Video Task Decathlon: Unifying Image and Video Tasks in Autonomous Driving Supplementary Material

Thomas E. Huang[1] Yifan Liu[1] Luc Van Gool[1,2] Fisher Yu[1]
[1]ETH Zürich [2]KU Leuven

https://www.vis.xyz/pub/vtd

The supplementary material is organized as follows:

- Section A: Additional base networks
- Section B: Full single-task comparison
- Section C: Experiments on KITTI dataset
- Section D: Additional ablation studies
- Section E: Full resource usage comparison
- Section F: VTD challenge details
- Section G: VTDNet details
- Section H: CPF training protocol details
- Section I: Training details
- Section J: Additional visualizations

A. Additional Base Networks

We provide comparisons of VTDNet against single-task baselines across additional base networks in Table S1, including ConvNeXt-T and ConvNeXt-B [17]. VTDNet maintains its advantage in overall performance across all base networks and achieves performance gains across most tasks. In particular, localization performance improves significantly as model capacity increases, reaching a high score of 36.3 detection AP and 29.4 instance segmentation AP (ConvNeXt-B). This also translates to 37.6 MOT AP and 34.7 MOTS AP. Furthermore, semantic segmentation also observes consistent improvements up to 65.9 mIoU. However, we observe that drivable area and lane detection do not noticeably benefit from the increased capacity, maintaining a similar performance across all base networks.

B. Full Single-Task Comparison

We compare VTDNet against single-task baselines, models from the official BDD100K model zoo[1], and state-of-the-art methods across all tasks. For MOT and MOTS, we use the metrics used by the official benchmarks, mMOTA [24] and mIDF1 [22]. All other tasks use the same metrics as in VTD. The results are shown in Table S2.

VTDNet can achieve SOTA performance on several benchmarks and competitive performance on the rest, despite using only a single model for all tasks with much simpler task-specific heads. For the ResNet-50 comparison, VTDNet achieves significantly better performance on MOTS compared to VMT [9], obtaining an improvement of 5 points in mMOTSA, 3.7 points in mIDF1, and 2.7 points in mAP without using extra tracking annotations or specialized modules. On instance segmentation, VTDNet obtains an improvement of 5.4 points over HTC [2], which utilizes a complex cascade structure. On semantic segmentation and drivable area segmentation, VTDNet achieves competitive performance with DeepLabv3+ [3], despite only using a simple FPN structure.

We also provide system-level comparisons with SOTA methods on established BDD100K benchmarks. By simply scaling up the capacity of the base network, VTDNet achieves performance gains across the board, outperforming other methods that utilize hand-designed task-specific modules with a simple unified structure. On semantic segmentation and object detection, VTDNet again achieves competitive performance with specialized models. On instance segmentation, VTDNet achieves an improvement of 1.9 points in AP over Cascade R-CNN [1] that uses a cascade structure for mask refinement. On MOT and MOTS, VTDNet obtains significantly higher mIDF1 of 1.2 points for MOT and 8.4 points for MOTS over Unicorn [32], demonstrating that it is much better at object association. However, since Unicorn uses a stronger detector, larger base network, and additional tracking training data, it achieves better mMOTA in MOT. Nevertheless, VTDNet still obtains an improvement of 5.7 points in mMOTSA in MOTS.

[1] https://github.com/SysCV/bdd100k-models

Table S1: Comparison of VTDNet against single-task (ST) baselines using additional base networks on VTD validation set. † denotes a separate model is trained for each task. VTDNet outperforms ST baselines on most tasks across all base networks and achieves significantly better VTDA.

Method	Base Network	Classification Tagging Acc^{Gw}	Acc^{Gs}	$VTDA_{cls}$	Segmentation Sem. IoU^{S}	Driv. IoU^{A}	Lane IoU^{L}	$VTDA_{seg}$	Localization Det. AP^{D}	Ins. AP^{I}	Pose AP^{P}	MOT AP^{T}	MOTS AP^{R}	$VTDA_{loc}$	Association Flow IoU^{F}	MOT $AssA^{T}$	MOTS $AssA^{R}$	$VTDA_{ass}$	VTDA
ST Baselines†	ConvNeXt-T	82.7	78.6	81.3	63.2	84.6	**27.6**	57.3	34.4	23.7	42.5	34.9	30.7	32.7	58.8	50.5	44.6	52.1	223.4
VTDNet	ConvNeXt-T	**83.2**	**80.0**	**82.1**	**64.8**	**86.3**	26.7	**57.9**	**36.0**	**28.4**	**42.8**	**36.2**	**33.4**	**34.8**	**60.4**	**52.1**	**45.3**	**53.5**	**228.3 (+4.9)**
ST Baselines†	ConvNeXt-B	83.0	78.8	81.6	64.9	85.8	**28.1**	**58.3**	35.2	24.7	**46.2**	35.0	31.4	33.6	59.2	50.8	46.1	52.8	226.3
VTDNet	ConvNeXt-B	**83.3**	**80.0**	**82.2**	**65.9**	**86.0**	27.1	58.1	**36.3**	**29.4**	45.5	**37.6**	**34.7**	**36.0**	**60.8**	**51.9**	**48.3**	**54.3**	**230.7 (+4.4)**

Table S2: Comparison of VTDNet to single-task models. Our single-task baselines are highlighted in gray, which use the same task decoders as VTDNet. † indicates results from the official BDD100K model zoo and ‡ indicates results from prior published works.

Method	Base Network	Tagging Acc^{Gw}	Acc^{Gs}	Sem. IoU^{S}	Driv. IoU^{A}	Lane IoU^{L}	Det. AP^{D}	Ins. AP^{I}	Pose AP^{P}	Flow IoU^{F}	MOT mMOTA	mIDF1	mAP	MOTS mMOTSA	mIDF1	mAP
ResNet-50																
ResNet [7]		81.9	77.9	–	–	–	–	–	–	–	–	–	–	–	–	–
Semantic FPN [12]		–	–	59.7	–	–	–	–	–	–	–	–	–	–	–	–
DNLNet [34]†		–	–	62.6	–	–	–	–	–	–	–	–	–	–	–	–
DeepLabv3+ [3]†		–	–	64.0	–	–	–	–	–	–	–	–	–	–	–	–
Semantic FPN [12]		–	–	–	83.9	–	–	–	–	–	–	–	–	–	–	–
DeepLabv3+ [3]†		–	–	–	84.4	–	–	–	–	–	–	–	–	–	–	–
DNLNet [34]†		–	–	–	84.8	–	–	–	–	–	–	–	–	–	–	–
Semantic FPN [12]		–	–	–	–	28.4	–	–	–	–	–	–	–	–	–	–
Faster R-CNN [21]		–	–	–	–	–	32.3	–	–	–	–	–	–	–	–	–
Deform. DETR [37]†		–	–	–	–	–	32.1	–	–	–	–	–	–	–	–	–
Cascade R-CNN [1]†	ResNet-50	–	–	–	–	–	33.8	–	–	–	–	–	–	–	–	–
Mask R-CNN [21]		–	–	–	–	–	–	20.2	–	–	–	–	–	–	–	–
Cascade R-CNN [21]†		–	–	–	–	–	–	21.4	–	–	–	–	–	–	–	–
HTC [2]†		–	–	–	–	–	–	21.7	–	–	–	–	–	–	–	–
Simple Baseline [30]		–	–	–	–	–	–	–	37.0	–	–	–	–	–	–	–
PWC-Net [25]		–	–	–	–	–	–	–	–	59.6	–	–	–	–	–	–
QDTrack [20]		–	–	–	–	–	–	–	–	–	36.6	50.8	32.6	–	–	–
Unicorn [32]‡		–	–	–	–	–	–	–	–	–	35.1	–	–	–	–	–
TETer [13]‡		–	–	–	–	–	–	–	–	–	39.0	53.6	–	–	–	–
QDTrack-MOTS [20]		–	–	–	–	–	–	–	–	–	–	–	–	23.5	44.5	25.5
PCAN [10]‡		–	–	–	–	–	–	–	–	–	–	–	–	27.4	45.1	26.6
VMT [9]‡		–	–	–	–	–	–	–	–	–	–	–	–	28.7	45.7	28.3
Unicorn [32]‡		–	–	–	–	–	–	–	–	–	–	–	–	30.8	–	–
VTDNet		83.2	79.7	63.8	85.4	27.8	33.4	27.1	39.7	60.3	36.4	51.5	34.7	33.7	49.4	31.6
System-level Comparison																
UPerNet [31]†	ConvNeXt-B	–	–	67.3	–	–	–	–	–	–	–	–	–	–	–	–
AFA-DLA [33]‡	DLA-169	–	–	67.5	–	–	–	–	–	–	–	–	–	–	–	–
Cascade R-CNN [1]†	ConvNeXt-B	–	–	–	–	–	36.2	–	–	–	–	–	–	–	–	–
Mask Transfiner [8]‡	ResNet-101	–	–	–	–	–	–	23.6	–	–	–	–	–	–	–	–
Cascade R-CNN [1]†	ConvNeXt-B	–	–	–	–	–	–	27.5	–	–	–	–	–	–	–	–
Unicorn [32]‡	ConvNeXt-L	–	–	–	–	–	–	–	–	–	41.2	54.0	–	–	–	–
Unicorn [32]‡	ConvNeXt-L	–	–	–	–	–	–	–	–	–	–	–	–	29.6	44.2	–
VTDNet	ConvNeXt-B	83.3	80.0	65.9	86.0	27.1	36.3	29.4	45.5	60.8	38.6	55.2	37.6	35.3	52.6	34.7

C. Experiments on KITTI Dataset

We conduct additional experiments on the KITTI dataset [5] to demonstrate the versatility of our proposed network and training protocol.

Dataset. KITTI is a real-world, autonomous driving benchmark suite that contains various vision tasks, including object detection, tracking, and segmentation. Compared to BDD100K [35], KITTI is much smaller in scale (∼10% of data) and uses fewer object categories (mainly pedestrians and cars). As only the training set is publicly available, we split the training set into train and validation sets for our experiments. Similar to BDD100K, the annotation density of each image set varies widely, and the statistics are shown in Table S3.

Evaluation Setting. We construct a similar heterogeneous multi-task setup on KITTI for training and evaluation. We use seven tasks that are consistent with VTD: image tagging, drivable area, semantic/instance segmentation, object detection, and MOT/MOTS. To compute VTDA, we simply drop the missing tasks from the averages. As we do not have estimates of task sensitivities across different base networks, we do not scale each task score and opt for a simple average.

Comparison to Baselines. We perform the same comparison to single-task (ST) and multi-task (MT) baselines, and we use the same architecture as with VTD but removing the missing tasks' decoders. As KITTI exhibits the same data imbalance issue, we use the same CPF training protocol as on VTD, but we do not find pseudo-labels to be necessary.

Table S3: Statistics of tasks and available annotations in KITTI [5].

Set	Images (Train / Val)	% Total Images	Tasks with Annotations
Detection	6.2K / 1.3K	54%	Image tagging, object detection
Segmentation	160 / 40	1%	Instance segmentation, semantic segmentation
Drivable	236 / 53	2%	Drivable area segmentation
Tracking	5K / 3K (=12 / 9 videos)	43%	MOT, MOTS, semantic segmentation

Table S4: Comparison of VTDNet against single-task (ST) and multi-task (MT) baselines on KITTI validation set. CPF denotes our training protocol, and † denotes a separate model is trained for each task. VTDNet achieves significantly better VTDA than both ST and MT baselines. **Black** / *blue* indicate best / second best.

Method	CPF	Class. Tag Acc^{G}	VTDA$_{cls}$	Segmentation Sem. IoU^{S}	Driv. IoU^{A}	VTDA$_{seg}$	Localization Det. AP^{D}	Ins. AP^{I}	MOT AP^{T}	MOTS AP^{R}	VTDA$_{loc}$	Association MOT AssA^{T}	MOTS AssA^{R}	VTDA$_{ass}$	VTDA
ST Baselines†		49.7		48.0	96.8	72.4	**60.5**	24.9	48.4	47.1	45.2	*61.8*	61.4	61.6	228.9
MT Baseline		*50.1*		42.9	94.9	68.9	53.4	*31.3*	*51.4*	52.3	47.1	60.9	61.6	61.3	227.4 (-1.5)
VTDNet	✗	**50.2**		**51.3**	**97.5**	**74.4**	*56.2*	31.2	49.2	*53.3*	*47.5*	61.1	*62.8*	*62.0*	*234.0* (+5.1)
	✓	*50.1*		*50.2*	*97.4*	*73.8*	52.1	**33.5**	**54.8**	**55.0**	**48.9**	**65.3**	**66.0**	**65.7**	**238.4 (+9.5)**

The results are shown in Table S4.

Compared to the single-task baselines, the multi-task baseline that jointly optimizes all tasks achieves worse performance on most tasks, demonstrating that a naive architecture and training protocol are not sufficient. VTDNet improves performance of most tasks and leads to much better segmentation and localization scores. With a better training protocol, CPF enables VTDNet to achieve significantly better performance on most tasks and an improvement of 9.5 points in VTDA. However, we note that there exists negative transfer between object detection, instance segmentation, and MOT localization on KITTI, *i.e.*, improvement in one score leads to a drop in the others. We attribute this to differences in annotation between each image set, as KITTI is not designed for multi-task learning. Nevertheless, these results demonstrate the generalizability and effectiveness of our proposed network and training protocol.

D. Additional Ablation Studies

We provide additional ablation studies on components of our optimization strategy. In these experiments, we use VTDNet with ResNet-50 [7] and use the default training parameters with CPF, unless otherwise stated.

Loss Weights. We provide full results of VTDNet with different loss weights configurations in Table S5 to complement Figure 6 in the main paper. For each configuration, we show the loss weights of each task in the first row and the task scores in the second row, and we increase the loss weights of a particular group of tasks to boost the performance of the network in that aspect. Doing so consistently improves the performance in each aspect at the cost of a drop in performance in other aspects. This enables prioritization of certain tasks over others through the choice of loss weights. The overall performance of VTDNet remains stable across all configurations.

Data Sampling Strategies. We additionally investigate different data sampling strategies used during joint optimization on all VTD tasks. Our default strategy, set-level round-robin, samples a batch of training data from each image set in order (section H.1). We also experiment with not using any sampling (None), uniform sampling (Uniform), and weighted sampling (Weighted), following [14]. Uniform and Weighted use a stochastic schedule that samples from a uniform and a weighted distribution (proportional to size of each image set). The results are shown in Table S6. We find that not using any sampling strategy achieves decent performance already and using a stochastic sampler does not achieve further performance gains. This is due to data-imbalance, and the aforementioned strategies are biased towards one image set over another. On the contrary, round-robin sampling better balances the data and obtains the best performance overall.

E. Compute Resource Comparison

We provide the full compute resource usage during inference comparison of VTDNet, single-task, and multi-task baselines with the ResNet-50 base network in Table S7 to complement Figure 5 in the main paper. We show the number of model parameters, number of multiply-accumulate operations (MACs), and number of floating-point operations (FLOPs). These are measured during model inference on a single GeForce RTX 2080 Ti GPU. The total resource utilization of VTDNet is less than that of the semantic segmentation baseline plus the MOTS baseline, showing that a unified network can drastically save computation. By sharing a majority of the network, VTDNet achieves much better computational efficiency compared to single-task baselines by tackling all ten tasks with only a single set of weights. Additionally, VTDNet only uses marginally more computation compared to the multi-task baseline, while achieving much better performance.

F. VTD Challenge Details

We present further details regarding our proposed VTD challenge, detailing each task and our VTDA metric.

Table S5: Ablation study on loss weights with VTDNet on VTD validation set. We show loss weights for each task (first row) and task-specific performance along with VTDA (second row). For loss weights, differences between settings are underlined. Increasing loss weights on a group of tasks boosts the performance of VTDNet in that aspect, enabling prioritization of task performance depending on application.

Loss weights	Classification			Segmentation				Localization						Association				
	Tagging		VTDA_{cls}	Sem.	Driv.	Lane	VTDA_{seg}	Det.	Ins.	Pose	MOT	MOTS	VTDA_{loc}	Flow	MOT	MOTS	VTDA_{ass}	VTDA
	Acc^{GW}	Acc^{GS}		IoU^{S}	IoU^{A}	IoU^{L}		AP^{D}	AP^{I}	AP^{P}	AP^{T}	AP^{R}		IoU^{F}	AssA^{T}	AssA^{R}		
Default	*0.05*	*0.05*		*1.0*	*1.0*	*2.0*		*2.0*	*2.0*	*800.0*	*1.0*			*1.0*	*1.0*			
	83.2	79.7	82.0	63.8	85.4	27.8	57.8	33.4	27.1	39.7	34.7	31.6	32.9	60.3	50.1	45.1	52.7	**225.3**
$2\lambda_G$	*0.1*	*0.1*		*1.0*	*1.0*	*2.0*		*2.0*	*2.0*	*800.0*	*1.0*			*1.0*	*1.0*			
	83.4	**79.9**	**82.2**	63.7	85.4	27.4	57.6	33.1	26.6	39.4	34.0	31.0	32.4	60.2	49.9	44.9	52.5	224.7
$2\lambda_S, 2\lambda_A, 2\lambda_L$	*0.05*	*0.05*		*2.0*	*2.0*	*4.0*		*2.0*	*2.0*	*800.0*	*1.0*			*1.0*	*1.0*			
	83.1	79.6	81.9	**64.0**	**85.5**	**28.0**	**58.0**	33.2	27.0	38.9	33.9	31.4	32.5	60.1	49.6	44.8	52.3	224.7
$2\lambda_D, 2\lambda_I, 2\lambda_P$	*0.05*	*0.05*		*1.0*	*1.0*	*2.0*		*4.0*	*4.0*	*1600.0*	*1.0*			*1.0*	*1.0*			
	83.3	79.7	82.1	63.4	85.1	27.0	57.3	**33.6**	**27.4**	**40.4**	**35.0**	**31.8**	**33.2**	60.0	49.9	45.0	52.5	225.0
$2\lambda_F, 2\lambda_T$	*0.05*	*0.05*		*1.0*	*1.0*	*2.0*		*2.0*	*2.0*	*800.0*	*2.0*			*2.0*	*2.0*			
	83.3	79.8	82.1	63.6	85.1	27.2	57.4	33.3	27.1	39.6	34.6	31.6	32.8	**60.6**	**50.4**	**45.4**	**53.0**	**225.3**

Table S6: Ablation study on data sampling strategies during joint training with VTDNet on VTD validation set.

Strategy	Classification			Segmentation				Localization						Association				
	Tagging		VTDA_{cls}	Sem.	Driv.	Lane	VTDA_{seg}	Det.	Ins.	Pose	MOT	MOTS	VTDA_{loc}	Flow	MOT	MOTS	VTDA_{ass}	VTDA
	Acc^{GW}	Acc^{GS}		IoU^{S}	IoU^{A}	IoU^{L}		AP^{D}	AP^{I}	AP^{P}	AP^{T}	AP^{R}		IoU^{F}	AssA^{T}	AssA^{R}		
None	**83.3**	**79.9**	**82.2**	**64.1**	84.8	27.2	57.4	33.0	26.5	39.1	34.2	31.4	32.4	**60.4**	49.4	44.8	52.4	224.3
Uniform	83.1	79.6	81.9	62.6	85.1	27.4	57.3	33.3	**27.5**	39.3	**34.8**	31.0	32.8	60.2	**50.5**	43.8	52.5	224.5
Weighted	83.2	79.7	82.0	62.6	84.9	27.7	57.4	**33.6**	27.0	39.6	34.4	31.3	32.8	60.2	49.7	44.4	52.3	224.4
Round-robin	83.2	79.7	82.0	63.8	**85.4**	**27.8**	**57.8**	33.4	27.1	**39.7**	34.7	**31.6**	**32.9**	60.3	50.1	**45.1**	**52.7**	**225.3**

Table S7: Full resource usage comparison during inference between VTDNet, single-task (ST), and multi-task (MT) baselines. VTDNet achieves much better computational efficiency compared to single-task baselines.

Model	Params (M)	MACs (G)	FLOPs (G)
Tagging	23.5	33.4	67.0
Detection	41.2	190.6	381.9
Instance Seg.	43.8	192.5	385.8
Pose Estimation	44.3	192.5	385.7
Semantic Seg.	28.6	133.5	267.4
Drivable Area	28.6	133.4	273.1
Lane Estimation	28.6	133.5	273.1
Optical Flow	5.7	166.8	334.8
MOT	56.6	192.2	385.2
MOTS	59.3	216.7	434.2
ST Sum	360.1	1585.1	3189.1
MT Baseline	73.4	292.1	586.5
VTDNet	73.3	309.9	622.1

F.1. Tasks

We first detail each task based on its definition in BDD100K [35] and modifications made to build our VTD challenge.

Image Tagging. There are two classification tasks, weather and scene classification. The weather conditions are rainy, snowy, clear, overcast, partly cloudy, and foggy (plus undefined). The scene types are tunnel, residential, parking lot, city street, gas stations, and highway (plus undefined).

Object Detection. BDD100K provides ten categories for detection: pedestrian, rider, car, truck, bus, train, motorcycle, bicycle, traffic light, and traffic sign. To be consistent with the segmentation and tracking sets, we only use the first eight categories for detection and treat the final two as stuff (background) categories.

Pose Estimation. Pedestrians and riders in BDD100K are labeled with 18 joint keypoints throughout the body.

Drivable Area Segmentation. For drivable area segmentation, the prediction of background is important to account for regions of the image that are not drivable. Thus, the network needs to predict three classes. Accuracy of the background prediction is not considered in the final score.

Lane Detection. Lanes in BDD100K are labeled with eight categories and two attributes, direction and style. Categories include road curb, crosswalk, double white, double yellow, double other color, single white, single yellow, and single other color. Directions include parallel and vertical, and styles include solid and dashed. Thus, each lane has three different labels. We treat lane detection as a contour detection problem for each of the three labels. During evaluation, the performance is averaged over the three labels. Similar to drivable area, background pixels are also required for prediction but not considered for evaluation. Before computing mIoU, we dilate the ground truth by five pixels to account for ambiguity during annotation. Due to the slow speed of computation, we use a subsample of 1K images for evaluation.

Semantic / Instance Segmentation. BDD100K has 19 categories for semantic segmentation, split into 8 thing (foreground) and 11 stuff categories. The 8 thing categories are

Table S8: Analysis of VTDA with VTDNet using ResNet-50 base network on VTD validation set. † denotes a separate model is trained for each task. We also show absolute and scaled differences in task-specific performance. VTDA better balances the contribution of each task score, leading to a more informative metric for our setting.

Method	Classification			Segmentation				Localization						Association				VTDA
	Tagging		VTDA$_{cls}$	Sem.	Driv.	Lane	VTDA$_{seg}$	Det.	Ins.	Pose	MOT	MOTS	VTDA$_{loc}$	Flow	MOT	MOTS	VTDA$_{ass}$	
	Acc^{Gw}	Acc^{Gs}		IoU^{S}	IoU^{A}	IoU^{L}		AP^{D}	AP^{I}	AP^{P}	AP^{T}	AP^{R}		IoU^{F}	AssA^{T}	AssA^{R}		
ST Baselines†	81.9	77.9	80.6	59.7	83.9	**28.4**	56.7	32.3	20.2	37.0	32.9	27.2	29.7	59.6	48.6	42.4	51.3	218.2
VTDNet	**83.2**	**79.7**	**82.0**	**63.8**	**85.4**	27.8	**57.8**	**33.4**	**27.1**	**39.7**	**34.7**	**31.6**	**32.9**	**60.3**	**50.1**	**45.1**	**52.7**	**225.3** (+7.1)
Absolute Δ	1.3	1.8	1.6	4.1	1.5	-0.6	1.7	1.1	6.9	2.7	1.8	4.4	3.4	0.7	1.3	2.7	1.6	–
σ_t	0.4	0.6	–	2.0	0.7	0.9	–	1.1	1.7	3.1	1.0	1.7	–	0.9	0.8	1.4	–	–
s_t	1.00	0.50	–	0.20	0.50	0.50	–	0.33	0.25	0.14	0.33	0.25	–	0.50	0.50	0.33	–	–
Scaled Δ	1.3	0.9	1.4	0.8	0.8	-0.3	1.1	0.4	1.7	0.4	0.6	1.1	3.2	0.3	0.7	0.9	1.4	–

consistent with the detection set. The 11 stuff categories include road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, and sky. Thing masks also include instance information used for instance segmentation.

MOT / MOTS. The object tracking categories are pedestrian, rider, car, truck, bus, train, motorcycle, and bicycle.

Optical Flow Estimation. We use a proxy evaluation method based on MOTS labels to evaluate optical flow estimation. Given segmentation masks of two consecutive frames M_t, $M_{t-1} \in \mathbb{R}^{H \times W}$ and the predicted optical flow $V \in \mathbb{R}^{H \times W \times 2}$, we use the flow to warp the second segmentation mask M_t with nearest sampling to obtain a synthesized mask of the first frame $\hat{M}_{t-1}(p) = M_t(p + V(p))$, where p are the pixel coordinates. The overlap of the warped mask $\hat{M}_{t-1}$ with the ground truth mask of the first frame M_{t-1} gives us an estimate of the flow accuracy for objects in the scene, and we use mean IoU as the metric.

Data Deduplication. We found there is an overlap of 454 images between the segmentation training set and the detection and tracking validation image sets. To maintain consistency in evaluation, we remove the overlapping images from the segmentation training set. Single-task baselines are still trained with the full training set. We found this to not noticeably affect the final results.

F.2. VTD Accuracy Metric

We provide additional details and analysis regarding our VTD Accuracy (VTDA) metric.

Details. VTDA first uses standard deviation estimates σ_t and scaling factors $s_t = 1/\lceil 2\sigma_t \rceil$ for each task t to normalize sensitivities of each metric. σ_t is measured over single-task baselines each trained with eight different base networks (ResNet-50/101 [7], Swin-T/S/B [16], and ConvNeXt-T/S/B [17]) and are provided in Table S8. By computing standard deviation over these baselines, we can get an estimate of how network performances vary across different architectures and capacity. Pose estimation and semantic segmentation have large variations in performance across different base networks. On the other hand, drivable area segmentation and optical flow estimation have smaller variances. Based on σ_t, we compute scaling factors s_t in order to scale each task accordingly. Pose estimation and semantic segmentation have a lower s_t, as improvements in these tasks are less significant. Drivable area and optical flow have larger s_t, as minor improvements are more significant. Each task score is multiplied by its corresponding s_t to obtain the final score.

Analysis. We compare absolute performance differences between VTDNet and single-task baselines (row 3) to s_t scaled performance differences (last row). We also provide VTDA metrics for each aspect, which are calculated in the same way but using Δ instead. With absolute difference, performance gains and losses in instance segmentation and pose estimation are large in magnitude and thus dominate the localization performance. On the other hand, VTDA scales down their values as they are more sensitive than other metrics, leading to more balanced scores across the board. Note that since we normalize the final scores of each aspect to the range $[0, 100]$, the magnitude of each score does not matter. Similarly, absolute difference of tasks with lower sensitivity (*i.e.*, drivable area segmentation) will not reflect the significance of the improvements in performance. Thus, we scale the scores of such tasks relatively more compared to other tasks.

G. VTDNet Details

We provide additional details regarding task decoders and feature interaction blocks.

G.1. Task Decoders

We first describe details about certain decoders and loss functions used for training.

Segmentation Decoders. The drivable area, lane detection, and semantic segmentation decoders use the same structure and employ convolutional layers to map the aggregated pixel features to the desired output. The only exception is the lane detection decoder, as it requires making per-pixel predictions for three separate labels. We replace the final convolutional layer with a convolutional layer for each label. We replace all convolutions with deformable ones [36] and use Group Normalization [29]. For the lane detection decoder, we additionally scale the foreground pixels by 10 to better balance their loss against background pixels. Cross entropy is used

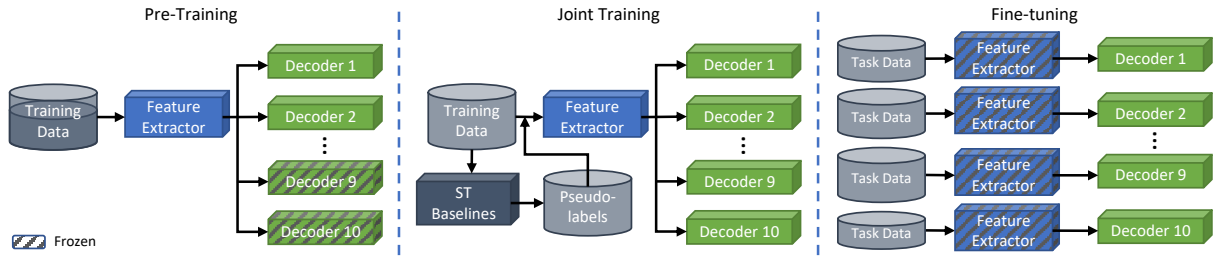

Figure S1: Our training protocol, CPF, including Curriculum training, Pseudo-labeling, and Fine-tuning. A subset of the network is first pre-trained on a portion of data. Then, the network is jointly trained on all tasks, using pseudo-labels for label-deficient tasks to avoid undertraining. Finally, each decoder is independently fine-tuned on its respective data while freezing the learned shared representation.

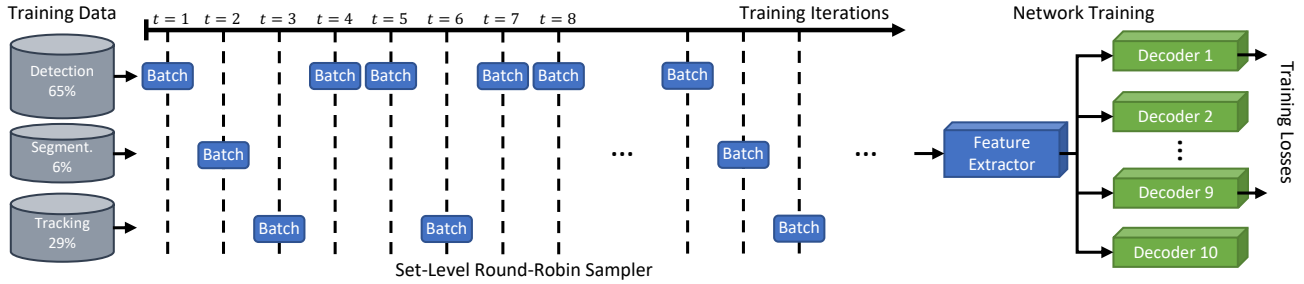

Figure S2: Illustration of our joint training protocol. We use a set-level round-robin data sampler for sampling batches of data during training. Each batch only contains annotations for a subset of the tasks, and the corresponding decoders are updated.

as the training loss for each decoder $L_{\mathtt{S}}$, $L_{\mathtt{A}}$, and $L_{\mathtt{L}}$.

Localization Decoders. The training loss of the localization decoders $L_{\mathtt{loc}}$ is a combination of multiple losses for the Region Proposal Network (RPN) [21] $L_{\mathtt{RPN}}$, bounding box decoder $L_{\mathtt{D}}$, mask decoder $L_{\mathtt{I}}$, and pose estimation decoder $L_{\mathtt{P}}$,

$$L_{\mathtt{loc}} = \lambda_{\mathtt{RPN}} L_{\mathtt{RPN}} + \lambda_{\mathtt{D}} L_{\mathtt{D}} + \lambda_{\mathtt{I}} L_{\mathtt{I}} + \lambda_{\mathtt{P}} L_{\mathtt{P}}, \qquad (1)$$

following Mask R-CNN [6]. $L_{\mathtt{RPN}}$ is a combination of a cross entropy loss for the classification branch of the RPN and a L1 loss for the regression branch. Similarly, $L_{\mathtt{D}}$ use the same losses for classification and regression. $L_{\mathtt{I}}$ uses a cross entropy loss on the instance mask predictions. For the pose estimation decoder, instead of a one-hot heatmap indicating the location of the joint in the Region of Interest (RoI), we use a Gaussian distribution as the training targets, following [30]. The keypoint localization loss $L_{\mathtt{P}}$ is then Mean Squared Error (MSE) on the predicted joint heatmaps. We use $\lambda_{\mathtt{RPN}} = 1.0$ in our experiments by default.

Association Decoders. The architecture of the optical flow estimation decoder follows PWCNet [25]. The flow decoder uses the first two pixel feature maps from the Feature Pyramid Network (FPN) [15] to create a feature pyramid. At each pyramid level, features of the second image are warped using the upsampled flow from the previous layer and then used to compute a cost volume through the correlation operation. Then, convolutional layers are used to predict the flow. We reduce the number of parameters in PWCNet by reducing the number of dense connections in the flow decoder and only using four pyramid levels. For occlusion estimation, we use the range map approach [28], which we found to greatly stabilize training.

The training loss of the flow decoder is a combination of a photometric consistency loss $L_{\mathtt{photo}}$ and a smoothness constraint loss $L_{\mathtt{smooth}}$, which are commonly used by unsupervised optical flow estimation methods,

$$L_{\mathtt{F}} = \lambda_{\mathtt{photo}} L_{\mathtt{photo}} + \lambda_{\mathtt{smooth}} L_{\mathtt{smooth}}. \qquad (2)$$

We use the Census loss [19] as $L_{\mathtt{photo}}$ and the edge-aware second order smoothness constraint [26] as $L_{\mathtt{smooth}}$ with an edge-weight of 150.0. We use $\lambda_{\mathtt{photo}} = 1.0$ and $\lambda_{\mathtt{smooth}} = 4.0$ in our experiments by default. We also experimented with using object segmentation masks as an additional training signal by enforcing consistency between the warped masks (similar to the evaluation protocol), but did not find it to be beneficial for performance.

The training loss of the MOT decoder is a combination of a multi-positive cross entropy loss $L_{\mathtt{embed}}$ and a L2 auxiliary loss $L_{\mathtt{aux}}$,

$$L_{\mathtt{T}} = \lambda_{\mathtt{embed}} L_{\mathtt{embed}} + \lambda_{\mathtt{aux}} L_{\mathtt{aux}}, \qquad (3)$$

following QDTrack [20, 4]. QDTrack uses an additional lightweight embedding head to extract features for each RoI from the RPN. Contrastive learning is used on the dense RoIs of two video frames (key and reference frames) for

Table S9: Training details of every single-task baseline and VTDNet using ResNet-50.

Model	lr	Optimzer	Batch Size	Epochs	Schedule	Augmentations
Image Tagging	0.1	SGD	48	60	Step decay at [30, 45]	Random crop and flip
Object Detection	0.04	SGD	32	36	Step decay at [24, 33]	Multi-scale, random flip
Instance Seg.	0.02		16			
Pose Estimation	0.02		16			
Drivable Area	0.01	SGD	16	∼18 (80K iters.)	Poly. decay with power = 0.9, min. lr = 0.0001	Random scale, crop, and flip; photometric distortion
Lane Detection				∼18 (80K iters.)		
Semantic Seg.				∼183 (80K iters.)		
MOT	0.02	SGD	16	12	Step decay at [8, 11]	Random flip
MOTS	0.01	SGD		12	Step decay at [8, 11]	Random flip
Optical Flow	0.0001	AdamW		36	Step decay at [24, 33]	Multi-scale, random flip
VTDNet	0.0001	AdamW	16	12	Step decay at [8, 11]	Multi-scale, random flip

feature learning. $L_{\texttt{embed}}$ is defined as,

$$L_{\texttt{embed}} = \log \left[1 + \sum_{\boldsymbol{k}^+} \sum_{\boldsymbol{k}^-} \exp \left(\boldsymbol{v} \cdot \boldsymbol{k}^- - \boldsymbol{v} \cdot \boldsymbol{k}^+ \right) \right], \quad (4)$$

where $\boldsymbol{v}$ is the feature embeddings of the training sample in the key frame and $\boldsymbol{k}^+$ and $\boldsymbol{k}^-$ are its positive and negative targets in the reference frame. The auxiliary loss is used to constrain the magnitude and cosine similarity of the vectors, which is defined as

$$L_{\texttt{aux}} = \log \left(\frac{\boldsymbol{v} \cdot \boldsymbol{k}}{||\boldsymbol{v}|| \cdot ||\boldsymbol{k}||} - c \right)^2, \quad (5)$$

where $c = 1$ if it is a positive match and $c = 0$ otherwise. We use $\lambda_{\texttt{embed}} = 0.25$ and $\lambda_{\texttt{aux}} = 1.0$ in our experiments by default.

For MOTS, we simply combine the outputs from the MOT and instance segmentation decoders, so there are no trainable parameters.

G.2. Feature Interaction Blocks

VTDNet utilizes two types of feature sharing modules, Intra-group (Intra-IB) and Cross-group (Cross-IB) Interaction Blocks. We use these blocks for the segmentation and localization task groups, but not the classification group as we found it does not require additional shared processing.

Intra-IB uses self-attention blocks to model feature interactions within a task group. For the segmentation task group, we use Window and Shifted Window Multi-Head Attention [16] on the high resolution feature maps to reduce computation costs. For the localization task group, we use the standard Multi-Head Attention [27] on the object features.

Cross-IB uses cross-attention blocks to model feature interactions between task groups. However, such attention is expensive as the resolution of pixel features is high and the number of object features can be high during training. To reduce the computational costs, we downsample the pixel features by a factor of 8 and average pool the object features into 1D vectors.

H. CPF Training Protocol Details

We provide additional details regarding each aspect of our training protocol CPF. The full protocol is illustrated in Figure S1.

H.1. Curriculum Training

Curriculum training involves pre-training a subset of the network first then joint-training the entire network.

Pre-Training. We first train the feature extractor and localization and object tracking decoders on all three image sets. This includes the object detection, instance segmentation, pose estimation, MOT, and MOTS tasks. We follow the training procedure of QDTrack-MOTS [20, 4], which first trains QDTrack on the detection and tracking sets then finetunes a instance segmentation decoder on the segmentation set and MOTS subset while freezing the rest of the network. We additionally train the pose estimation decoder along with the instance segmentation decoder.

Joint Training. We provide a detailed illustration of our joint training protocol in Figure S2. We use a set-level round-robin data sampler, which samples a batch of training data from each image set in order. By default, we do not oversample the data in each set and spread out the samples to avoid many training iterations without gradients for a particular group of tasks. For tracking, we use the MOTS subset instead of the full tracking set for better data balance, which is only 10% the size. This does not compromise MOT performance as we already pre-trained the MOT decoder on the full tracking set. The loss weights used for joint training is provided in Table S5 under the default setting. The MOTS decoder has no trainable parameters, so there is no corresponding loss weight.

H.2. Pseudo-Labeling

We use single-task baselines to generate pseudo-labels for VTDNet. For consistency, we use single-task baselines with the same base network as VTDNet to generate the pseudo-labels. Such pseudo-labels are only used for pose estimation

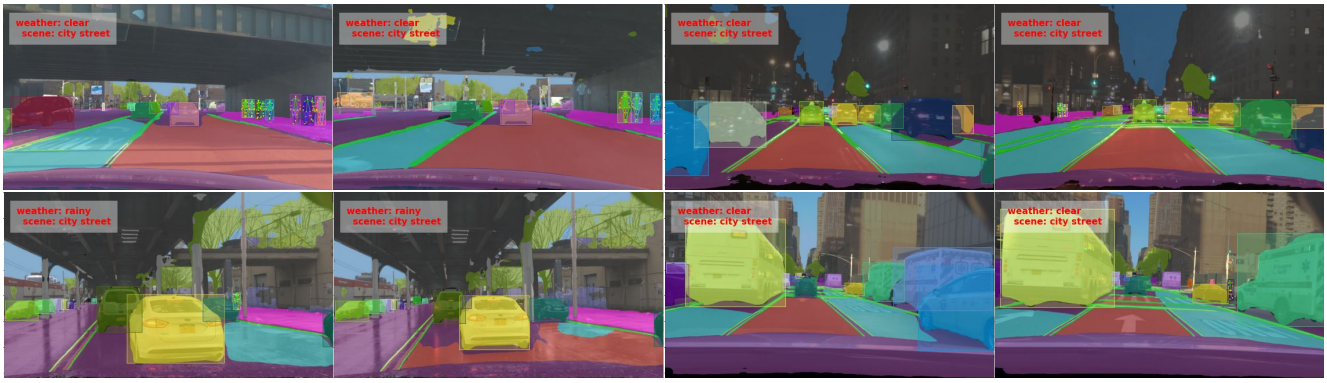

Figure S3: Additional visualizations of VTDNet predictions on all tasks (excluding flow). Best viewed in color.

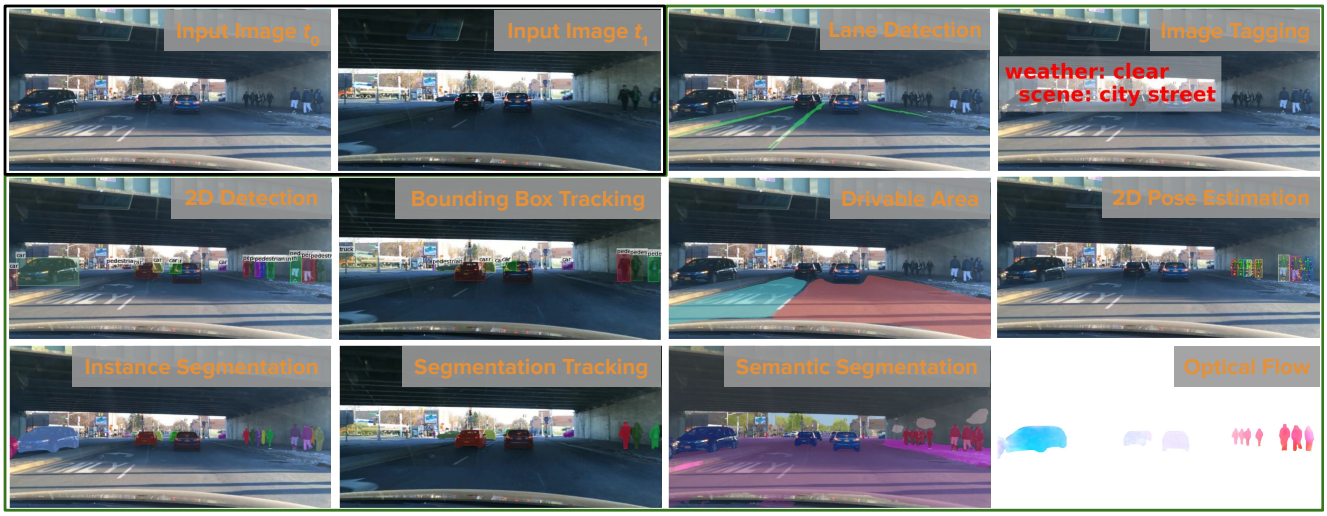

Figure S4: Visualization of VTDNet predictions on all ten VTD tasks for a pair of input images. Best viewed in color.

and semantic segmentation as the proportion of their data is the lowest.

Pose Estimation. We generate pose estimation pseudo-labels following standard inference procedure. We use a visibility threshold of 0.2 to filter the predictions, where predicted joints with a confidence lower than this threshold are removed.

Semantic Segmentation. We generate semantic segmentation pseudo-labels using standard inference procedure. We use a confidence threshold of 0.3 to filter the predictions, where prediction pixels with a confidence lower than this threshold are set to unknown and ignored.

H.3. Fine-Tuning

After joint training, we fine-tune each task-specific decoder on the corresponding data for six epochs while freezing the rest of the network. We decrease the learning rate by a factor of 10. After fine-tuning, we combine the weights of each decoder and the rest of the network to obtain our final network weights.

I. Training Details

In this section, we show full training details for VTDNet and the single-task baselines. All models are trained on either 8 GeForce RTX 2080 Ti or 8 GeForce RTX 3090 GPUs. We use half-precision floating-point format (FP16) for all models to speed up training. We use the same codebase and environment for training all models to ensure consistency in the training setting. For each task, the baseline is trained only on data from the particular task. The baseline uses task-specific augmentations and training schedules that are optimized for single-task performance, which we detail in Table S9. For SGD [23], we use a momentum of 0.9 and weight decay of 10^{-4}. For AdamW [11, 18], we use $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 0.05. For the multi-scale augmentation, we sample an image height from $[600, 624, 648, 672, 696, 720]$ and scale the image while keeping the aspect ratio the same. For all models using Swin Transformer [16] or ConvNeXt [17] as the base network, we use AdamW with a learning rate of 0.0001.

J. Visualizations

We provide additional visualizations of VTDNet predictions on the VTD tasks in Figure S3. We also visualize each task prediction separately in Figure S4. The color of each object indicate the predicted instance identity. For drivable area segmentation, red areas on the road indicate drivable regions, and blue areas indicate alternatively drivable areas. The green lines represent predicted lanes on the road. For optical flow estimation, we segment the flow using the instance segmentation mask predictions to extract object-level flow to be consistent with the evaluation protocol. VTDNet can produce high quality predictions for all ten tasks.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 1, 2

[2] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4974–4983, 2019. 1, 2

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 1, 2

[4] Tobias Fischer, Thomas E Huang, Jiangmiao Pang, Linlu Qiu, Haofeng Chen, Trevor Darrell, and Fisher Yu. Qdtrack: quasi-dense similarity learning for appearance-only multiple object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 6, 7

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2, 3

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 6

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2, 3, 5

[8] Lei Ke, Martin Danelljan, Xia Li, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Mask transfiner for high-quality instance segmentation. In *CVPR*, 2022. 2

[9] Lei Ke, Henghui Ding, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Video mask transfiner for high-quality video instance segmentation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 731–747. Springer, 2022. 1, 2

[10] Lei Ke, Xia Li, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Prototypical cross-attention networks for multiple object tracking and segmentation. *NeurIPS*, 34, 2021. 2

[11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[12] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 2

[13] Siyuan Li, Martin Danelljan, Henghui Ding, Thomas E. Huang, and Fisher Yu. Tracking every thing in the wild. In *ECCV*, 2022. 2

[14] Valerii Likhosherstov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. Polyvit: Co-training vision transformers on images, videos and audio. *arXiv preprint arXiv:2111.12993*, 2021. 3

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 5, 7, 8

[17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022. 1, 5, 8

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *iclr*, 2019. 8

[19] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018. 6

[20] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *CVPR*, 2021. 2, 6, 7

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28, 2015. 2, 6

[22] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV*, pages 17–35. Springer, 2016. 1

[23] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. 8

[24] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, John Garofolo, Djamel Mostefa, and Padmanabhan Soundararajan. The clear 2006 evaluation. In *Multimodal Technologies for Perception of Humans*, pages 1–44, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 1

[25] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 2, 6

[26] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846, 1998. 6

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017. 7

[28] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. Occlusion aware unsupervised learning of optical flow. In *CVPR*, pages 4884–4893, 2018. 6

[29] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, pages 3–19, 2018. 5

[30] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, September 2018. 2, 6

[31] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 2

[32] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. In *ECCV*, 2022. 1, 2

[33] Yung-Hsu Yang, Thomas E Huang, Min Sun, Samuel Rota Bulò, Peter Kontschieder, and Fisher Yu. Dense prediction with attentive feature aggregation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 97–106, 2023. 2

[34] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks, 2020. 2

[35] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, pages 2636–2645, 2020. 2, 4

[36] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, pages 9308–9316, 2019. 5

[37] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2