Real-time Object Detection for Streaming Perception

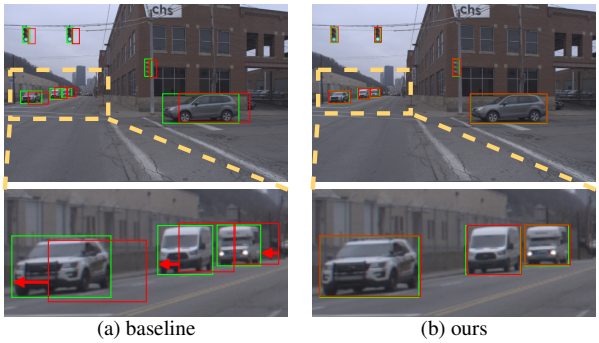
Jinrong Yang[1], Songtao Liu[2]*, Zeming Li[2], Xiaoping Li[1], Jian Sun[2]
[1]Huazhong University of Science and Technology, [2]Megvii Technology
yangjinrong@hust.edu.cn;liusongtao@megvii.com;lizeming@megvii.com;
lixiaoping@hust.edu.cn;sunjian@megvii.com

Abstract

Autonomous driving requires the model to perceive the environment and (re)act within a low latency for safety. While past works ignore the inevitable changes in the environment after processing, streaming perception is proposed to jointly evaluate the latency and accuracy into a single metric for video online perception. In this paper, instead of searching trade-offs between accuracy and speed like previous works, we point out that endowing real-time models with the ability to predict the future is the key to dealing with this problem. We build a simple and effective framework for streaming perception. It equips a novel Dual-Flow Perception module (DFP), which includes dynamic and static flows to capture the moving trend and basic detection feature for streaming prediction. Further, we introduce a Trend-Aware Loss (TAL) combined with a trend factor to generate adaptive weights for objects with different moving speeds. Our simple method achieves competitive performance on Argoverse-HD dataset and improves the AP by 4.9% compared to the strong baseline, validating its effectiveness. Our code will be made available at https://github.com/yancie-yjr/StreamYOLO.

1. Introduction

One critical factor for autonomous safe driving is to perceive its environment and (re)act within a low latency. Recently, several real-time detectors [3, 13, 18, 31, 33, 41–43] achieve competitive performance under the low latency restriction. But they are still explored in an *offline* setting [26]. In a real-world vision-for-online scenario, no matter how fast the model becomes, the surrounding environment has changed once the model finishes processing the latest frame. As shown in Fig. 1(a), the inconsistency between perceptive results and the changed state may trigger unsafe decisions for autonomous driving. Thus for online perception, detectors are imposed to have the ability of future forecasting.



(a) baseline (b) ours

Figure 1. Illustration of visualization results of base detector and our method. The green boxes are ground truth, while the red ones are predictions. The red arrows mark the shifts of the prediction boxes caused by the processing time delay while our approach alleviates this issue.

To tackle this issue, [26] firstly proposes a new metric named streaming accuracy, which integrates latency and accuracy into a single metric for real-time online perception. It jointly evaluates the output of the entire perception stack at every time instant, forcing the perception to forecast the state where the model finishes processing. With this metric, [26] shows a significant performance drop of several strong detectors [6, 21, 28] from offline setting to streaming perception. Further, [26] proposes a meta-detector named Streamer that can incorporate any detector with decision-theoretic scheduling, asynchronous tracking, and future forecasting to recover much of the performance drop. Following this work, Adaptive streamer [16] adopts numerous approximate executions based on deep reinforcement learning to learn a better trade-off online. These works focus on searching for a better trade-off policy between speed and accuracy for some existing detectors, while a novel streaming perception model design is not well studied.

One more thing ignored by the above works is the existing real-time object detectors [13, 18]. By strong data augmentation and delicate architecture design, they achieve competitive performance and can run faster than 30 FPS. With these "fast enough" detectors, there is no space for ac-

*Corresponding author

curacy and latency trade-off on streaming perception as the current frame results from the detector are always matched and evaluated by the next frame. These real-time detectors can narrow the performance gap between streaming perception and offline settings. In fact, both the 1st [59] and 2nd [20] place solution of Streaming Perception Challenge (Workshop on Autonomous Driving at CVPR 2021) adopt real-time models YOLOX [13] and YOLOv5 [18] as their base detectors. Standing on the shoulder of the real-time models, we find that now the performance gap all comes from the fixed inconsistency between the current processing frame and the next matched frame. Thus the key solution for streaming perception is to predict the results of the *next* frame at the *current* state.

Unlike the heuristic methods such as Kalman filter [25] adopted in [26], in this paper, we directly endow the real-time detector with the ability to predict the future of the next frame. Specifically, we construct triplets of the last, current, and next frame for training, where the model gets the last and current frames as input and learns to predict the detection results of the next frame. We propose two crucial designs to improve the training efficiency: i) For model architecture, we conduct a Dual-Flow Perception (DFP) module to fuse the feature map from the last and current frames. It consists of a dynamic flow and a static flow. Dynamic flow pays attention to the moving trend of objects for forecasting while static flow provides basic information and features of detection through a residual connection. ii) For the training strategy, we introduce a Trend Aware Loss (TAL) to dynamically assign different weights for localizing and forecasting each object, as we find that objects within one frame may have different moving speeds.

We conduct comprehensive experiments on Argoverse-HD [5, 26] dataset, showing significant improvements in the stream perception task. In summary, the contributions of this work are as three-fold as follows:

- With the strong performance of real-time detectors, we find the key solution for streaming perception is to predict the results of the *next* frame. This simplified task is easy to be structured and learned by a model-based algorithm.
- We build a simple and effective streaming detector that learns to forecast the next frame. We propose two adaptation modules, *i.e.*, Dual-Flow Perception (DFP) and Trend Aware Loss (TAL), to perceive the moving trend and predict the future.
- We achieve competitive performance on Argoverse-HD [5, 26] dataset without bells and whistles. Our method improves the mAP by +4.9% compared to the strong baseline of the real-time detector and shows robust forecasting under the different moving speeds of the driving vehicle.

2. Related Works

Image object detection. In the era of deep learning, detection algorithms can be split into the two-stage [17, 27, 34, 44, 53, 60] and the one-stage [14, 28, 31, 32, 35, 38, 39, 41, 51, 61] frameworks. Some works, such as YOLO series [3, 13, 18, 41–43], adopt a bunch of training and accelerating tricks to achieve strong performance with real-time inference speed. Our work is based on the recent real-time detector YOLOX [13] which achieves strong performance among real-time detectors.

Video object detection. Streaming perception also relates to video object detection. Some recent methods [1, 7, 11, 62] employ attention mechanism, optical flow, and tracking method, aiming to aggregate rich features for the complex video variation, *e.g.*, motion blur, occlusion, and out-of-focus. However, they all focus on the offline setting, while streaming perception considers the online processing latency and needs to predict the future results.

Video prediction. Video prediction tasks aim to predict the results for the unobserved future data. Current tasks include future semantic/instance segmentation. For semantic segmentation, early works [2, 37] construct a mapping from past segmentation to future segmentation. Recent works [9, 30, 46, 47] convert to predict intermediate segmentation features by employing deformable convolutions, teacher-student learning, flow-based forecasting, LSTM-based approaches, etc. For instance segmentation prediction, some approaches predict the pyramid features [36] or the feature of varied pyramid levels jointly [23, 49]. The above prediction methods do not consider the misalignment of prediction and environment change caused by processing latency, leaving a gap to real-world application. In this paper, we focus on the more practical task of streaming perception.

Streaming perception. Streaming perception task coherently considers latency and accuracy. [26] firstly proposes sAP to evaluate accuracy under the consideration of time delay. Facing latency, non-real-time detectors will miss some frames. [26] proposes a meta-detector to alleviate this problem by employing Kalman filter [25], decision-theoretic scheduling, and asynchronous tracking [1]. [16] lists several factors (e.g., input scales, switchability of detectors, and scene aggregation.) and designs a reinforcement learning-based agent to learn a better combination for a better trade-off. Fovea [50] employs a KDE-based mapping to raise the upper limit of the offline performance. In this work, instead of searching better trade-off or enhancing base detector, we simplify the steaming perception to the task of "predicting the next frame" by a real-time detector.

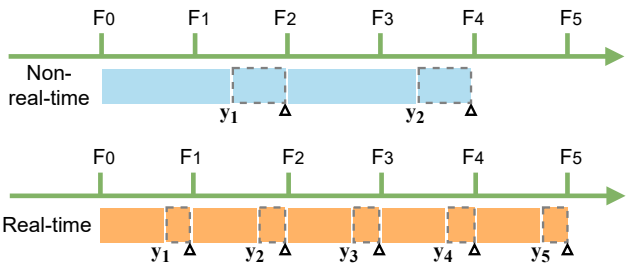


Figure 2. Comparison on different detectors in streaming perception evaluation framework. Each block represents the process of the detector for one frame and its length indicates the running time. The dashed block indicates the time until the next frame data is received.

3. Methods

3.1. Streaming Perception

Streaming perception organizes data as a set of sensor observations. To take the model processing latency into account, [26] proposes a new metric named streaming AP (sAP) to simultaneously evaluate time latency and detection accuracy. As shown in Fig. 2, the streaming benchmark evaluates the detection results over a continuous time frame. After receiving and processing an image frame, sAP simulates the time latency among the streaming flow and examines the processed output with a ground truth of the actual world state.

For the example of a non-real-time detector, the output y_1 of the frame F_1 is matched and evaluated with the ground truth of F_3 and the result of F_2 is missed. Thus for the task of streaming perception, non-real-time detectors may miss many image frames and produce long-time shifted results, significantly hurting the performance of *offline* detection.

For real-time detectors (the total processing time of one image frame is less than the time interval of image streaming), the task of streaming perception becomes easy and clear. As we can see in Fig. 2, a real-time detector avoids the shifting problem with a fixed pattern of matching the next frame to the current prediction. This fixed matching pattern not only eradicates the missed frames but also reduces the time shift for each matched ground truth.

In Fig. 3, we compare two detectors, Mask R-CNN [21] and YOLOX [13], with several image scales and study the performance gap between streaming perception and offline settings. In the case of low-resolution input, the performance gaps are small for two detectors as they are all running in a real-time manner. However, with the resolution increasing, the performance drop of Mask R-CNN gets larger as it runs slower. For YOLOX, its inference speed maintains real-time with the resolution increasing, so that the gap is not correspondingly widened.

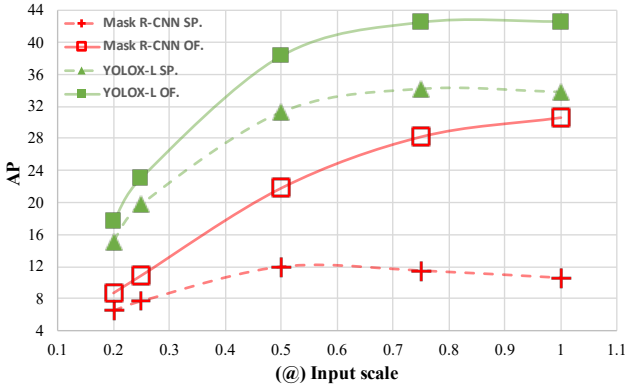


Figure 3. The performance gap between *offline* and streaming perception setting brings about on Argoverse-HD dataset. 'OF' and 'SP' indicate *offline* and streaming perception setting respectively. The number after @ is the input scale (the full resolution is 1200×1920).

3.2. Pipeline

The fixed matching pattern from real-time detectors also enables us to train a learnable model to dig the potential moving trend and predict the objects of the next image frames. Our approach includes a basic real-time detector, an offline training schedule, an online inference strategy, which are described next.

Base detector. We choose the recent proposed YOLOX [13] as our base detector. It inherits and carries forward YOLO series [41–43] to an anchor-free framework with several tricks, *e.g.*, decoupled heads [48, 56], strong data augmentations [15, 58], and advanced label assigning [12], achieving strong performance among real-time detectors. It is also the 1st place solution [59] of Streaming Perception Challenge in the Workshop on Autonomous Driving at CVPR 2021. Different from [59], we remove some engineering speedup tricks such as TensorRT and change the input scale to the half resolution (600×960) to ensure the real-time speed without TensorRT. We also discard the extra datasets used in [59], *i.e.*, BDD100K [57], Cityscapes [10], and nuScenes [4] for pre-training. These shrinking changes definitely decrease the detection performance compared to [59], but they alleviate the executive burden and allow extensive experiments. We believe the shrinking changes are orthogonal to our work and can be equipped to further improve the performance.

Training. We visualize our total training pipeline in Fig. 4. We construct the last, the current frames and next gt boxes to a triplet (F_{t-1}, F_t, G_{t+1}) for training. The main

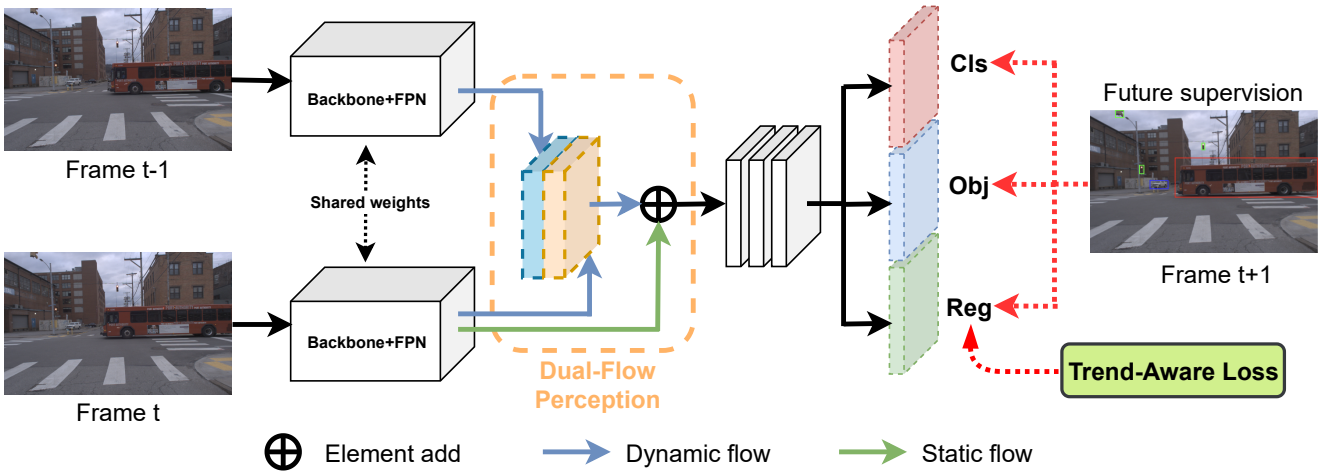


Figure 4. The training pipeline. First, we adopt a shared weight CSPDarknet-53 with PANet to extract FPN features of the current and last image frames. Second, we use the proposed Dual-Flow Perception module (DFP) to aggregate feature maps and feed them to classification, objectness and regression head. Third, we directly utilize the ground truth of the next frame to conduct supervision. We also design a Trend-Aware Loss (TAL) applied to the regression head for efficient training.

reason for this design is simple and direct: in order to predict the future position of objects, it is inevitable to know the moving status for each object. We thus take two adjacent frames (F_{t-1}, F_t) as input and train the model to directly predict the detection results of the next frame, supervised by the ground truth of F_{t+1}. Based on the triplets of inputs and supervision, we rebuild the training dataset to the formulate of $\{(F_{t-1}, F_t, G_{t+1})\}_{t=1}^{n_t}$, where n_t is the total sample number. The first and last frame of each video streaming is excluded. With this rebuilt dataset, we can keep a random shuffling strategy for training and improve the efficiency with distributed GPU training as normal.

To better capture the moving trend between two input frames, we propose a Dual-Flow Perception Module (DFP) and a Trend-Aware Loss (TAL), introduced in the next subsection, to fuse the FPN feature maps of two frames and adaptively catch the moving trend for each object.

We also study another indirect task which parallelly predicts the current gt boxes G_t and the offsets of object transformations from G_t to G_{t+1}. However, according to some ablation experiments, described in the next section (Sec. 4.2), we find that predicting the additional offsets always falls into a suboptimal task. One reason is that the value of the transformative offsets between two adjacent frames is small, involving some noise of numerical instability. It also has some bad cases where the label of the corresponding object is sometimes not reachable (new objects come or current objects disappear in the next frame).

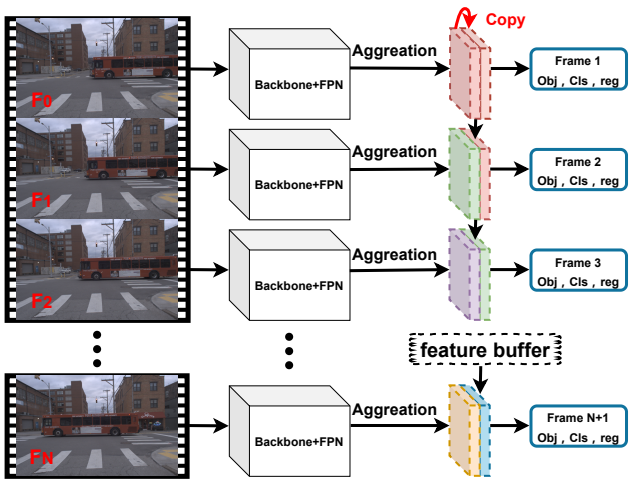


Figure 5. The inference pipeline. We employ a feature buffer to save the historical features of the latest frame and thus only need to extract current features. By directly aggregating the features stored at the last moment, we save the time of handling the last frame again. For the beginning of the vedio, we copy the current FPN features as pseudo historical buffers to predict results.

Inference. The proposed model takes two image frames as input, bringing nearly twice computational cost and time consumption compared to the original detector. As shown in Fig. 5, to eliminate the dilemma, we employ a feature buffer to store all the FPN feature maps of the previous frame F_{t-1}. At inference, our model only extracts the feature of the current image frame and then aggregates the historical features from the buffer. With this strategy, our model runs almost at the same speed as the base detector. For the beginning frame F_0 of the stream, we duplicate the FPN feature maps as pseudo historical buffers to predict re-

sults. This duplication actually means "no moving" status and the static results are inconsistent with F_1. Fortunately, the influence on performance is trivial as this case is rare.

3.3. Dual-Flow Perception Module (DFP)

Given the FPN feature maps of the current frame F_t and the historical frame F_{t-1}, we suppose two critical pieces of information the feature should have for predicting the next frame. One is the moving tendency to capture the moving state and estimate the magnitude of movement. The other is the basic semantic information for the detector to localize and classify the corresponding objects.

We thus design a Dual-Flow Perception (DFP) module to encode the expected features with the dynamic flow and static flow, as seen in Fig. 4. Dynamic flow fuses the FPN feature of two adjacent frames to learn the moving information. It first employs a shared weight 1×1 convolution layer followed by the batchnorm and SiLU [40] to reduce the channel to half numbers for both two FPN features. Then, it simply concatenates these two reduced features to generate the dynamic features. We have studied several other fusing operations like add, non-local block [52], STN [24] based on squeeze-and-excitation network [22], where concatenation shows the best efficiency and performance (see Tab. 1c). As for static flow, we reasonably add the original feature of the current frame through a residual connection. In the later experiments, we find the static flow not only provides the basic information for detection but also improves the predicting robustness across different moving speeds of the driving vehicle.

3.4. Trend-Aware Loss (TAL)

We notice an important fact in streaming perception, in which the moving speed of each object within one frame is quite different. The variant trends come from many aspects: different sizes and moving states of their own, occlusions, or the different topological distances.

Motivated by the observations, we introduce a Trend-Aware Loss (TAL) which adopts adaptive weight for each object according to its moving trend. Generally, we pay more attention to the fast-moving objects as they are more difficult to predict the future states. To quantitatively measure the moving speed, we introduce a trend factor for each object. We calculate an IoU (Intersection over Union) matrix between the ground truth boxes of F_{t+1} and F_t and then conduct the max operation on the dimension of F_t to get the matching IoU of the corresponding objects between two frames. The small value of this matching IoU means the fast-moving speed of the object and vice versa. If a new object comes in F_{t+1}, there is no box to match it and its matching IoU is much smaller than usual. We set a threshold τ to handle this situation and formulate the final trend factor ω_i for each object in F_{t+1} as:

$$mIoU_i = \max_j(\{IoU(box_i^{t+1}, box_j^t)\}) \tag{1}$$

$$\omega_i = \begin{cases} 1/mIoU_i & mIoU_i \geq \tau \\ 1/\nu & mIoU_i < \tau \end{cases}, \tag{2}$$

where $\max_j$ represents the max operation among boxes in F_t, ν is a constant weight for the new coming objects. We set ν as 1.4 (bigger than 1) to reduce the attention according to hyper-parameters grid searching.

Note that simply applying the weight to the loss of each object will change the magnitude of the total losses. This may disturb the balance between the loss of positive and negative samples and decrease the detection performance. Inspired by [54, 55], we normalize ω_i to $\hat{\omega}_i$ intending to keep the sum of total loss unchanged:

$$\hat{\omega}_i = \omega_i \cdot \frac{\sum_{i=1}^{N} \mathcal{L}_i^{reg}}{\sum_{i=1}^{N} \omega_i \mathcal{L}_i^{reg}}, \tag{3}$$

where $\mathcal{L}_i^{reg}$ indicates the regression loss of object i. Next, we re-weight the regression loss of each object with $\hat{\omega}_i$ and the total loss is exhibited as:

$$\mathcal{L}_{total} = \sum_{i \in positive} \hat{\omega}_i \mathcal{L}_i^{reg} + \mathcal{L}_{cls} + \mathcal{L}_{obj}. \tag{4}$$

4. Experiments

4.1. Settings

Datasets. We conduct the experiments on video autonomous driving dataset Argoverse-HD [5, 26] (High-frame-rate Detection), which contains diverse urban outdoor scenes from two US cities. It has multiple sensors and high frame-rate sensor data (30 FPS). Following [26], we only use the center RGB camera and the detection annotations provided by [26]. We also follow the train/val split in [26], where the validation set contains 24 videos with a total of 15k frames.

Evaluation metrics. We use sAP [26] (the streaming perception challenge toolkit [45]) to evaluate all experiments. sAP is a metric for streaming perception. It simultaneously considers latency and accuracy. Similar to MS COCO metric [29], it evaluates average mAP over IoU (Intersection-over-Union) thresholds from 0.5 to 0.95 as well as $\mathrm{AP_s}$, $\mathrm{AP_m}$, $\mathrm{AP_l}$ for small, medium and large object.

Implementation details. If not specified, we use YOLOX-L [13] as our default detector. All of our experiments are fine-tuned from the COCO pre-trained model by 15 epochs. We set batch size at 32 on 8 GTX 2080ti GPUs. We use stochastic gradient descent (SGD) for training. We

Method	sAP	sAP_{50}	sAP_{75}
Baseline	31.2	54.8	29.5
Offsets	31.0 (-0.2)	52.2	30.7
Next	34.2 (+3.0)	54.6	34.9

(a) **Prediction task**. Comparisons on two types of prediction tasks.

Method	sAP	sAP_{50}	sAP_{75}	Latency
Baseline	31.2	54.8	29.5	18.23 ms
Input	30.3 (-0.9)	50.5	29.2	18.33 ms
Backbone	30.5 (-0.7)	50.5	30.5	18.76 ms
FPN	34.2 (+3.0)	54.6	34.9	18.98 ms

(b) **Fusion feature**. Comparisons on three different patterns of features to fuse.

Operation	sAP	sAP_{50}	sAP_{75}	Latency
Baseline	31.2	54.8	29.5	18.23 ms
Add	30.8 (-0.4)	54.8	29.6	18.81 ms
NL	32.7 (+1.5)	56.1	30.7	26.11 ms
STN	34.0 (+2.8)	55.8	32.9	24.32 ms
Concatenation	34.2 (+3.0)	54.6	34.9	18.98 ms

(c) **Fusion operator**. Comparisons on different fusion operations.

Table 1. Ablation experiments for building a strong pipeline. We employ a basic YOLOX-L detector as the baseline for all experiments.

adopt a learning rate of $0.001 \times BatchSize/64$ (linear scaling [19]) and the cosine schedule with a warm-up strategy for 1 epoch. The weight decay is 0.0005 and the SGD momentum is 0.9. The base input size of the image is 600×960 while the long side evenly ranges from 800 to 1120 with 16 strides. We do not use any data augmentation (such as Mosaic [18], Mixup [58], horizontal flip, etc.) since the feeding adjacent frames need to be aligned. For inference, we keep the input size at 600×960 and measure the processing time on a Tesla V100 GPU.

4.2. Ablations for Pipeline

We conduct ablation studies for the pipeline design on three crucial components: the task of prediction, the feature used for fusion, and the operation of fusion. We employ a basic YOLOX-L detector as the baseline for all experiments and keep the other two components unchanged when ablating one. In particular, all entries work in real-time (30 FPS) so that the comparison is fair.

Prediction task. We compare the two types of prediction tasks mentioned in Sec. 3.2. As shown in Tab. 1a, indirectly predicting current bounding boxes with corresponding offsets gets even worse performance than the baseline. In contrast, directly forecasting future results achieves significant improvement (+3.0 AP). This demonstrates the supremacy of directly predicting the results of the next frame.

Fusion feature. Fusing the previous and current information is important for the streaming task. For a general detector, we can choose three different patterns of features to fuse: input, backbone, and FPN pattern respectively. Technically, the input pattern directly concatenates two adjacent frames together and adjusts the input channel of the first layer. The backbone and FPN pattern adopt a 1×1 convolution followed by batch normalization and SiLU to reduce half channels for each frame and then concatenate them together. As shown in Tab. 1b. The results of the input and backbone pattern decrease the performance by 0.9 and 0.7 AP. By contrast, the FPN pattern significantly boosts 3.0 AP, turning into the best choice. These results indicate that the fusing FPN feature may get a better trade-off between capturing the motion and detecting the objects.

Fusion operation. We also explore the fusion operation for FPN features. We seek several regular operators (*i.e.*, element-wise add and concatenation) and advanced ones (*i.e.*, spatial transformer network [24] (STN)[1] and non-local network [52] (NL)[2]. Tab. 1c shows the performance among these operations. We can see that the element-wise add operation drops performance by 0.4 AP while other ones achieve similar gains. We suppose that adding element-wise values may break down the relative information between two frames and fail to learn trending information. And among effective operations, concatenation is prominent because of its light parameters and high inference speed.

4.3. Ablations for DFP and TAL

Effect of DFP and TAL. To validate the effect of DFP and TAL, we conduct extensive experiments on YOLOX detectors with different model sizes. In Tab. 2, "Pipe." denotes our basic pipeline containing basic feature fusion and future prediction training. Compared to the baseline detector, the proposed pipelines have already improved the performance by 1.3 to 3.0 AP across different models. Based on these high-performance baselines, DFP and TAL can boost the accuracy of sAP by $\sim$1.0 AP independently, and their combinations further improve the performance by nearly 2.0 AP. These facts not only demonstrate the effectiveness of DFP and TAL but also indicate that the contributions of the two modules are almost orthogonal.

Indeed, DFP adopts dynamic flow and static flow to extract the moving state feature and basic detection feature separately and enhances the FPN feature for streaming perception. Meanwhile, TAL employs adaptive weight for each object to predict different trending. We believe the two modules cover different points for streaming perception: architecture and optimization. We hope that our simple design of the two modules will lead to future endeavors in these two under-explored aspects.

Value of τ and ν. As depicted in Eq. 2, the value of τ acts as a threshold to monitor newly emerging objects while

[1]To implement STN for variable inputs, we adopt a SE [22] block to calculate the transformation parameters instead of using flatten operation and fully connected layers in the original STN.

[2]For NL, we use the current feature to calculate the values and queries and use the previous feature to generate keys.

Model	Pipe.	DFP	TAL	Off AP	sAP	sAP_{50}	sAP_{75}
YOLOX-S				32.0	26.3	48.1	24.0
	✓				27.6 $_{\uparrow 1.3}$	48.3	26.1
	✓	✓			28.2 (+0.6)	49.4	27.4
	✓		✓		28.1 (+0.5)	49.1	27.0
	✓	✓	✓		28.8 (+1.2)	50.3	27.6
YOLOX-M				34.5	29.2	51.9	27.7
	✓				31.2 $_{\uparrow 2.0}$	51.1	31.9
	✓	✓			32.3 (+1.1)	52.9	32.5
	✓		✓		31.8 (+0.6)	53.1	31.8
	✓	✓	✓		32.9 (+1.7)	54.0	32.5
YOLOX-L				38.3	31.2	54.8	29.5
	✓				34.2 $_{\uparrow 3.0}$	54.6	34.9
	✓	✓			35.5 (+1.3)	56.4	35.3
	✓		✓		35.1 (+0.9)	55.5	35.6
	✓	✓	✓		36.1 (+1.9)	57.6	35.6

Table 2. The effect of the proposed pipeline, DFP, and TAL. 'Off AP' means the corresponding AP using the base detector on the offline setting. 'Pipe.' denotes the proposed pipeline, marked in gray, while '↑' indicates the corresponding improvements. '(·)' indicates the relative improvements based on the strong pipeline.

τ	ν	sAP	sAP_{50}	sAP_{75}	sAP_s	sAP_m	sAP_l
0.2	1.3	35.6	57.1	36.0	13.0	36.9	62.8
0.2	1.4	35.8	57.4	35.0	13.2	37.0	62.5
0.2	1.5	35.8	57.2	35.4	12.8	36.8	**63.6**
0.3	1.3	35.8	56.8	35.3	13.7	36.3	62.6
0.3	1.4	**36.1**	**57.6**	35.6	13.8	**37.1**	**63.3**
0.3	1.5	35.9	57.6	35.0	13.2	36.8	63.3
0.4	1.3	35.3	56.9	35.2	12.8	35.1	61.6
0.4	1.4	35.7	57.1	**36.0**	13.2	36.6	61.0
0.4	1.5	35.4	56.7	35.3	13.6	35.9	61.8

(a) $\nu > 1$.

ν	sAP
1.5	35.9
1.4	**36.1**
1.3	35.8
1.2	35.8
1.1	35.5
1.0	35.6
0.9	35.4
0.8	35.4
0.7	35.0

(b) $\nu < 1$.

Table 3. Grid search of τ and ν in Eq. 2 for TAL.

ν controls the degree of attention on the new objects. We set ν larger than 1.0 so that the model pays less attention to the new-coming objects. We conduct a grid search for the two hyperparameters in Tab. 3a, where τ and ν achieve the best performance at 0.3 and 1.4 respectively. When ν is less than 1, we will pay more attention to new-coming objects and decrease the performance as shown in Tab. 3b.

4.4. Further Analysis

Robustness at different speeds. We further test the robustness of our model at different moving speeds of the driving vehicle. To simulate the static (0× speed) and faster speed (2×) environments, we re-sample the video frames to build new datasets. For 0× speed setting, we treat it as a special driving state and re-sample the frames to the triplet (F_t, F_t, G_t). It means the previous and current frames have no change and the model should predict the non-changed results. For 2× speed setting, we re-build the triplet data as (F_{t-2}, F_t, G_{t+2}). This indicates the faster moving speed of both the ground truth objects and the driving vehicle.

Results are listed in Tab. 4. For 0× speed, the predicting results are supposed to be the same as the offline setting. However, if we only adopt the basic pipeline, we can see a significant performance drop (-1.9 AP) compared to the offline, which means the model fails to deduce the static state. By adopting the DFP module into the basic pipeline, we recover this reduction and achieve the same results as the offline performance. It reveals that DFP, especially the static flow, is a key to extracting the right moving trend and assisting in prediction. It is also worth noting that at 0× speed, all the weights in TAL are one thus it has no influence. For 2× speed, as the objects move faster, the gap between offline and streaming perception is further expanded. Meanwhile, the improvements from our models, including the basic pipeline, DFP, and TAL, are also enlarged. These robustness results further manifest the superiority of our method.

Model	Pipe.	DFP	TAL	Off AP	sAP_{0x}	sAP_{1x}	sAP_{2x}
YOLOX-L				38.3	38.3	31.2	24.9
	✓				36.4	34.2	31.3
	✓	✓			38.3	35.5	32.9
	✓	✓	✓		38.3	36.1	33.3

Table 4. Results on different moving speed settings. The 0× static setting actually equals to the offline setting. Subscripts indicate different moving speeds.

Forecasting manner	sAP_{1x}	sAP_{2x}	Extra Latency
Offline Det	31.2	24.9	0 ms
KF Forecasting	35.5	31.8	3.11 ms
Ours (E2E)	**36.1**	**33.3**	**0.8 ms**

Table 5. Comparison results on different forecasting manners.

Method	sAP	sAP_{50}	sAP_{75}	sAP_s	sAP_m	sAP_l
Non-real-time methods						
Streamer (size=900) [26]	18.2	35.3	16.8	4.7	14.4	34.6
Streamer (size=600) [26]	20.4	35.6	20.8	3.6	18.0	47.2
Streamer + AdaScale [8, 16]	13.8	23.4	14.2	0.2	9.0	39.9
Adaptive Streamer [16]	21.3	37.3	21.1	4.4	18.7	47.1
Real-time methods						
1st place (size=1440)† [59]	40.2	68.9	39.4	21.5	42.9	53.9
2nd place (size=1200)† [20]	33.2	58.6	30.9	13.3	31.9	40.0
Ours-S	28.8	50.3	27.6	9.7	30.7	53.1
Ours-M	32.9	54.0	32.5	12.4	34.8	58.1
Ours-L	36.1	57.6	35.6	13.8	37.1	63.3
Ours-L (size=1200)†	42.3	64.5	46.4	23.9	45.7	68.1

Table 6. Performance comparison with state-of-the-art approaches on Argoverse-HD dataset. Size means the shortest side of input image and the input image resolution is 600×960 for our models. '†' means using extra dataset and TensorRT.

Comparison with Kalman Filter based forecasting. We follow the implementation of [26] and report the advanced baseline of Kalman Filter based forecasting in Tab. 5. For ordinary sAP (1×), our end-to-end method still outperforms

Figure 6. Visualization results of the baseline detector and the proposed method. The green boxes represent ground truth boxes, while red ones represent prediction results.

the advanced baseline by 0.5 AP. Further, when we simulate and evaluate them with faster moving (2×), our model shows more superiority of robustness (33.3 sAP v.s. 31.8 sAP). Besides, our model brings less extra latency (0.8 ms v.s. 3.1 ms taking the average of 5 tests).

Visualization results As shown in Fig. 6, we present the visualization results. For the baseline detector, the predicting bounding boxes encounter severe time lag. The faster the vehicles and pedestrians move, the larger the predictions shift. For small objects like traffic lights, the overlap between predictions and ground truth becomes small and is even non. In contrast, our method alleviates the mismatch and fits accurately between the predicting boxes and moving objects. It further confirms the effectiveness of our method.

Comparison with state-of-the-art. We compare our method with other state-of-the-art detectors on Argoverse-HD dataset. As shown in Fig. 6, real-time methods show absolute advantages over non-real-time detectors. We also report the results of the 1st and 2nd place in Streaming Perception Challenge. They involve extra datasets and accelerating tricks, while our methods get competitive performance and even surpass the accuracy of the 2nd place without any tricks. Once we adopt the same tricks, our method outperforms the 1st place by a significant margin (2.1 sAP).

5. Conclusion

This paper focuses on a streaming perception task that takes the processing latency into account. Under this metric, we reveal the superiority of using a real-time detector with the ability of future prediction for online perception. We further build a real-time detector with Dual-Flow Perception module and Trend-Aware Loss, alleviating the time lag problem in streaming perception. Extensive experiments show that our simple framework achieves state-of-the-art performance. It also obtains robust results on different speed settings. We hope that our simple and effective design will motivate future efforts in this practical and challenging perception task.

Limitations In real-world scenarios, the assumption of real-time processing may be violated due to limited hardware resources or high-resolution input. Moreover, we can see that the gap between *offline* setting and our model of online perception still exists by a large margin, indicating that there is still unexplored room for streaming perception.

Acknowledge This paper is supported by the National Key R&D Plan of the Ministry of Science and Technology (Project No. 2020AAA0104400). It was also funded by China Postdoctoral Science Foundation (2021M690375) and Beijing Postdoctoral Research Foundation.

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019. 2

[2] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Bayesian prediction of future street scenes using synthetic likelihoods. In *In International Conference on Learning Representations*, 2018. 2

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1, 2

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 3

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 2, 5

[6] 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

[7] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10337–10346, 2020. 2

[8] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. In *In Proceedings of Machine Learning and Systems*, 2019. 7

[9] Hsu-kuang Chiu, Ehsan Adeli, and Juan Carlos Niebles. Segmenting the future. *IEEE Robotics and Automation Letters*, 5(3):4202–4209, 2020. 2

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 3

[11] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. Relation distillation networks for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7023–7032, 2019. 2

[12] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. Ota: Optimal transport assignment for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 303–312, 2021. 3

[13] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 1, 2, 3, 5

[14] Zheng Ge, Jianfeng Wang, Xin Huang, Songtao Liu, and Osamu Yoshie. Lla: Loss-aware label assignment for dense pedestrian detection. *Neurocomputing*, 462:272–281, 2021. 2

[15] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928, 2021. 3

[16] Anurag Ghosh, Akshay Nambi, Aditya Singh, Harish YVS, and Tanuja Ganu. Adaptive streaming perception using deep reinforcement learning. *arXiv preprint arXiv:2106.05665*, 2021. 1, 2, 7

[17] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[18] glenn jocher. yolov5. *https://github.com/ultralytics/yolov5*, 2021. 1, 2, 6

[19] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[20] Yongxiang Gu, Qianlei Wang, and Xiaolin Qin. Real-time streaming perception system for autonomous driving. *arXiv preprint arXiv:2107.14388*, 2021. 2, 7

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 3

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 5, 6

[23] Jian-Fang Hu, Jiangxin Sun, Zihang Lin, Jian-Huang Lai, Wenjun Zeng, and Wei-Shi Zheng. Apanet: Auto-path aggregation for future instance segmentation prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[24] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 5, 6

[25] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering 82(Series D)*, pages 35–45, 1960. 2

[26] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020. 1, 2, 3, 5, 7

[27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[28] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[30] Zihang Lin, Jiangxin Sun, Jian-Fang Hu, Qizhi Yu, Jian-Huang Lai, and Wei-Shi Zheng. Predictive feature learning for future segmentation prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7365–7374, 2021. 2

[31] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 385–400, 2018. 1, 2

[32] Songtao Liu, Di Huang, and Yunhong Wang. Adaptive nms: Refining pedestrian detection in a crowd. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6459–6468, 2019. 2

[33] Songtao Liu, Di Huang, and Yunhong Wang. Learning spatial fusion for single-shot object detection. *arXiv preprint arXiv:1911.09516*, 2019. 1

[34] Songtao Liu, Di Huang, and Yunhong Wang. Pay attention to them: deep reinforcement learning-based cascade object detection. *IEEE transactions on neural networks and learning systems*, 31(7):2544–2556, 2019. 2

[35] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[36] Pauline Luc, Camille Couprie, Yann Lecun, and Jakob Verbeek. Predicting future instance segmentation by forecasting convolutional features. In *Proceedings of the european conference on computer vision (ECCV)*, pages 584–599, 2018. 2

[37] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 648–657, 2017. 2

[38] Yuchen Ma, Songtao Liu, Zeming Li, and Jian Sun. Iqdet: Instance-wise quality distribution sampling for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1717–1725, 2021. 2

[39] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. Borderdet: Border feature for dense object detection. In *European Conference on Computer Vision*, pages 549–564. Springer, 2020. 2

[40] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7:1, 2017. 5

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2, 3

[42] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1, 2, 3

[43] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 2, 3

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 2

[45] sAP. Code for towards streaming perception. https://github.com/wkentaro/labelme, 2021. 5

[46] Josip Šarić, Marin Oršić, Tonći Antunović, Sacha Vražić, and Siniša Šegvić. Single level feature-to-feature forecasting with deformable convolutions. In *German Conference on Pattern Recognition*, pages 189–202. Springer, 2019. 2

[47] Josip Saric, Marin Orsic, Tonci Antunovic, Sacha Vrazic, and Sinisa Segvic. Warp to the future: Joint forecasting of features and feature motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10648–10657, 2020. 2

[48] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11563–11572, 2020. 3

[49] Jiangxin Sun, Jiafeng Xie, Jian-Fang Hu, Zihang Lin, Jianhuang Lai, Wenjun Zeng, and Wei-Shi Zheng. Predicting future instance segmentation with contextual pyramid convlstms. In *Proceedings of the 27th acm international conference on multimedia*, pages 2043–2051, 2019. 2

[50] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15539–15548, 2021. 2

[51] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 2

[52] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 5, 6

[53] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *European conference on computer vision*, pages 456–472. Springer, 2020. 2

[54] Shengkai Wu, Jinrong Yang, Xinggang Wang, and Xiaoping Li. Iou-balanced loss functions for single-stage object detection. *Pattern Recognition Letters*, 2022. 5

[55] Shengkai Wu, Jinrong Yang, Hangcheng Yu, Lijun Gou, and Xiaoping Li. Gaussian guided iou: A better metric for balanced learning on object detection. *arXiv preprint arXiv:2103.13613*, 2021. 5

[56] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10186–10195, 2020. 3

[57] 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*, 2020. 3

[58] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3, 6

[59] Songyang Zhang, Lin Song, Songtao Liu, Zheng Ge, Zeming Li, Xuming He, and Jian Sun. Workshop on autonomous driving at cvpr 2021: Technical report for streaming perception challenge. *arXiv preprint arXiv:2108.04230*, 2021. 2, 3, 7

[60] Yangtao Zheng, Di Huang, Songtao Liu, and Yunhong Wang. Cross-domain object detection through coarse-to-fine feature adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13766–13775, 2020. 2

[61] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. Autoassign: Differentiable label assignment for dense object detection. *arXiv preprint arXiv:2007.03496*, 2020. 2

[62] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017. 2