20220628

一、Multi-level Domain Adaptation for Lane Detection

【来源】CVPR2022 Workshop of Autonomous Driving

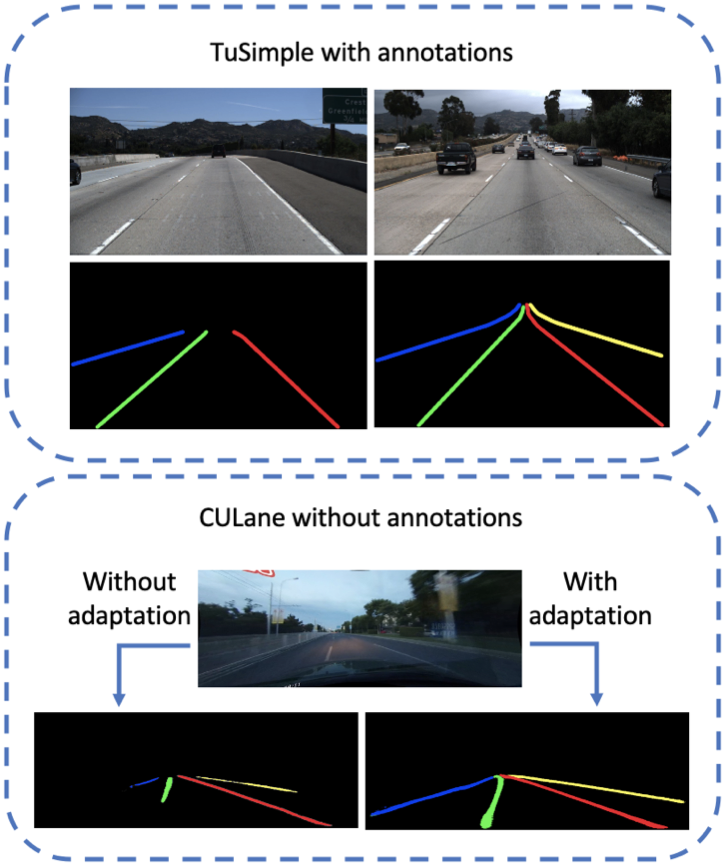
【arxiv】http://arxiv.org/abs/2206.10692

Multi-level Domain Adaptation for Lane Detection

Chenguang Li*[1], Boheng Zhang*†[1,2], Jia Shi†[1,3], Guangliang Cheng‡[1,4]
[1]SenseTime Research [2]Tsinghua University [3]Robotics Institute, Carnegie Mellon University
[4]Shanghai AI Laboratory
lichenguang@senseauto.com, zbh17@mails.tsinghua.edu.cn,
jiashi@andrew.cmu.edu, guangliangcheng2014@gmail.com

Motivation

不同车道线数据集之间有一定的gap，直接迁移会造成性能低下。如：TuSimple上训练好的车道线检测模型，直接迁移到CULane上漏报较多。



Method

采用基于**语义分割**的车道线检测范式和基于**伪标签**的UDA方法，分别提出pixel-wise、instance-wise和category-wise的domain adaptation，简称Multi-level domain adaptation （MLDA）

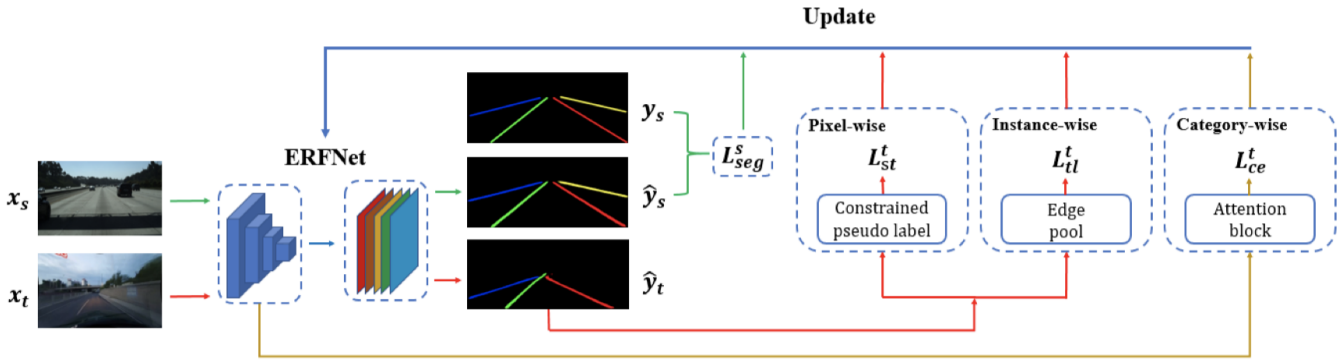


Figure 2. **Approach overview (MLDA).** We use ERFNet [40] as backbone. x_s and y_s refer to the input image and groundtruth in source domain respectively. x_t is the input image in target domain. $\hat{y}_s$ and $\hat{y}_t$ are segmentation results of x_s and x_t respectively.

1） Pixel–wise Adaptation

Target domain数据生成的伪标签中，绝大部分像素都是背景，前背景区域显著不平衡。提出**probability constrained self–training**，只选取低熵的pixel作为伪标签训练target domain：

$$y_t(i,j) = \begin{cases} argmax_C \hat{y}_t(i,j) & if\ \hat{y}_t(i,j,c) > \alpha_c \\ null & otherwise \end{cases} \quad (2)$$

2）Instance–wise Adaptation

利用Tripplet loss，在instance层面区分车道线特征和非车道线特征：

$$L^t_{tl} = \lambda_{tl} \frac{1}{M(M-1)N} \sum_i^M \sum_{\substack{j \\ j \neq i}}^M \sum_k^N \max\{0, \|f(x_i{}^a) - f(x_j{}^p)\|_2^2 - \|f(x_i{}^a) - f(x_k{}^n)\|_2^2 + \beta\} \quad (3)$$

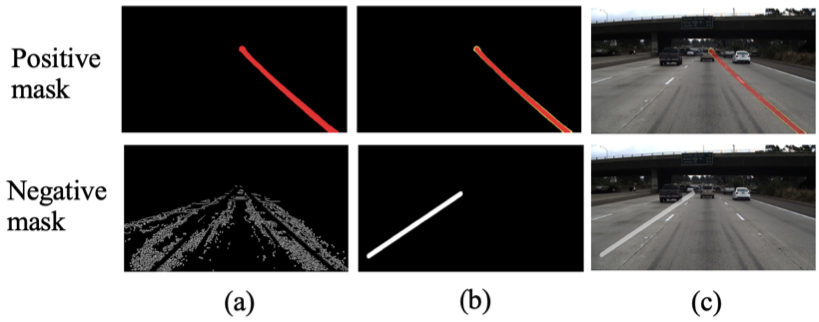
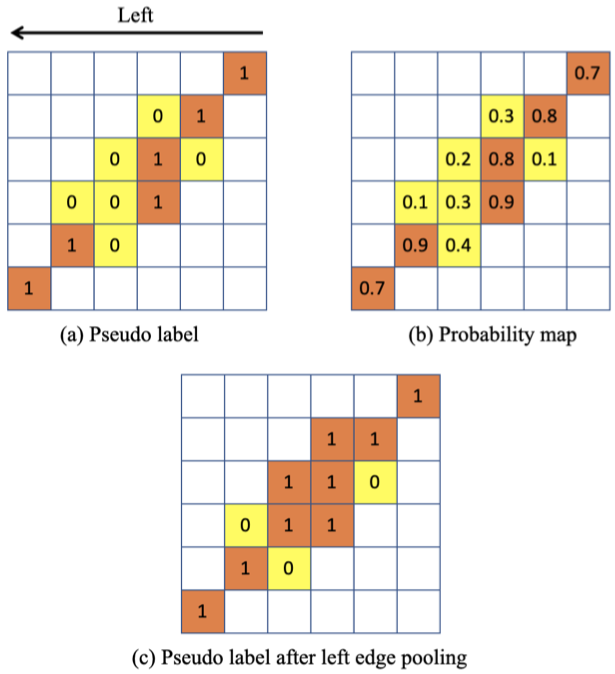


Figure 6. **Examples of positive instances and negative instances.** In the top row, from left to right are (a) pseudo labels, (b) expanded lane mask with edge pooling (yellow area) and (c) visualization on original image. In the bottom row, from left to right are ROI edges, negative mask and visualization.

同时提出edge pooling，补全伪标签中mask不完整的地方：



3）Category-wise Adaptation

基于语义分割的车道线检测方法对车道线的位置关系（category）有一定要求，设计损失函数来检测第i条车道线的存在与否：

$$z_t = \begin{cases} c & if\ f_s(\hat{z}_t(c)) > \eta \\ null & otherwise \end{cases} \tag{8}$$

$$L_{ce}^{t} = \lambda_{ce} \sum_{t \in T} CE(z_t, \hat{z}_t) \tag{9}$$

同时，利用spatial、channel attention来引入位置先验、减少错误分类：

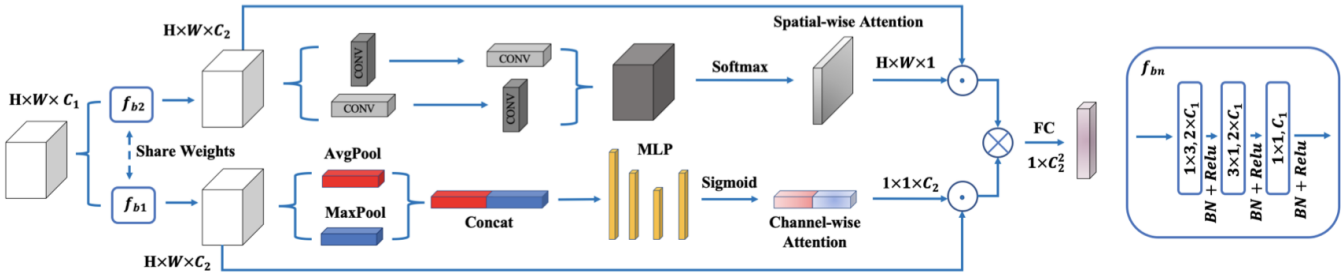


Figure 5. **Adaptive inter-domain embedding module.** The multi-layer perceptron (MLP) includes three layers (FC + Relu). Three FC layers squeeze feature maps to 1/2, 1/16 and 16 times respectively.

Experiment

分别将TuSimple和CuLane作为S/T：

Table 1. Quantitative comparison on "TuSimple to CULane". "Source only" denotes directly applying the model trained on TuSimple [45] to CULane [37] without adaptation. "Target only" denotes supervised training on target domain. For crossroad, it only shows FP.

Experiment Setting	Normal	Crowded	Night	No line	Shadow	Arrow	Dazzle light	Curve	Crossroad	Total
Source only	50.0	25.5	19.6	18.5	16.8	36.0	25.4	34.2	7405	30.5
Target only	91.9	71.0	69.1	46.6	74.2	87.3	65.6	69.7	2632	73.8
Advent [47]	49.3	24.7	20.5	18.4	16.4	34.4	26.1	34.9	6527	30.4
PyCDA [31]	41.8	19.9	13.6	15.1	13.7	27.8	18.2	29.6	**4422**	25.1
Maximum Squares [6]	50.5	27.2	20.8	19.0	20.4	40.1	27.4	38.8	10324	31.0
Ours (MLDA)	**61.4**	**36.3**	**27.4**	**21.3**	**23.4**	**49.1**	**30.3**	**43.4**	11386	**38.4**

Table 2. Quantitative comparison on "CULane to TuSimple". "Source only" denotes directly applying the model trained on CULane [37] to TuSimple [45] without adaptation.

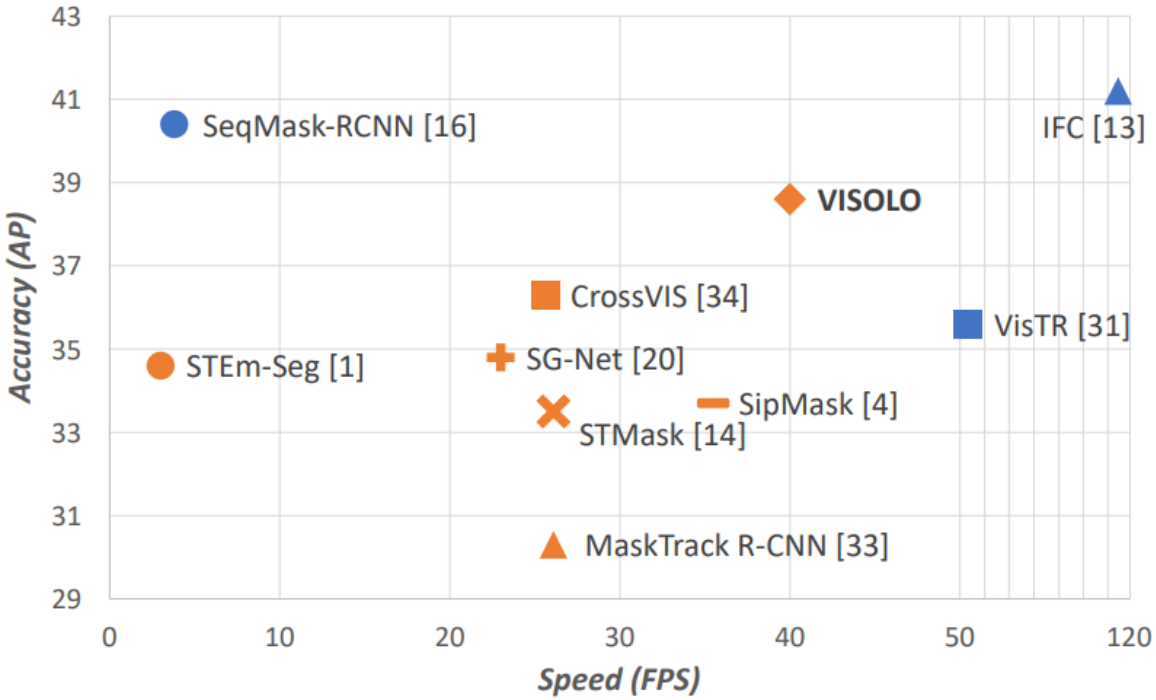
Experiment Setting	FP	FN	Accuracy
Source only	31.6	55.2	60.9
Target only	19.3	4.1	95.6
Advent [47]	39.7	43.9	77.1
PyCDA [31]	51.9	45.1	80.9
Maximum Squares [6]	38.2	42.8	76.0
Ours (MLDA)	**29.5**	**18.4**	**89.7**

Table 3. Ablation study of MLDA on "CULane to TuSimple".

	PL	IL	CL	FP	FN	Accuracy
Source only				31.6	55.2	60.9
	√			40.7	34.8	84.6
	√	√		29.7	25.0	86.9
	√	√	√	**29.5**	**18.4**	**89.7**

二、VISOLO: Grid-Based Space-Time Aggregation for Efficient Online Video Instance Segmentation

[paper] [code]



YouTube-VIS2019数据集，橙色online，蓝色offline

模型是单阶段、online，基于网格结构的特征表示，网格结构特征允许我们利用FCN来实时处理，也容易和不同组件共享特征

相关工作

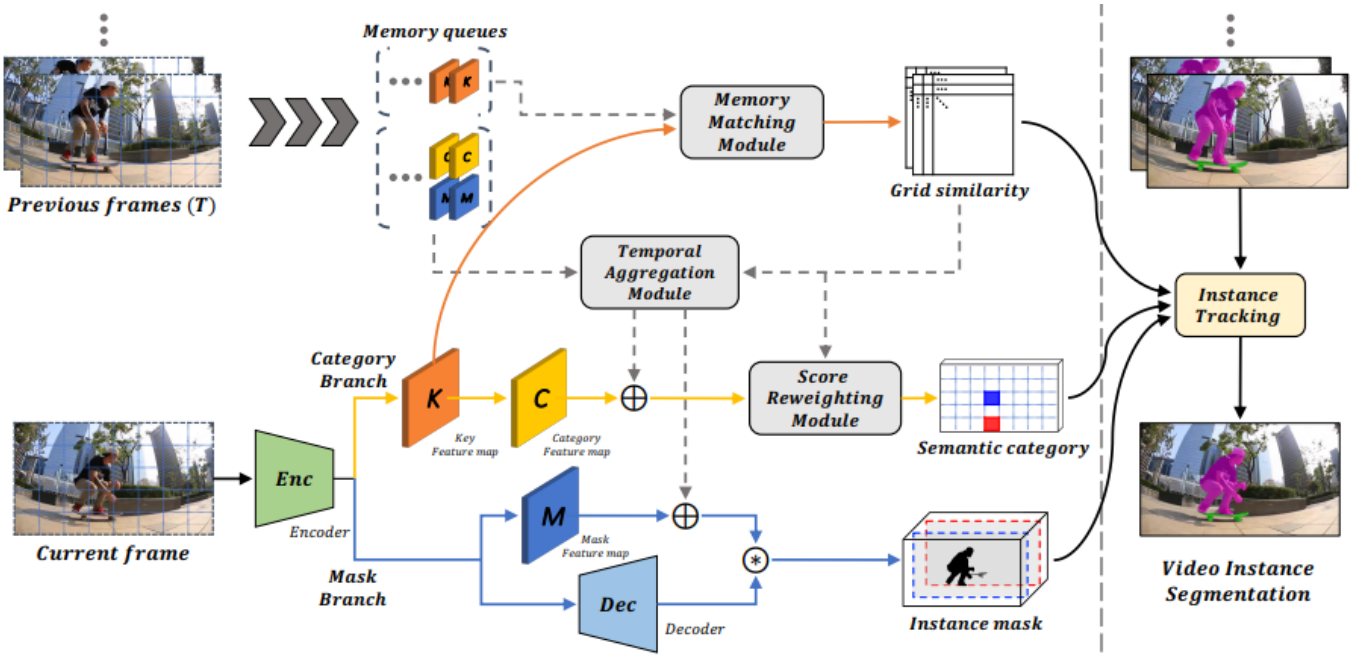
概述

2 利用掩码传播进行VIS 13 IFC 16 SeqMask RCNN 31 VisTR	通过mask propagation和TR。表现好，offline不能实时，因为它要求在预测前处理整个视频
online算法，以下都是	很多执行帧-level分类分割，没有完全利用先前帧的信息，对于遮挡等不好
33 masktrack RCNN	使用mask RCNN的分类、分割结果，仅仅用前一帧信息用于tracking
4 SipMask 20 SG-net	使用先前帧的时间线索改善图像级分割
34 CrossVIS	在训练时使用时间信息加强特征，推断和分类时没用
14 STMask	使用前一帧的信息，但是只使用一个相邻帧for分割
9 CompFeat	提出时空注意力模块，聚合信息for分割和分类with非局部操作，计算量大，基于Mask RCNN的两阶段，还需要额外的encoder来获取key和value，像24 STM
我们的VISOLO	利用以往帧信息，不仅for跟踪，还for分类、分割 基于SOLO单阶段图像实例分割
29,30 SOLO	将输入图像划分为统一的网格，每个网格输出语义类别分数和实例mask

汇总

图像实例分割	7 8 10 12 15 21	两阶段方法，首先用RPN生成目标候选，然后用聚合的RoI特征执行box回归、分类、分割
	3 6 26 29 30 32	不用候选生成，用全卷积网络结构直接预测bbox和mask
	VIS: 2 9 16 33	大多数VIS通过添加tracking头或者mask传播头扩展实例分割（在mask RCNN基础上）
VIS	offline	用mask传播和TR， maskRrop通过传播实例mask产生多个重叠clip，然后clip级track被聚合来为整个视频产生instance序列 seqMask RCNN通过传播多个关键帧的实例mask为整个视频产生实例序列，减少冗余序列 最近基于TR的如VisTR和IFC拓展了DETR
	online	masktrack RCNN添加tracking分支，把实例标签赋给每个候选box sipMask提出spatial preservation（sp）改善mask预测表现，也添加分支产生tracking特征图，然后用那些特征图with和masktrack RCNN相似的指标匹配帧的实例 SG-Net动态划分一个目标实例为子区域，在每个区域上执行分割，添加tracking头，track实例的中心 CrossVIS提出crossover学习策略，用当前帧的实例特征，以像素级定位其他帧中相同实例 STMask提出spatial calibration校准来获得更多anchor box的精确空间特征，也添加了时间混合模块，获得时间相关性between相邻帧，以推断实例mask和tracking CompFeat提出时空注意力模块，聚合时间特征来获得分割和分类结果with非局部matching，也提出了基于相关性的tracking模块，产生空间likelihood和目标相似性for tracking

pipeline



VISOLO的架构，backbone：ResNet50

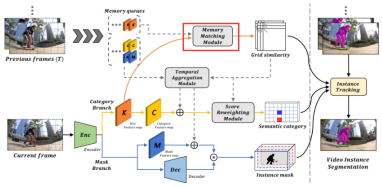
两个分支：category分支、mask分支

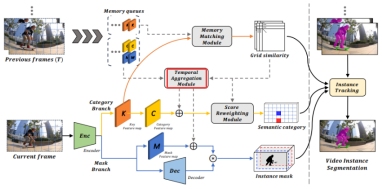
三个模块：memory match模块、temporal聚合模块、score reweighting模块

来自2分支的特征图key的K，category的C，mask的M，被存储在memory queues

虚线箭头表示利用先前帧的信息，+为元素及加和，*为卷积

网格状特征表示		摆脱中间阶段像RPN、ROI–Align，用FCN就可以，速度快 更容易管理存储网格结构中的特征，允许添加额外模块，多子任务共享特征，提升整体性能					
memoeyqueu		存储K C M特征图					

es							
memory matching 模块	希望能找到和*当前帧实例对应的网格*相同的*先前帧的网格*，匹配这些网格的实例来执行tracking 通过比较帧间key feature K来预测**网格相似性Sim**（被用来tracing跨帧	通过迭代pair-wise比较预测当前帧和先前帧的网格之间的Sim 输入是两个key特征图，分别经过卷积层，然后通过矩阵内积聚合来产生Sim  					

	实例、从先前帧中聚集信息）						
时序聚合模块	C和M特征用网格相似性聚合，给两个分支提供先前帧的丰富信息（和STM类似，用soft权重从先前帧中取回外观信息，但STM	对每个grid，使用Sim聚集过去帧的外观信息： 输入是memory队列 $(\mathbf{C}_T \& \mathbf{M}_T \in \mathbb{R}^{T\times S_h\times S_w\times E})$，用卷积层操作reshape到 $\mathbb{R}^{(T\cdot S_h\cdot S_w)\times E}$，然后通过加权聚合这些特征， 权重 $(\mathbf{W}_T \in \mathbb{R}^{(S_h\cdot S_w)\times(T\cdot S_h\cdot S_w)})$ 通过在Sim上用softmax函数计算 	输出 $\mathbf{C}_A = \mathbf{W}_T \otimes \mathbf{C}_T,$ $\mathbf{M}_A = \mathbf{W}_T \otimes \mathbf{M}_T$ x表示矩阵内积，这两个特征被add with 当前帧的category和mask特征图				

	需要重新编码每个额外的memory帧通过resnet编码器来获得value特征，我们更高效，重用分支的特征）						
score reweighting模块	动态调整校准输出的**score map**来加强类别branch的**分类表现**。	输入：也就是用Sim加权之后的category分支输出为 $\mathbf{Cat} \in \mathbb{R}^{S_h \times S_w \times C}$ 用两个Sim，当前帧和**两个先前帧**的Sim1，Sim2 每个Sim矩阵 $(\mathbf{Sim} \in \mathbb{R}^{(S_h \cdot S_w) \times (S_h \cdot S_w)})$ 每行最大值，shape到 $S_h \times S_w$ Cat的每个网格被如下乘法得到最后**分类分数**： $\mathbf{P} = \mathbf{Cat} \odot \text{AVG}(\tilde{\mathbf{Sim}}_1, \tilde{\mathbf{Sim}}_2)$						

	也用到网格相似性	，表示每个通道的元素级乘法					

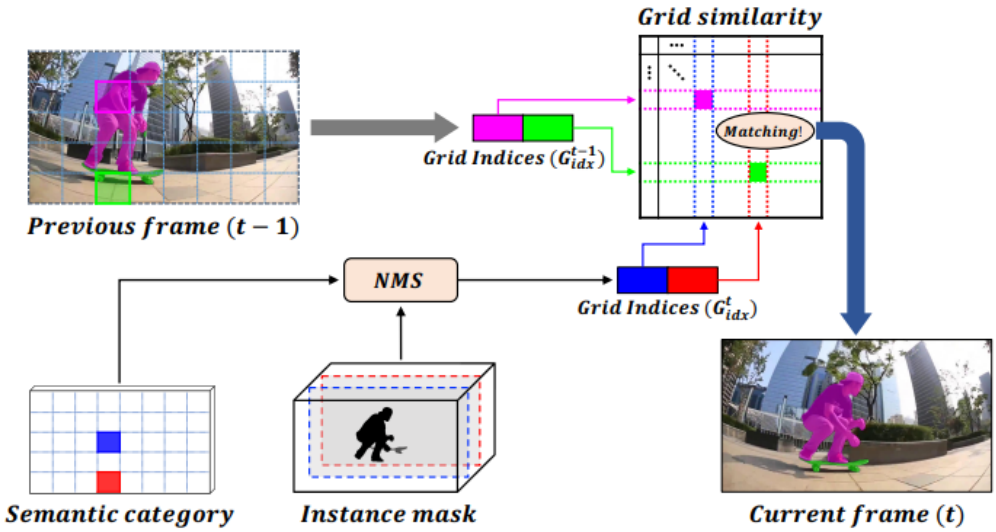
我们框架中组件和模块紧密耦合，从先前帧到当前帧的信息流仅增加了边际开销，因为它利用memory matching模块已经算出来的网格结构特征和网格相似性。

SR	TA (Category)	TA (Mask)	AP	AP_{50}	AP_{75}
			34.6	51.5	36.8
✓			35.6	53.8	37.9
	✓		36.4	54.4	39.3
✓	✓		37.7	56.6	40.3
✓	✓	✓	38.6	56.3	43.7

分数重加权模块 时序聚合模块

Table 3. Ablation study of the **S**core **R**eweighting module (SR) and the **T**emporal **A**ggregation module (TA), estimated on YouTube-VIS 2019 dataset.

instance tracking



> 实例分割模块，当前帧和上一帧，Gidx是包含实例中心的grid索引

在当前帧t，首先对mask分支和category分支的输出用矩阵NMS，获得**包含实例中心的网格索引** G_{idx}^{t}

比较当前帧 G_{idx}^{t} 和前一帧 G_{idx}^{t-1} 的Sim值来实现tracking：

> 当前实例被连接到先前的实例with最高相似性值，若两个Gidx的相似性值都低于阈值0.1，就进入**下一个先前帧**，并使用该帧的Sim来查看当前instance是否和该帧中任何实例匹配。
>
> 如果都不匹配，声明为新instance
>
> （我们保存用来计算网格相似性的特征图，以及没能匹配的实例 G_{idx}^{fail} ，解决遮挡等问题）

训练和推断

$$\mathbf{L} = \mathbf{L}_{class} + \lambda \mathbf{L}_{mask} + \mathbf{L}_{grid},$$

端到端训练，类别和网格相似性损失用Focal loss，mask loss用Dice Loss

Lmask和Lgrid只在GT物体存在的时候计算，λ=3

推断中，使用更多帧信息更丰富，但是写入所有帧的特征低效，默认总保存前两帧，对于中间帧，我们每5个帧保存特征，对于输入视频的第一个帧，放到内存用它

实验

resnet50，youtube-vis 19和21，在验证集上衡量，分别302,421个视频

我们测量了平均视频平均精度(AP)、IoU阈值为50%和75%的视频平均精度（AP50、AP75）、每个视频1和10个实例的平均召回（AR1、AR10）和每秒帧数（FPS）。

首先在COCO预训练，将网格的数量设为输入视频宽高比 $(S = (12, 21))$

	Methods	Backbone	FPS	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
Offline	MaskProp [2]	ResNet-50	–	40.0	–	42.9	–	–
	SeqMask-RCNN [16]	ResNet-50	3.8	40.4	63.0	43.8	41.1	49.7
	VisTR [31]	ResNet-50	51.1	35.6	56.8	37.0	35.2	40.2
	IFC [13]	ResNet-50	107.1	41.2	65.1	44.6	42.3	49.6
Near Online	STEm-Seg [1]	ResNet-101	3.0	34.6	55.8	37.9	34.4	41.6
Online	MaskTrack-RCNN [33]	ResNet-50	26.1	30.3	51.1	32.6	31.0	35.5
	SipMask [4]	ResNet-50	35.5	33.7	54.1	35.8	35.4	40.1
	SG-Net [20]	ResNet-50	23.0*	34.8	56.1	36.8	35.8	40.8
	SG-Net [20]	ResNet-101	19.8*	36.3	57.1	39.6	35.9	43.0
	CompFeat [9]	ResNet-50	–	35.3	56.0	38.6	33.1	40.3
	CrossVIS [34]	ResNet-50	25.6	36.3	56.8	38.9	35.6	40.7
	CrossVIS [34]	ResNet-101	23.3	36.6	57.3	39.7	36.0	42.0
	STMask [14]	ResNet-50†	26.1	33.5	52.1	36.9	31.1	39.2
	STMask [14]	ResNet-101‡	22.4	36.8	56.8	38.0	34.8	41.8
	Our VISOLO	ResNet-50	40.0	38.6	56.3	43.7	35.7	42.5

Table 1. Quantitative evaluation on **YouTube-VIS 2019** [33] validation set. [20] does not provide official checkpoints, so we infer the speed reported in [20] (FPS with superscript "∗"). "†" and "‡" indicate the ResNet-50-DCN and ResNet-101-DCN, respectively.

Methods	AP	AP_{50}	AP_{75}	AR_1	AR_{10}
MaskTrack-RCNN	28.6	48.9	29.6	26.5	33.8
SipMask	31.7	52.5	34.0	30.8	37.8
CrossVIS	34.2	54.4	37.9	30.4	38.2
STMask	30.6	49.4	32.0	26.4	36.0
Our VISOLO	36.9	54.7	40.2	30.6	40.9

Table 2. Quantitative evaluation on **YouTube-VIS 2021** validation set. We refer the results reported in [34]. All models use the ResNet-50 [11] as the backbone network, except for STMask [14] that uses ResNet-50-DCN.

Memory frames	FPS	AP	AP_{50}	AP_{75}
2 frames	40.4	36.7	54.2	40.4
10 frames	39.5	37.5	55.3	41.3
20 frames	38.7	37.7	55.4	41.4
Every 5 frames	40.0	38.6	56.3	43.7

Table 4. The number of reference frames for temporal aggregation module analysis on the validation sets of YouTube-VIS 2019 dataset [33]. We compare results by different memory storing rules.

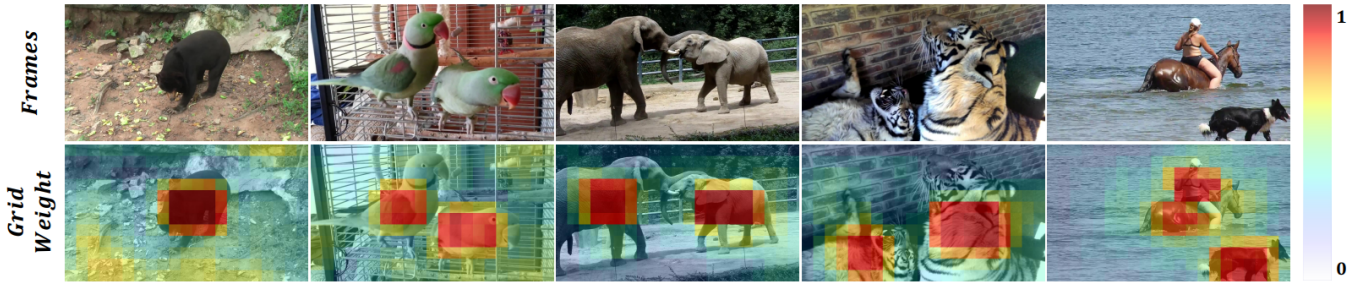


Figure 6. Visualization of weights for each grid in the score reweighting module at the second row. The first row shows the original frames.

评分重加权模块的可视化

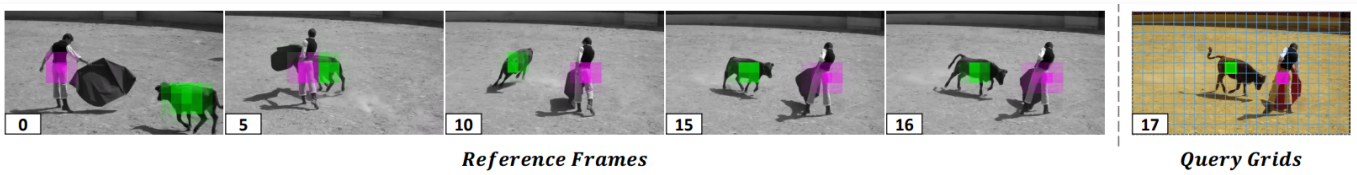


Figure 7. Visualization of our temporal aggregation module operation. We first compute the grid similarities between query grids and all grids of reference frames, and obtain the soft weight by a softmax operation. Then, we visualize the normalized soft weights of the reference frames. The query grids and weights of each grid of reference frames with respect to the query grids are assigned with different colors.

我们的时间聚合模块操作的可视化。我们首先计算查询网格与参考帧的所有网格之间的网格相似性，并通过softmax运算获得软权重。然后，我们可视化参考帧的归一化软权重。参考帧的每个网格相对于查询网格的查询网格和权重被指定为不同的颜色

局限之处

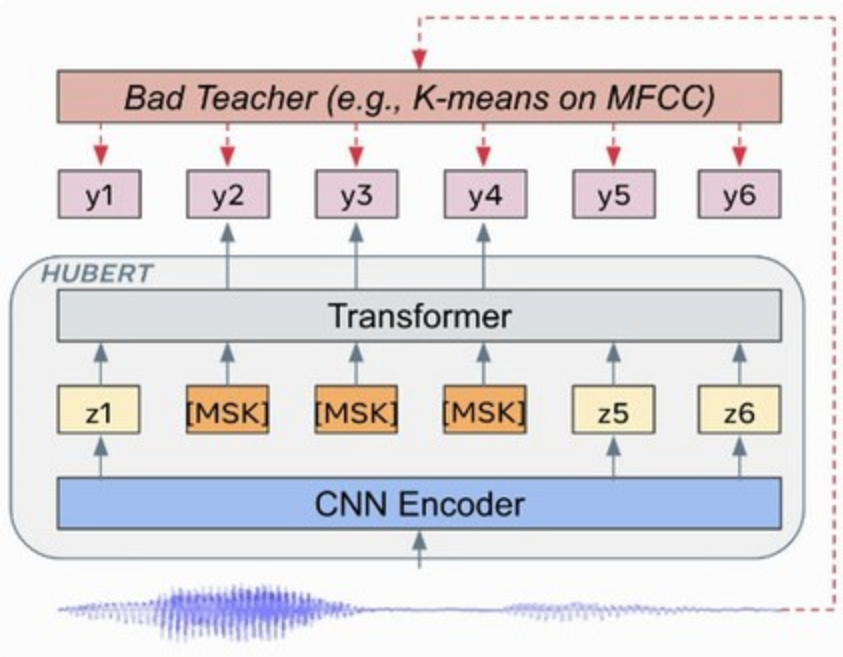
- 相同外观的新实例容易被混淆身份

- 难以检测小物体，SOLO用FPN，在不同尺寸设置不同数量的网格，考虑到VIS的特点，只用单一尺度改善效率
- 高度依赖网格相似性，如果存在问题，性能可能下降

三、HUBERT & AV-HUBERT

- 《Hubert: How Much Can a Bad Teacher Benefit ASR Pre-Training?》HuBERT
- 《Learning Audio-Visual Speech Representation by Masked Multimodal Cluster Prediction》Audio-Video HuBERT

HUBERT



与 wav2vec 的在线聚类方式不同，HUBERT 获取 target 采取离线聚类的方式，具体做法为：首先在 39 维的 MFCC 特征上进行 K-means 与 GMM 聚类，聚类中心数为 {50, 100, 500}。聚类之后可以获取每一帧语音的聚类中心作为 target。

每一帧语音获取 target 后，进行与 wav2vec 2.0 类似的预训练，然后在 mask 位置与未被 mask 的位置计算损失函数。

预训练完成后，已经有比较好的预训练模型，可以将预训练模型作为 teacher 生成表征，第二次进行聚类，从而发现更有意义的聚类单元。

Audio–Video HUBERT

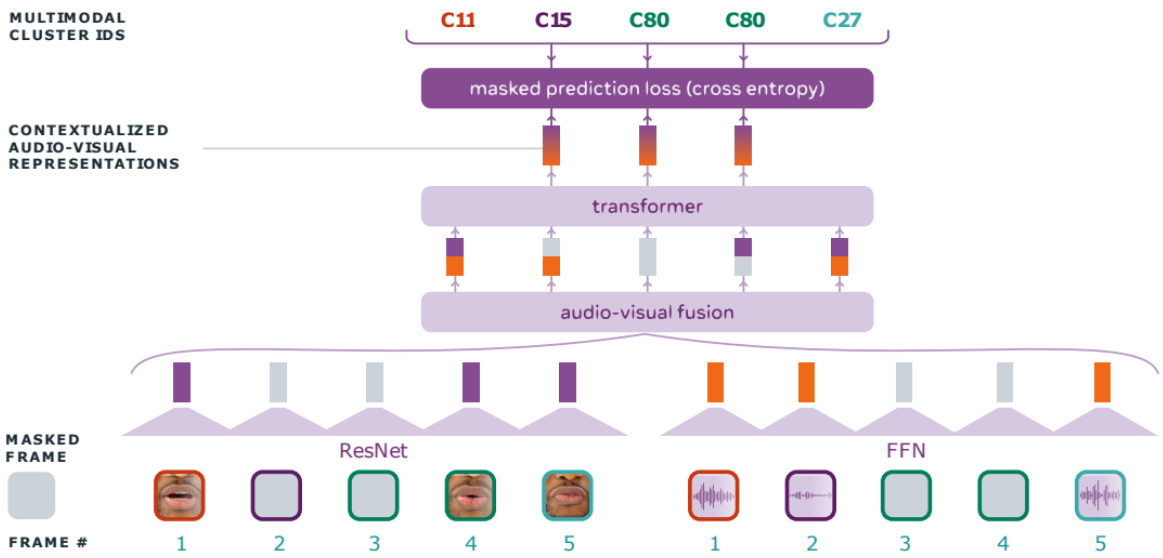


Figure 1: Illustration of AV-HuBERT. Masked prediction losses are only computed for the three middle frames, because at least one modality is masked for those frames. See section A for its comparison between single-modal and cross-modal visual HuBERT.

- 跨模态信息被分别 encode，融合后输入 transformer backbone，最后进行聚类获得标签
- 模态 dropout：在进行识别时，语音模态通常占据主导地位。为了平衡两个模态的关系，聚类标签先由语音模态取得，但为了防止模型对语音的过度依赖，对语音只使用线性层进行编码，使模型只能学习语音的简单特征。
- 在进行 dropout 时，以 p_m 的概率两个模态都使用；如果不是两个模态都使用，以 p_a 的概率使用语音模态：

$$\mathbf{f}_t^{av} = \begin{cases} \texttt{concat}(\mathbf{f}_t^a, \mathbf{f}_t^v) & \text{with } p_m \\ \texttt{concat}(\mathbf{f}_t^a, \mathbf{0}) & \text{with } (1-p_m)p_a \\ \texttt{concat}(\mathbf{0}, \mathbf{f}_t^v) & \text{with } (1-p_m)(1-p_a) \end{cases}$$

- 在微调与 inference 阶段，模型只输入视觉信息。这样，无论向模型输入哪一个模态的信息，都能得到有效的表示。

Table 1: WER (%) of our models and prior work on the LRS3 dataset. †We re-implemented Ma et al. (2021a) with the same architecture since the author source code was not provided.

Method	Backbone	Criterion	Labeled iso (hrs)	Labeled utt (hrs)	Unlabeled data (hrs)	WER (%)
Supervised						
Afouras et al. (2020)	CNN	CTC	157	433	-	68.8
Zhang et al. (2019b)	CNN	S2S	157	698	-	60.1
Afouras et al. (2018a)	Transformer	S2S	157	1,362	-	58.9
Xu et al. (2020)	RNN	S2S	157	433	-	57.8
Shillingford et al. (2019)	RNN	CTC	-	3,886	-	55.1
Ma et al. (2021b)	Conformer	CTC+S2S	-	433	-	46.9
Ma et al. (2021b)	Conformer	CTC+S2S	157	433	-	43.3
Makino et al. (2019)	RNN	Transducer	-	31,000	-	**33.6**
Semi-Supervised & Self-Supervised						
Afouras et al. (2020)	CNN	CTC	157	433	334	59.8
Ma et al. (2021a)†	Transformer-BASE	S2S	-	30	433	71.9
			-	433	1,759	49.6
Proposed (Self-Supervised & Self-Supervised + Semi-Supervised)						
AV-HuBERT	Transformer-BASE	S2S	-	30	-	94.3
			-	30	433	51.8
			-	30	1,759	**46.1**
			-	433	-	60.3
			-	433	433	44.0
			-	433	1,759	**34.8**
	Transformer-LARGE	S2S	-	30	-	92.3
			-	30	433	44.8
			-	30	1,759	**32.5**
			-	433	-	62.3
			-	433	433	41.6
			-	433	1,759	**28.6**
AV-HuBERT + Self-Training	Transformer-LARGE	S2S	-	30	1,759	**28.6**
			-	433	1,759	**26.9**