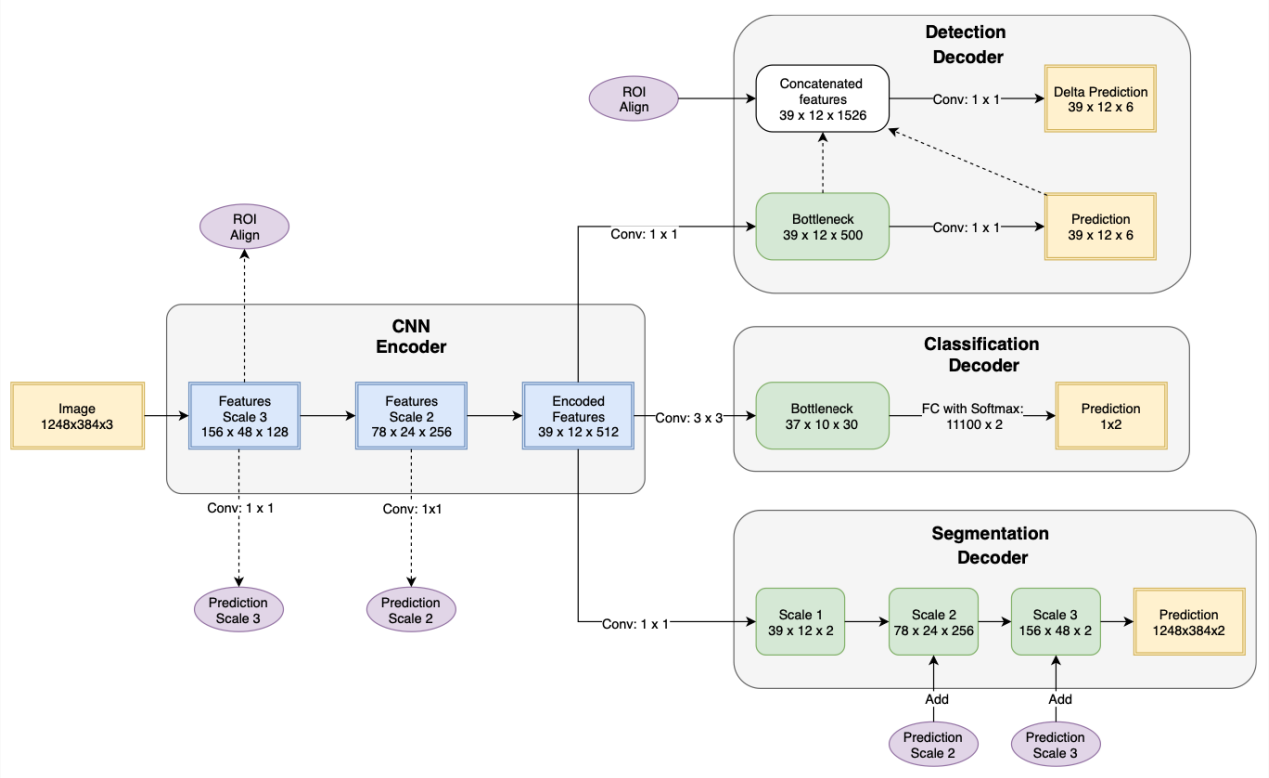
- Preliminary predictions are made for four tasks, then these predictions are re-combined and used to compute final, refined predictions for two output tasks.



[1]Xu et al., Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing, IEEE Conference on Computer Vision and Pattern Recognition 2018

MultiNet

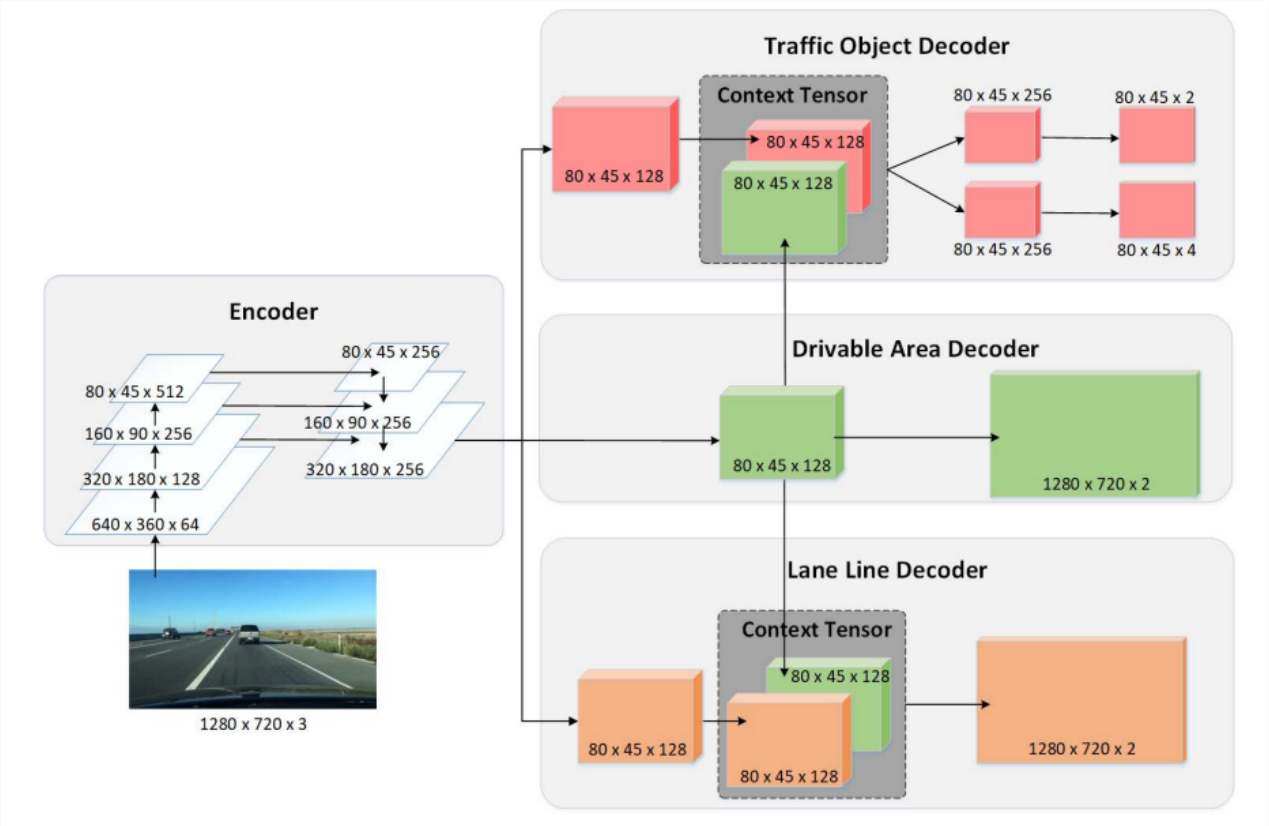
- Limitation: Fixed input size due to cell-based method.



[1] Marvin Teichmann et al., MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving, IEEE Intelligent Vehicles Symposium 2018

DLT-Net

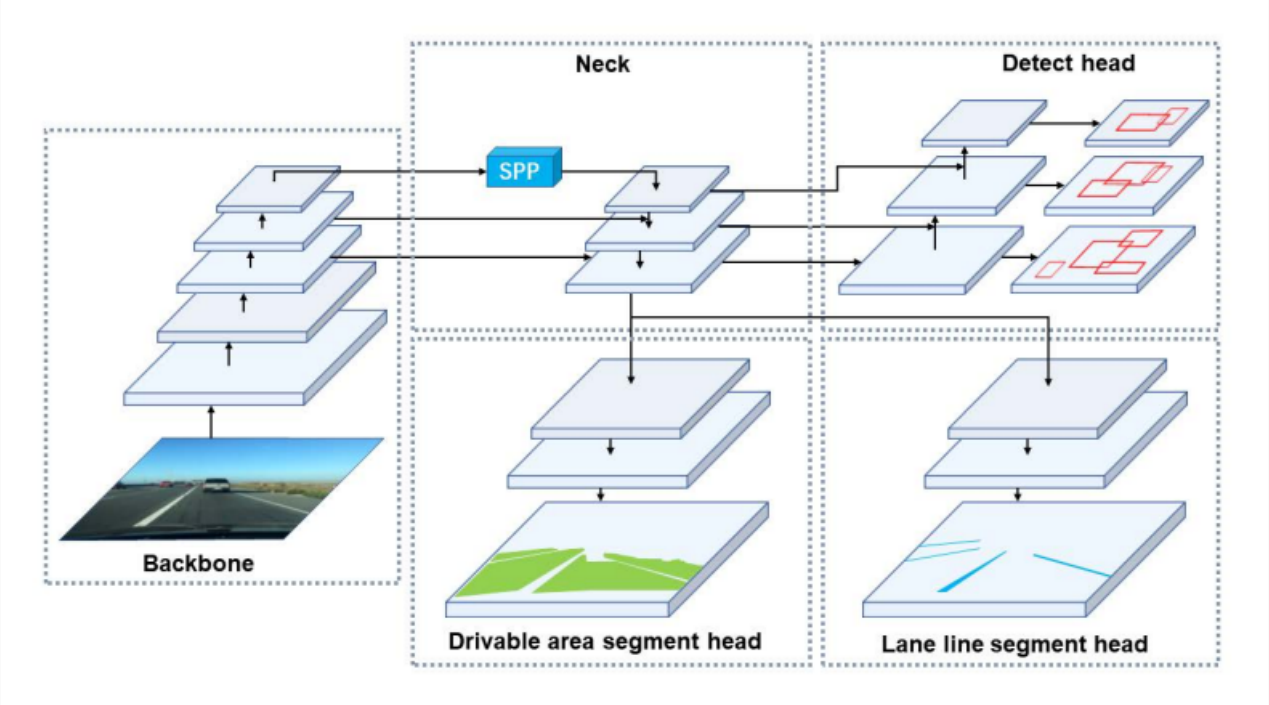
- Limitation: Shared context tensor from drivable area to other decoders, making finetuning of specific tasks harder.



[1]Yeqiang Qian, John M. Dolan, Ming Yang, DLT-Net: Joint Detection of Drivable Areas, Lane Lines, and Traffic Object, IEEE Transactions on Intelligent Transportation Systems 2020

YOLOP

- Limitation: Two different segmentation heads for drivable area and lane line.

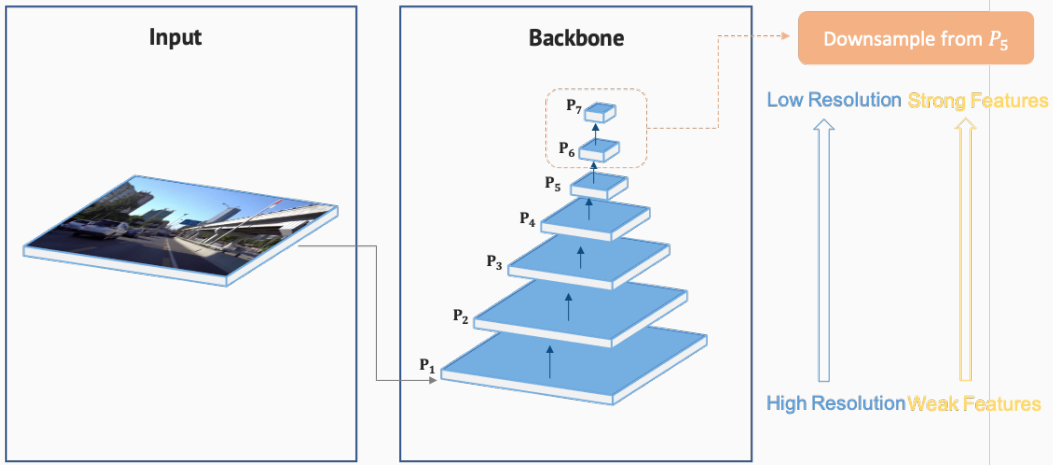


[1]Dong Wu et al., YOLOP: You Only Look Once for Panoptic Driving Perception, arXiv 2021

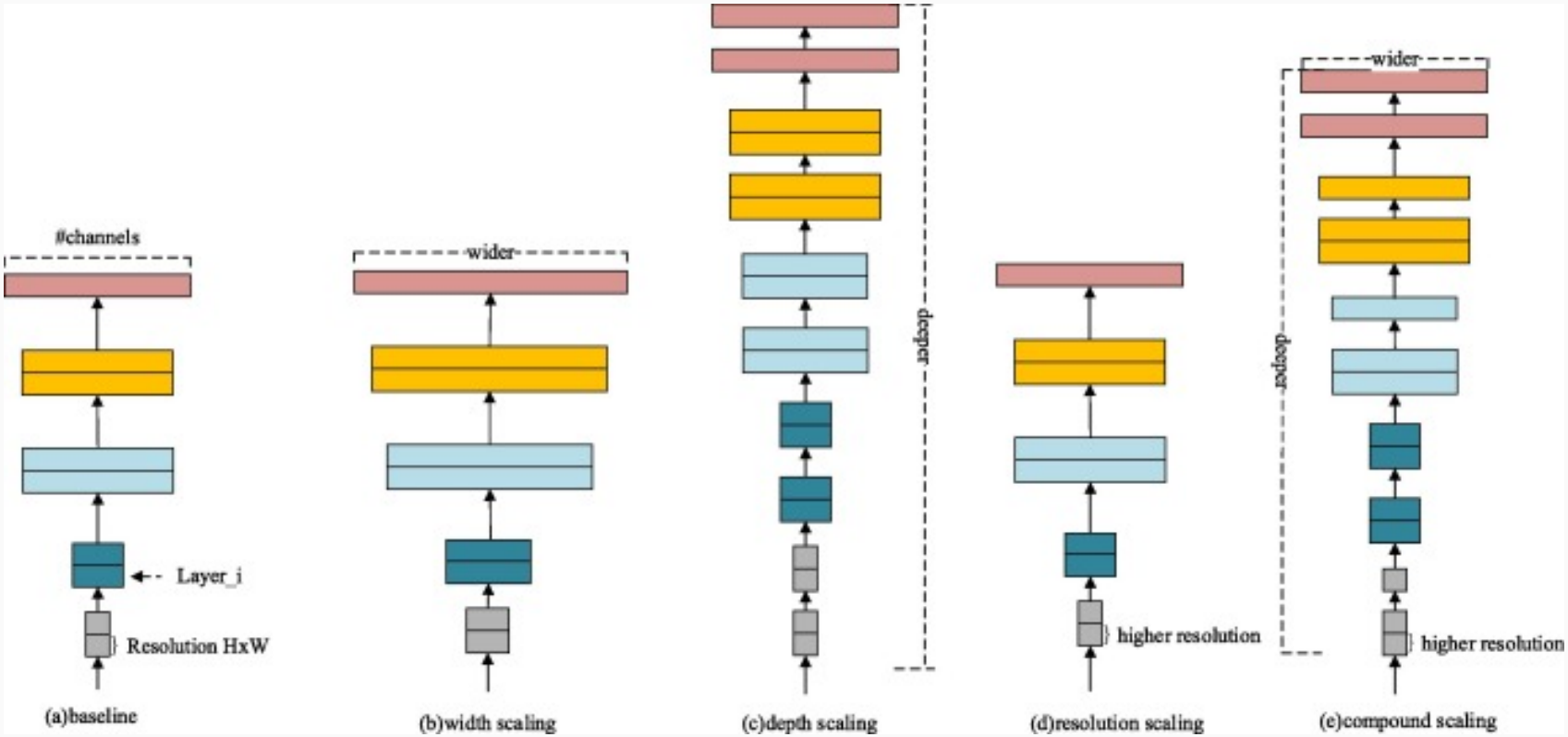
Proposed Methods

HybridNets Architecture

Backbone Network



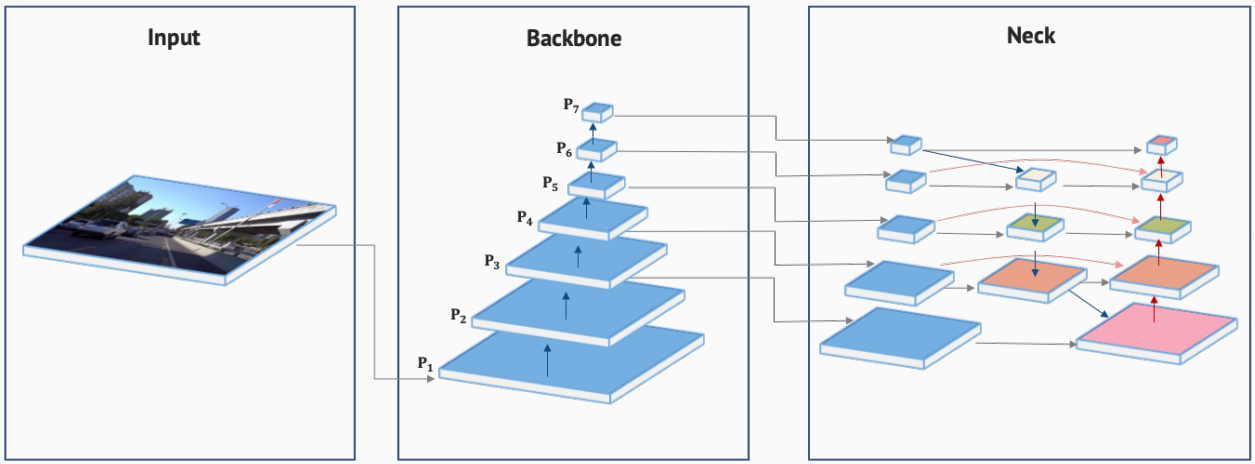
EfficientNet



[1]Mingxing Tan, Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, ICML 2019

Neck Network

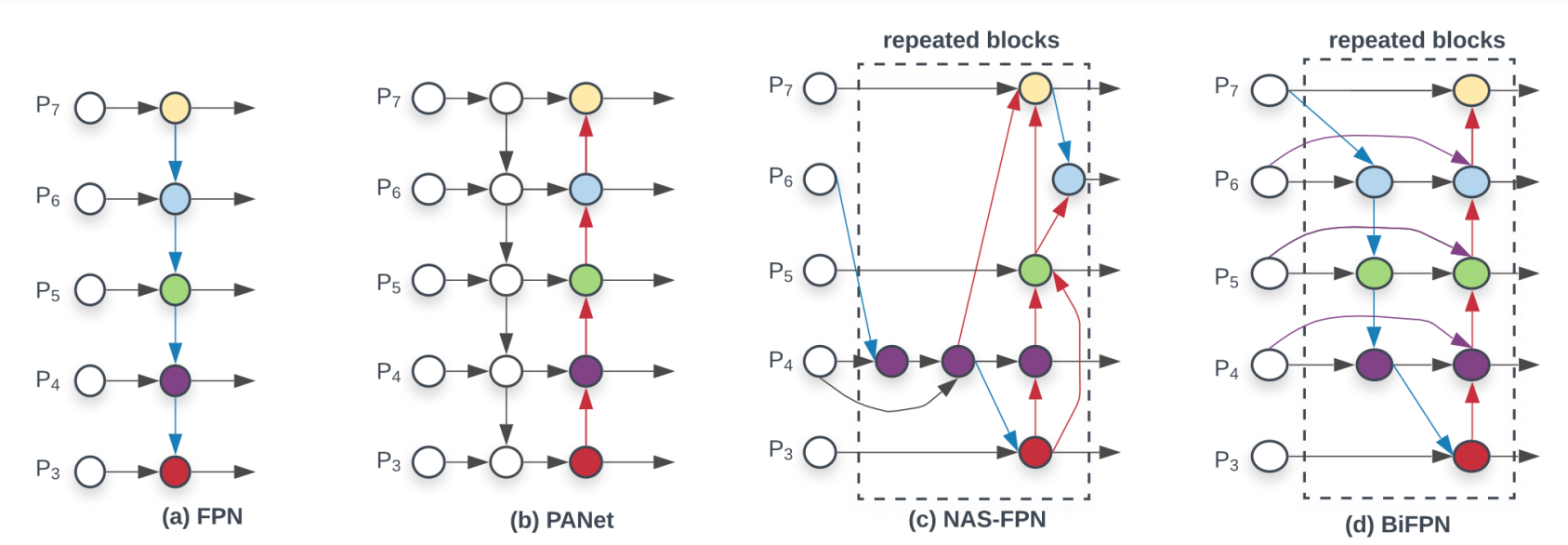
Why this research used Bi-FPN instead of FPN, PA-Net, NAS-FPN?



Bi-FPN [1]

[1] Mingxing Tan, Ruoming Pang, Quoc V. Le, EfficientDet: Scalable and Efficient Object Detection, CVPR 2020

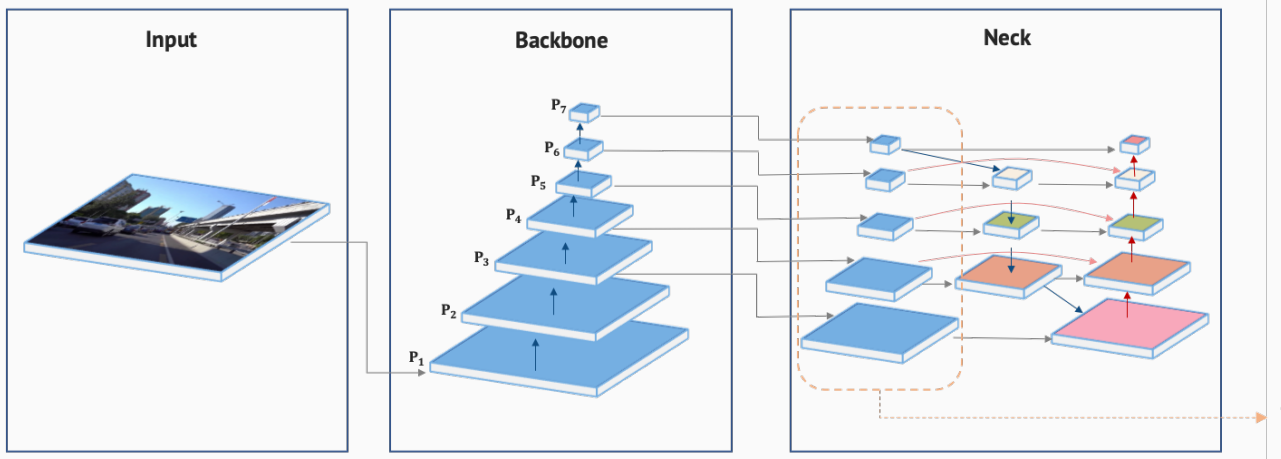
Feature Network Design



[1]Mingxing Tan, Ruoming Pang, Quoc V. Le, EfficientDet: Scalable and Efficient Object Detection, CVPR 2020

Neck Network

Why this research used Bi-FPN instead of FPN, PA-Net, NAS-FPN?

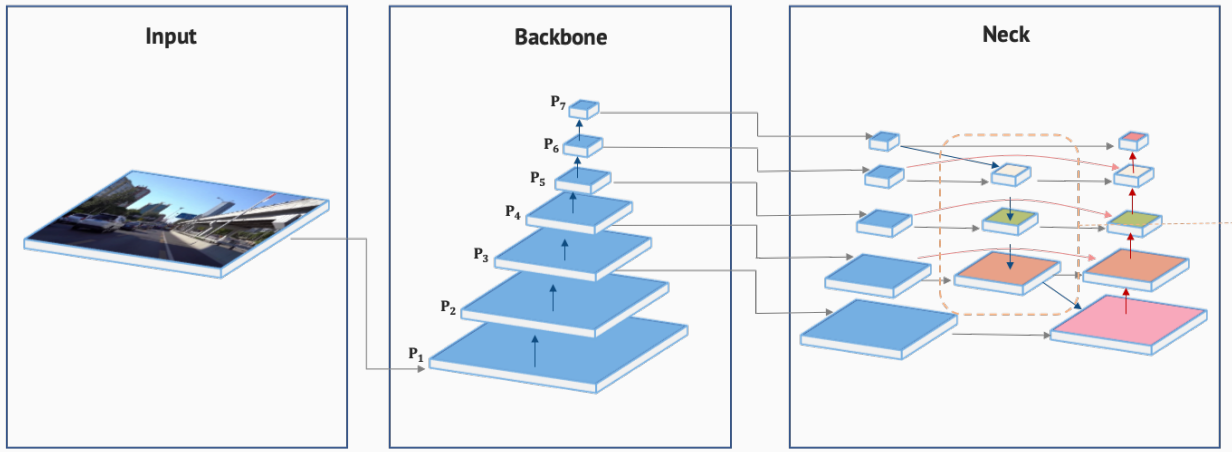


Bi-FPN [1]

Remove all those nodes that have only one input edge

Neck Network

Why this research used Bi-FPN instead of FPN, PA-Net, NAS-FPN?

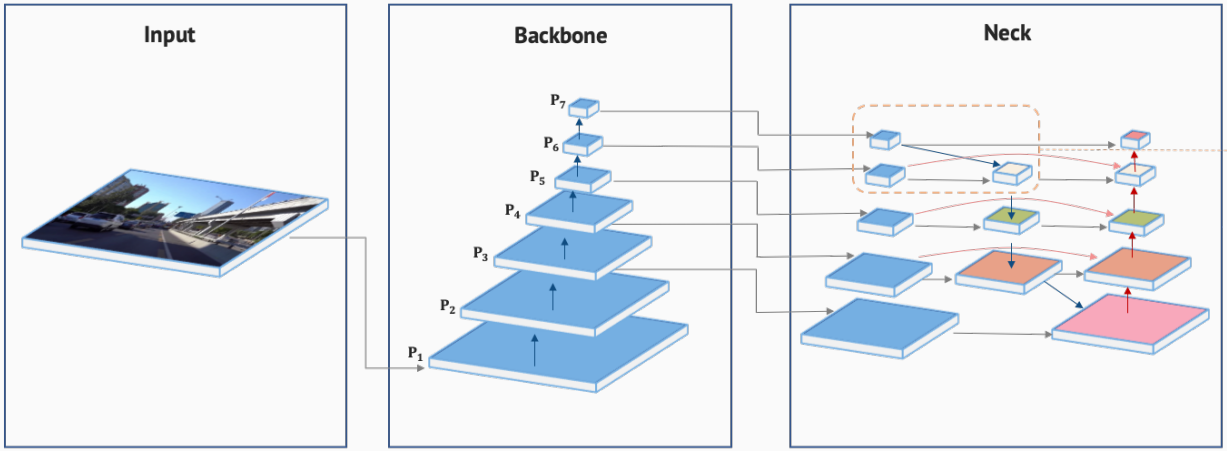


Bi-FPN [1]

Treat each bidirectional path as one single layer and have multiple of these to enable more high-level feature fusion.

Neck Network

Why this research used Bi-FPN instead of FPN, PA-Net, NAS-FPN?



Bi-FPN [1]

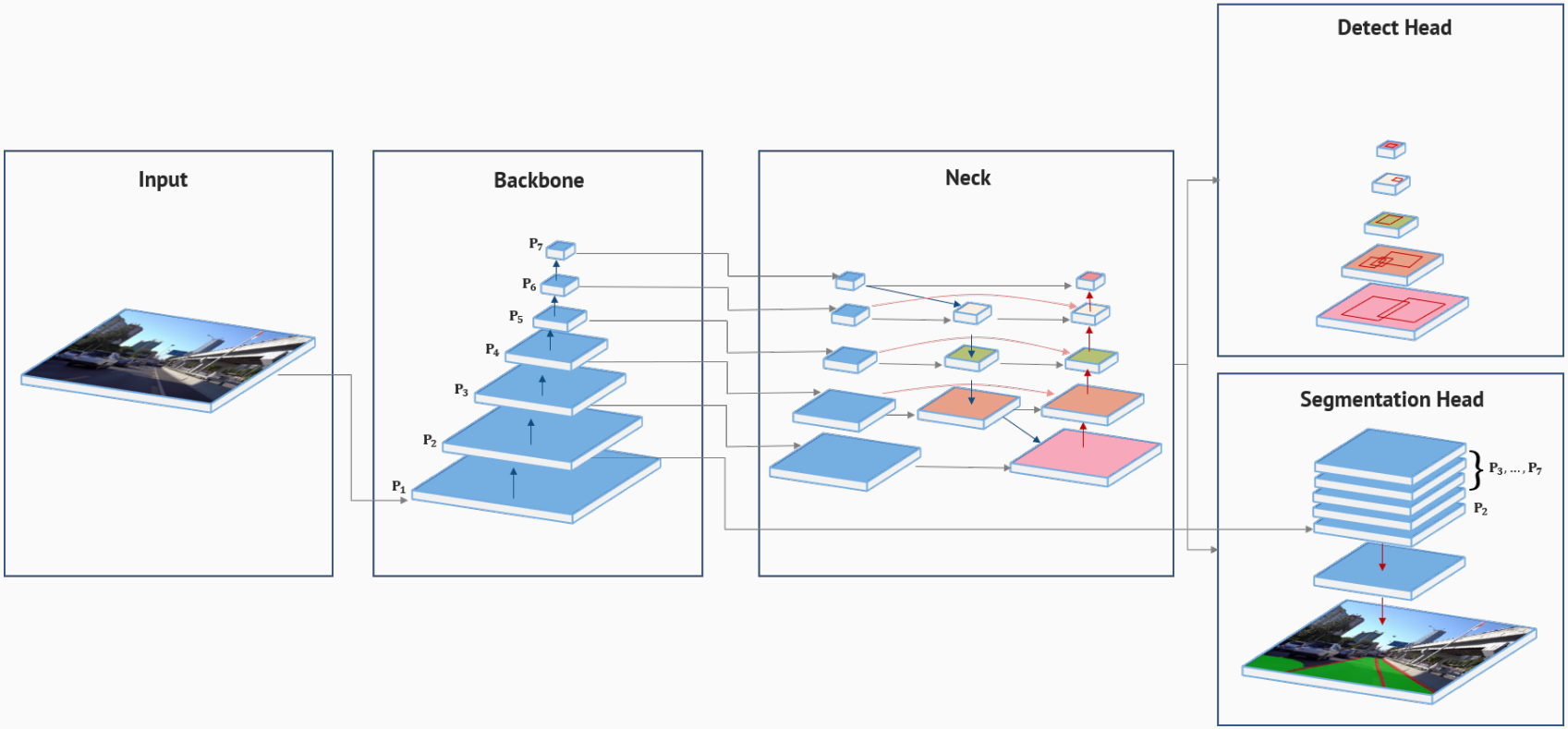
$$P_6^{td} = Conv\left(\frac{w_1 \cdot P_6^{in} + w_2 \cdot Resize(P_7^{in})}{w_1 + w_2 + \epsilon}\right)$$

$$P_6^{out} = Conv\left(\frac{w_1' \cdot P_6^{in} + w_2' \cdot P_6^{td} + w_3' \cdot Resize(P_5^{out})}{w_1' + w_2' + w_3' + \epsilon}\right)$$

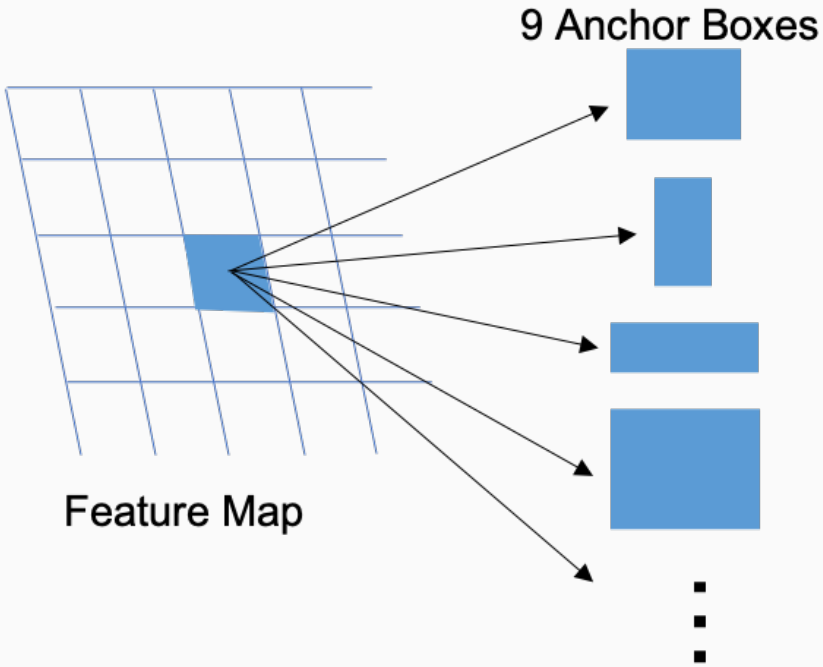
When fusing features with different resolutions, a common way is to first resize them to the same resolution and then simply sum them up treating all input features equally without distinction, as named

Weighted Feature Fusion

HybridNets Architecture Overview



Detection Head



Anchor Boxes is calculated by Anchor Scales and Anchor Ratios based on K-Means algorithm.

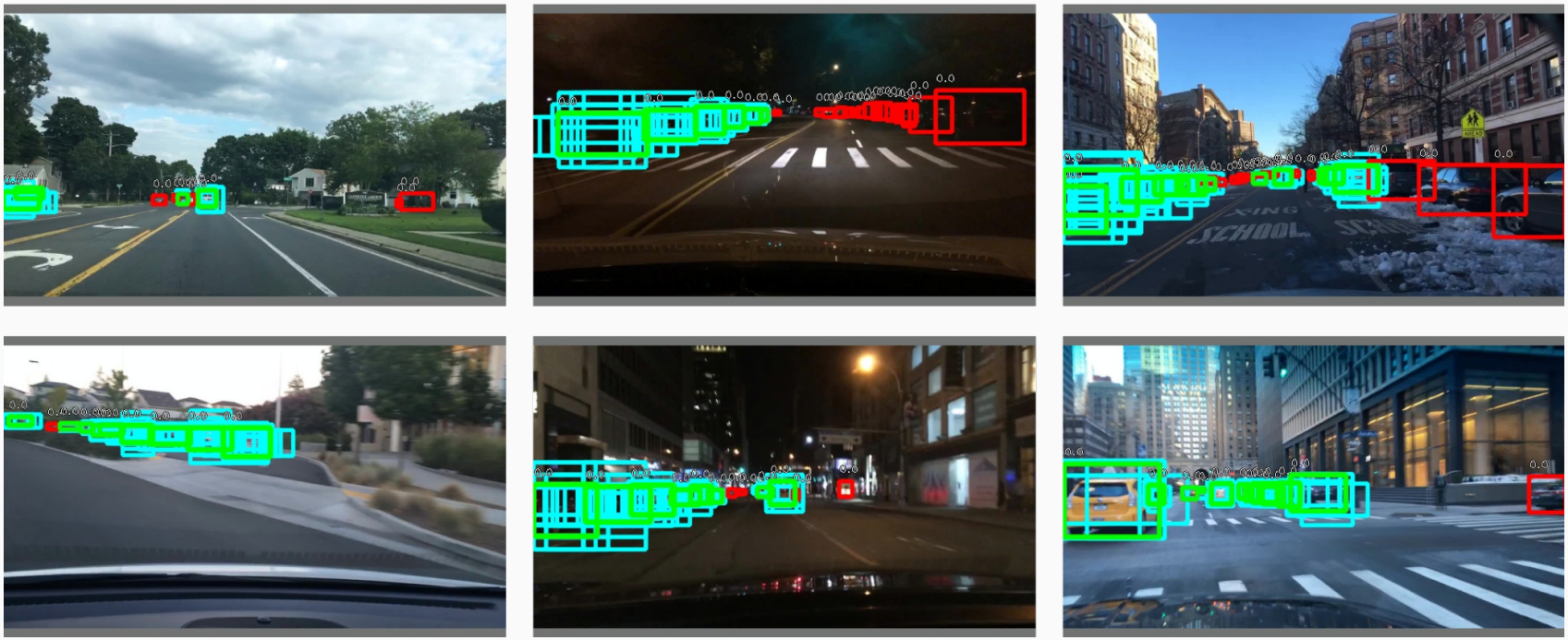
Anchor Boxes Calculation

Anchor Boxes each level:

2**level*base_scale
*scale*ratio

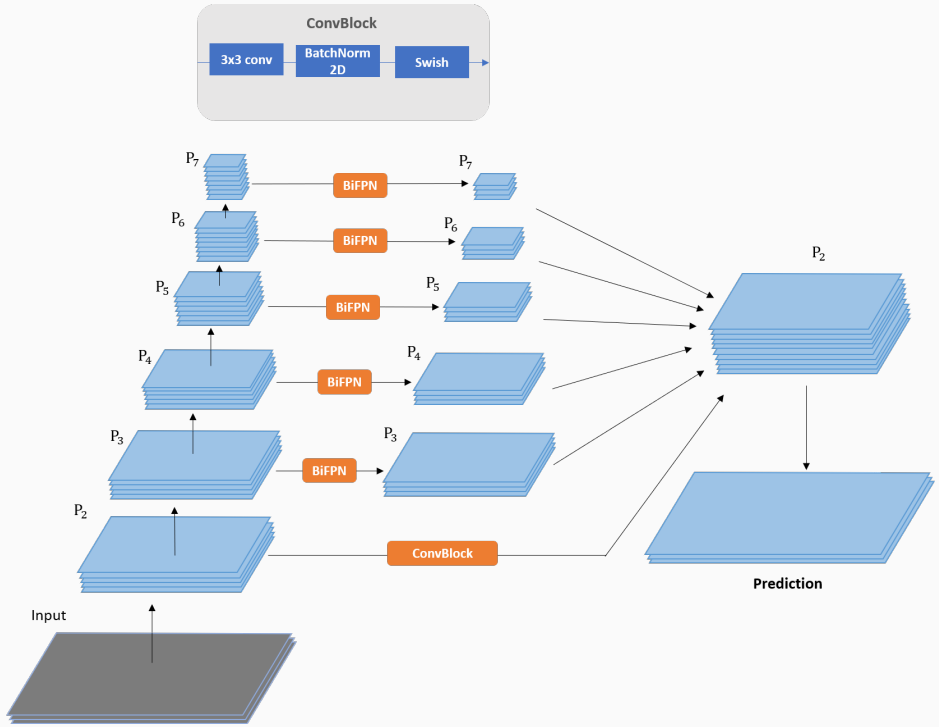
Base scale = 1.25 for each feature map level
Anchor Scales = [2**0, 2**0.70, 2**1.32]
Anchor Ratios = [(0.62, 1.58), (1.0, 1.0), (1.58, 0.62)]

Anchor Boxes Visualization



- This method work wells with small and huge objects.
- Become a good base to help network convergence fast.

Segmentation Head



- Why our work up scale on P2 Level and add P2 feature map from backbone into output feature map?

Object Detection Loss Function

$$\mathcal{L}_{all} = \alpha \mathcal{L}_{det} + \beta \mathcal{L}_{seg}$$

$$\mathcal{L}_{det} = \alpha_1 \mathcal{L}_{class} + \alpha_2 \mathcal{L}_{obj} + \alpha_3 \mathcal{L}_{box}$$

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t).$$

$$\mathrm{smooth}_{L1}(\mathrm{x}) = \begin{cases} \delta_1 x^2 & if \quad x < \delta_2 \\ x - \delta_1 \end{cases}$$

$$\mathrm{x} = \delta b_p \cdot \left| b_x - \hat{b}_x \right| + \left| b_y - \hat{b}_y \right| + \left| b_w - \hat{b}_w \right| + \left| b_h - \hat{b}_h \right|$$

Segmentation Loss Function

$$\mathcal{L}_{all} = \alpha \mathcal{L}_{det} + \beta \mathcal{L}_{seg}$$

$$\mathcal{L}_{det} = \alpha_1 \mathcal{L}_{class} + \alpha_2 \mathcal{L}_{obj} + \alpha_3 \mathcal{L}_{box}$$

$$\mathcal{L}_{seg} = \mathcal{L}_{Tversky} + \lambda \mathcal{L}_{Focal}$$

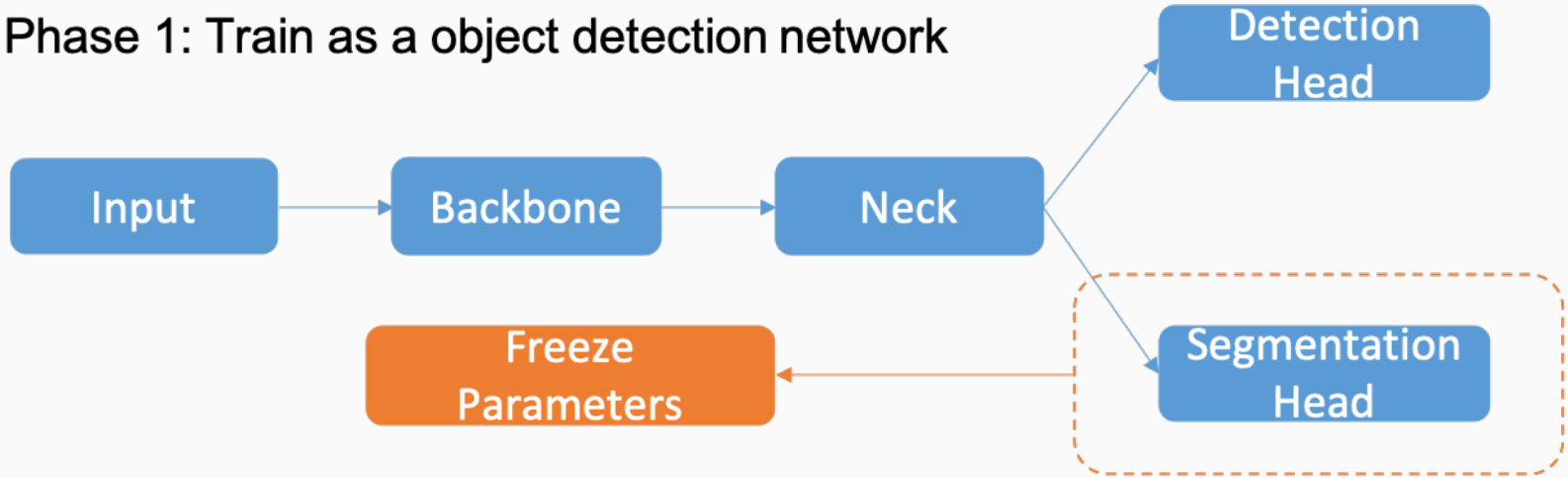
$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t).$$

$$\mathcal{L}_{Tversky} = C - \sum_{c=0}^{C-1} \frac{TP_p(c)}{TP_p(c) + \varphi FN_p(c) + (1-\varphi) FP_p(c)}$$

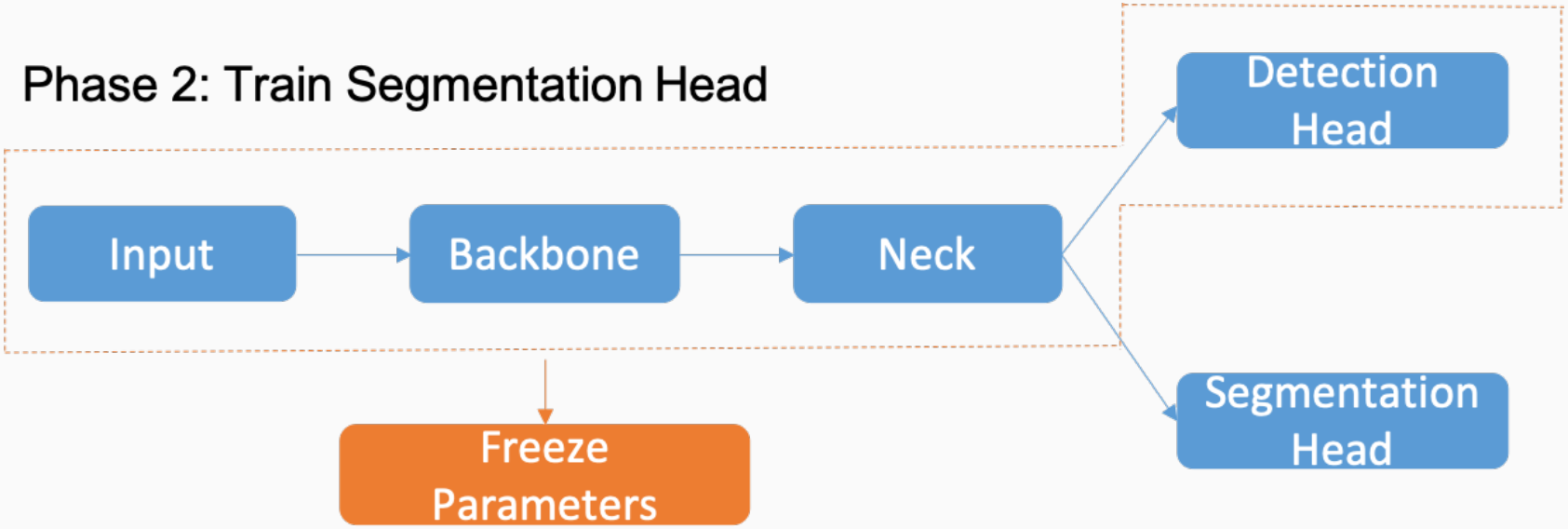
$$\mathrm{smooth}_{L1}(\mathrm{x}) = \begin{cases} \delta_1 x^2 & if \quad x < \delta_2 \\ x - \delta_1 \end{cases}$$

$$\mathrm{x} = \delta b_p \cdot \left| b_x - \hat{b}_x \right| + \left| b_y - \hat{b}_y \right| + \left| b_w - \hat{b}_w \right| + \left| b_h - \hat{b}_h \right|$$

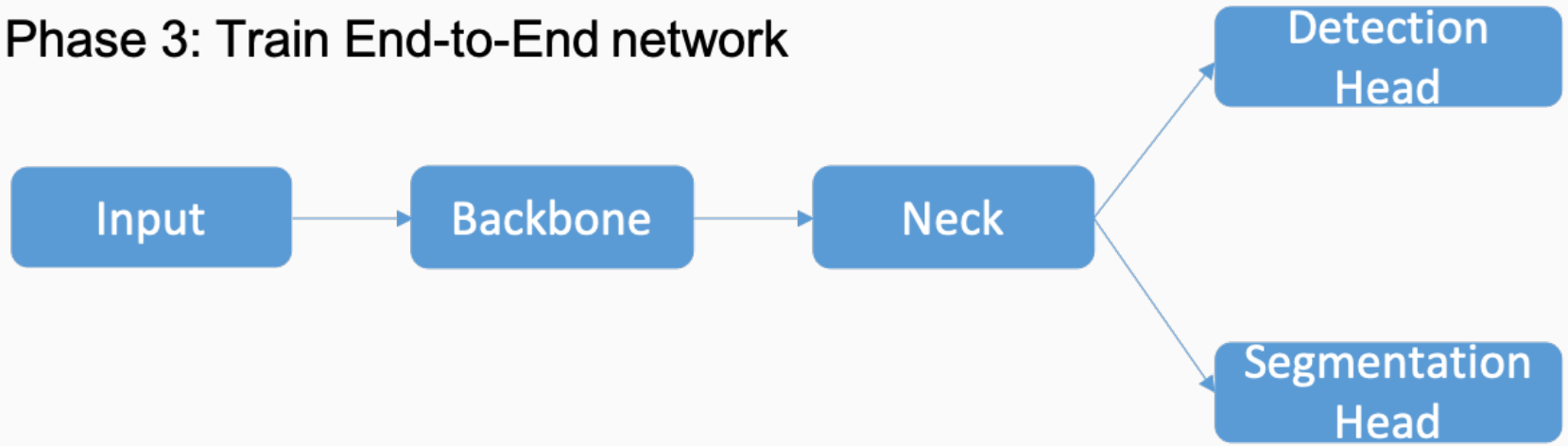
Phase 1


Phase 1: Train as a object detection network
Input
Backbone
Neck
Detection Head
Segmentation Head
Freeze Parameters

Phase 2


Phase 2: Train Segmentation Head
Input
Backbone
Neck
Detection Head
Segmentation Head
Freeze Parameters

Phase 3


Phase 3: Train End-to-End network
Input
Backbone
Neck
Detection Head
Segmentation Head

Training Strategy Overview

Input: Target end-to-end network $\mathcal{F}$ with parameter group

$\Theta = \{\theta_{enc}, \theta_{det}, \theta_{seg}\}$;

Training dataset $\mathcal{T}$;

Threshold for convergence $\gamma = \{\gamma_1, \gamma_2, \gamma_3\}$;

Total loss function $\mathcal{L}$;

Pivot strategy $\mathcal{P} = \{\{\theta_{enc}, \theta_{det}\}, \{\theta_{seg}\}, \{\theta_{enc}, \theta_{det}, \theta_{seg}\}\}$

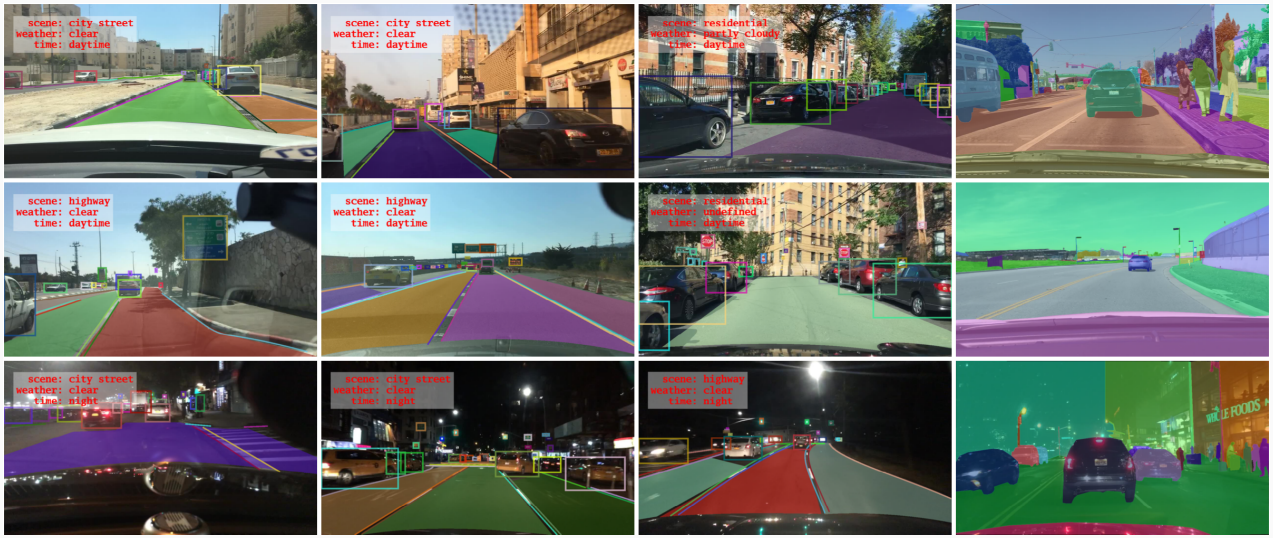
Output: Proposed network: $\mathcal{F}(\mathcal{X}, \Theta)$

procedure Train $(\mathcal{F}, \mathcal{T})$
  for i = 0 to length$(\mathcal{P})$ - 1
    $\Theta \leftarrow \Theta \cap \mathcal{P}[i]$ // Freeze parameters
    repeat
      Sample a mini-batch $(\mathbf{x}_m, \mathbf{y}_m)$ from training dataset $\mathcal{T}$
      $\ell \leftarrow \mathcal{L}_{all}(\mathcal{F}(\mathbf{x}_m; \Theta), \mathbf{y}_m)$
      $\Theta \leftarrow \arg\min_\theta \ell$
    until $\ell < \gamma[i]$
    if i < length$(\mathcal{P})$ - 1 **then**
      $\Theta \leftarrow \Theta \cup \mathcal{P}[i+1]$
    endif
  end for
end procedure
Train $(\mathcal{F}, \mathcal{T})$
return Proposed network $\mathcal{F}(\mathcal{X}, \Theta)$

Experiments

Dataset

- Dataset: BDD100K.



- Merge four classes: Car, truck, bus, train into a single class (vehicle).
- Merge two drivable area {direct, alternative} into drivable, re-labelled two lane line annotations into a central one line.
- Basic augmentation techniques: rotating, flipping, HSV shifiting, Mosaic, Mixup.

HybridNets Training Details

- Data augmentation:

Phase 1

Phase 2

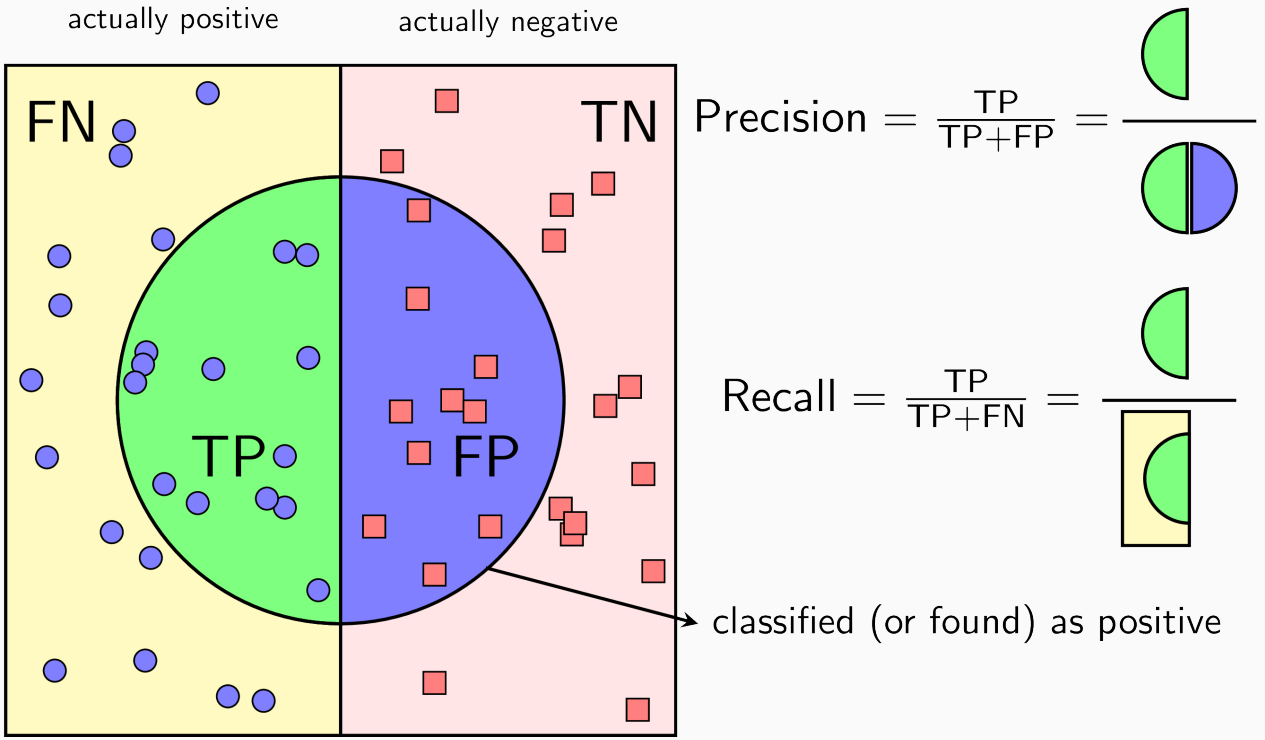
Phase 3

Phase 1: Mosaic + Mixup + Flip + Affine
Phase 2: Flip
Phase 3: Flip + Random(Blur, Gray, Brightness)

- Training stats:
  - Training time: 750 hours (1 GPU 3090 RTX)
  - GPU memory usage: 24 GB
  - Batch size: 16

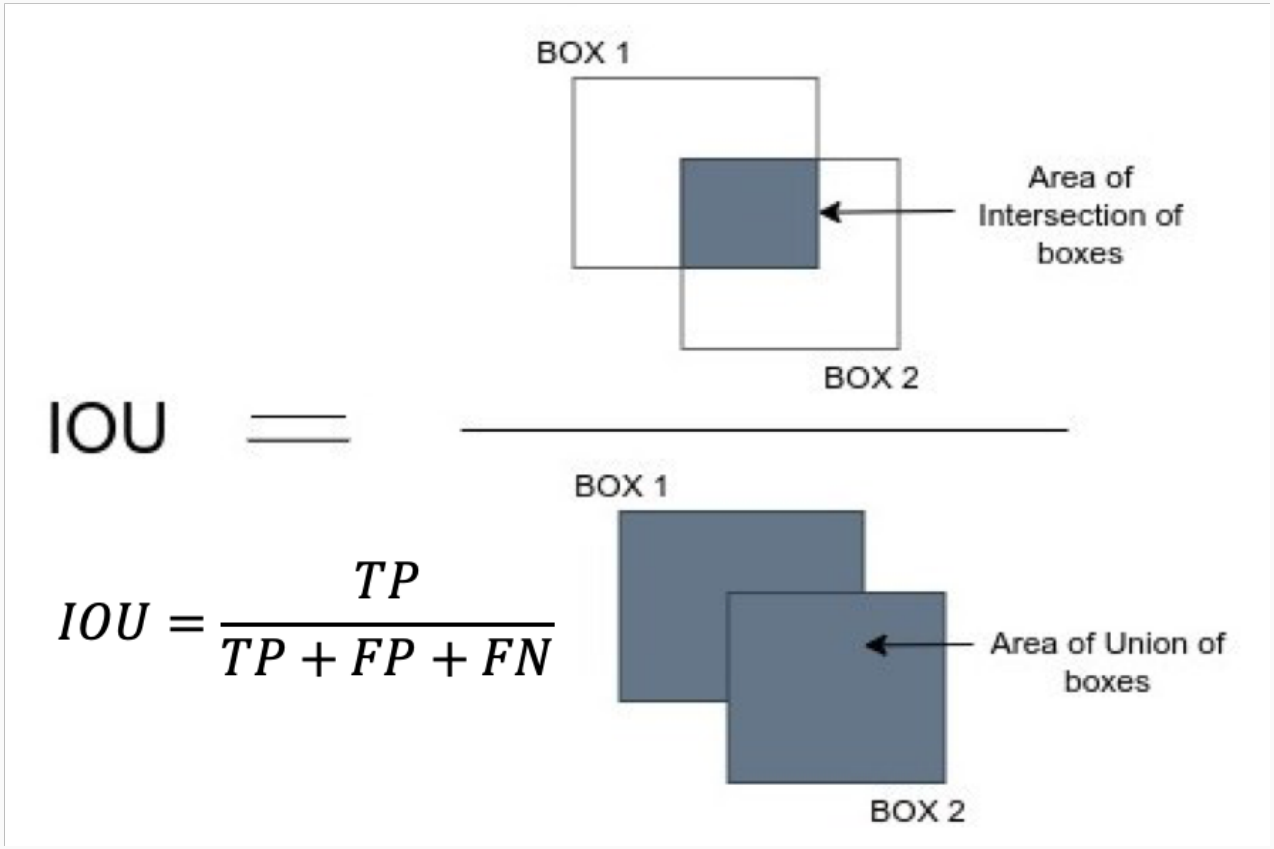
Evaluation metrics

- Precision and Recall



Evaluation metrics

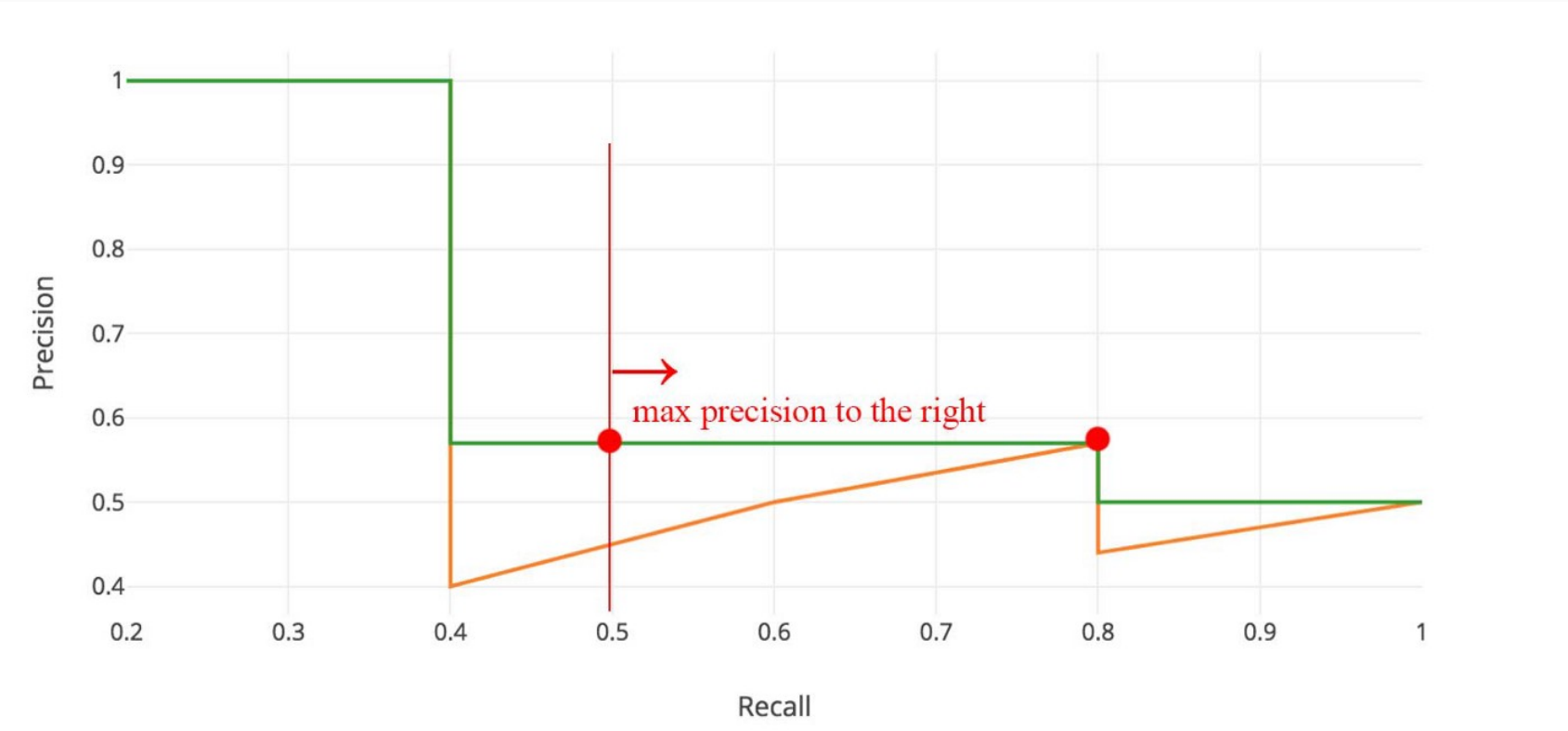
- IoU metric for Drivable Area Segmentation and Lane Line Segmentation.



$$IOU = \frac{TP}{TP + FP + FN}$$

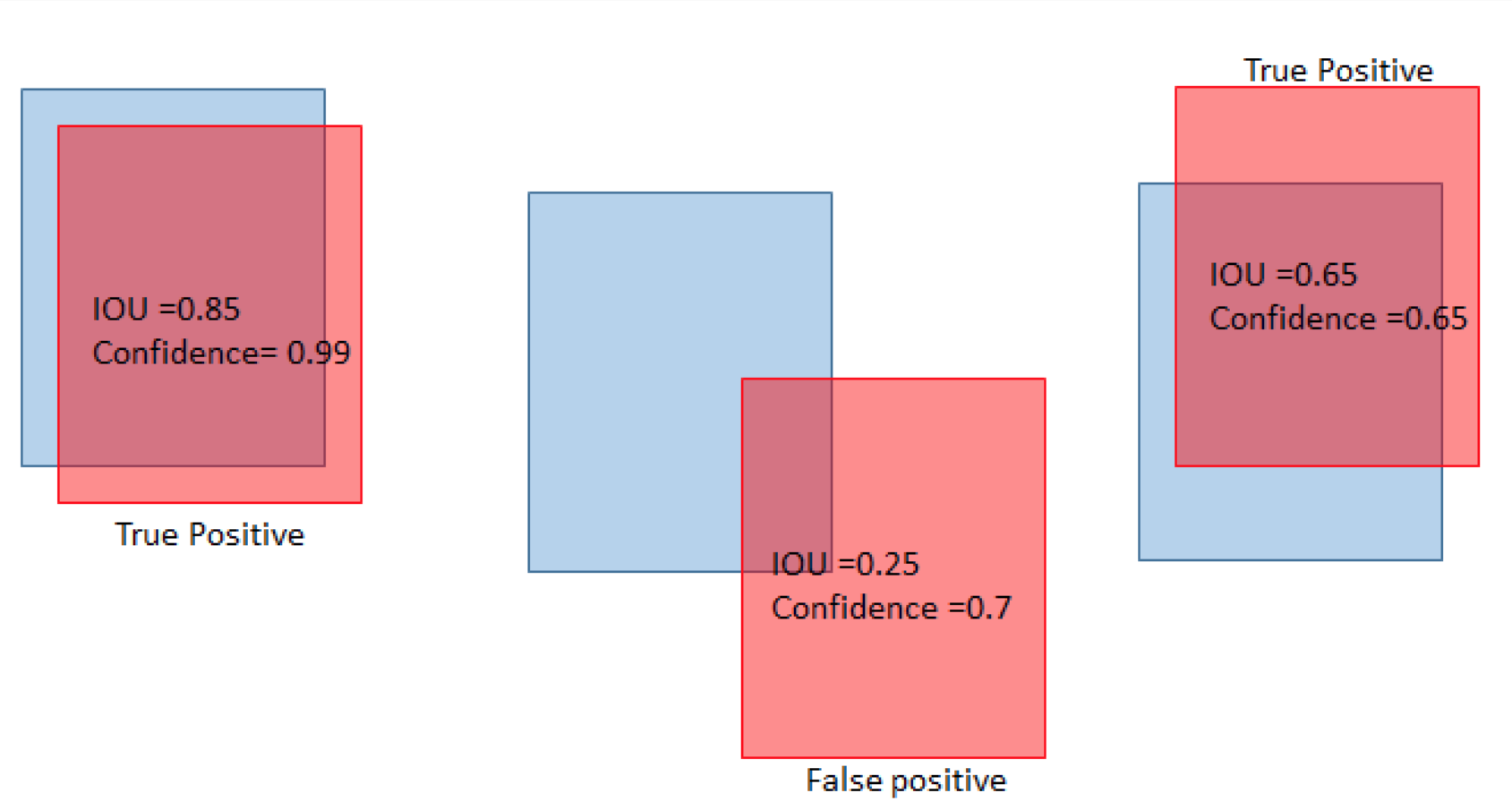
Evaluation metrics

- $AP = \sum_{k=0}^{k=n-1} [\text{Recalls(k)} - \text{Recalls(k+1)}] * \text{Precisions(k)}$.
- Recalls(n) = 0, Precisions(n) = 1 where n = Number of thresholds.

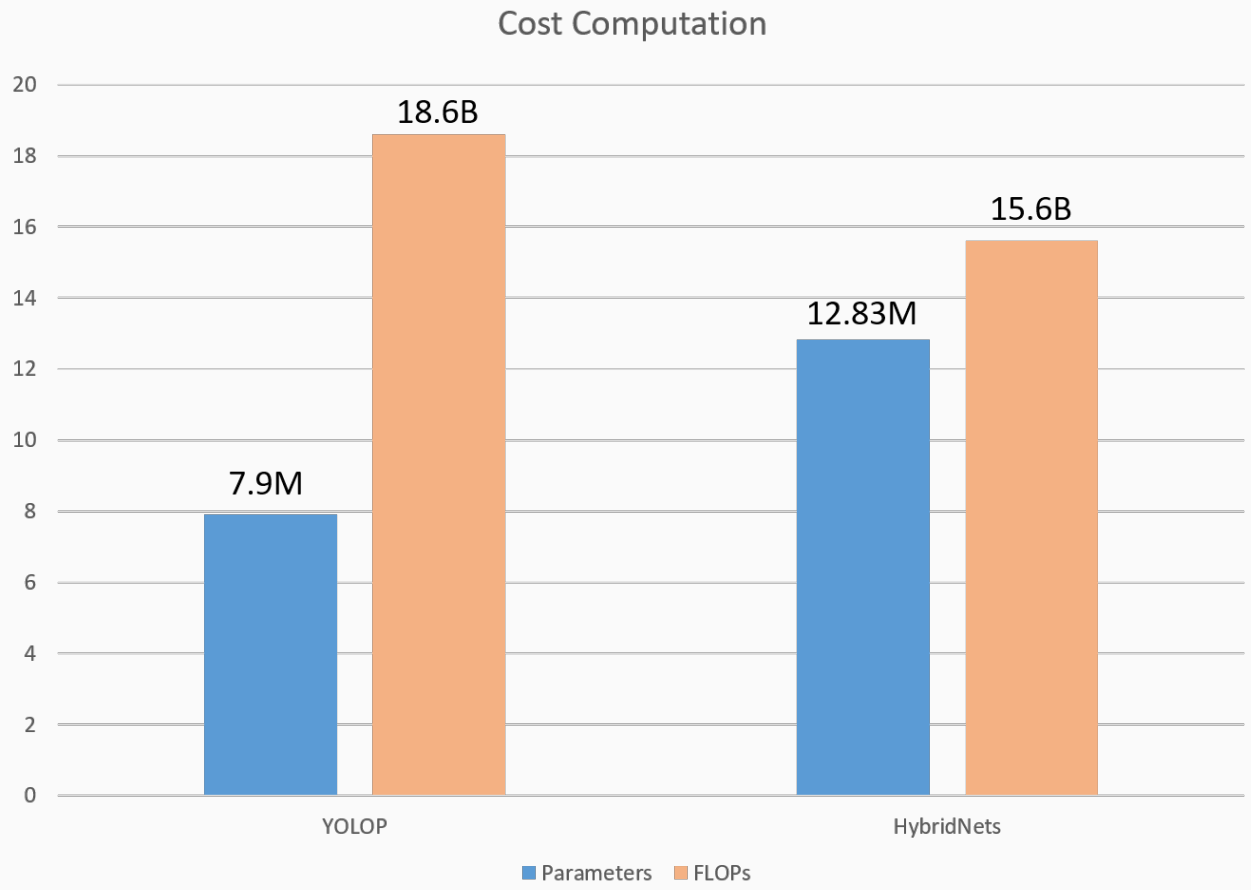


Evaluation metrics

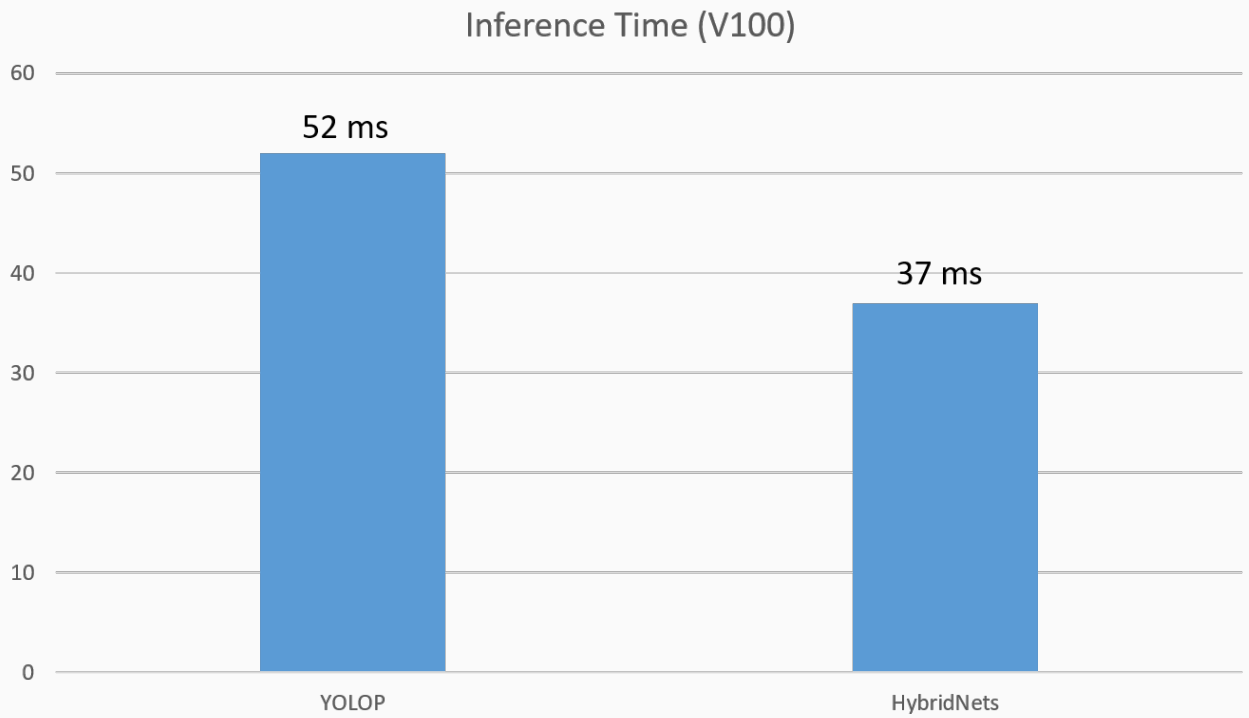
- mAP50 is used for Traffic Object Detection task.



Cost computation

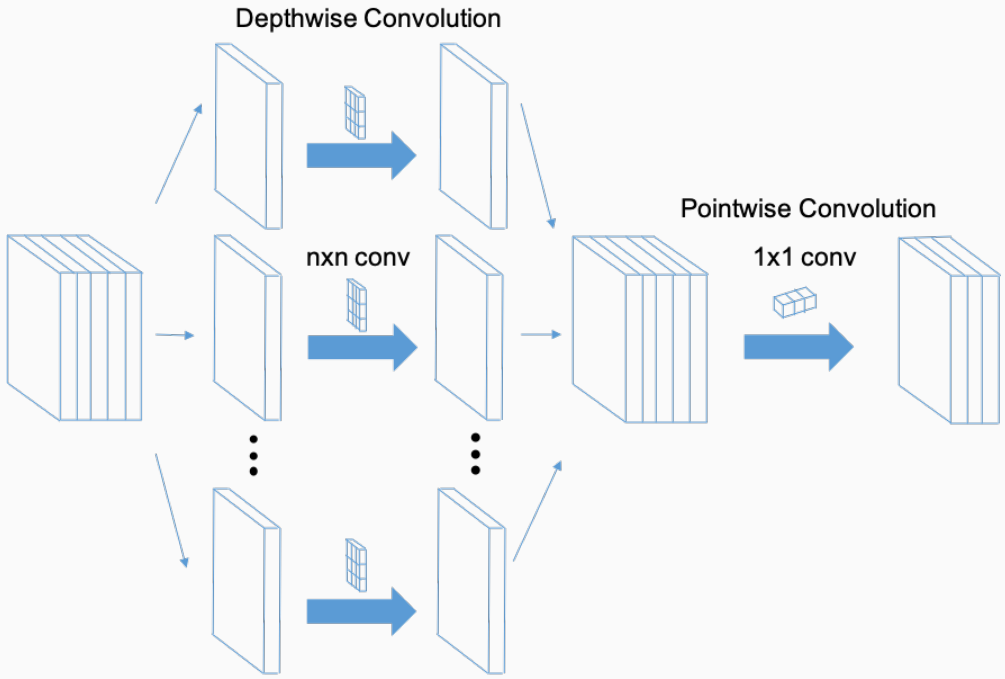


Inference Time



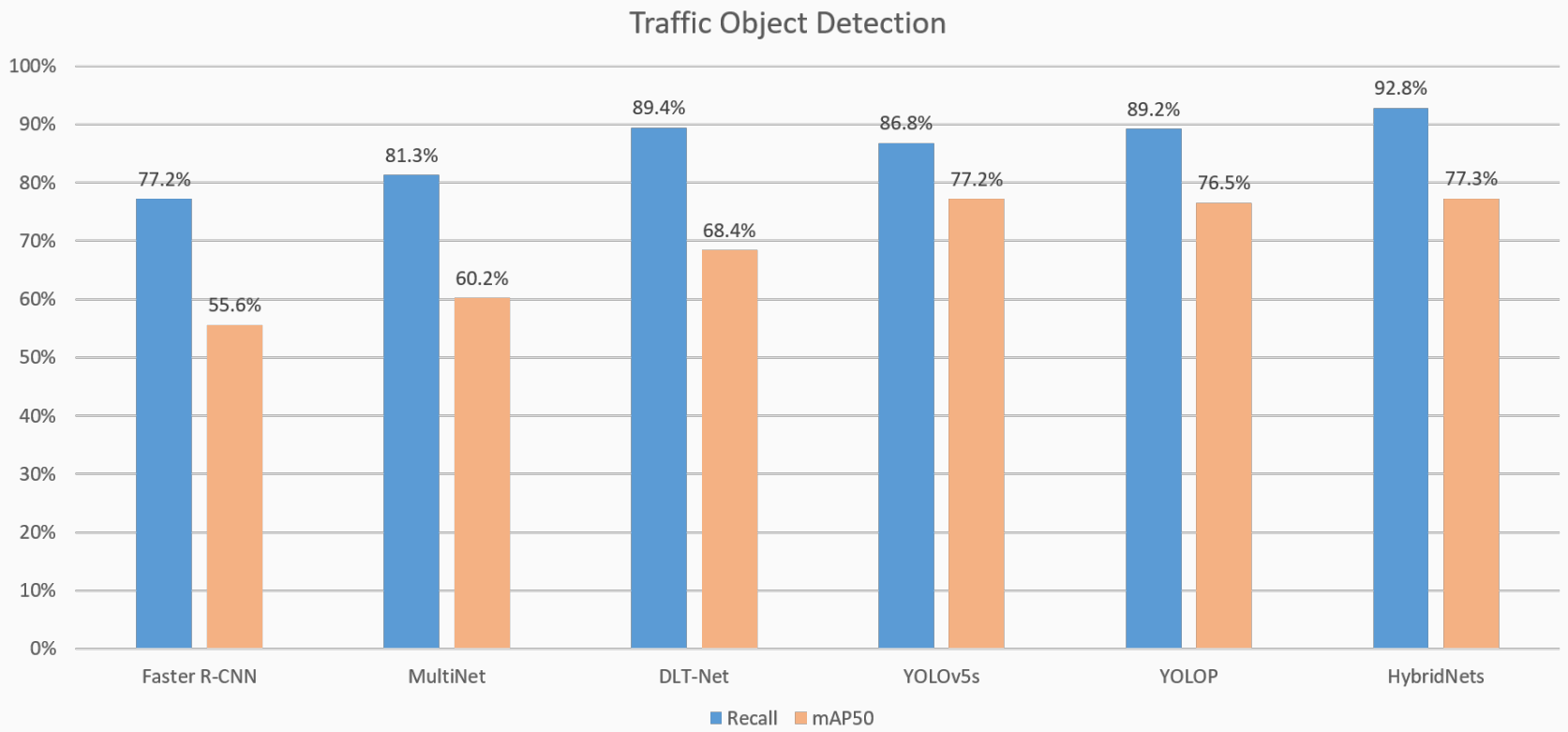
Why HybridNets has less cost computation than YOLOP?

Depthwise Separable Convolutions



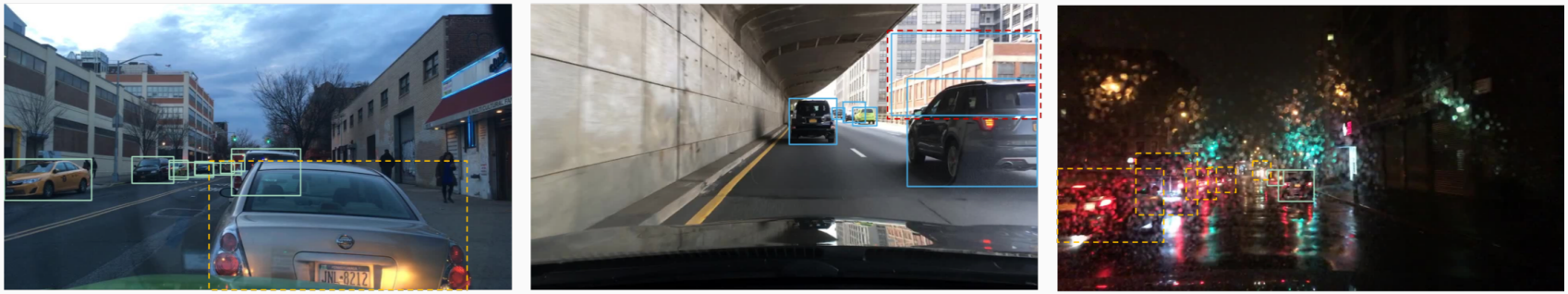
- In depth-wise convolution, we use each filter channel only at one input channel.
- Point-wise Convolution is a type of convolution that uses a 1x1 kernel: a kernel that iterates through every single point.

Traffic Object Detection Task

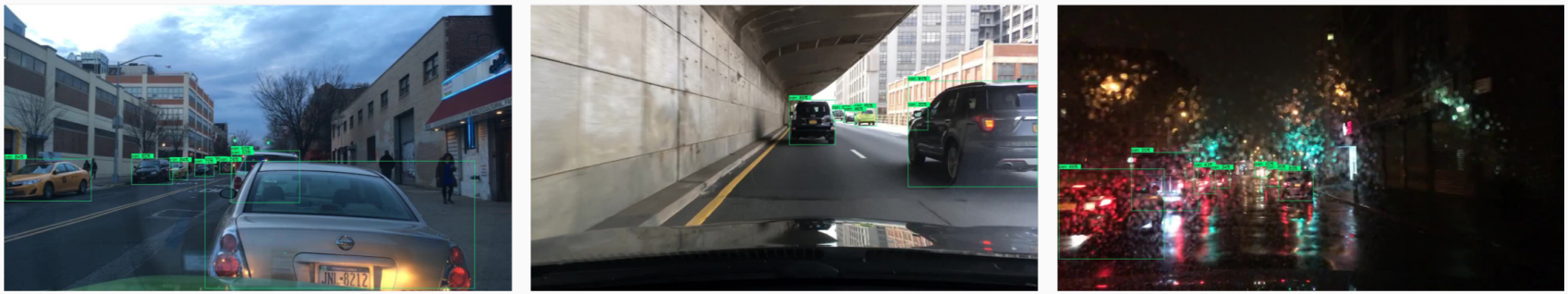


Traffic Object Detection Visualization

False Positive
False Negative

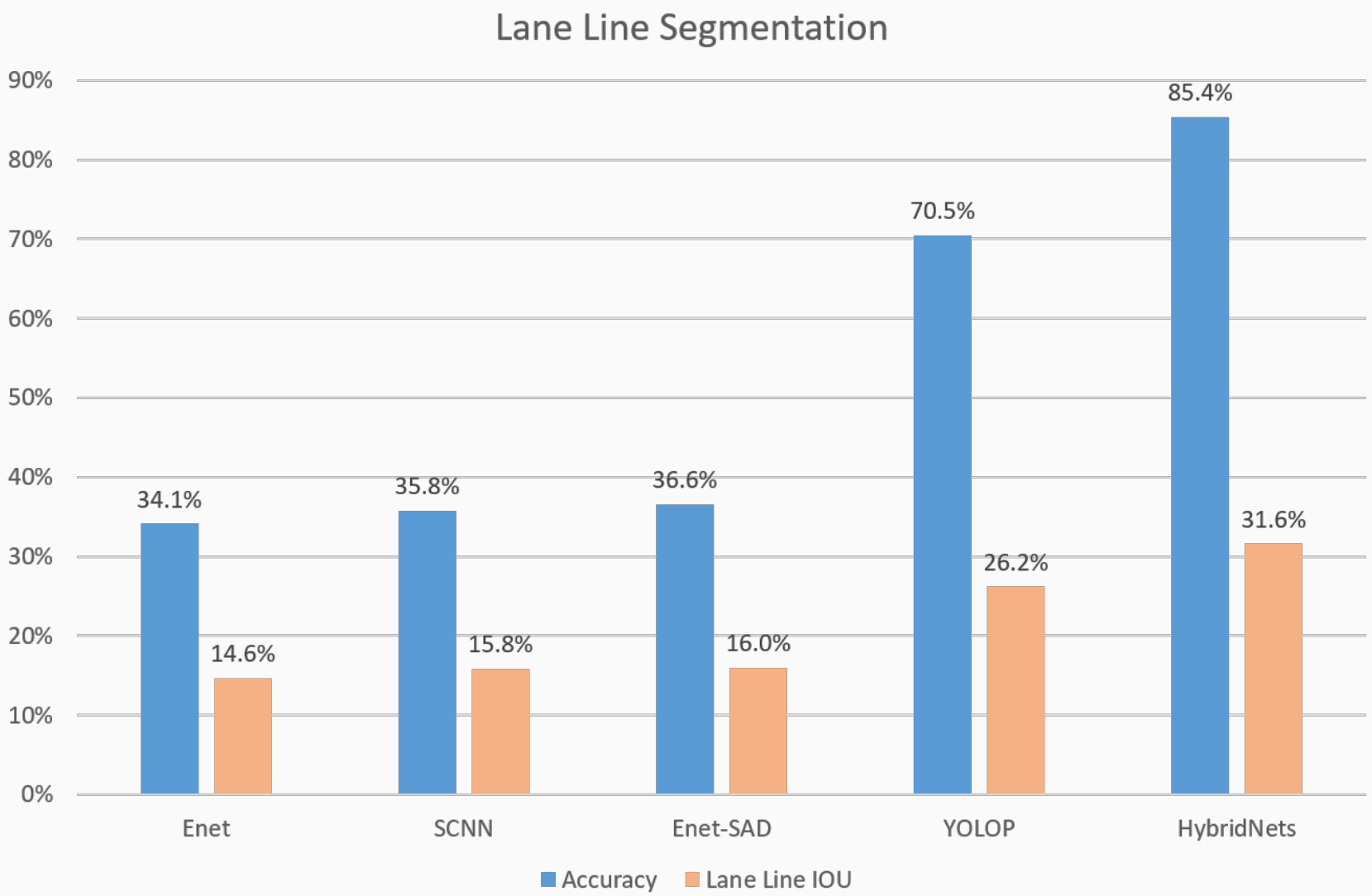


a) YOLOP



b) HybridNets

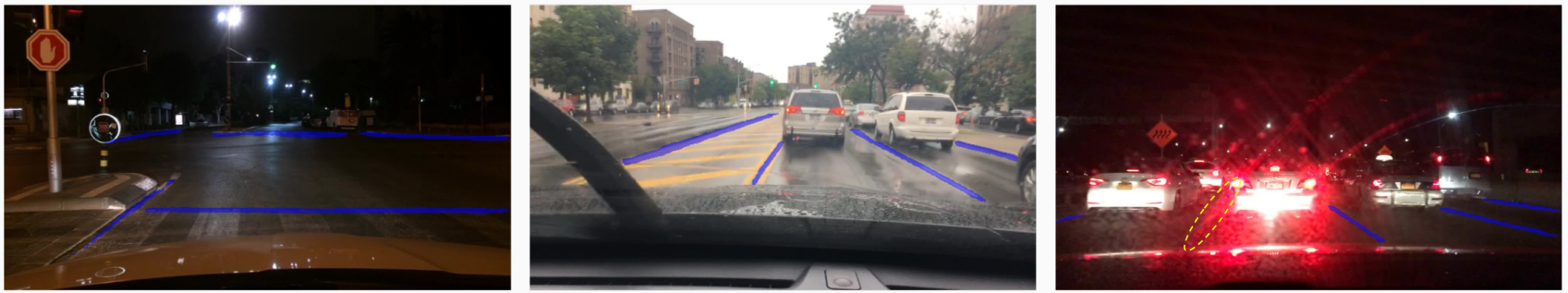
Lane Segmentation Task



Lane Segmentation Visualization

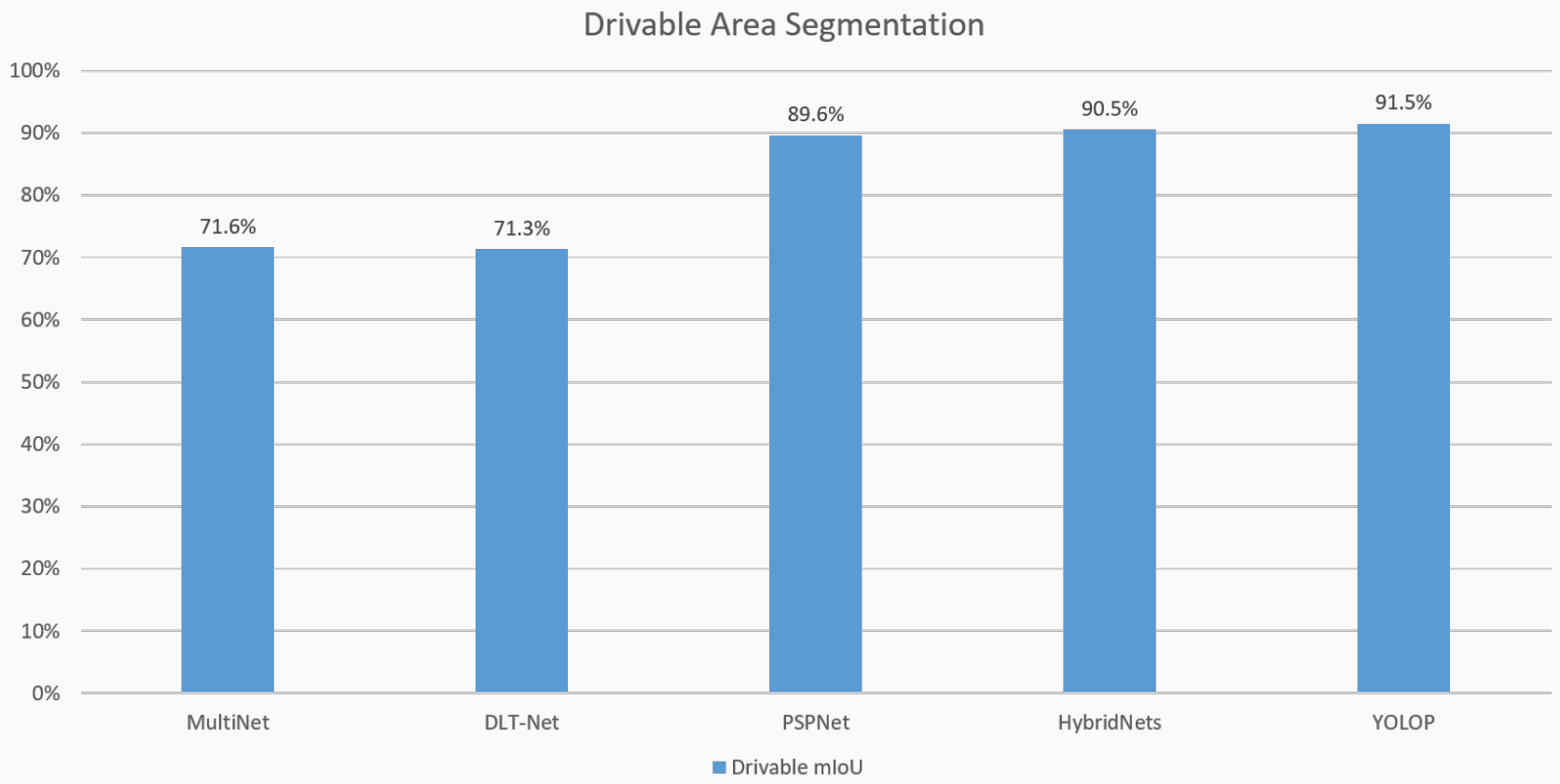
False Positive
False Negative

a) YOLOP



b) HybridNets

Drivable Segmentation Task



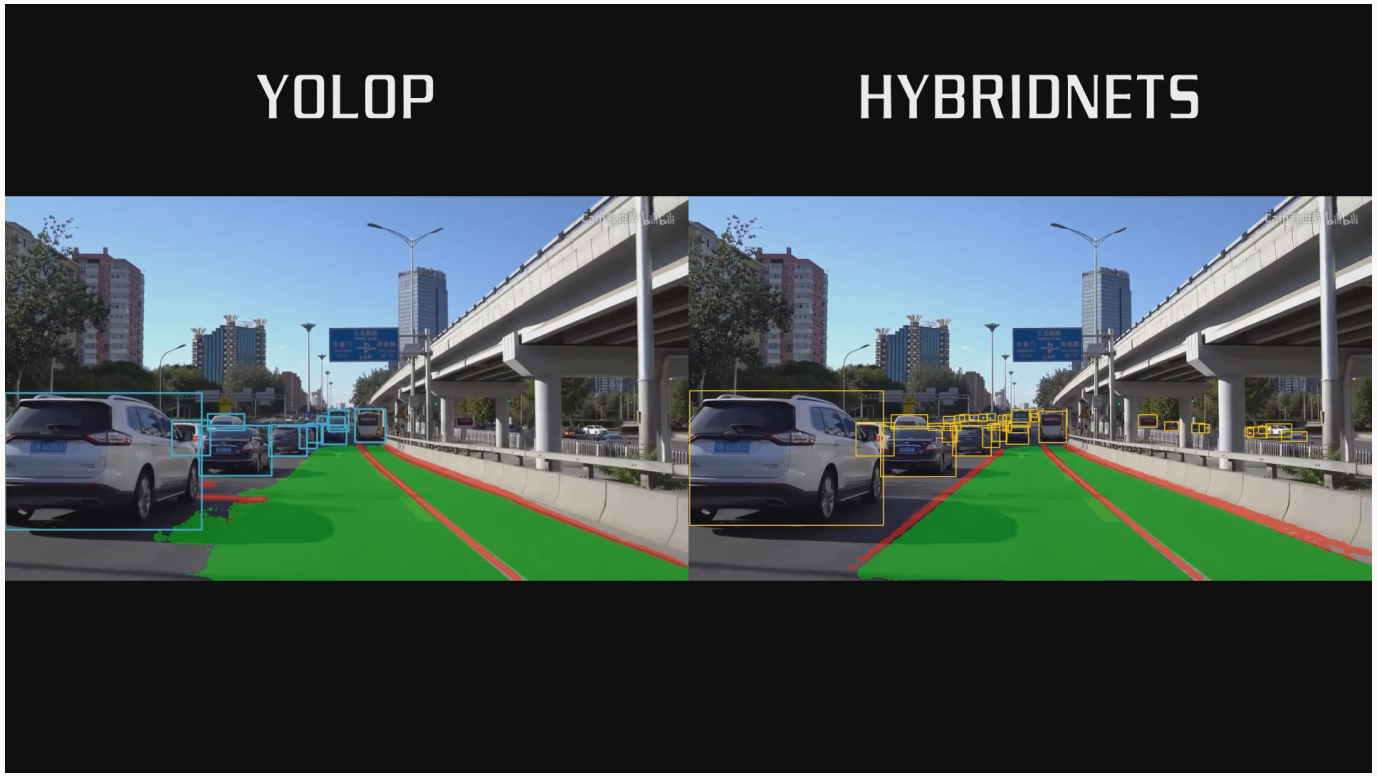
Drivable Segmentation Visualization

False Positive
False Negative

a) YOLOP

b) HybridNets

Comparison Between HybridNets and Prior SOTA YOLOP



Discussion

Our Contribution

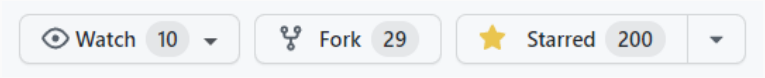
Research

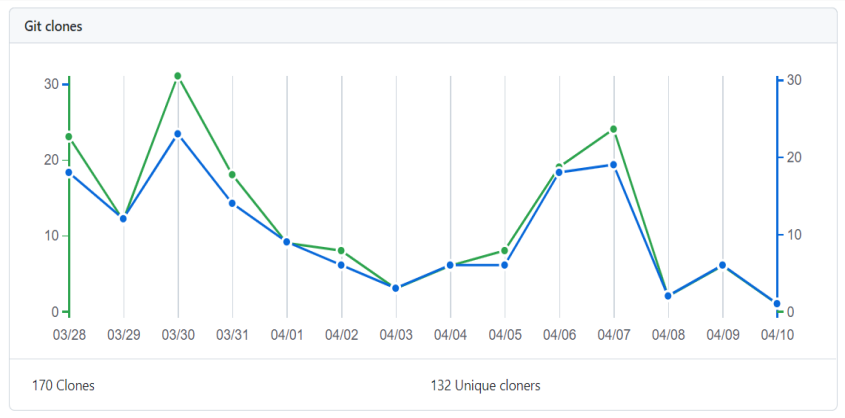
- Show once again that multi-tasking is an underexplored field with untapped potential in autonomous driving.
- Currently, the preprint on arXiv have yet to receive a citation.

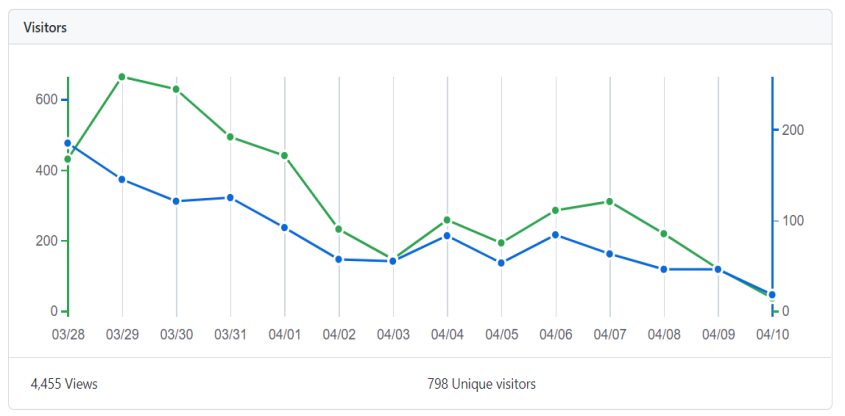
Production

- Create a scalable network on a wide range of embedded systems, from industrial-grade edge computing to off-the-shelf mobile phones.
- Porting to TensorRT, allowing streamlined user experience.

Community Interaction







Github statistics as of April 10th

Community Contribution

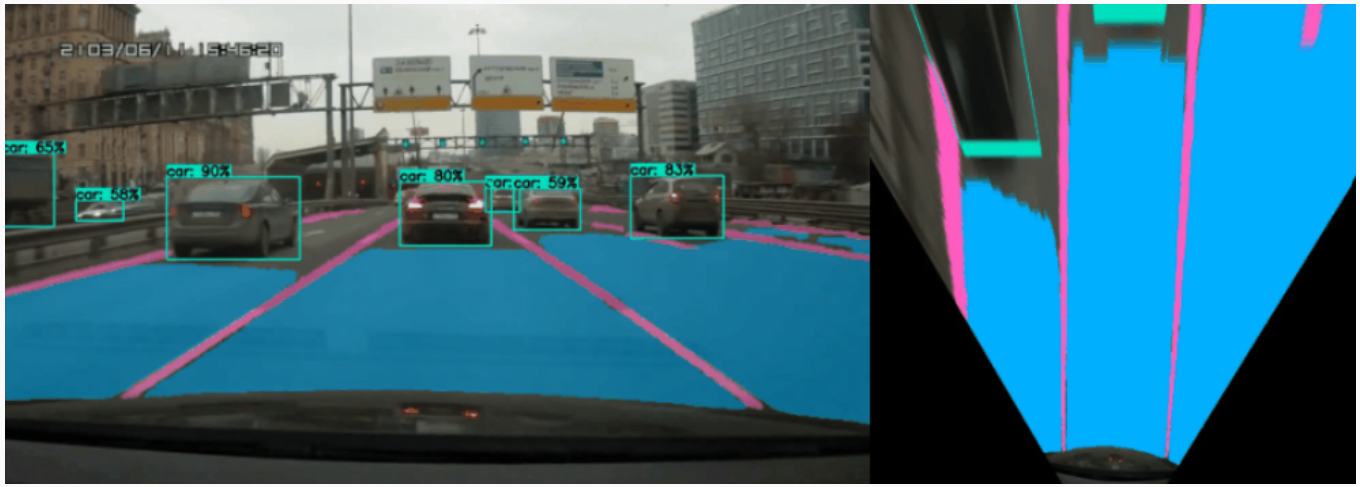
- Github.com/PINTO0309
- Provided a script to convert model to TFLite, ONNX, OpenVINO, CoreML, TFJS.
- Stress-tested on a foggy video clip with real-time performance (ONNX 384x512, Input 720x1280, FP16).

Community Contribution

- Github.com/iwatake2222
- Created a demo environment entirely in C++, using ported model from PINTO0309.
- Stress-tested on Pixel 4a, only achieved 6 FPS maximum (TFLite 384x640, Input 720x1280, FP16).

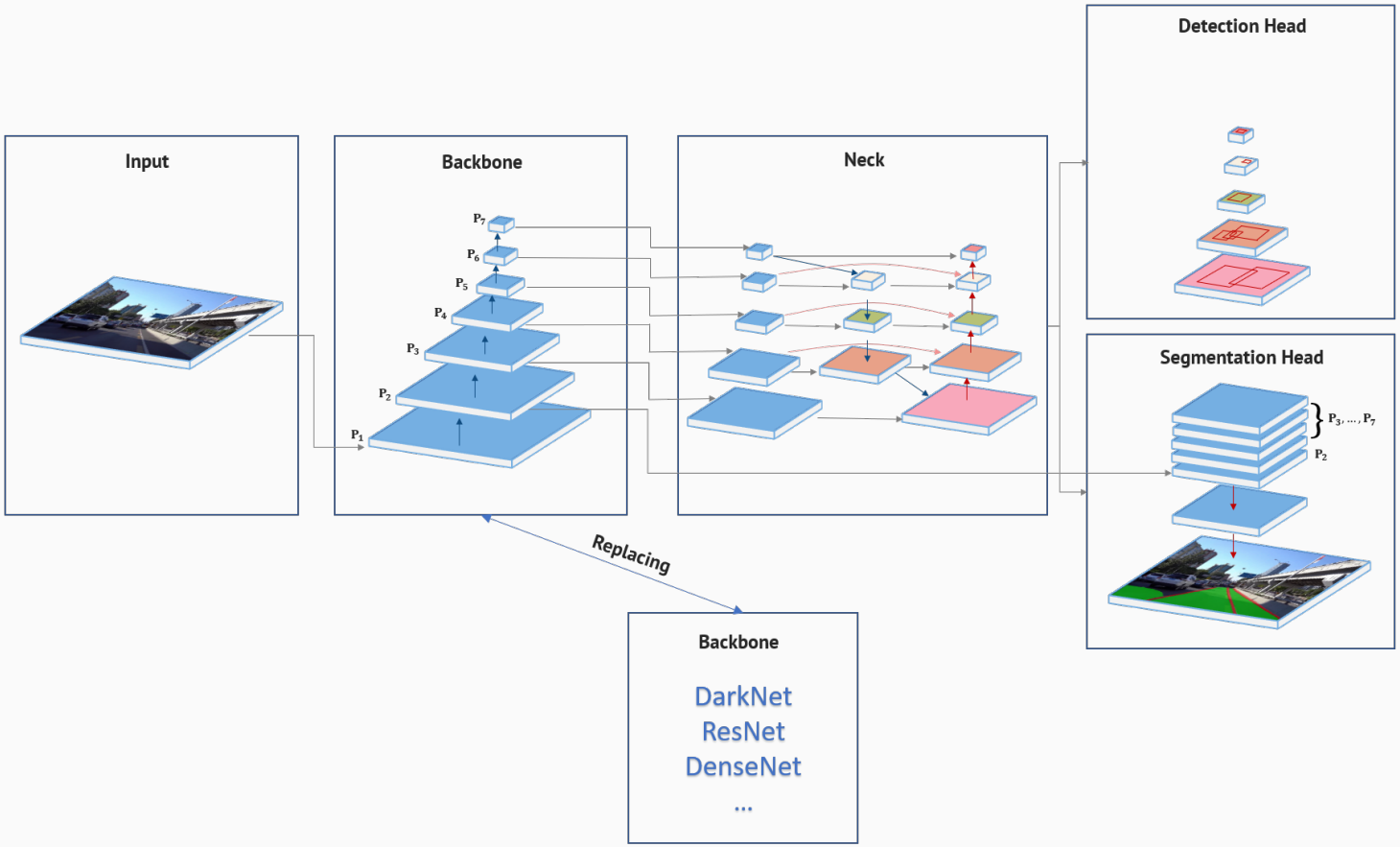
Community Contribution

- Github.com/ibaiGorordo
- Created a Python script to infer Youtube videos with customizable bird eye view, using ported model from PINTO0309.



Advantages

- Vehicle's perception in a single pass.
- Scalable thanks to EfficientNet backbone.
- Switchable backbone.



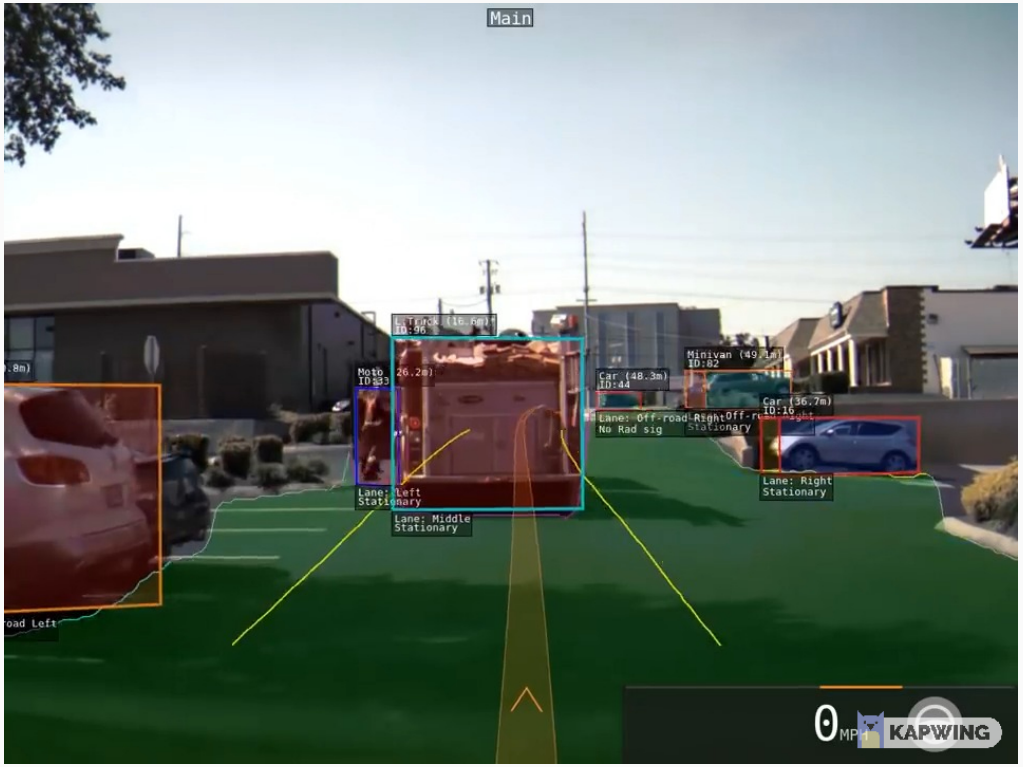
Limitations

- Fixed to 2 heads for 2 specific tasks of object detection and segmentation.
- Lack of ablation studies.

Conclusion

Future Work

- Perform various tasks related to perception.



Future Work

- Perform various tasks related to perception.
- Improve parameters and FLOPs of network for edge devices.
- Build a decoder network detecting 3D Object Detection with only one input.

The End

Questions?