Chapter 2 - Section 8

Image Segmentation

Dr. Li Hongyang

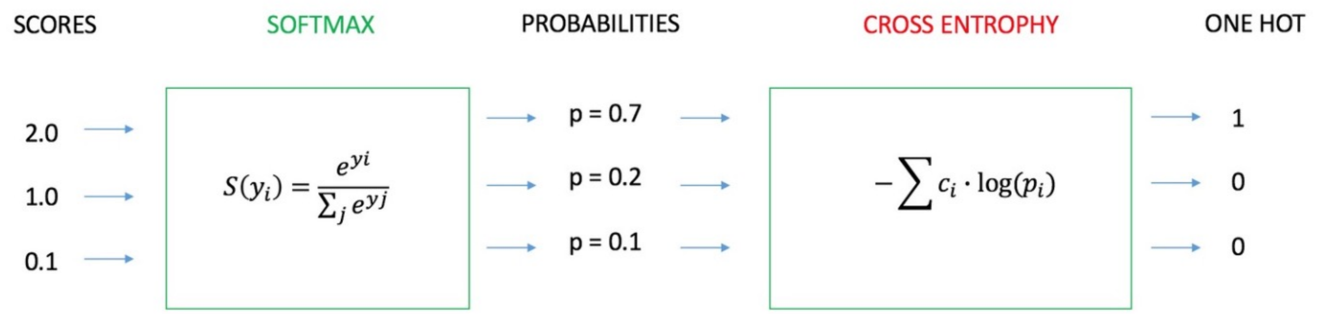
Thursday, April 14, 2022

Partial credit by : Wang Cheng

Outline

Recap: Image Classification

- Classification Loss
  - Cross entropy or MSE?



For MSE loss, we have

$$L_{mse,i} = \frac{1}{2}(y_i - p_i)^2$$

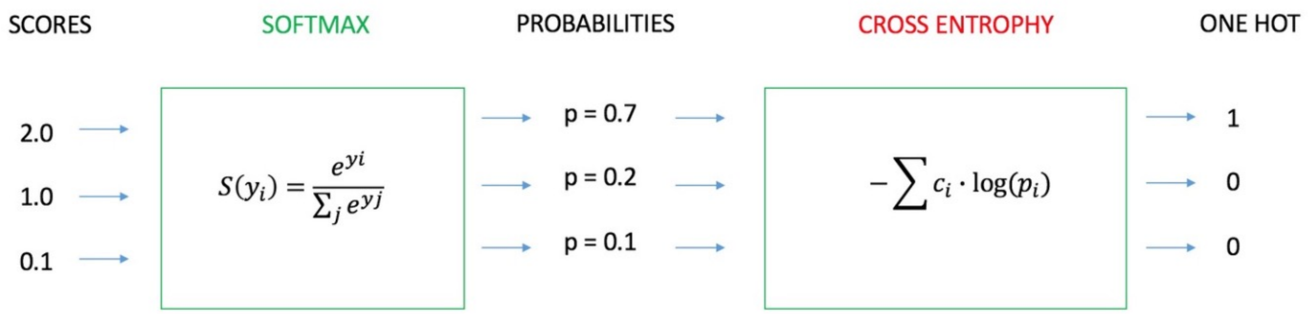
$p_i = \sigma(Wx + b)$ 函数$\sigma(\cdot)$是`Sigmoid`函数

The grad for MSE loss is: $\partial L_{mse,i} / \partial W = (y_i - p_i) p_i (1 - p_i) x$

- *Wx+b* could be very large/small in the first few iterations; since parameters are randomly initialized.
- Gradient vanishing!

Recap: Image Classification

- Classification Loss
    - Cross entropy or MSE?



For Cross entropy loss, we have

$$L_{ce,i} = -[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

Equivalent

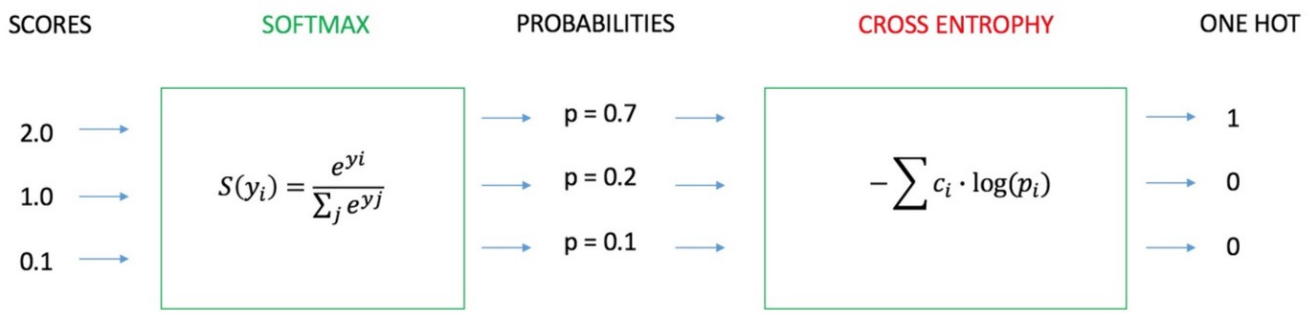
$$L_{ce,i} = \begin{cases} -\log(p_i), & y_i = 1 \\ -\log(1-p_i), & y_i = 0 \end{cases}$$

The grad for CE loss is: $\partial L_{ce,i} / \partial W = -(y_i - p_i)x$

- Gradient vanishing! ***resolved***

Recap: Image Classification

- Classification Loss
  - Cross entropy or MSE?



Multi-classification CE loss:

$$p_{i,k} = \exp(q_{i,k}) / \sum_j^M \exp(q_{i,j})$$

$$L = \sum_i^N L_{ce,i} = -\sum_i^N \sum_j^M y_{i,j} \log(p_{i,j})$$

<u>Further extension:</u>

From cross entropy loss to Focal loss and Circle loss
What’s the relationship?

Pseudo-Labelling

- Background
  - Consistency Regularization

$$\|\mathrm{p_{model}}(y \mid \mathrm{Augment}(x); \theta) - \mathrm{p_{model}}(y \mid \mathrm{Augment}(x); \theta)\|_2^2$$

模型在无标签数据***增广前***和***增广后***的预测应该一致

  - Entropy Minimization
    - Minimizes the entropy of $p_{model}(y|x;\theta)$

要求分类器对无标签样本输出熵较少的结果

  - Traditional Regularization

加入一些常量干扰，使得模型难以记住训练样本，来达到泛化要求

Pseudo-Labelling

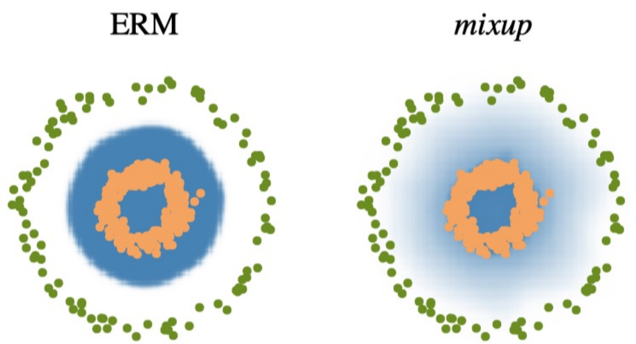
- ***Mixup***: Beyond Empirical Risk Minimization (经验风险最小化；训练误差越小越好)
  - A simple and data-agnostic data augmentation method

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Zhang H, Cisse M, Dauphin YN, Lopez-Paz D. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412. 2017 Oct 25.

Pseudo-Labelling

- mixup: Beyond Empirical Risk Minimization
  - Why use beta distribution?
  - Similar to label smooth?
  - Why mixup works? (Generalization gap between training data and real data)

https://www.zhihu.com/question/67472285

如何评价mixup: BEYOND EMPIRICAL RISK MINIMIZATION?

这篇paper是ICLR2018的投稿，直接对raw data和 label interpolation，在很多数据集上取得了SoTA。arxiv.org/pdf/1710.0941...

关注问题 | 写回答 | 邀请回答 | 好问题 4 | 添加评论 | 分享 | ··· | 收起

20 个回答 默认排序

张宏毅
MIT 机器学习 / 认知科学
456 人赞同了该回答

谢流远
深度学习（Deep Learning）话题下的优秀答主
77 人赞同了该回答

Mixup超好用的，轻松提高一个点，参见我们的paper：

arxiv.org/abs/1812.0118...

编辑于 2018-12-07

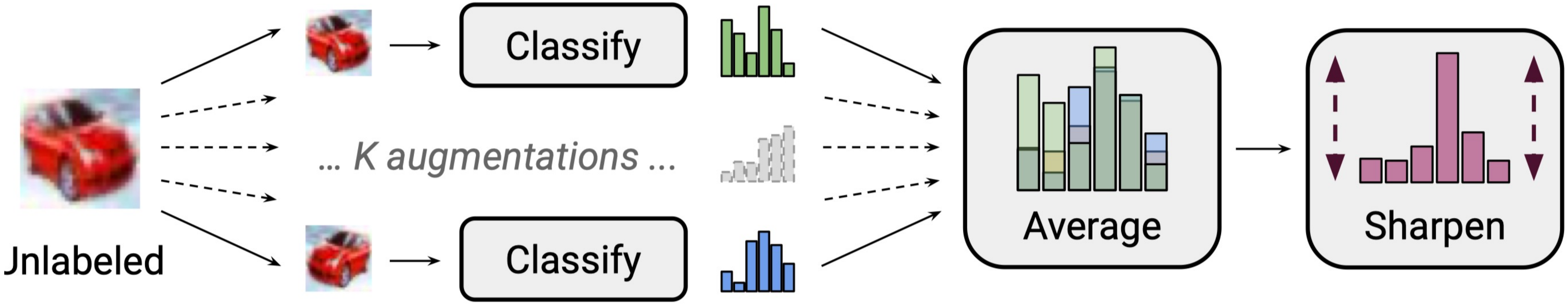
Pseudo-Labelling

- mixup: Beyond Empirical Risk Minimization

Further reading

- MixMatch
- ReMixMatch
- FixMatch

- MixMatch: A Holistic Approach to Semi-Supervised Learning
  - Stochastic data augmentation is applied to an unlabeled image K times
  - The average of these K predictions is "sharpened" by adjusting the distribution's temperature



Berthelot D, Carlini N, Goodfellow I, Papernot N, Oliver A, Raffel C. Mixmatch: A holistic approach to semi-supervised learning. arXiv preprint arXiv:1905.02249. 2019 May 6.

Pseudo-Labelling

- MixMatch: A Holistic Approach to Semi-Supervised Learning

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \ldots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \ldots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.

for $b = 1$ **to** B **do**

 $\hat{x}_b = \text{Augment}(x_b)$ *// Apply data augmentation to x_b*

 for $k = 1$ **to** K **do**

  $\hat{u}_{b,k} = \text{Augment}(u_b)$ *// Apply k^{th} round of data augmentation to u_b*

 end for

 $\bar{q}_b = \frac{1}{K} \sum_k \mathrm{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of u_b*

 $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. $\boxed{7}$)*

end for

$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \ldots, B)\big)$ *// Augmented labeled examples and their labels*

$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \ldots, B), k \in (1, \ldots, K)\big)$ *// Augmented unlabeled examples, guessed labels*

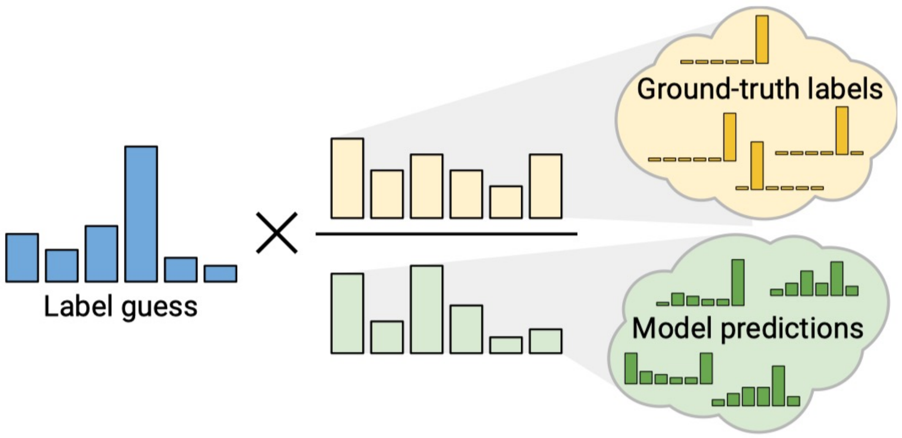
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*

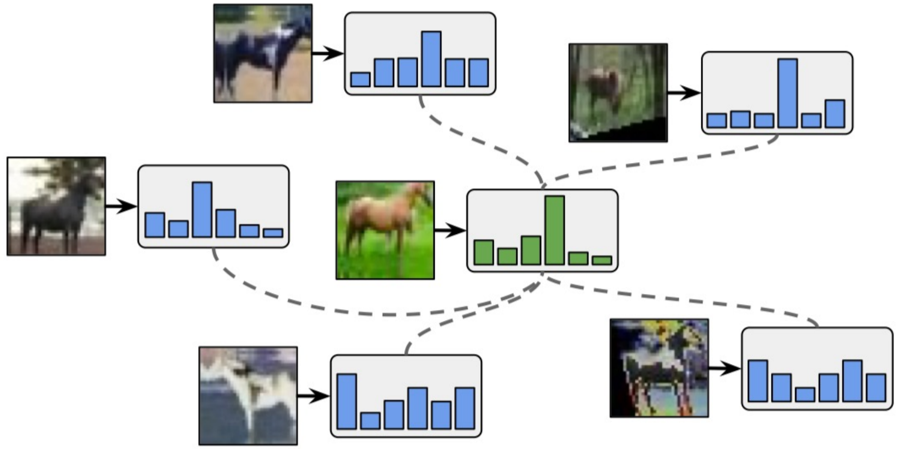
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \ldots, |\hat{\mathcal{X}}|)\big)$ *// Apply MixUp to labeled data and entries from $\mathcal{W}$*

$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \ldots, |\hat{\mathcal{U}}|)\big)$ *// Apply MixUp to unlabeled data and the rest of $\mathcal{W}$*

return $\mathcal{X}', \mathcal{U}'$

- ReMixMatch: Semi-Supervised Learning with Distribution Matching and Augmentation Anchoring
  - Improved version of MixMatch
  - Distribution Alignment (left) and Augmentation Anchor (right)

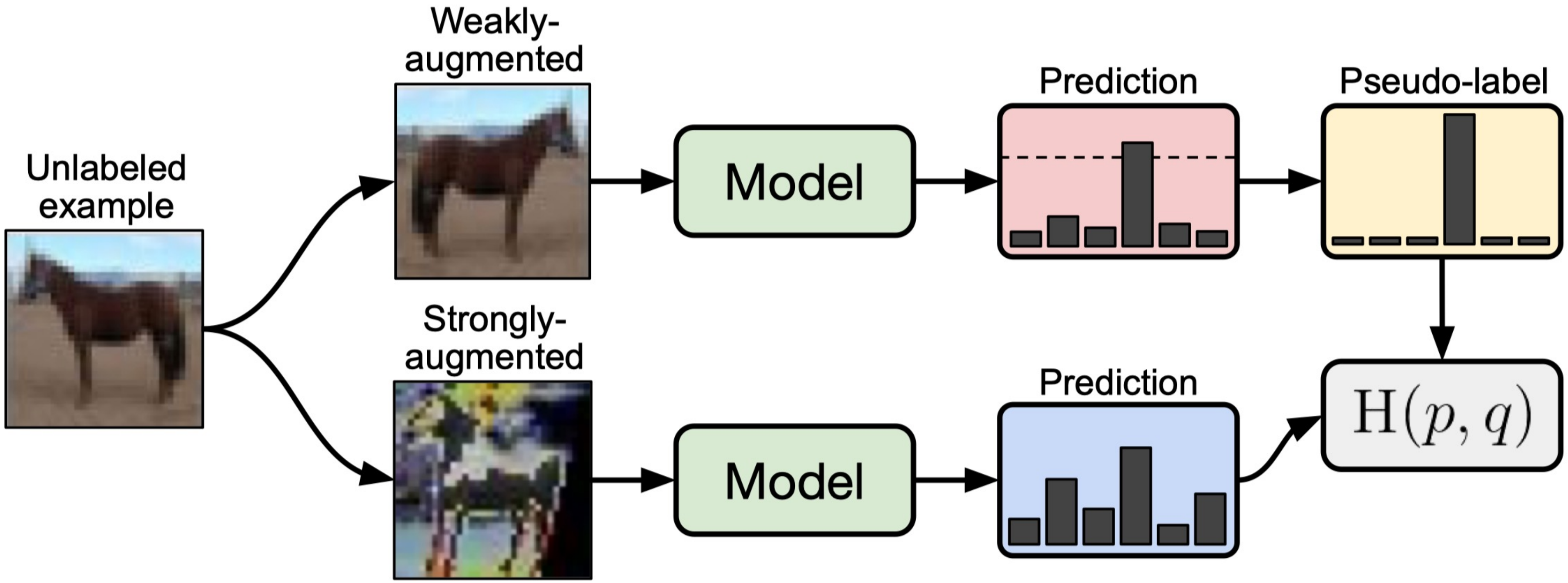




Berthelot D, Carlini N, et al. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. arXiv preprint arXiv:1911.09785. 2019 Nov 21.

Pseudo-Labelling

- FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence
  - Combination of Consistency regularization and pseudo-labeling.



Sohn K, Berthelot D, et al. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. arXiv preprint arXiv:2001.07685. 2020 Jan 21.

Pseudo-Labelling

- FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence
  - FixMatch consists of two loss terms: a supervised loss ℓ_s and an unsupervised loss ℓ_u
  - ℓ_s is the standard cross-entropy loss

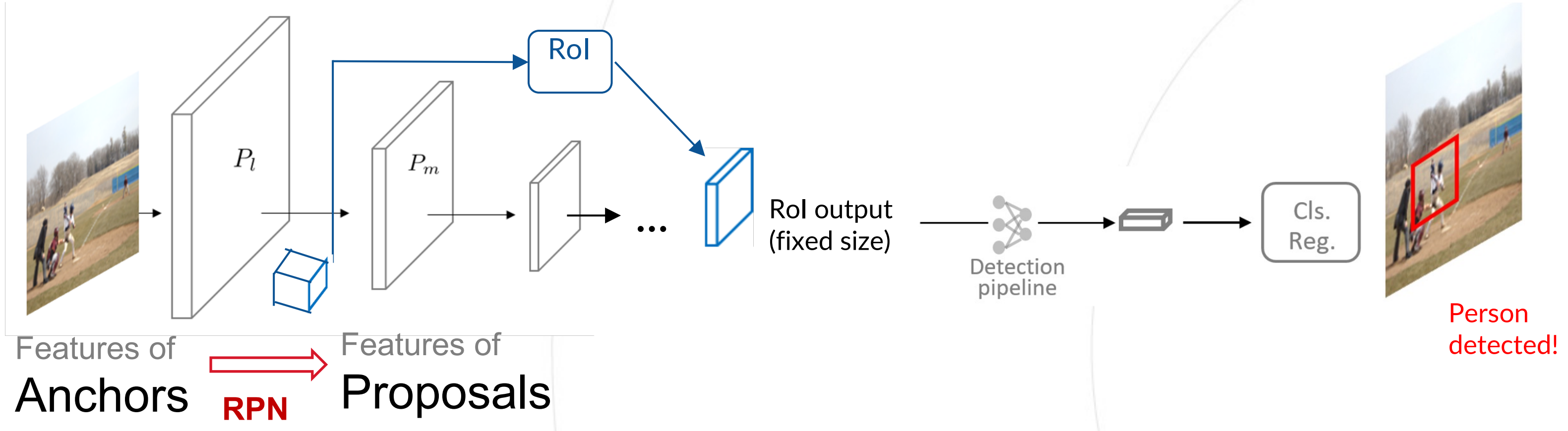
$$\ell_s = \frac{1}{B}\sum_{b=1}^{B} \mathrm{H}(p_b, p_{\mathrm{m}}(y \mid \alpha(x_b)))$$

  - Convert the prediction on the weakly-augmented image to a one-hot pseudo-label
  - ℓ_u is the cross-entropy loss against the model' s output for the strongly-augmented image

$$\ell_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \mathbb{1}(\max(q_b) \geq \tau)\, \mathrm{H}(\hat{q}_b, p_{\mathrm{m}}(y \mid \mathcal{A}(u_b)))$$

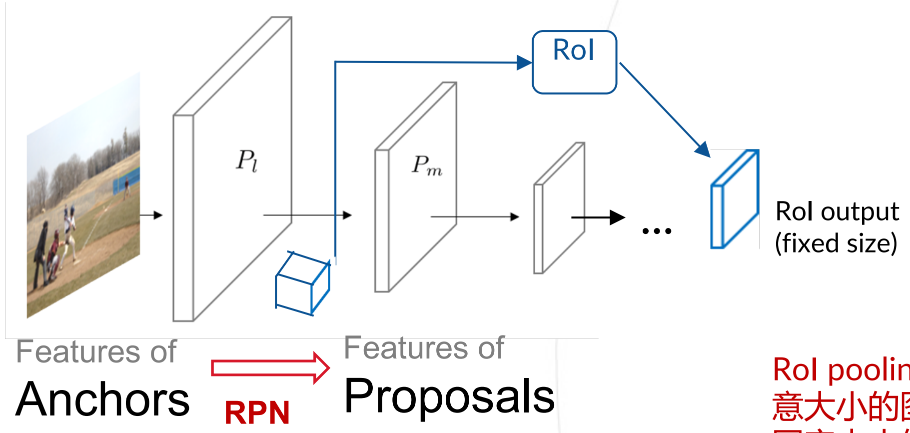
Recap: Image Detection

- **Two stage** vs One-stage pipeline
  - Anchor placement: large anchors should be put in the low-level layers or high-level layers?



Two stage vs One-stage pipeline

- **Two stage** vs One-stage pipeline
  - Loss of RoI pooling



$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r, j)] \frac{\partial L}{\partial y_{rj}}. \qquad (4)$$

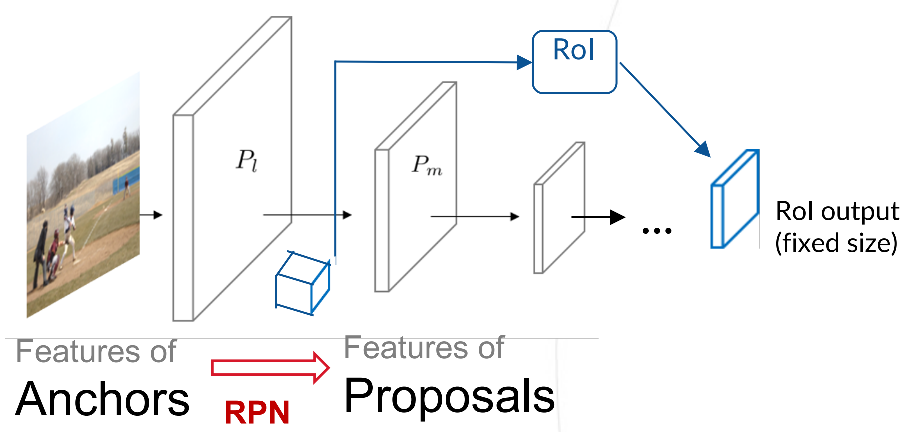
In words, for each mini-batch RoI r and for each pooling output unit y_{rj}, the partial derivative $\partial L / \partial y_{rj}$ is accumulated if i is the argmax selected for y_{rj} by max pooling. In back-propagation, the partial derivatives $\partial L / \partial y_{rj}$ are already computed by the `backwards` function of the layer on top of the RoI pooling layer.

RoI layer的BP计算。
详见Fast RCNN paper.

RoI pooling 的出现，让我们能用任意大小的图像作为输入，总能产生固定大小的输出。

Recap: Image Detection

- **Two stage** vs One-stage pipeline
  - Loss of RoI pooling



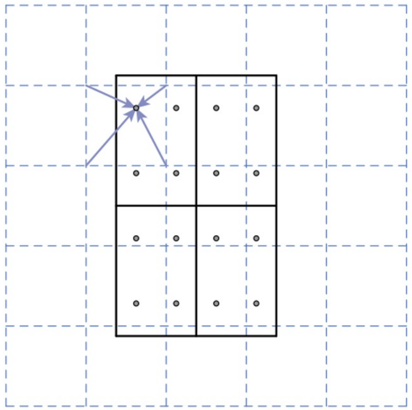
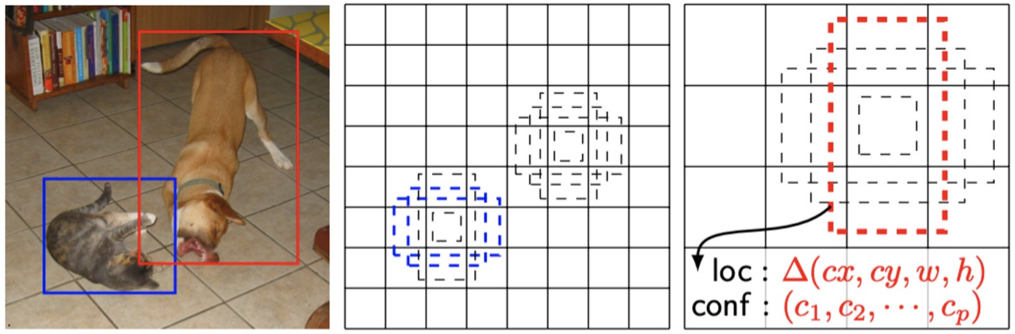


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

- Two stage vs **One-stage pipeline**
  - No RPN, No RoI-pooling



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

What's the output feature map in this (c) example?

Answer: **4 x 4** x (**4** * **4** + ***p***)

Within each grid cell:

- Regress from each for the **B base boxes** (aka anchors) to a final box with **(dx, dy, dh, dw)**
- Predict scores for each of ***p*** classes

Recap: Image Detection

- Detection loss

https://arxiv.org/pdf/1506.01497.pdf

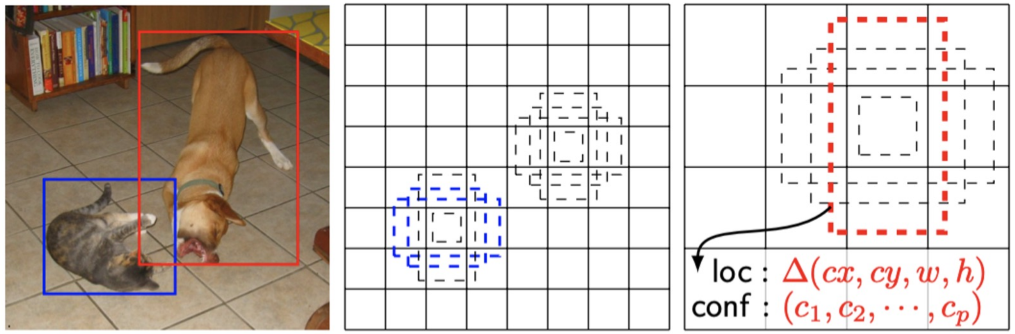
$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*)$$
$$+\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

*p** and *t** are the ground truth for classification and localization/regression

- Note that regression loss is only for positive samples.

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$$

R is the smoothed L1 loss

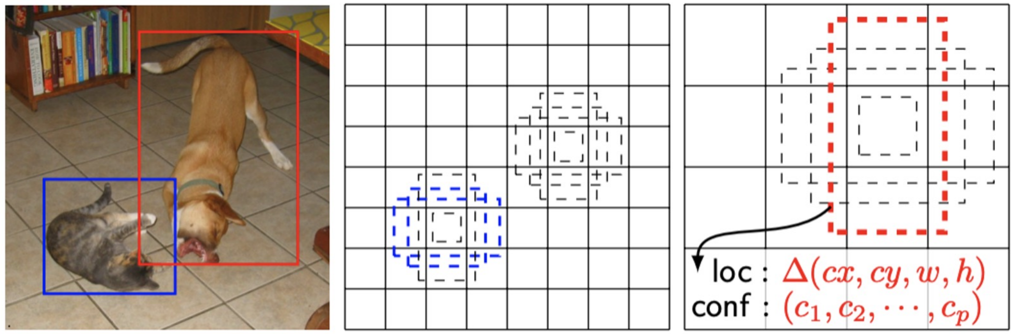


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

- Detection loss

https://arxiv.org/pdf/1506.01497.pdf



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

*p** and *t** are the ground truth for classification and localization/regression

- Note that regression loss is only for positive samples.

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$$

R is the smoothed L1 loss

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

A bounding-box regression from **an anchor box** to a nearby ground-truth box.

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

- YOLO vs SSD

SSD:
- Smaller input size
- Faster FPS

SSD:
- More templates/anchors from various depth in the network
- Higher mAP

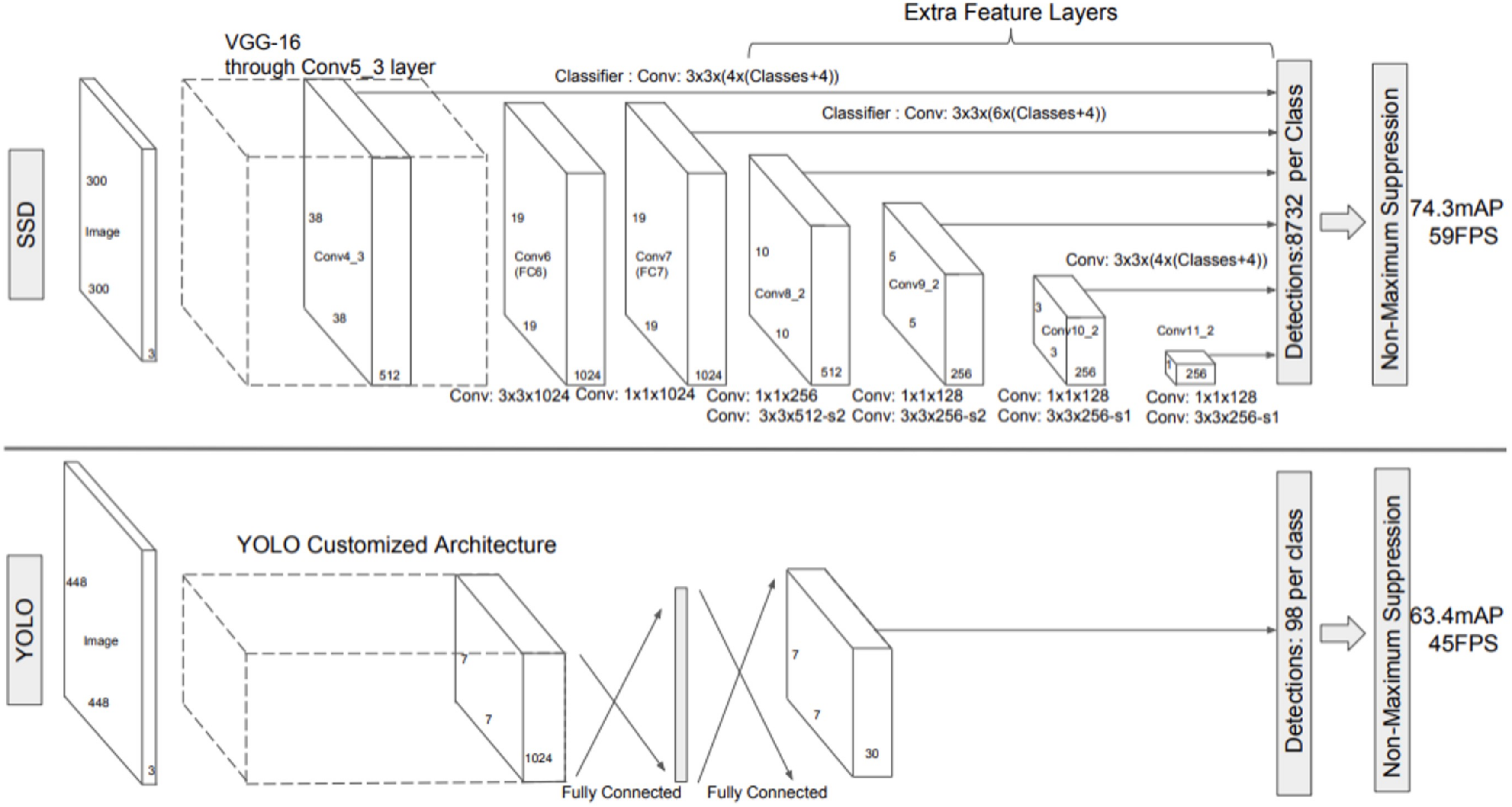


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 × 300 input size significantly outperforms its 448 × 448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Recap: Image Detection

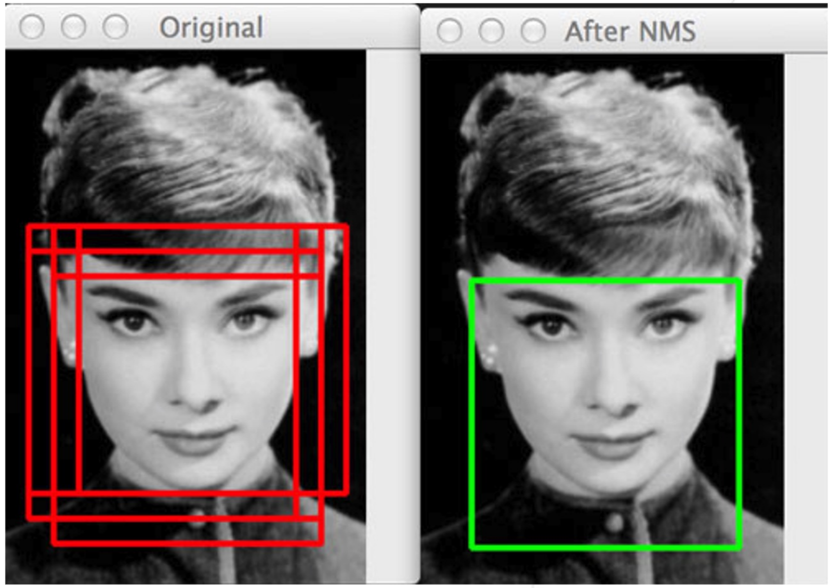
- One-stage detector: open-sourced repos

SSD Demo
https://github.com/hli2020/object_detection#testing-ssd
Or
(ipython notebook例子)
https://github.com/amdegroot/ssd.pytorch/blob/master/demo/demo.ipynb

How to implement a **YOLO (v3)** object detector from scratch in PyTorch
https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

SSD，YOLO这些方法都是one-stage detector.
没有RPN过程，直接生成检测结果。

- NMS



一种post-processing 方式。
用在**所有**检测系统里。

物体检测的指标里，不允许出现多个重复的检测，即使这些结果和真值都比较近。

那么如何删除多余的检测结果呢？
Non-maximum suppression (NMS)

做法：
把所有检测结果按照分值(conf. score)从高到底排序，保留最高分数的box，那么和它距离上最近的那个box, 就没有必要保留了。

以此类推。

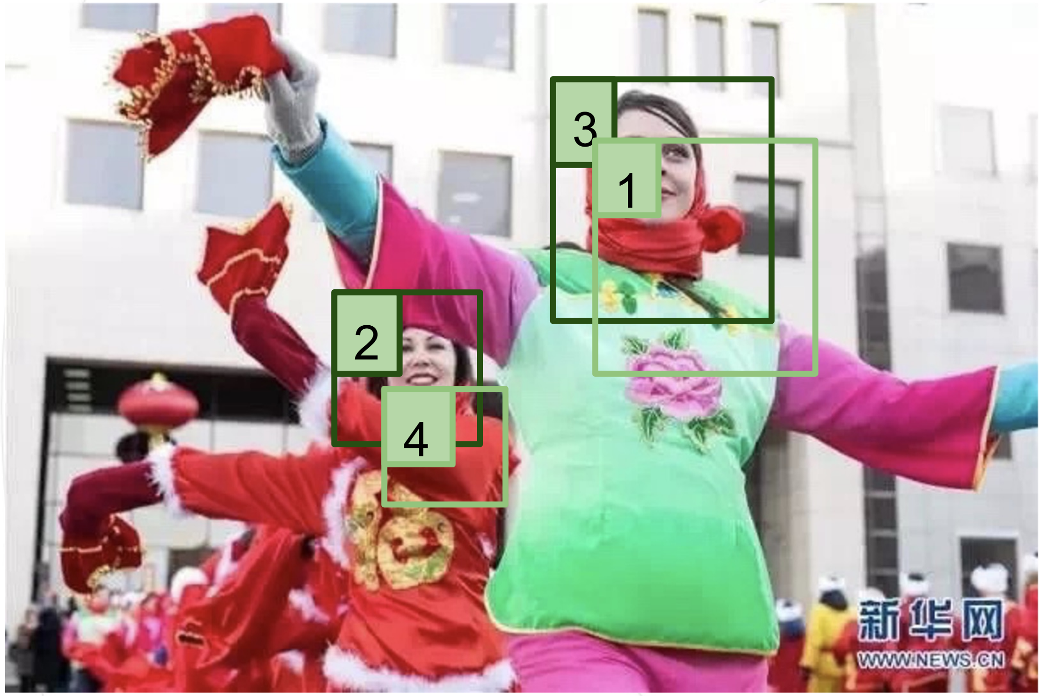
- NMS

按照类别来做的。

右图例子（检测人脸），

1-4分别是分数由高到低的4个目标框，假设1，3被判为距离较近，2，4距离很近，

哪些框保留，哪些要删除？



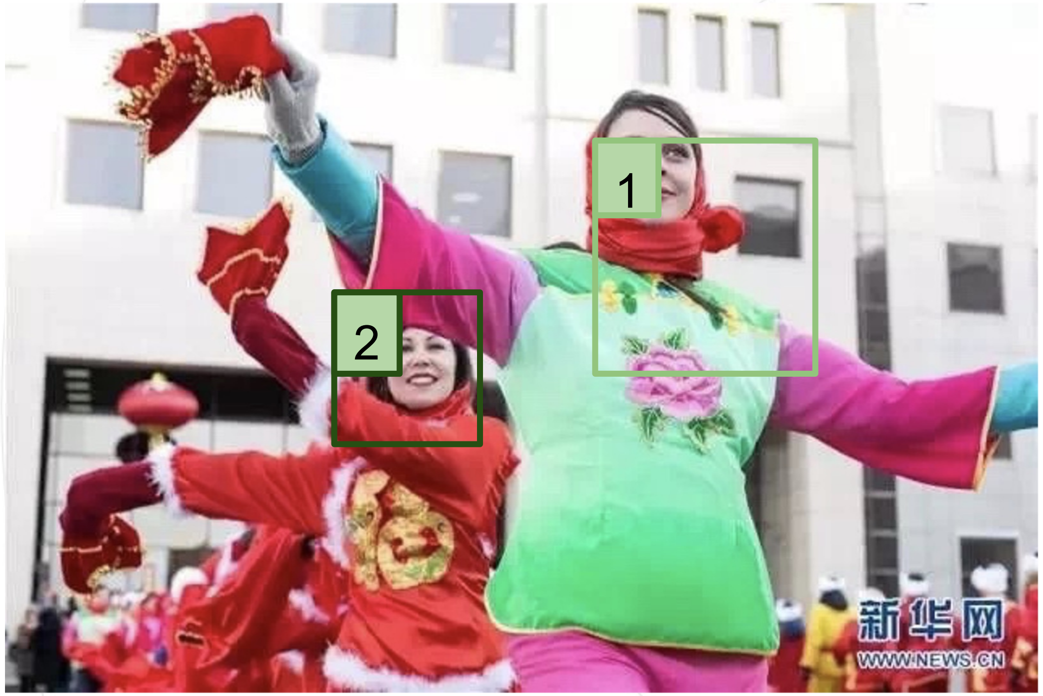
Recap: Image Detection

- NMS

NMS是按照***每一个类别***做的

https://github.com/hli2020/feature_intertwiner/blob/master/lib/layers.py#L664

3, 4 removed!

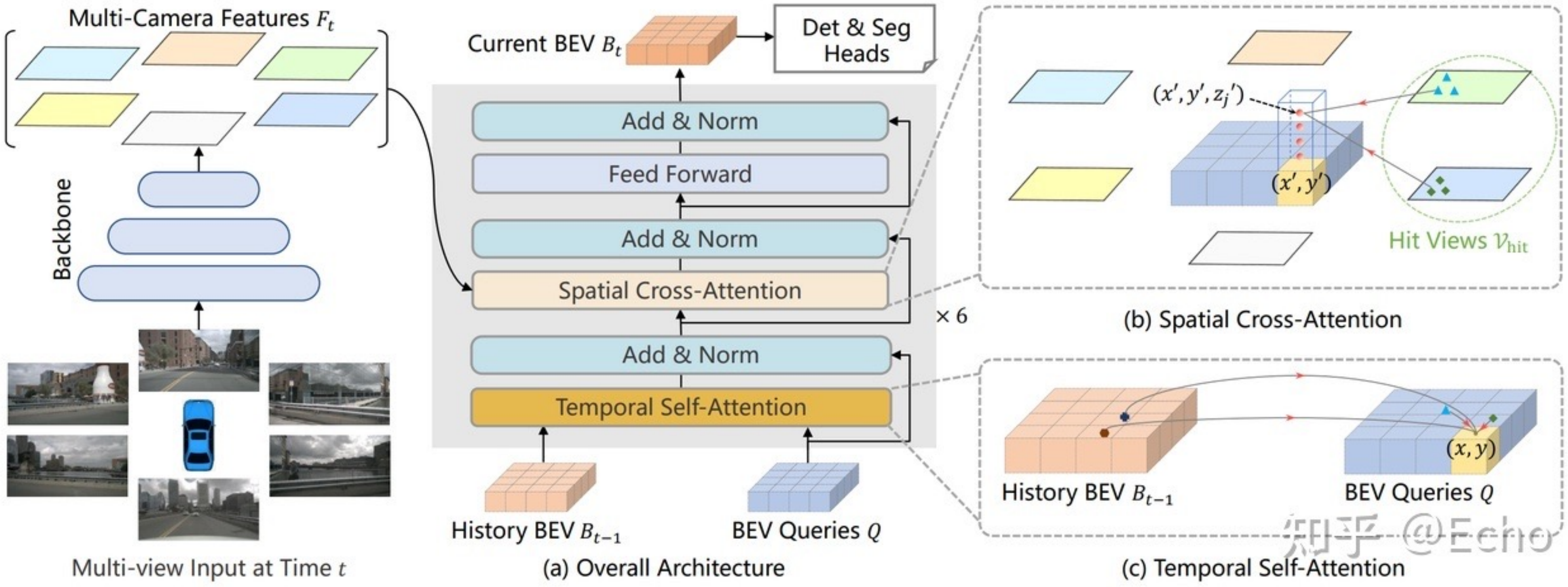


Outline

Bird’s-eye-view Perception

Leaderboard:
https://www.nuscenes.org/object-detection?externalData=all&mapData=no&modalities=Camera

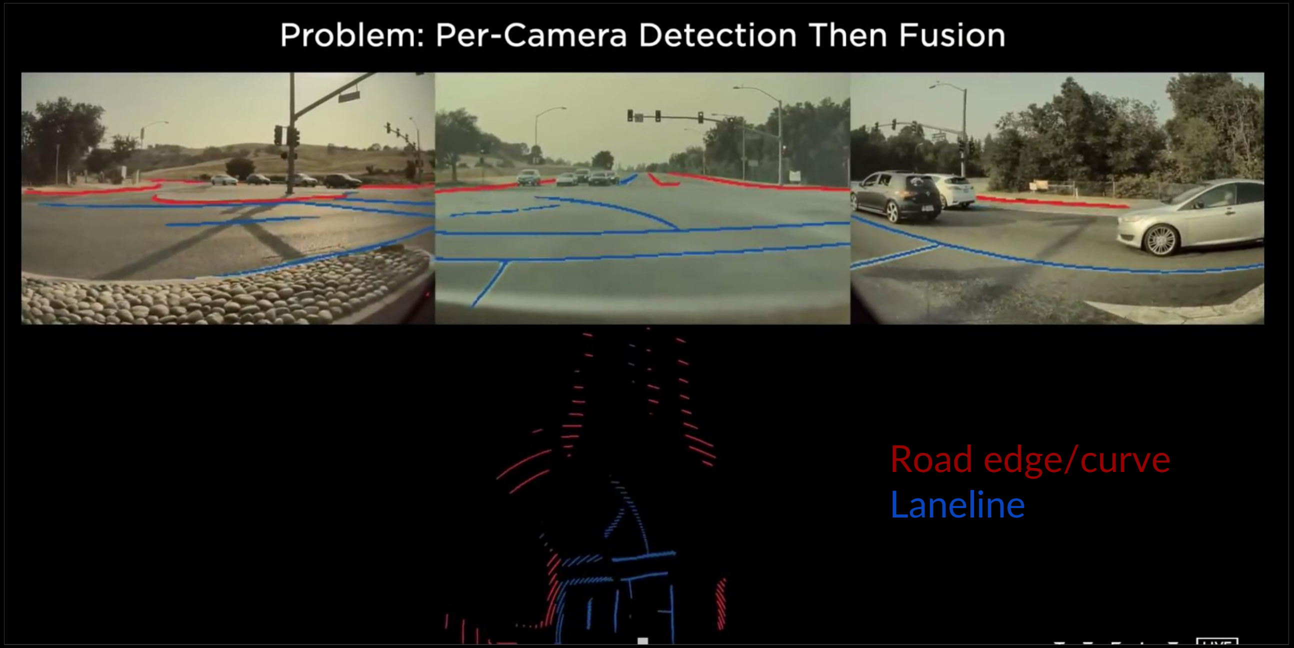
Tech blog:
https://zhuanlan.zhihu.com/p/495819042

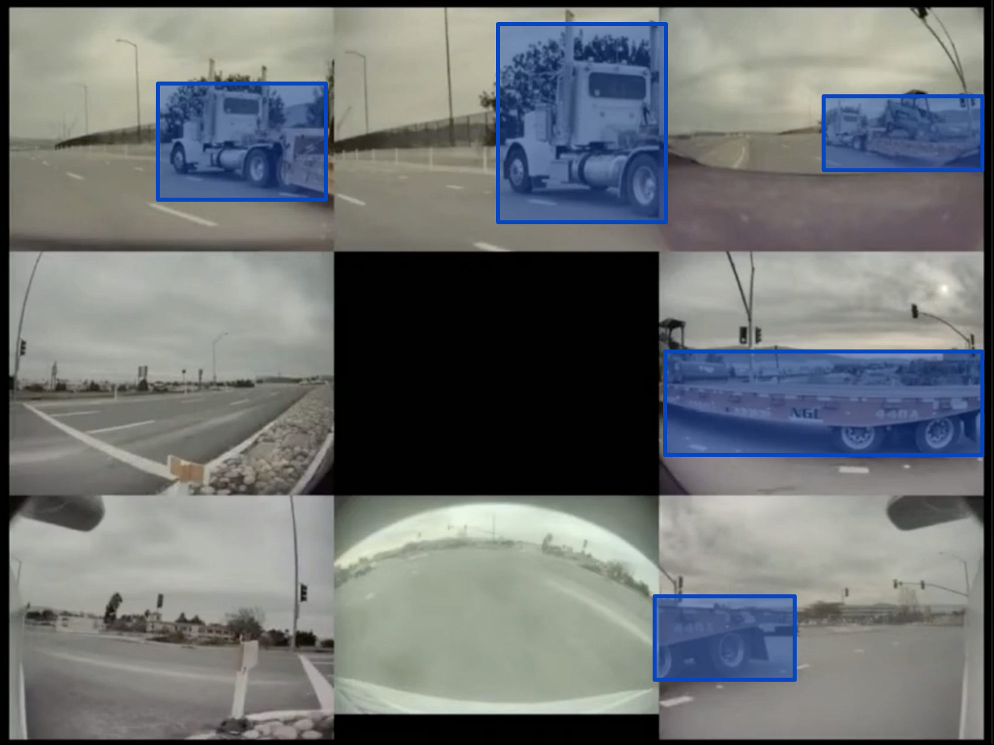


Smart Summon - Per-camera detection then fusion (nV)

Goal: summon vehicle to the person nearby

Cast out image-space predictions onto vector space





Traditional method:

- project from image plane to vector space. → Don't have depth per pixel
- assumption ground is horizontal. → which is not ture

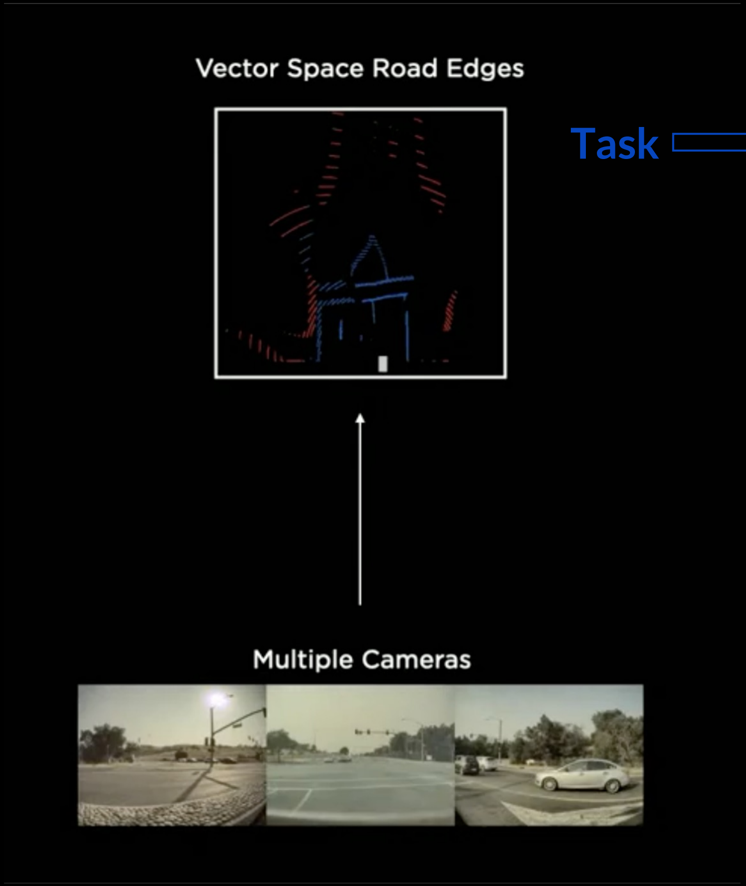
Fusion is difficult as objects span **differently** across images.

Smart Summon - Per-camera detection then fusion (nV)

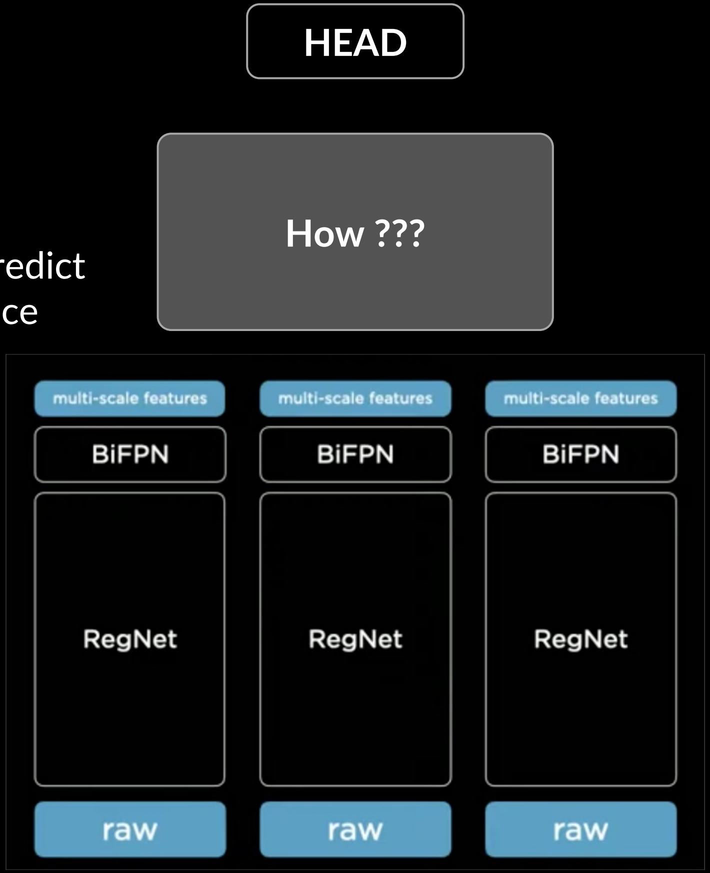
CORE

Goal: summon vehicle to the person nearby

Cast out image-space predictions onto vector space



Task ⟹ **Solution:**
Directly predict vector space results



Caveats

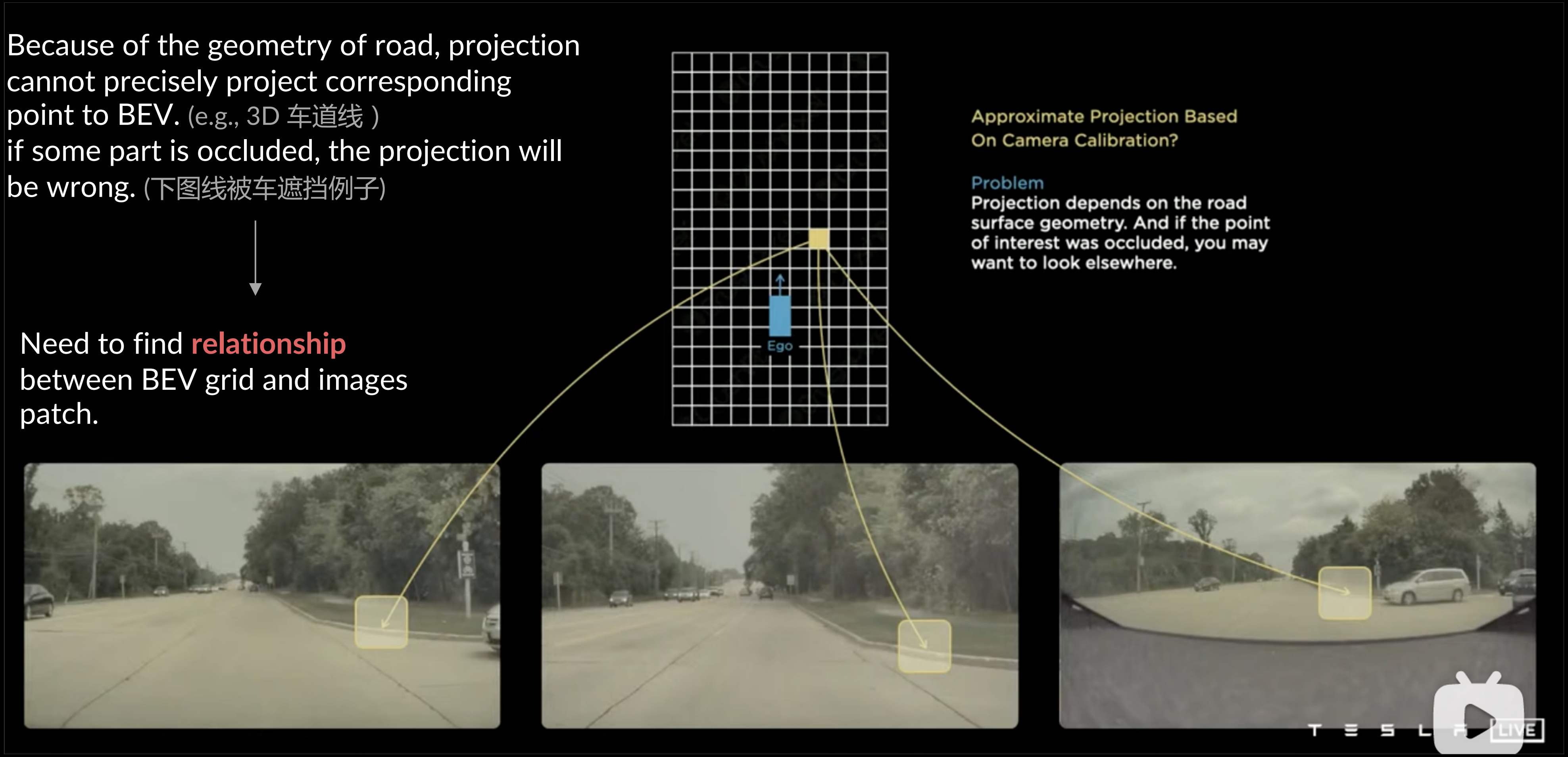
1. How to transform features from image-space to vector space?
   a. differentiable, e2e
   b. camera pose **varies**

1. Vector space dataset
   a. massive labelling (coming up)

Caveat 1

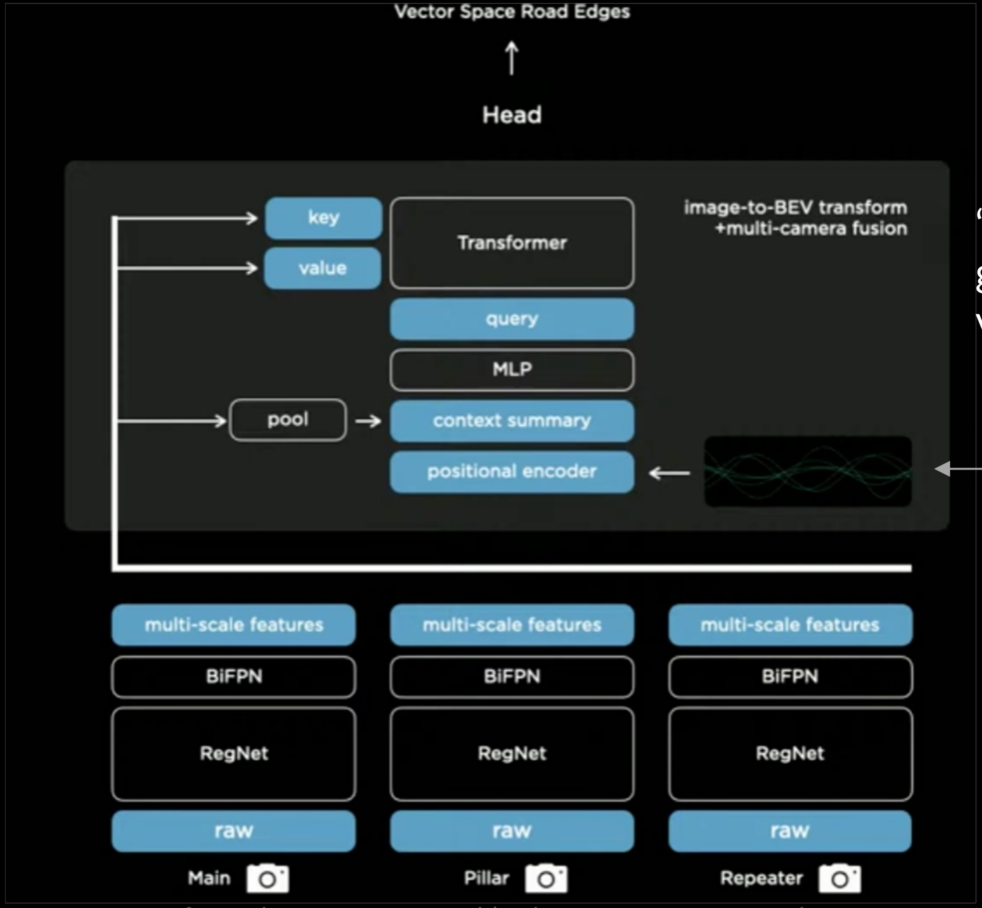
- Because of the geometry of road, projection cannot precisely project corresponding point to BEV. (e.g., 3D 车道线)
- if some part is occluded, the projection will be wrong. (下图线被车遮挡例子)

Need to find **relationship** between BEV grid and images patch.



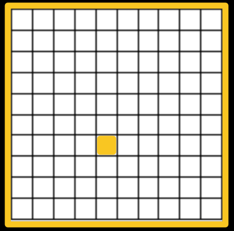
Solution to Caveat 1: Transformer

Attention is all you need, https://arxiv.org/pdf/1706.03762.pdf 2017



front view side view rear view

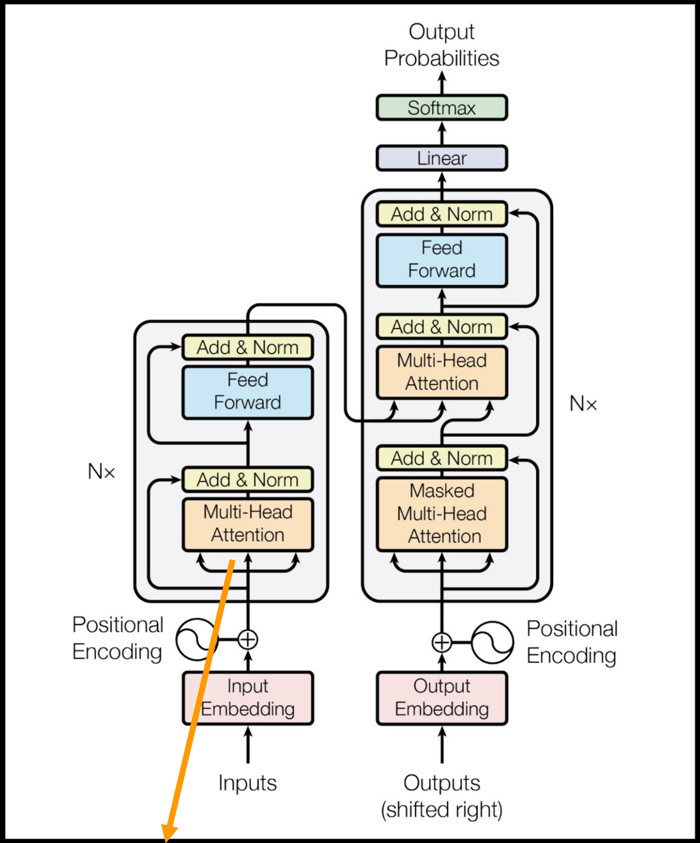
“Raster” position encoding, generate a position encoding vector for every grid on raster.

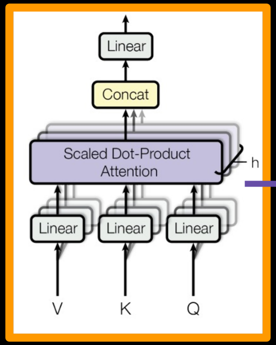


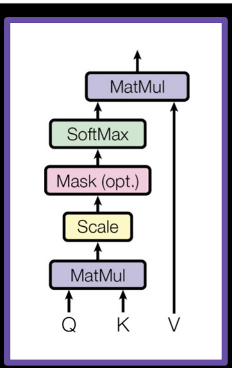
for location *(i,j)* in BEV view

$$A(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Transformer - Model structure





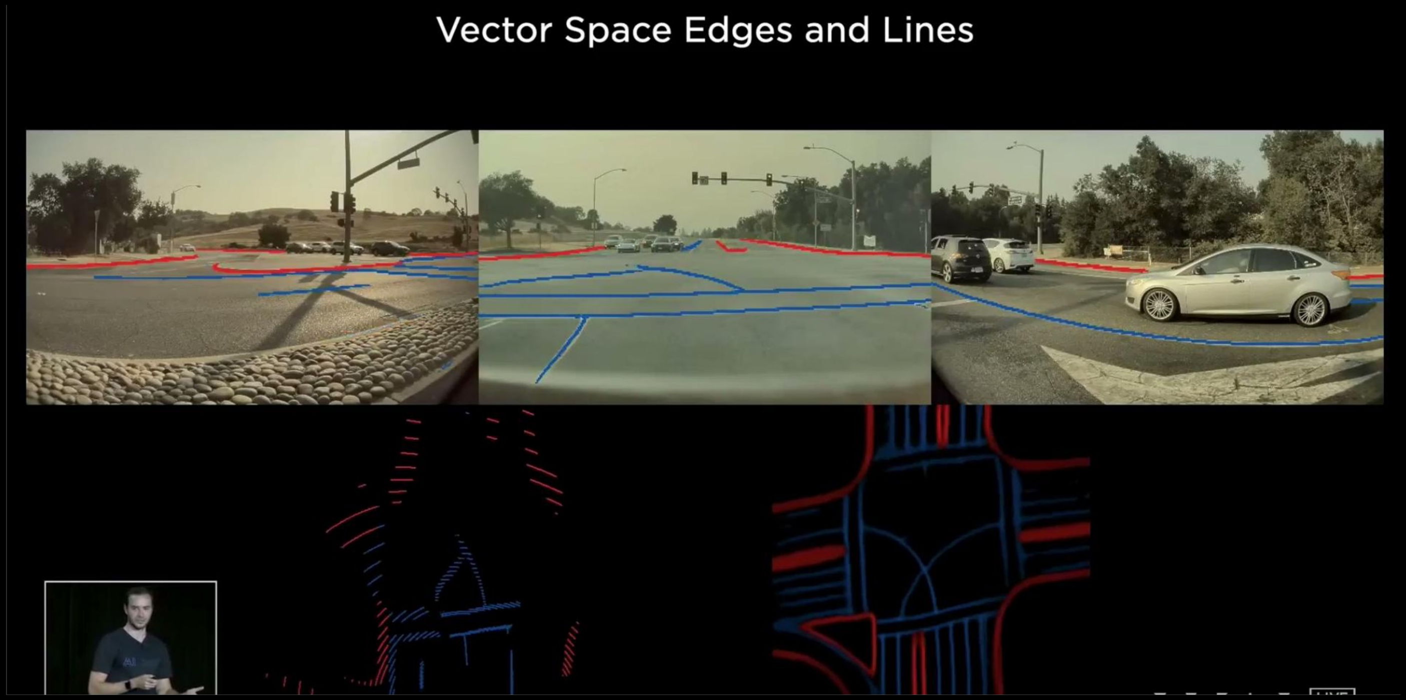


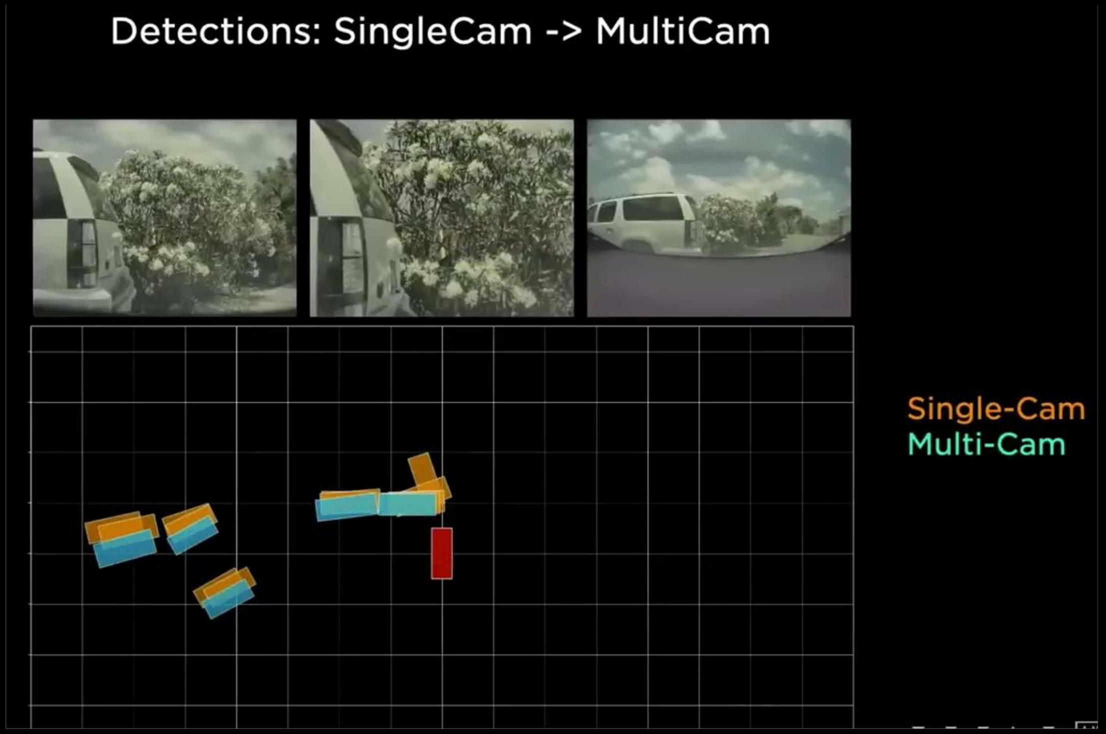
Multi-head attention Dot-product Attention

Background on Transformer

- What: a query and a set of key-value pairs to an output
- The output: a **weighted** sum of the **values**, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Improvement after Transformer and Rectification





Before

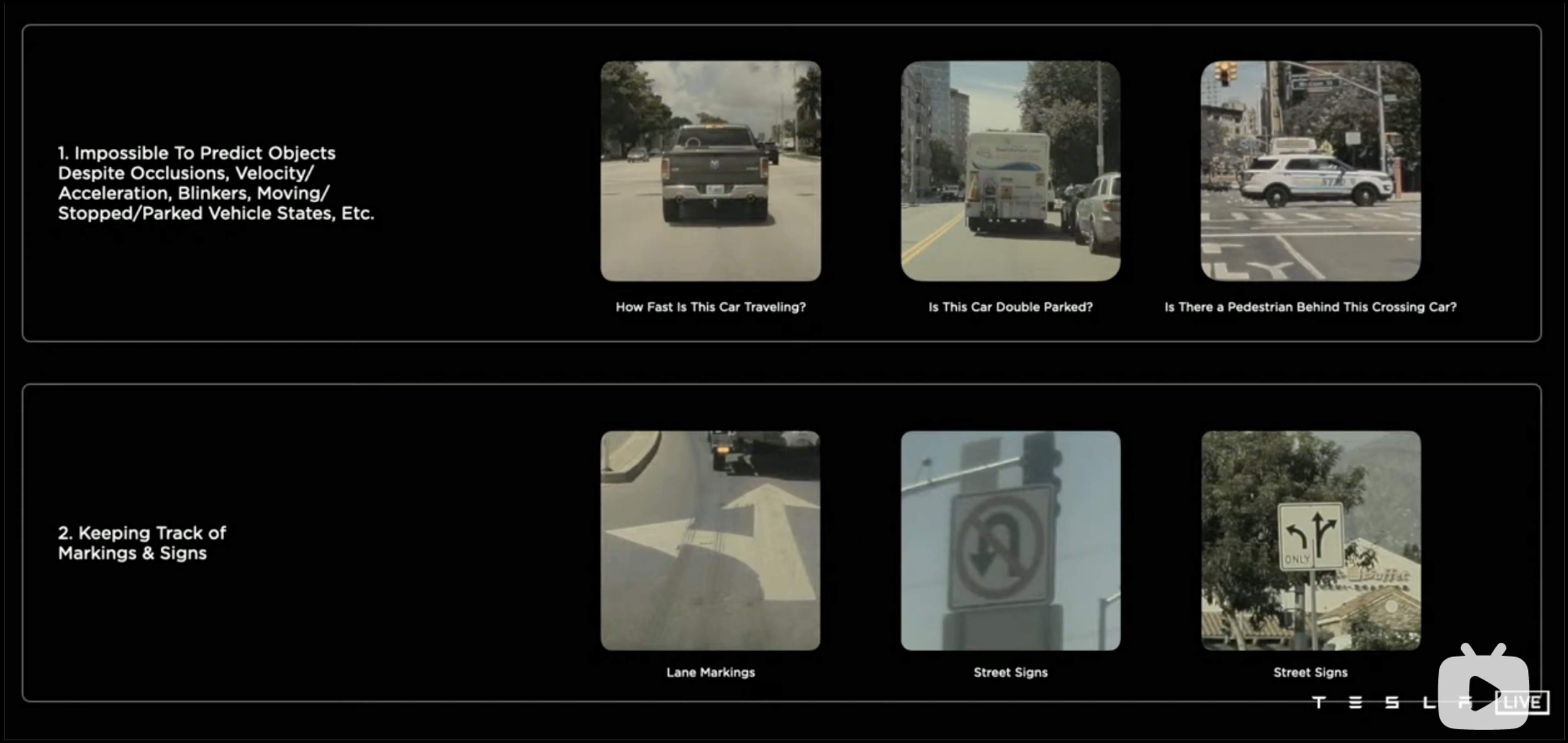
Road edge/curve
Laneline

After (nV, Transformer + Rectify)

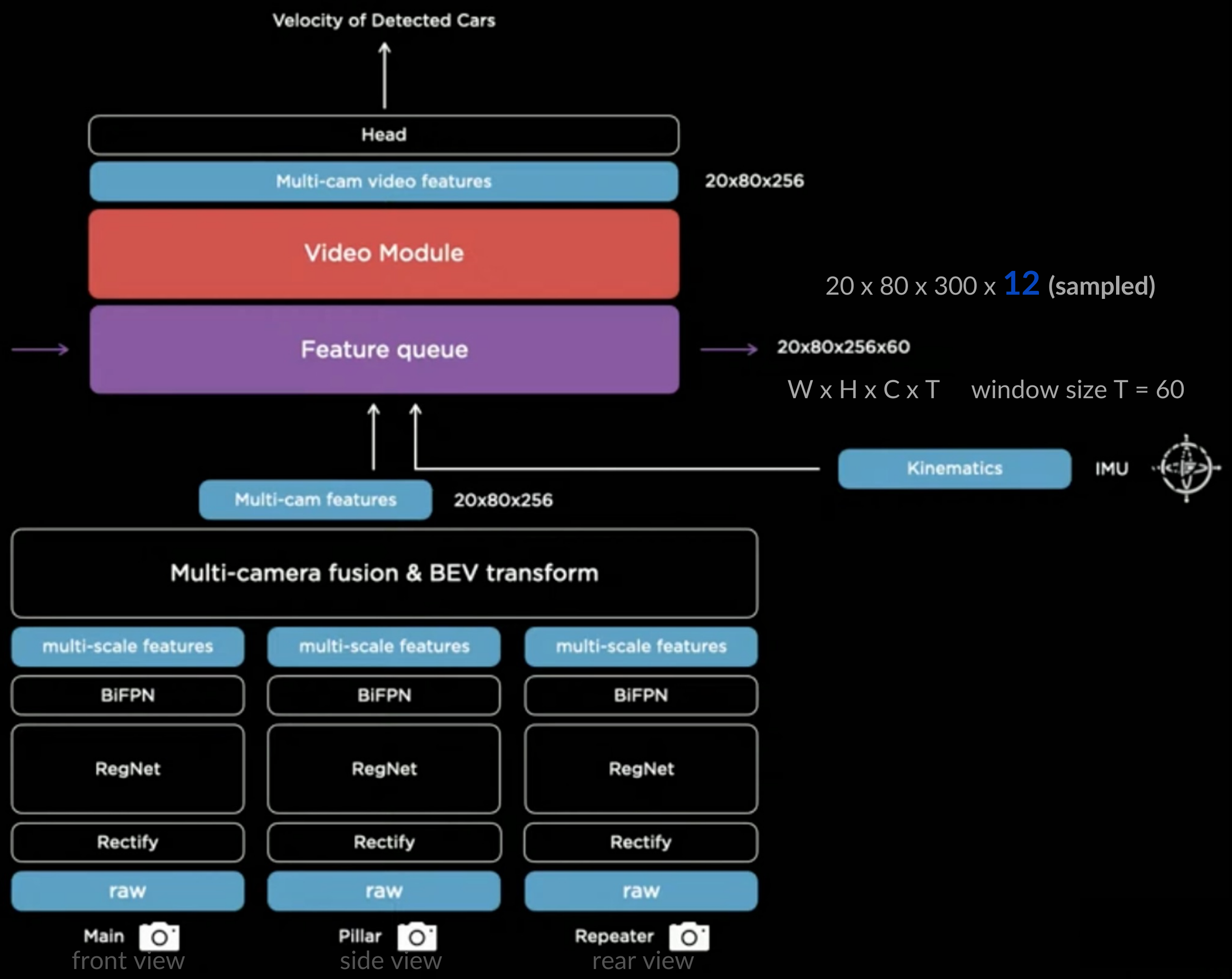
It’s basically night and day (天差地别)

Stepping further - Motivation: Lack of memory

Introducing temporal info

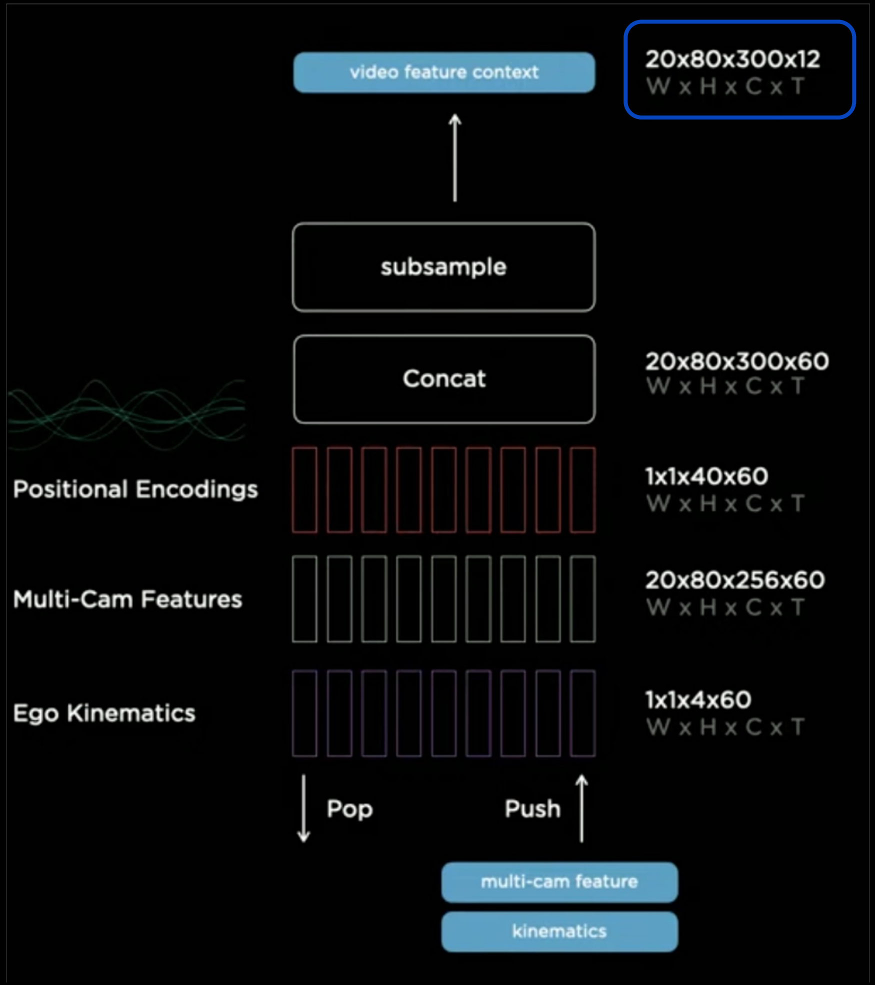


Video Neural Net Architecture



feature queue

Input to video module

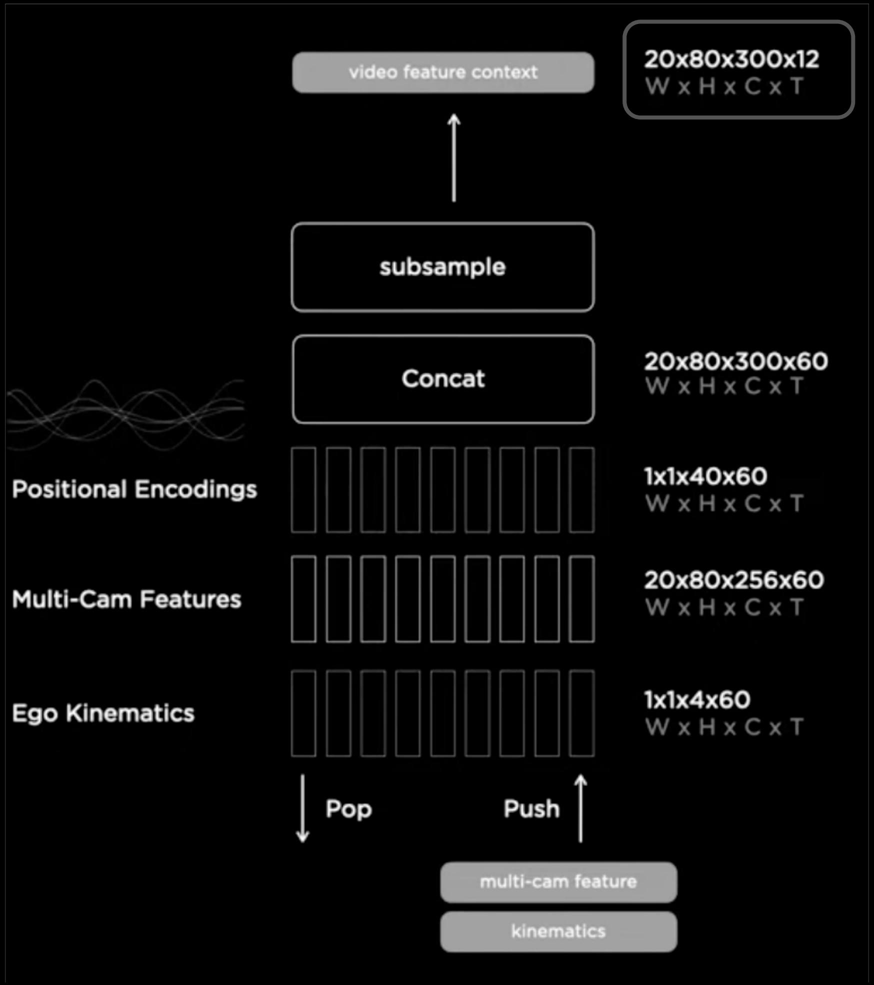


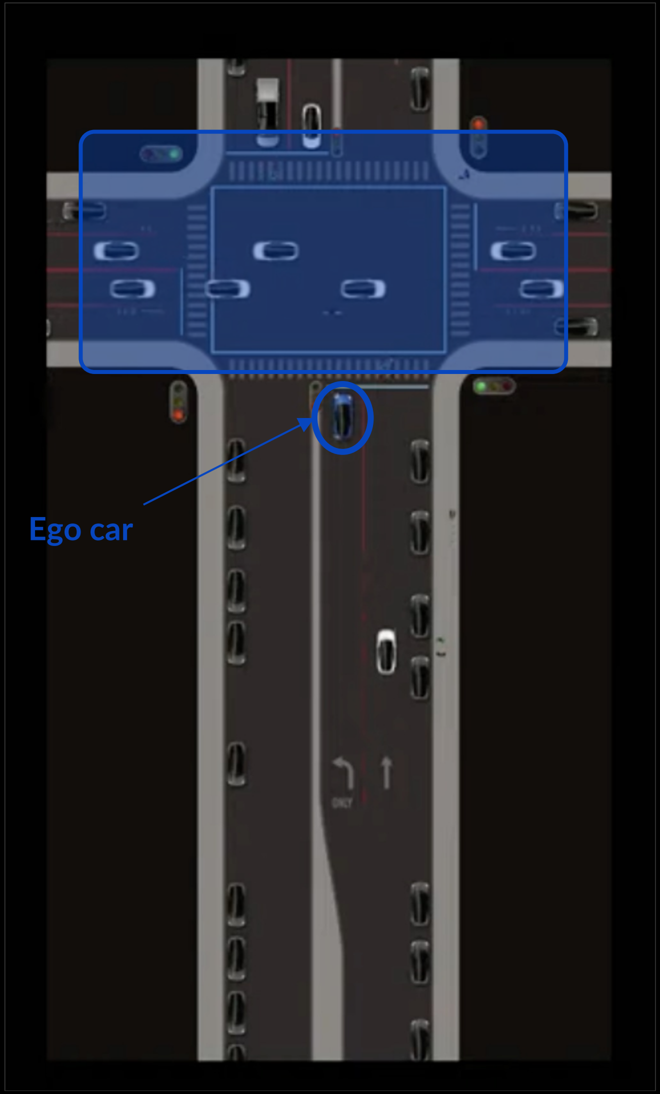
Positional encoding **(40)**: encode (x,y) as does in Transformer paper
Ego kinematics **(4)**: velocity, acc. etc

feature queue

Why use/push the queue?

Input to video module





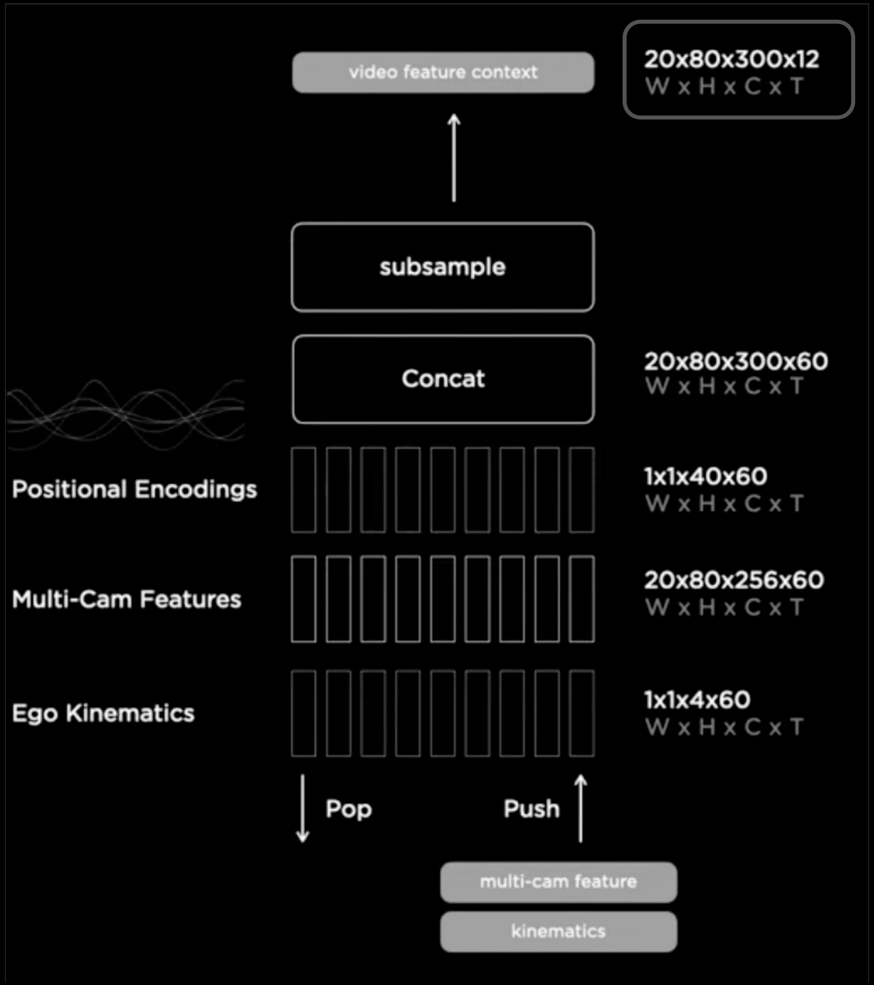
1. Temporary occlusions
=> time-based queue
(e.g. push every 27ms)

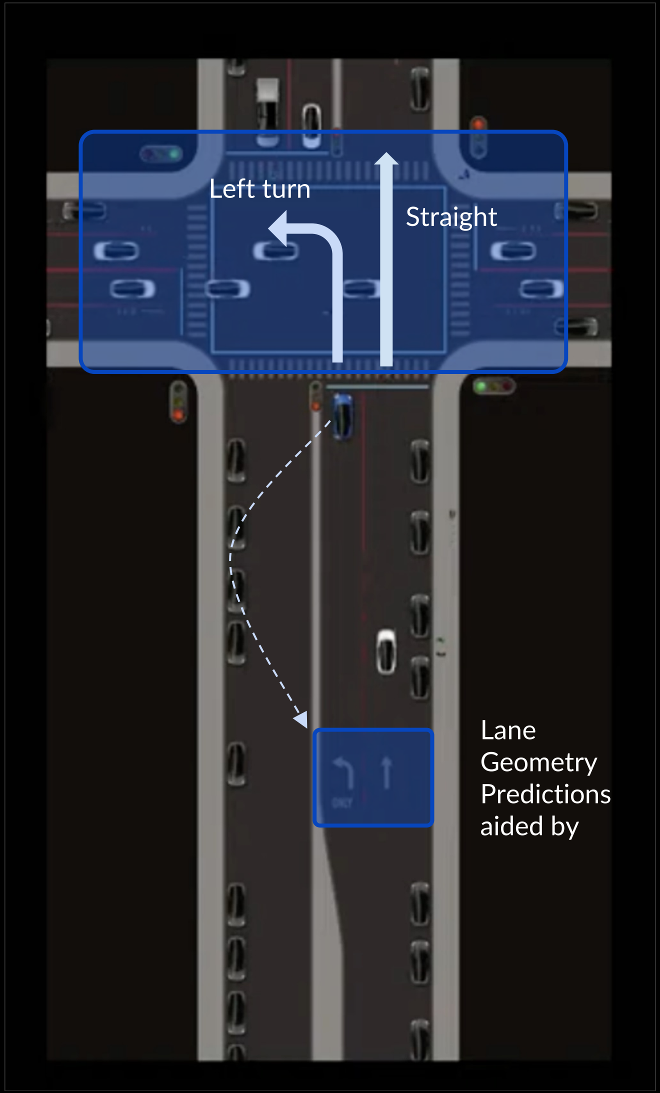
Positional encoding **(40)**: encode (x,y,z) to higher frequency [1]
Ego kinematics **(4)**: velocity, acc. etc

feature queue

Why use/push the queue?

Input to video module





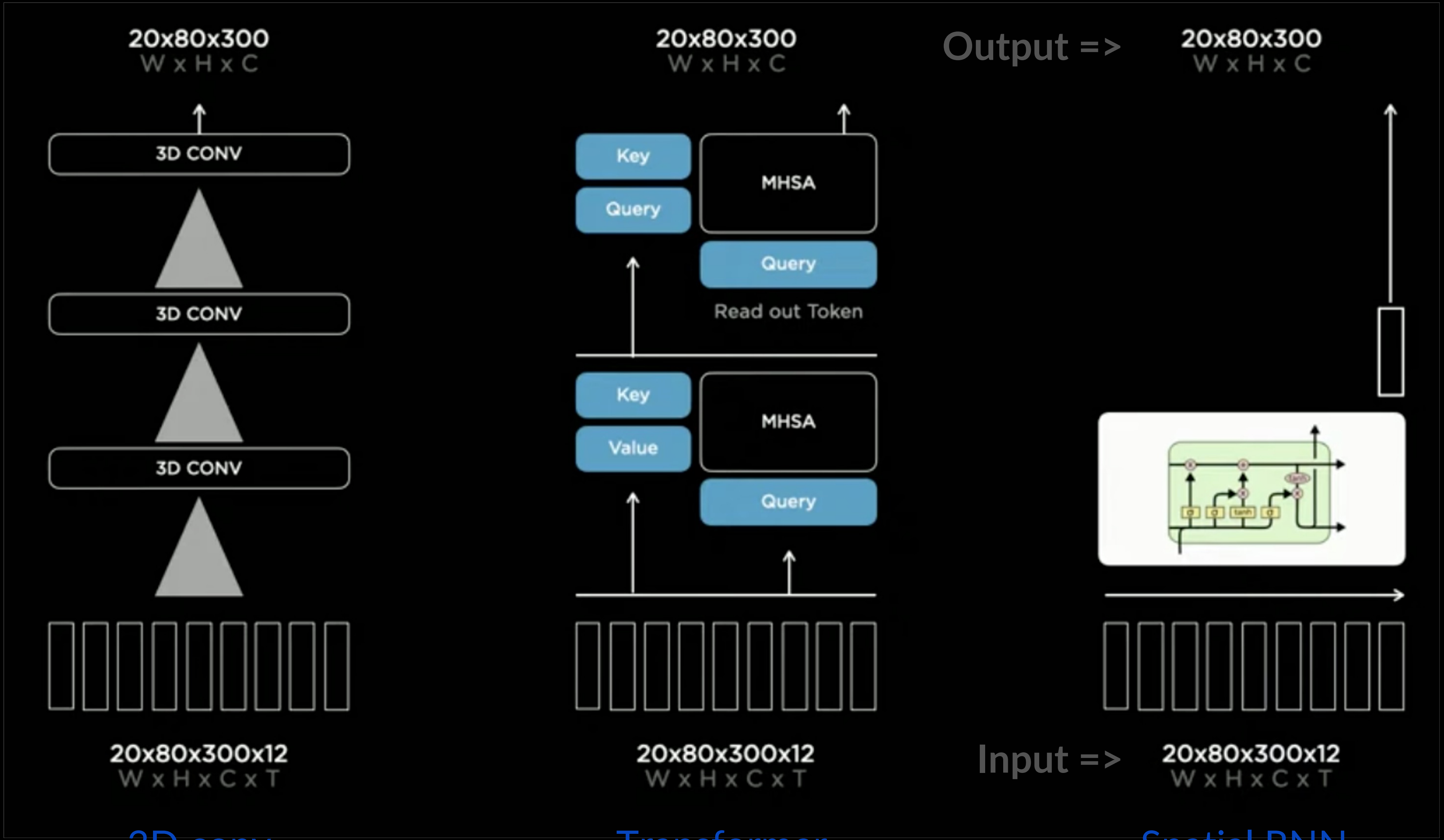
1. Temporary occlusions
=> time-based queue
(e.g. push every 27ms)

2. Signs & Markings Earlier on the Road
=> space-based queue
(e.g. push every 1 meter)

Positional encoding **(40)**: encode (x,y,z) to higher frequency [1]
Ego kinematics **(4)**: velocity, acc. etc

video module

Possible candidates



3D conv | Transformer | Spatial RNN

Axial Transformer [1]

[1] https://arxiv.org/abs/1912.12180

video module

Spatial RNN

Hidden state h(t-1)
W x H x C

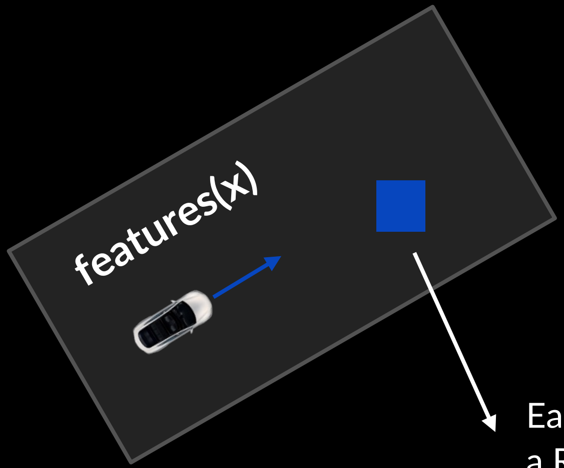
Input x(t)

kinematics

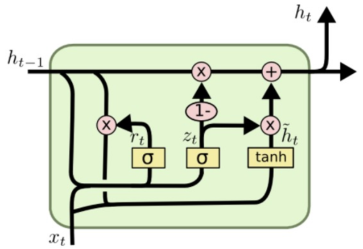
features

20 x 80 x 256
Ego Coordinate System

Spatial Feature Grid: h(t)
W x H x C

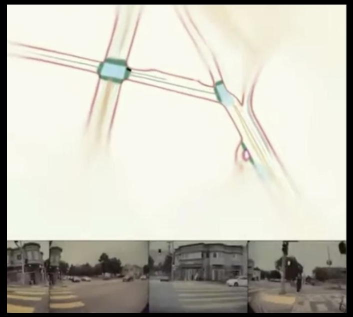


Each cell is a RNN



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

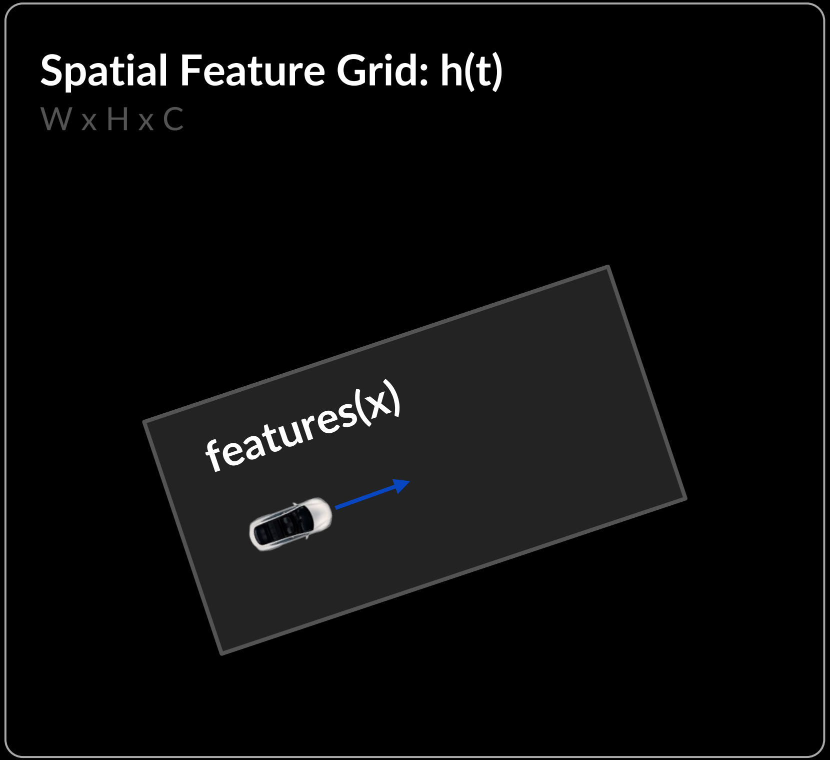
Output h(t)
W x H x C

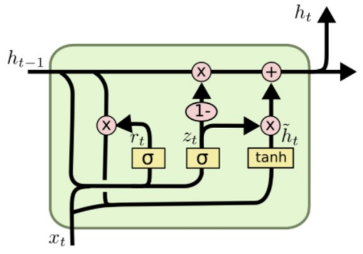


- 尺寸不一致（300/256）
- 20 x 80 - 我们的理解

video module

Spatial RNN





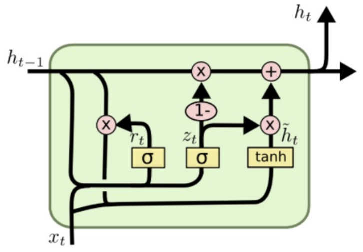
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Only update RNN at the points where they are **nearby** the ego car

- to save computational cost

video module

Spatial RNN



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

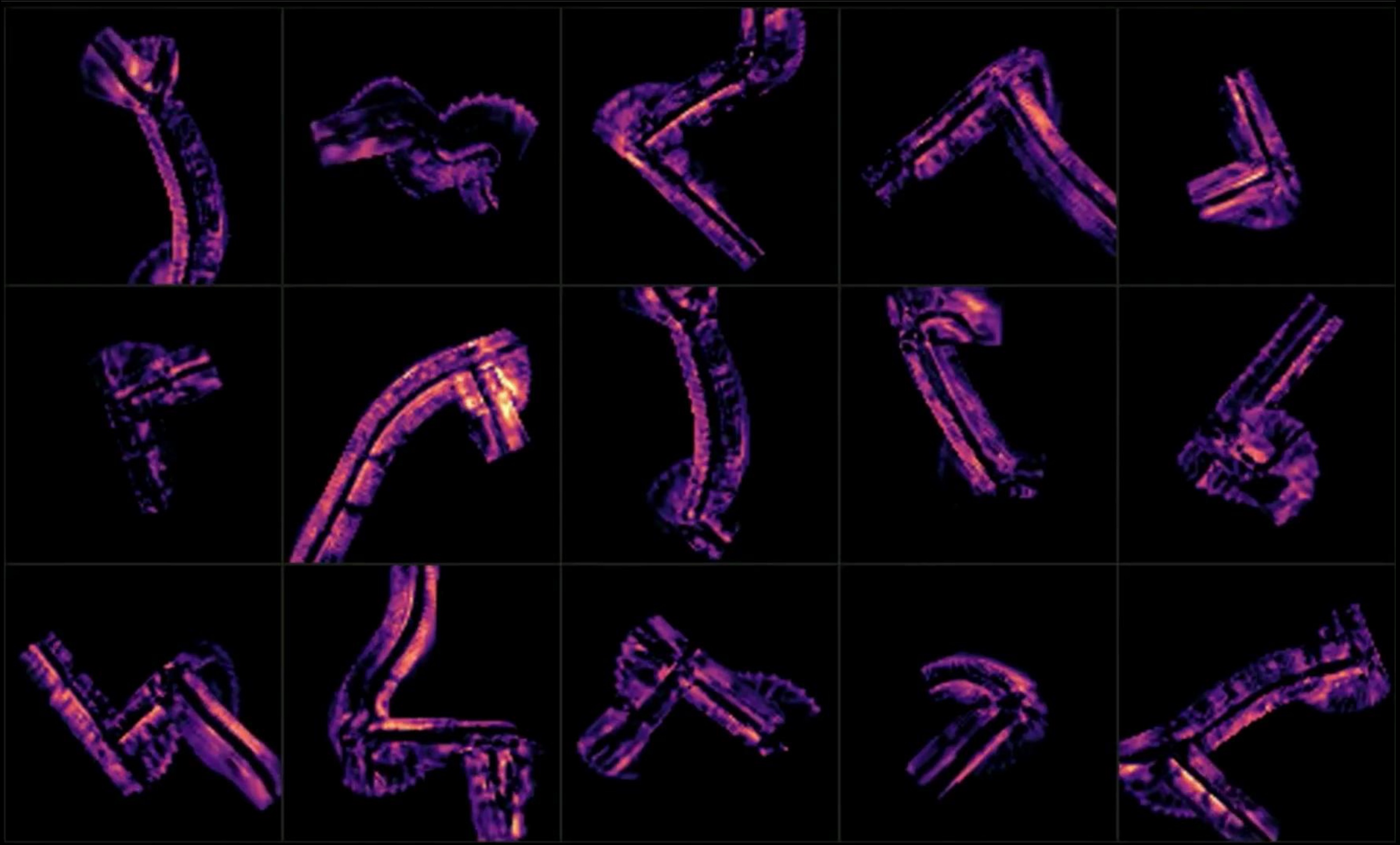
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Only update RNN at the points where they are **nearby** the ego car

- to save computational cost

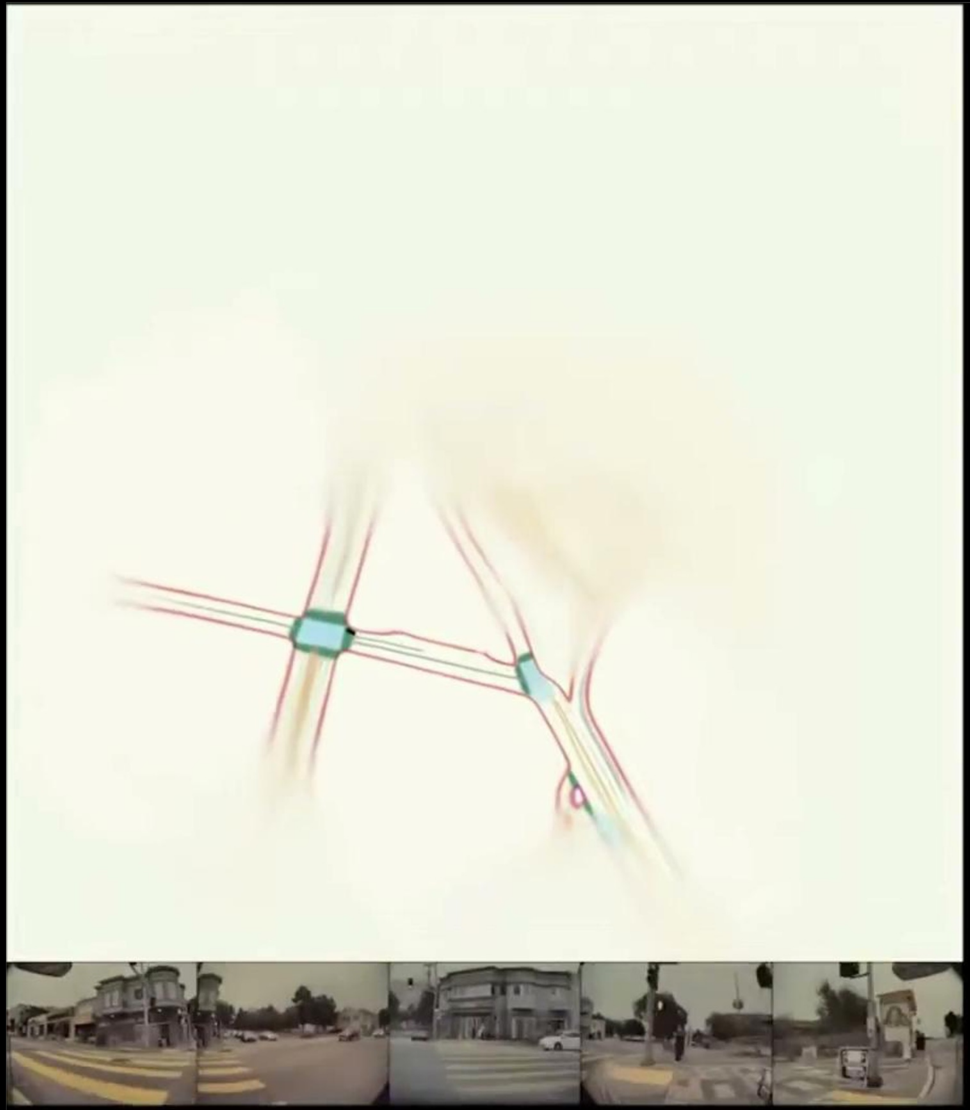
video module

Spatial RNN - Feature Channel Visualization

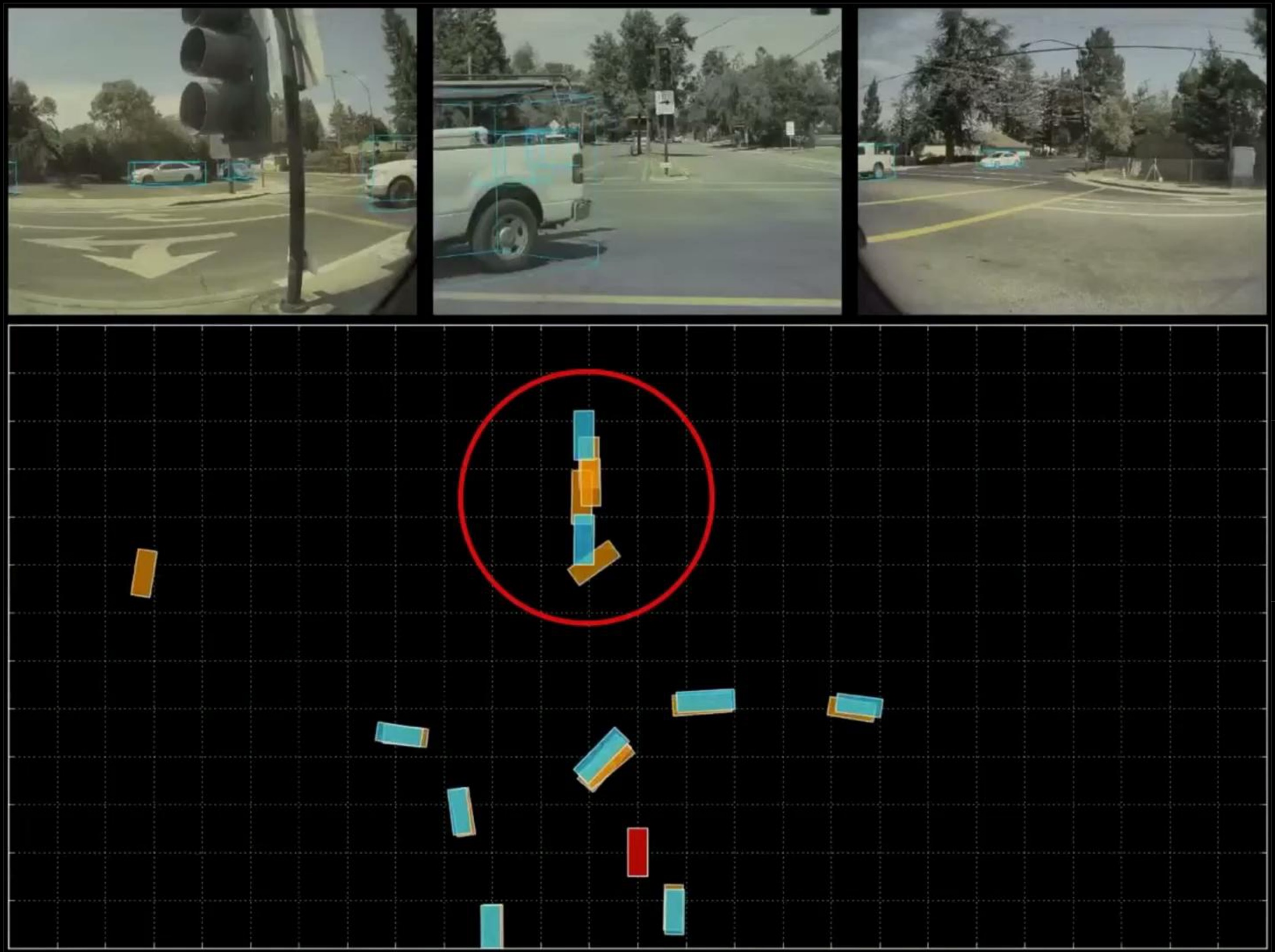


video module

Spatial RNN - Road reconstruction

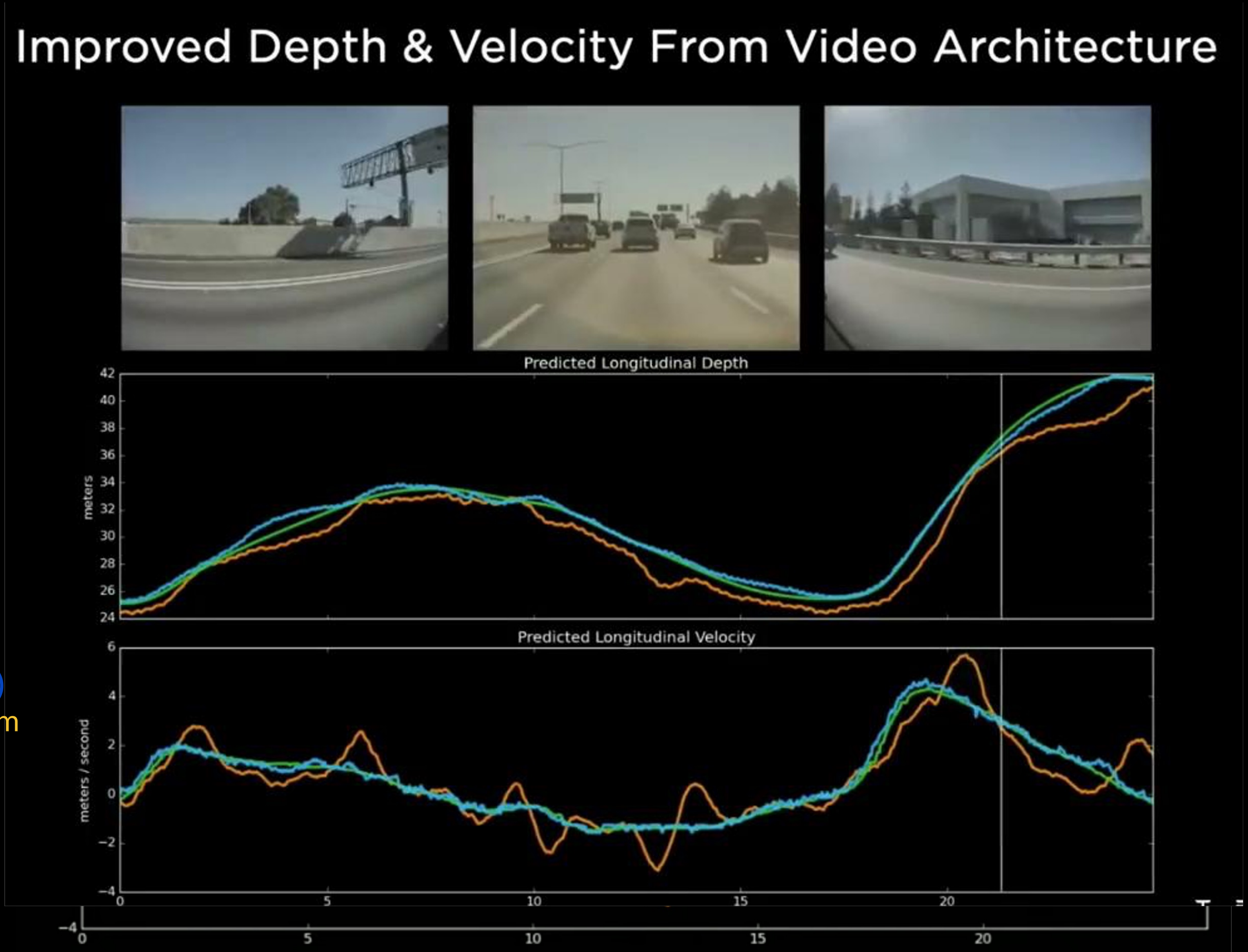


Object Detection - Improved Robustness to Temporary Occlusion



Single-Frame
Video

Improved Depth & Velocity from Video Architecture



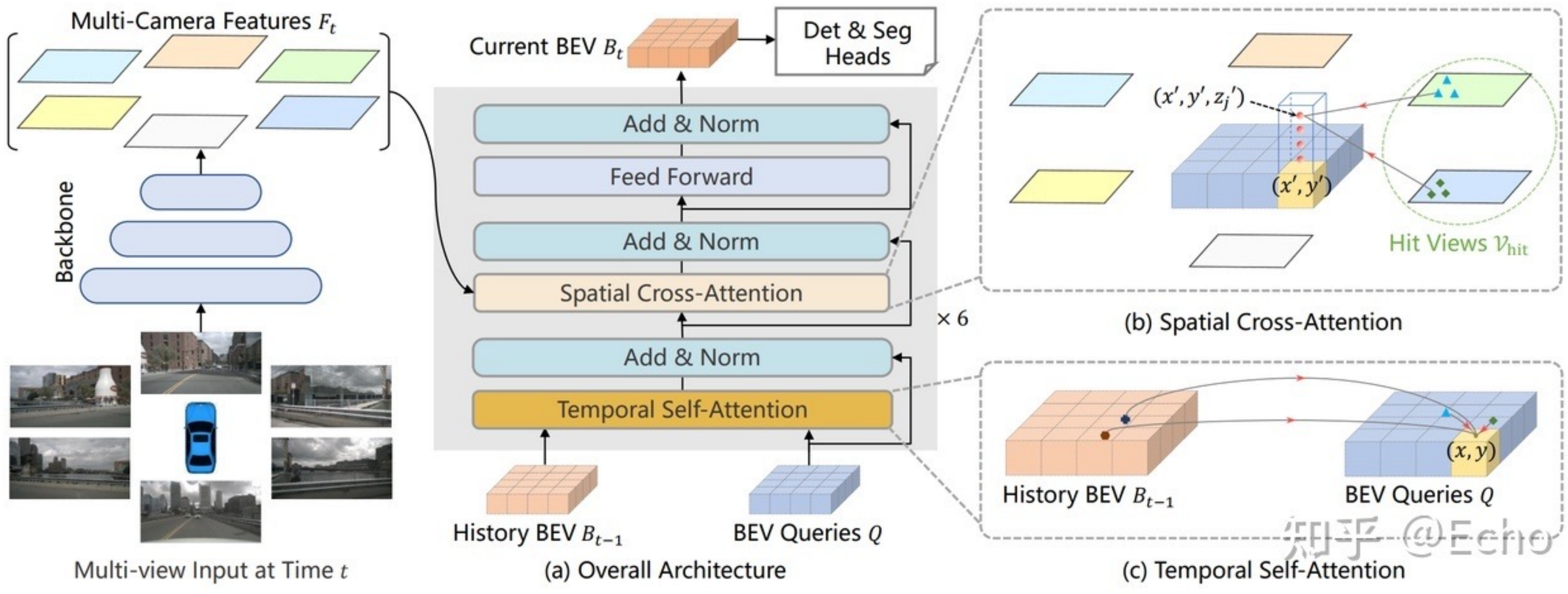
LEGEND

Radar signal (GT)
Video architecture (Ours)
Single frame (velocity from differentiable)

Bird's-eye-view Perception

[Leaderboard: https://www.nuscenes.org/object-detection?externalData=all&mapData=no&modalities=Camera](https://www.nuscenes.org/object-detection?externalData=all&mapData=no&modalities=Camera)

Tech blog:
https://zhuanlan.zhihu.com/p/495819042



Bird’s-eye-view Perception

Table 1: **3D Detection Results on nuScenes `test` set.** * notes that VoVNet-99 (V2-99) [21] was pre-trained on the depth estimation task with extra data [31]. “BEVFormer-S” does not leverage temporal information in the BEV encoder. “L” and “C” indicate LiDAR and Camera, respectively.

Method	Modality	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
SSN [54]	L	-	0.569	0.463	-	-	-	-	-
CenterPoint-Voxel [51]	L	-	0.655	0.580	-	-	-	-	-
PointPainting [43]	L&C	-	0.581	0.464	0.388	0.271	0.496	0.247	0.111
FCOS3D [45]	C	R101	0.428	0.358	0.690	0.249	0.452	1.434	**0.124**
PGD [44]	C	R101	0.448	0.386	**0.626**	**0.245**	0.451	1.509	0.127
BEVFormer-S	C	R101	0.462	0.409	0.650	0.261	0.439	0.925	0.147
BEVFormer	C	R101	**0.535**	**0.445**	0.631	0.257	**0.405**	**0.435**	0.143
DD3D [31]	C	V2-99*	0.477	0.418	**0.572**	**0.249**	**0.368**	1.014	**0.124**
DETR3D [47]	C	V2-99*	0.479	0.412	0.641	0.255	0.394	0.845	0.133
BEVFormer-S	C	V2-99*	0.495	0.435	0.589	0.254	0.402	0.842	0.131
BEVFormer	C	V2-99*	**0.569**	**0.481**	0.582	0.256	0.375	**0.378**	0.126

Table 2: **3D Detection Results on nuScenes `val` set.** “C” indicates Camera.

Method	Modality	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D [45]	C	R101	0.415	0.343	0.725	0.263	0.422	1.292	**0.153**
PGD [44]	C	R101	0.428	0.369	0.683	**0.260**	0.439	1.268	0.185
DETR3D [47]	C	R101	0.425	0.346	0.773	0.268	0.383	0.842	0.216
BEVFormer-S	C	R101	0.448	0.375	0.725	0.272	0.391	0.802	0.200
BEVFormer	C	R101	**0.517**	**0.416**	**0.673**	0.274	**0.372**	**0.394**	0.198

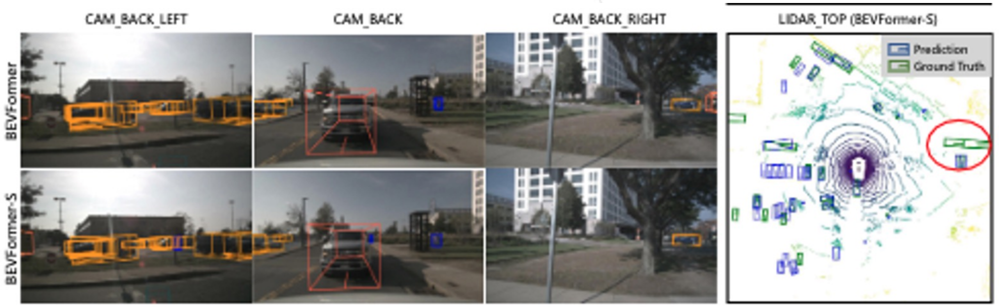
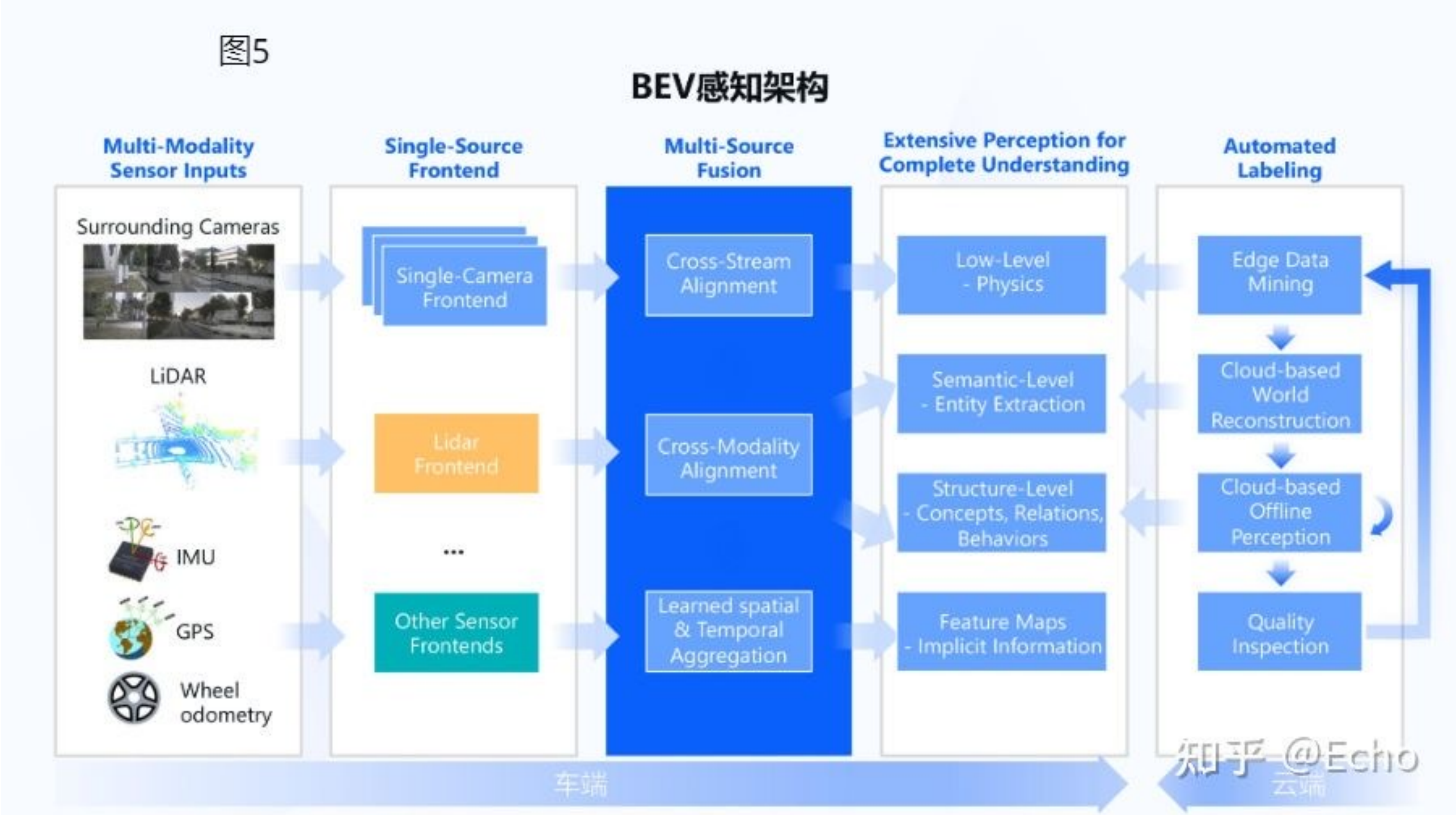


Figure 7: **Comparision of BEVFormer and BEVFormer-S on nuScenes val set.** We can observe that BEVFormer can detect highly occluded objects, and these objects are missed in the prediction results of BEVFormer-S (in red circle).

Bird’s-eye-view Perception

slide credit: 毫末

Bird's-eye-view Perception

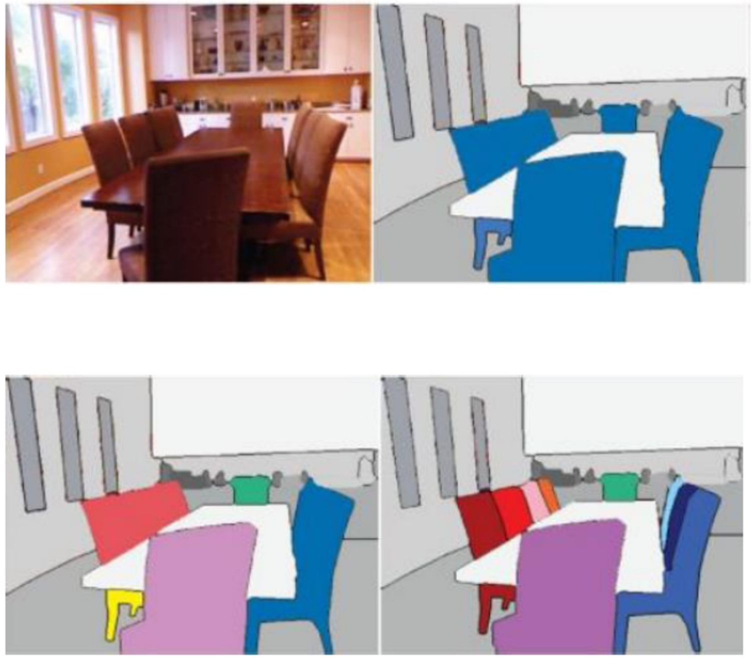


slide credit: 地平线

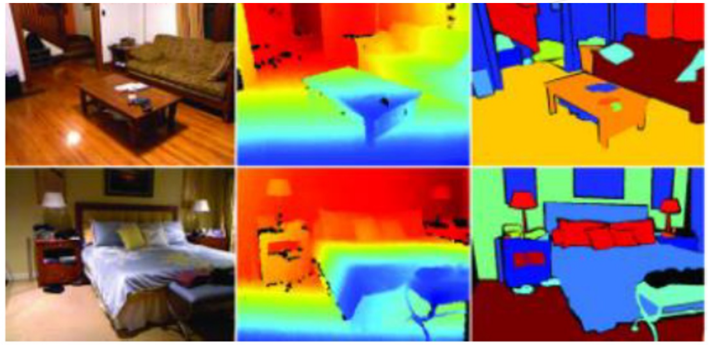
Outline

Image Segmentation: An Introduction

语义(semantic) or 实例



2D, 2.5D (depth est.) and 3D



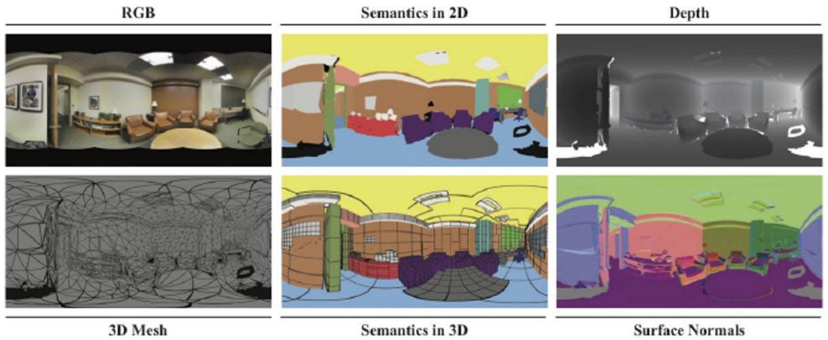


Image Segmentation – popular methods

Semantic segmentation

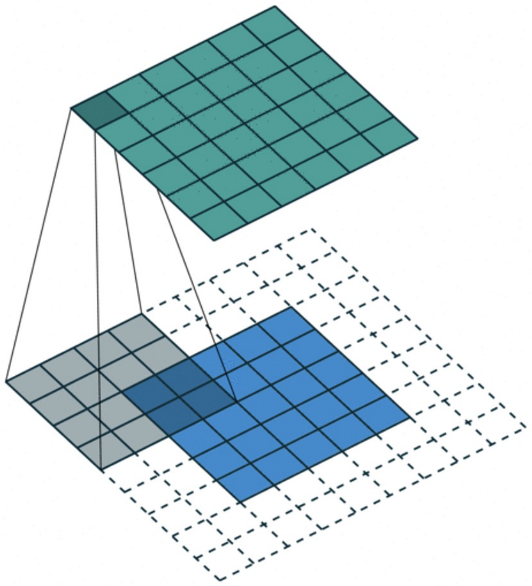
- **FCN**
- SegNet
- Dilation
- DeconvNet
- ENet (速度快)
- **Deeplab V1 V2 V3**
- ParseNet
- RefineNet
- Large Kernel Matters

Instance segmentation

- SDS
- DeepMask
- SharpMask
- MultiPathNet
- MNC
- **Mask-RCNN**

Prerequisite: convolution operation

卷积 (Convolution)



Class `torch.nn.Conv2d`*(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)*
https://pytorch.org/docs/stable/nn.html#torch.nn.Conv2d

输出大小是多少？

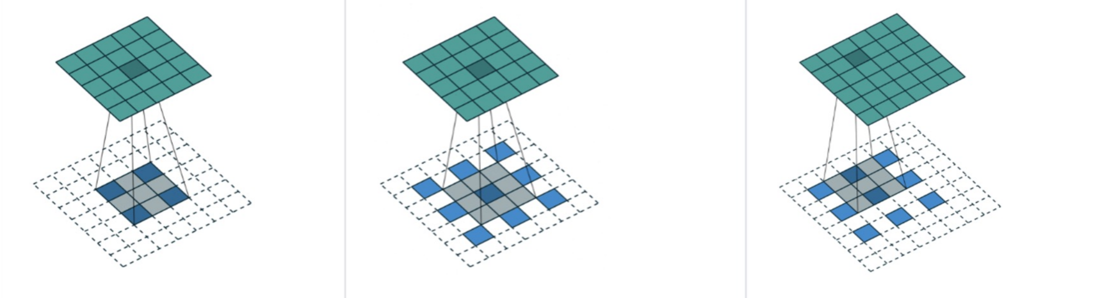
Output size = 20, 33, 25, 50

公式：(W - kernel + 2 pad) / stride + 1 向下取整

https://github.com/vdumoulin/conv_arithmetic

Prerequisite: convolution operation

反卷积 (Deconvolution) - upsample



No padding, strides, transposed	Padding, strides, transposed	Padding, strides, transposed (odd)

普通卷积

W_out = (W - kernel + 2 pad) / stride + 1

反函数

W = (W_out - 1) * stride - 2pad + kernel

反卷积公式

W_out = (W_in - 1) * stride - 2pad + kernel

蓝色的是输入feature map (较小)，绿色的是输出（较大）

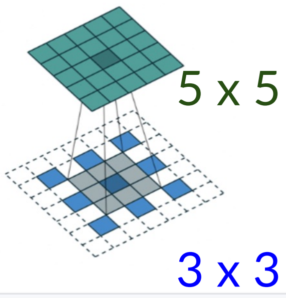
有stride 版本的反卷积是
先 **up-sample输入**(蓝色)，然后移动filter，正常卷积，得出结果

Prerequisite: convolution operation

反卷积 (Deconvolution) - upsample

$$W\_out = (W\_in - 1) * stride - 2pad + kernel$$

k=3, 问padding 和 stride是多少？



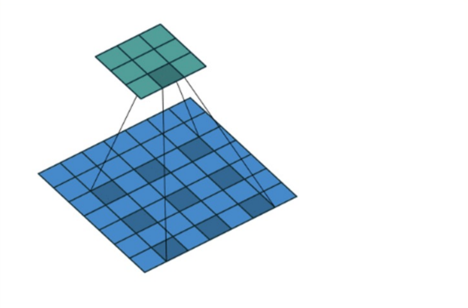
Padding, strides, transposed

k=3,
stride=2,
pad=1

Class **`torch.nn.ConvTranspose2d`** *(in_channels, out_channels, kernel_size, stride=1, padding=0, output_padding=0, groups=1, bias=True, dilation=1)*

Prerequisite: convolution operation

空洞卷积 (Dilated convolution) - 正常卷积(downsample)的一个细节



No padding, no stride, dilation

- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

默认值1, 即没有dilation

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \boxed{\text{dilation}[0]} \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

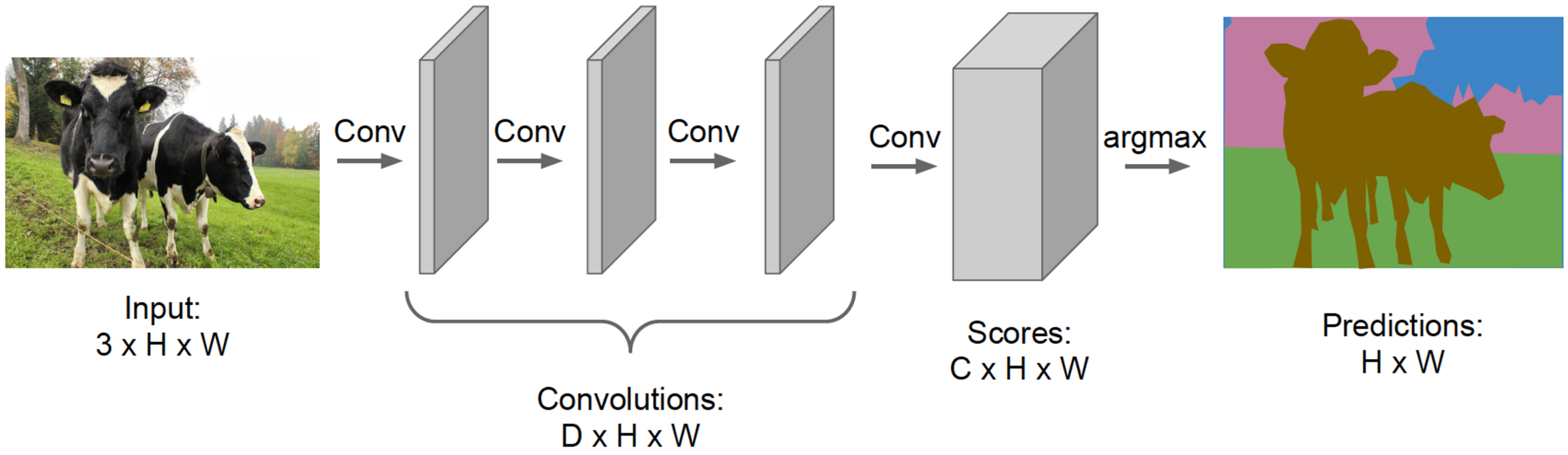
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

稀疏化filter - 扩大视野(receptive field)

注意：和反卷积不同！

Semantic Segmentation: a naïve approach

应用一堆卷积，保持feature map和原始图片一致。(channel=D)
最后一层卷积的channel个数是类别个数即可。(channel=C)

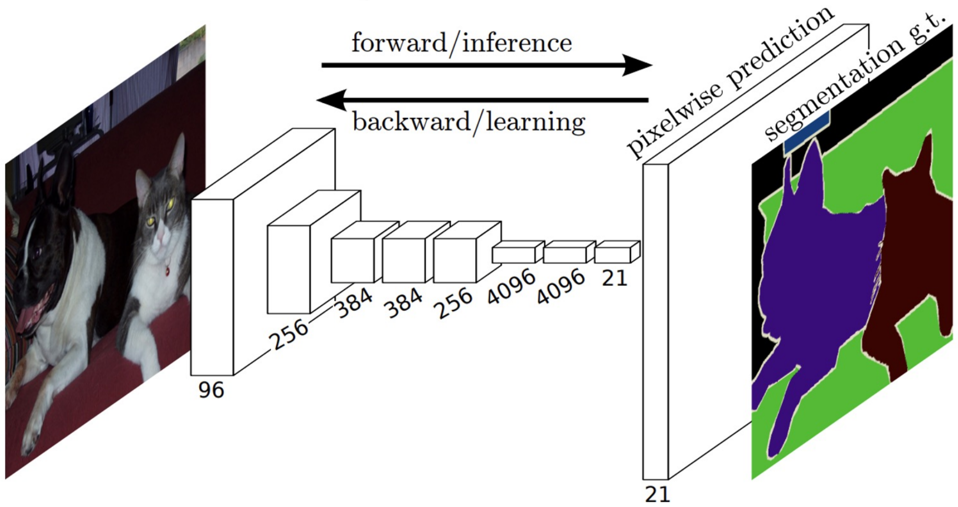


但很显然，计算量较大

Semantic Segmentation: FCN

This work is quite like the milestone of RCNN in detection

1. 训练问题：端到端学习
2. 连接层问题：全连接改为全卷积，支持可变输入
3. 特征图变小问题：利用反卷积向上放大特征图
4. 特征融合问题：利用skip connection融合多层特征提高上采样精细度

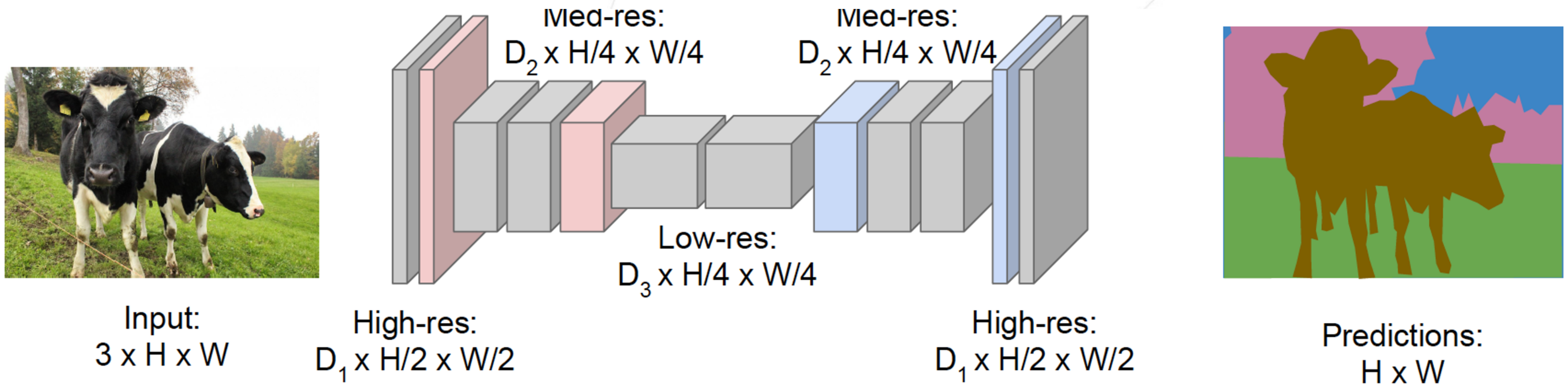


Long, Shelhamer, and Darrell, "**Fully Convolutional Networks** for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

This work is quite like the milestone of RCNN in detection

Downsampling:
Pooling, strided convolution

Up-sampling:
Unpooling or strided transpose convolution



Long, Shelhamer, and Darrell, "**Fully Convolutional Networks** for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation: DeepLab

V1: ICLR 2015

https://arxiv.org/pdf/1412.7062.pdf

Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs
Code: https://bitbucket.org/deeplab/deeplab-public/src/master/

V2: arXiv:1606.00915

DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
Code: https://bitbucket.org/aquariusjay/deeplab-public-ver2/src/master/

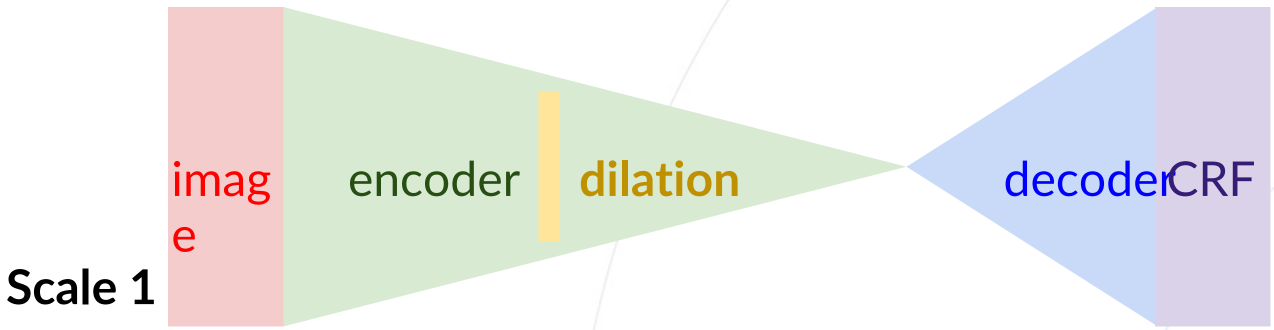
V3: Rethinking Atrous Convolution for Semantic Image Segmentation

https://arxiv.org/pdf/1706.05587.pdf

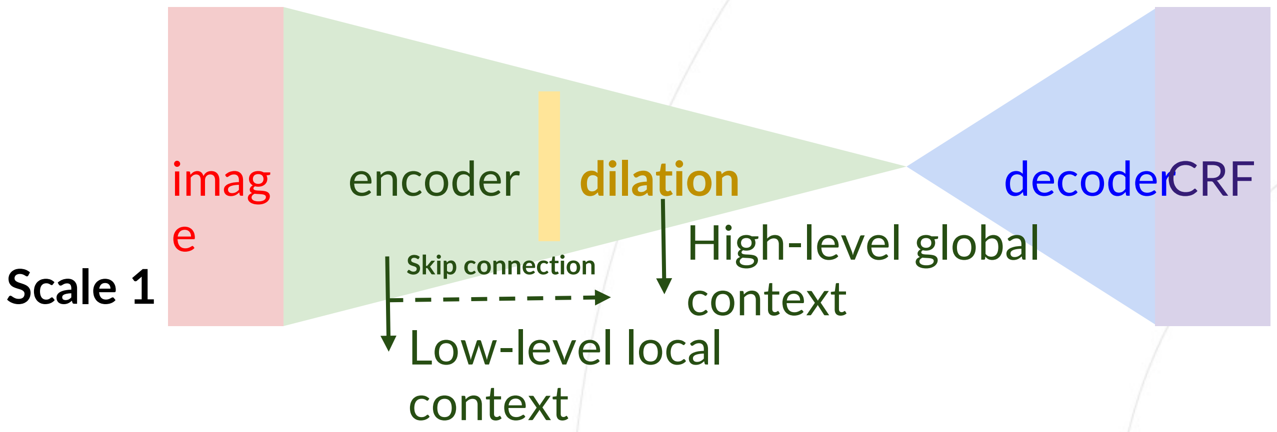
Website: http://liangchiehchen.com/projects/DeepLab.html

Blog: https://towardsdatascience.com/review-deeplabv3-atrous-convolution-semantic-segmentation-6d818bfd1d74

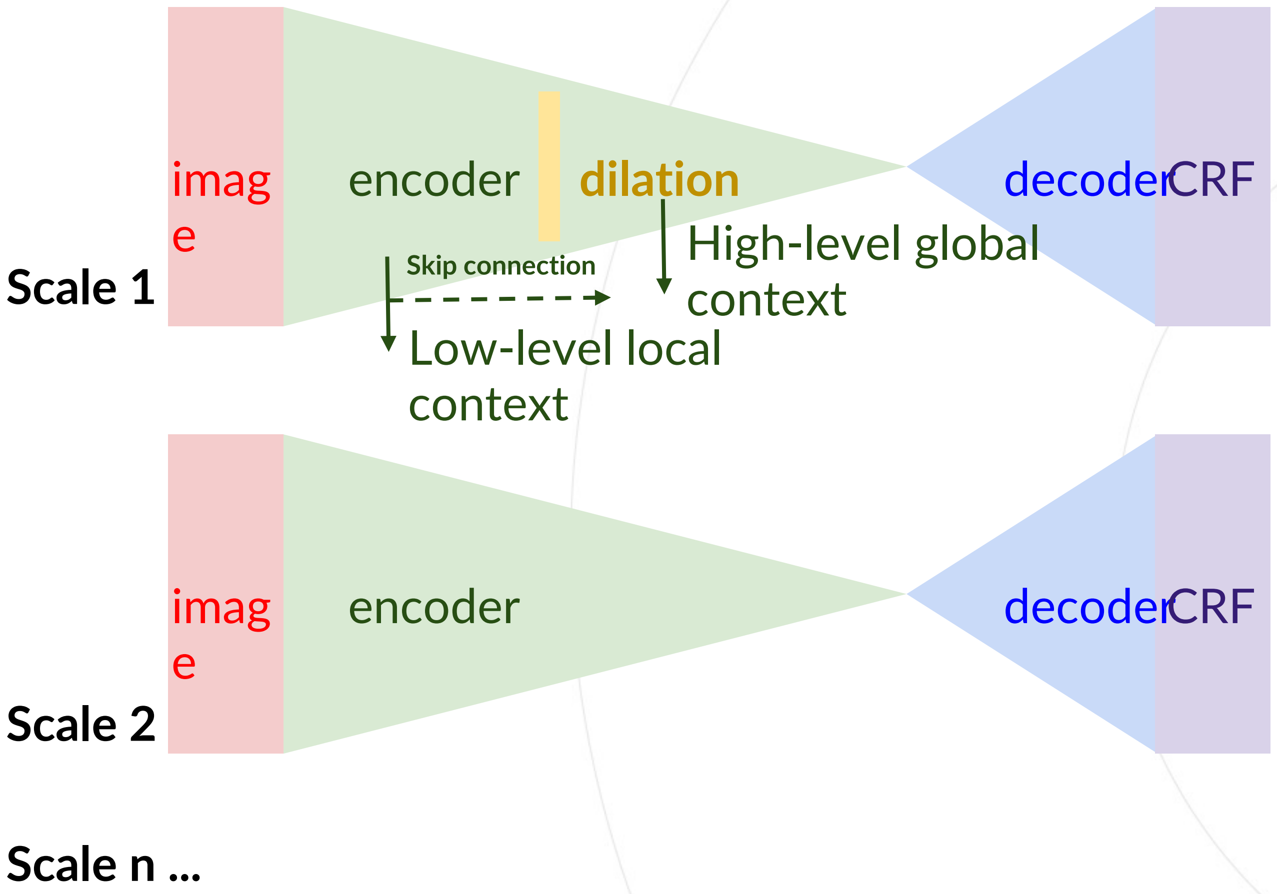
Semantic Segmentation: general pipeline



Semantic Segmentation: general pipeline



Semantic Segmentation: general pipeline



Semantic Segmentation: key issues

1. 上采样问题 SegNet, DeconvNet, SharpMask, RefineNet
    a. Encoder-decoder
    b. Deconvolution
    c. Unpooling
    d. Interpolation
2. 底层特征融合 U-net/hourglass structure in pose estimation
    a. Skip connect
    b. Refine block
    c. CRF

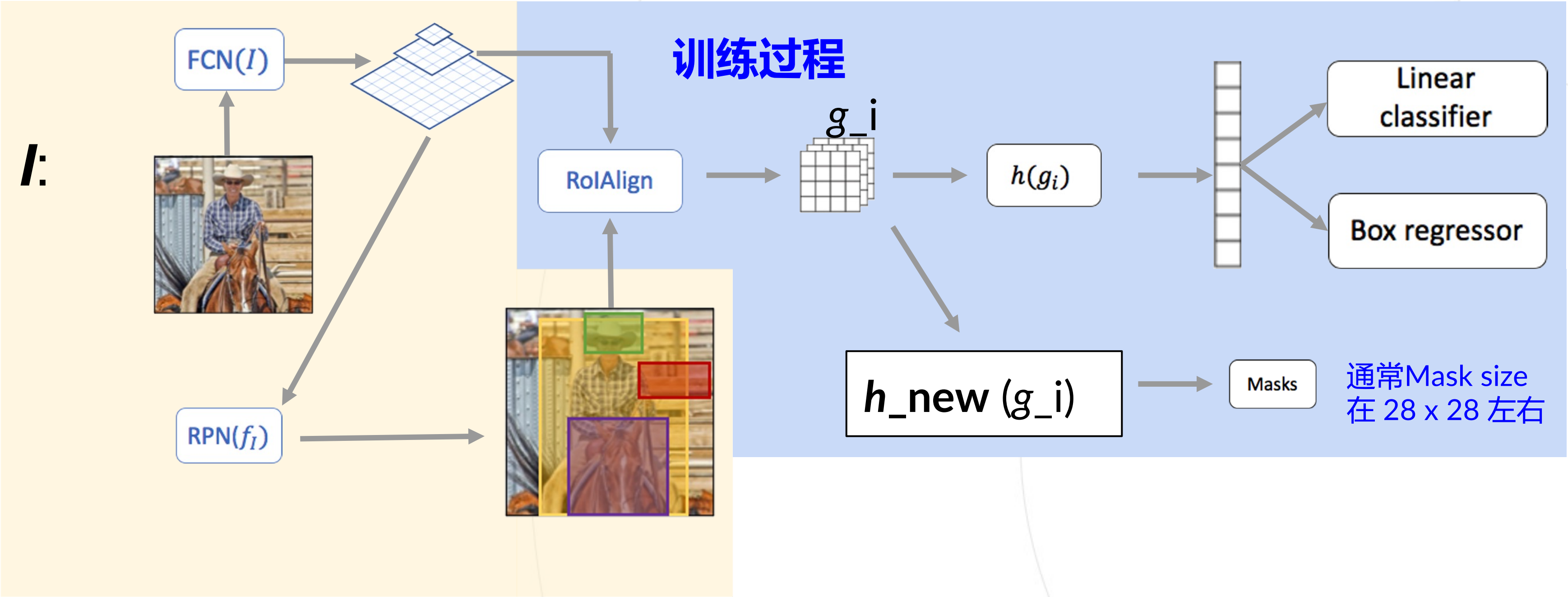
Deeplab, ParseNet, PSPNet/ICNet

3. Receptive field
    a. Dilation /hole
    b. Global pooling
4. 多尺度 Deeplab
    a. Multi-scale train/test
    b. high/low layer feature fusion
    c. Spatial pyramid pooling

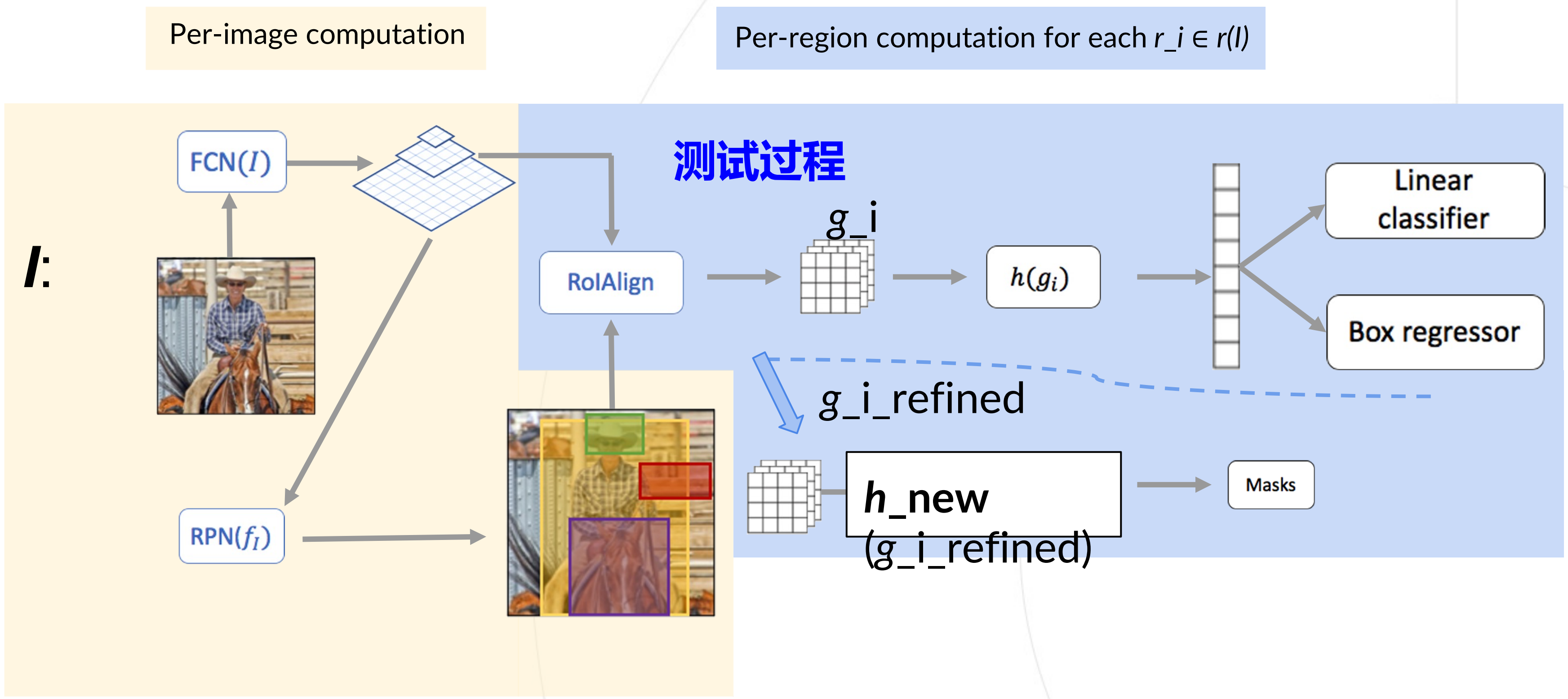
Instance Segmentation: Mask RCNN

Per-image computation

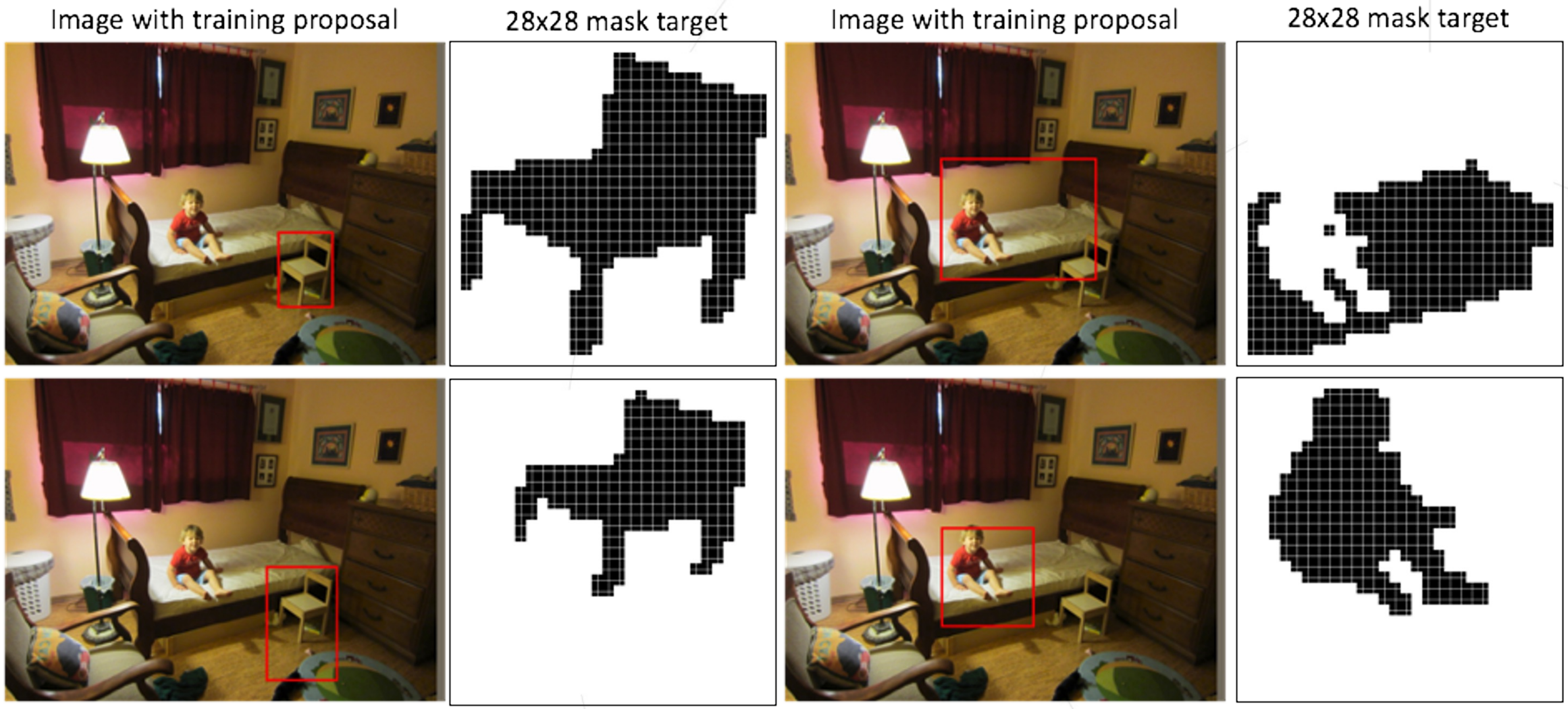
Per-region computation for each *r_i ∈ r(I)*



Instance Segmentation: Mask RCNN



Instance Segmentation: Mask RCNN - results



Instance Segmentation: Mask RCNN - results

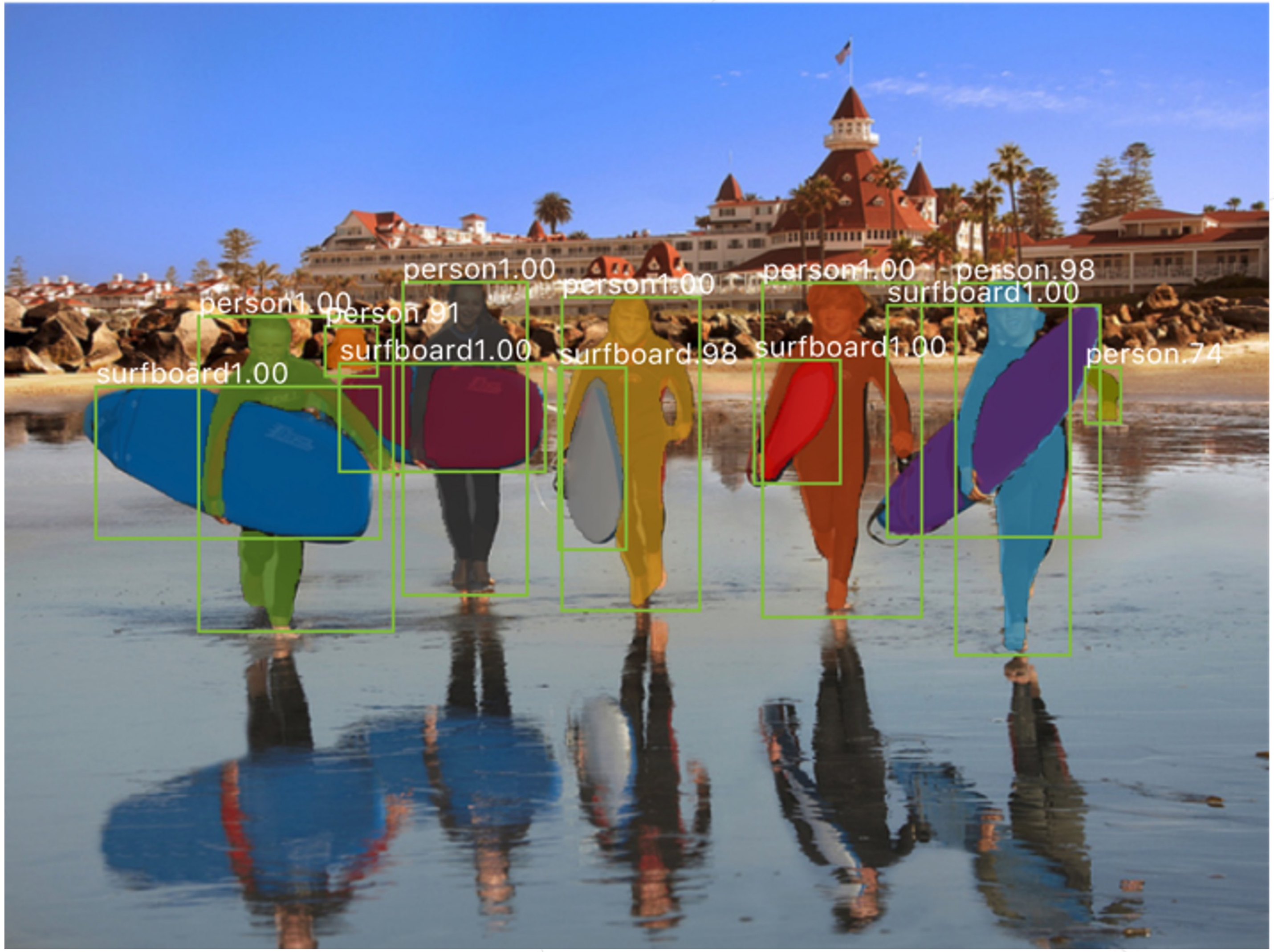


Image Segmentation: datasets

TABLE 1: Popular large-scale segmentation datasets.

Name and Reference	Purpose	Year	Classes	Data	Resolution	Sequence	Synthetic/Real	Samples (training)	Samples (validation)	Samples (test)
PASCAL VOC 2012 Segmentation [27]	Generic	2012	21	2D	Variable	✗	R	1464	1449	Private
PASCAL-Context [28]	Generic	2014	540 (59)	2D	Variable	✗	R	10103	N/A	9637
PASCAL-Part [29]	Generic-Part	2014	20	2D	Variable	✗	R	10103	N/A	9637
SBD [30]	Generic	2011	21	2D	Variable	✗	R	8498	2857	N/A
Microsoft COCO [31]	Generic	2014	+80	2D	Variable	✗	R	82783	40504	81434
SYNTHIA [32]	Urban (Driving)	2016	11	2D	960 × 720	✗	S	13407	N/A	N/A
Cityscapes (fine) [33]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	2975	500	1525
Cityscapes (coarse) [33]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	22973	500	N/A
CamVid [34]	Urban (Driving)	2009	32	2D	960 × 720	✓	R	701	N/A	N/A
CamVid-Sturgess [35]	Urban (Driving)	2009	11	2D	960 × 720	✓	R	367	100	233
KITTI-Layout [36] [37]	Urban/Driving	2012	3	2D	Variable	✗	R	323	N/A	N/A
KITTI-Ros [38]	Urban/Driving	2015	11	2D	Variable	✗	R	170	N/A	46
KITTI-Zhang [39]	Urban/Driving	2015	10	2D/3D	1226 × 370	✗	R	140	N/A	112
Stanford background [40]	Outdoor	2009	8	2D	320 × 240	✗	R	725	N/A	N/A
SiftFlow [41]	Outdoor	2011	33	2D	256 × 256	✗	R	2688	N/A	N/A
Youtube-Objects-Jain [42]	Objects	2014	10	2D	480 × 360	✓	R	10167	N/A	N/A
Adobe's Portrait Segmentation [26]	Portrait	2016	2	2D	600 × 800	✗	R	1500	300	N/A
MINC [43]	Materials	2015	23	2D	Variable	✗	R	7061	2500	5000
DAVIS [44] [45]	Generic	2016	4	2D	480p	✓	R	4219	2023	2180
NYUDv2 [46]	Indoor	2012	40	2.5D	480 × 640	✗	R	795	654	N/A
SUN3D [47]	Indoor	2013	–	2.5D	640 × 480	✓	R	19640	N/A	N/A
SUNRGBD [48]	Indoor	2015	37	2.5D	Variable	✗	R	2666	2619	5050
RGB-D Object Dataset [49]	Household objects	2011	51	2.5D	640 × 480	✓	R	207920	N/A	N/A
ShapeNet Part [50]	Object/Part	2016	16/50	3D	N/A	✗	S	31,963	N/A	N/A
Stanford 2D-3D-S [51]	Indoor	2017	13	2D/2.5D/3D	1080 × 1080	✓	R	70469	N/A	N/A
3D Mesh [52]	Object/Part	2009	19	3D	N/A	✗	S	380	N/A	N/A
Sydney Urban Objects Dataset [53]	Urban (Objects)	2013	26	3D	N/A	✗	R	41	N/A	N/A
Large-Scale Point Cloud Classification Benchmark [54]	Urban/Nature	2016	8	3D	N/A	✗	R	15	N/A	15

Pascal, COCO, Cityspace (cars and all),
KITTI

Image Segmentation: evaluation metrics

假设K+1类，下标从0到 k。pij 表示属于 i 类的样本被预测为 j 类。

Pixel Accuracy(PA) = (预测对的像素个数)/(总的像素个数)

$$PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k}\sum_{j=0}^{k} p_{ij}}$$

Mean Pixel Accuracy(MPA)= 平均每类的准确率

$$MPA = \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k} p_{ij}}$$

Mean IoU=平均每类的IOU

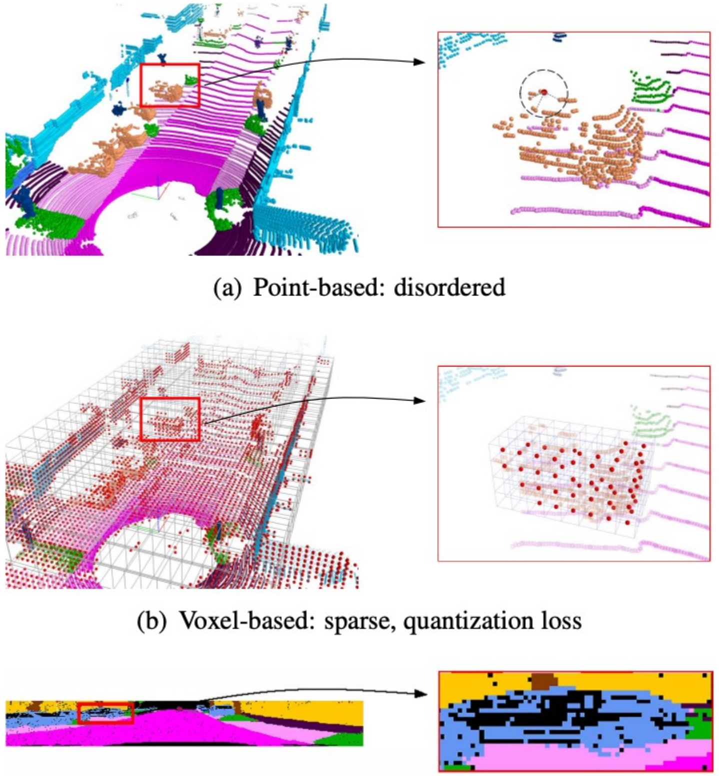
$$MIoU = \frac{1}{k+1}\sum_{i=0}^{k}\frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$

Frequency weighted IoU=加权后每类的IOU

$$FWIoU = \frac{1}{\sum_{i=0}^{k}\sum_{j=0}^{k} p_{ij}}\sum_{i=0}^{k}\frac{\sum_{j=0}^{k} p_{ij}p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$$

LiDAR Semantic Segmentation

http://www.semantic-kitti.org/tasks.html#semseg



(a) Point-based: disordered

(b) Voxel-based: sparse, quantization loss

(c) Range-based: physical dimensions distorted

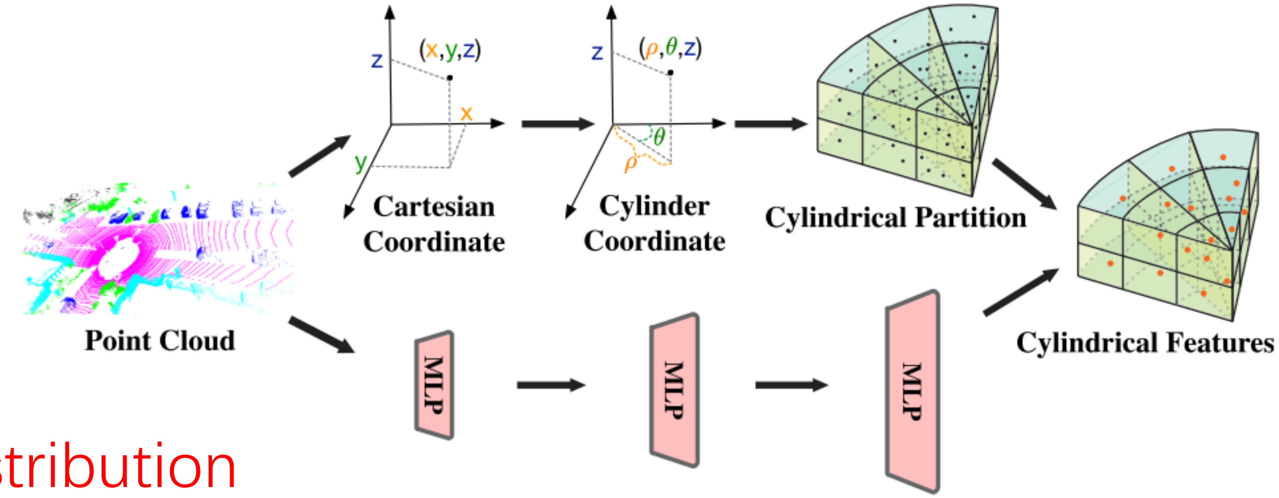
Leaderboard. Following leaderboard contains only published approaches, where we at least can provide an arXiv link. (Last updated: August 24, 2021.)

To avoid confusion between the numbers reported in the paper and post-publication results, we report here the numbers from the paper. Please contact us if we missed an updated version with different numbers.

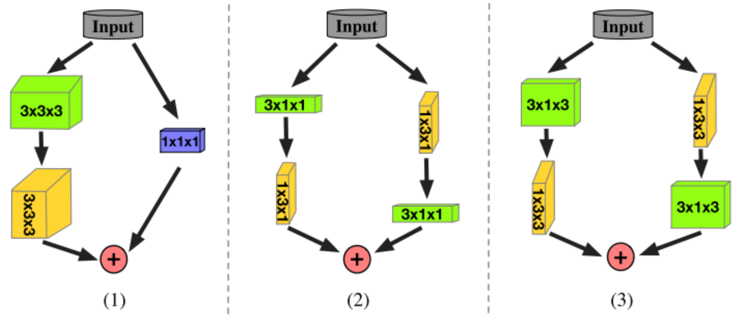
Single Scan | Multiple Scans

Approach	Paper	Code	mIoU	Classes (IoU)	Details
RPVNet			70.3		
AF2S3Net			69.7		
Cylinder3D			67.8		
SPVNAS			66.4		
JS3C-Net			66.0		
AMVNet			65.3		
Lite-HDSeg			63.8		
TORNADONet			63.1		
KPRNet			63.1		

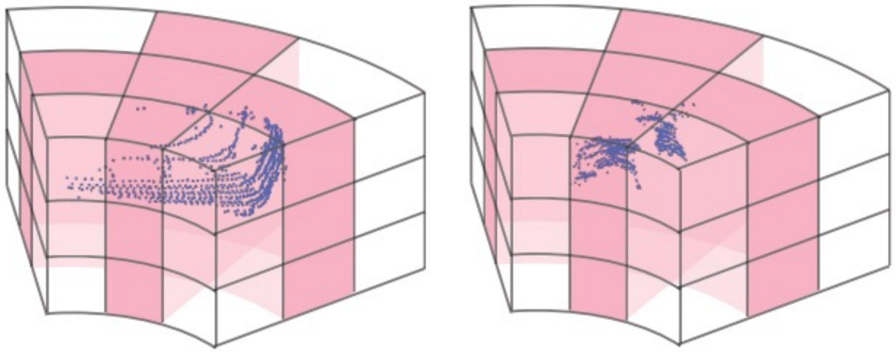
Cylinder3D



- Cylindrical Partition
  - varying-density, imbalanced distribution
  - cylinder coordinate
- Asymmetrical 3D Convolution Network
  - specific object shape distribution (cubic objects)
  - asymmetrical residual block (match ~)

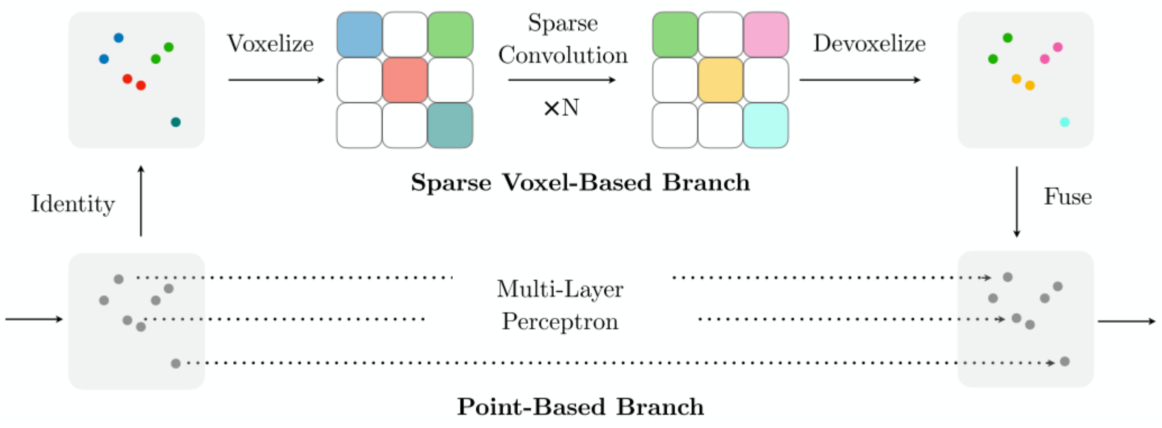


- Summary
  - outdoor LiDAR point cloud
  - distribution: point cloud & specific object
  - cylinder coordinate

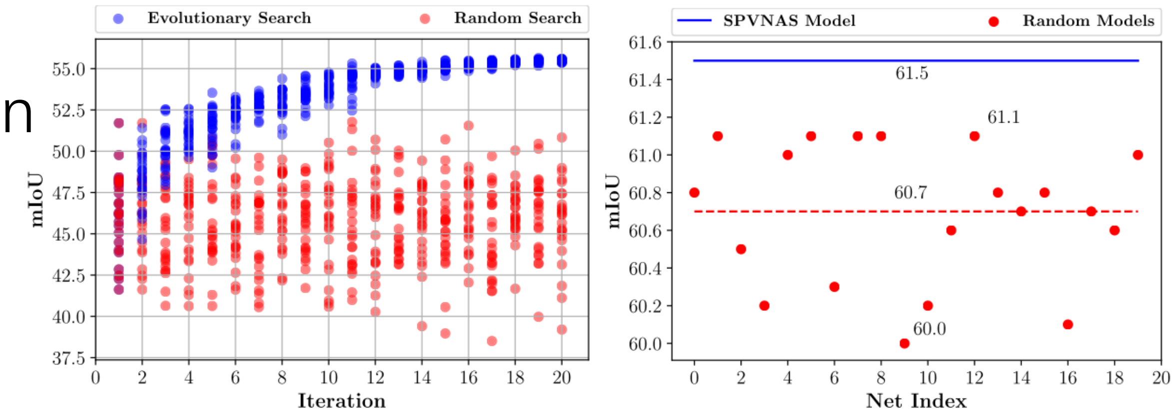


(a) Car (b) Motorcycle

SPVNAS



- Sparse Point-Voxel Convolution
  - Sparse Convolution cannot always keep high-resolution
  - Point-Voxel Convolution does not scale up to large 3D scenes
- 3D Neural Architecture Search
  - architecture search framework for 3D scene
  - improves the efficiency and performance of SPVCNN
- Summary
  - SPVNC: large scenes & high-resolution
  - NAS (evolutionary search)
  - lightweight, fast and powerful



(a) Search curves of ES and RS. (b) Comparison with random models.

END

Reach me at lihongyang@senseauto.com