DaDe: Delay-Adaptive Detector for Streaming Perception

Wonwoo Jo[1,*], Kyungshin Lee[2,†], Jaewon Baik[1,*], Sangsun Lee[1,*], Dongho Choi[1,*]
and Hyunkyoo Park[1,*]
[1]*C&BIS Co.,Ltd., Republic of Korea*
[2]*Independent Researcher, Republic of Korea*

Keywords: Streaming Perception, Real-Time Processing, Object Detection.

Abstract: Recognizing the surrounding environment at low latency is critical in autonomous driving. In real-time environment, surrounding environment changes when processing is over. Current detection models are incapable of dealing with changes in the environment that occur after processing. Streaming perception is proposed to assess the latency and accuracy of real-time video perception. However, additional problems arise in real-world applications due to limited hardware resources, high temperatures, and other factors. In this study, we develop a model that can reflect processing delays in real time and produce the most reasonable results. By incorporating the proposed feature queue and feature select module, the system gains the ability to forecast specific time steps without any additional computational costs. Our method is tested on the Argoverse-HD dataset. It achieves higher performance than the current state-of-the-art methods(2022.12) in various environments when delayed . The code is available at https://github.com/danjos95/DADE.

1 INTRODUCTION

Recognizing the surrounding environment and reacting with low latency is critical in autonomous driving for a safe and comfortable driving experience. In practice, the gap between the input data and the surrounding environment widens as the processing latency increases. As shown in Figure 1, surrounding environment changes from sensor input time t to $t+n$ when the model finishes processing. To address this issue, some detectors (Redmon and Farhadi, 2018)(Ge et al., 2021) are focused on lowering the latency. They can complete the entire processing before the next sensor input. It appears reasonable but the processing delay caused a gap between the results and the changing environment. Furthermore, current image detection metrics such as average precision, and mean average precision do not consider a real-time online perception environment. It causes detectors to prioritize accuracy over the delay. To evaluate streaming performance, (Li et al., 2020) proposed a new metric, "streaming accuracy" to integrate latency and accuracy into a single metric for real-time online perception. Strong detectors (He et al., 2017)(Cai and Vasconcelos, 2018) showed a significant performance drop in streaming perception. StreamYOLO (Yang et al., 2022) created the model to predict future frames by combining the previous and current frames.

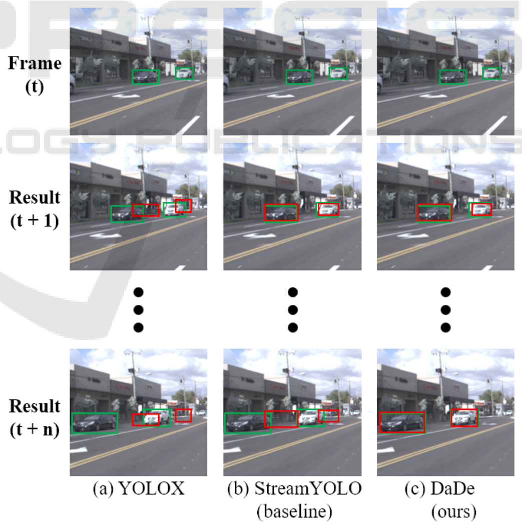

Figure 1: Visualization of the results of detectors (red boxes). Ground Truth (green boxes) changes due to image flow. Our method can detect objects correctly even after multiple time-step passes during processing.

It achieved state-of-the-art streaming AP performance by successfully detecting future frame-keeping real-time performance. In the real-world environment, processing time varies for various reasons(e.g. Temperature, Multi-modal Multi-task and sequential processing) while running the model. It requires detectors with the ability to perform multi-time-step predictions. For example, in Figure 1(b), StreamYOLO outperforms the baseline detector in predicting the next time-step frame when the processing time is shorter than the inter-frame time. When processing time exceeds inter-frame time, the system begins to collapse and produces poor prediction results, as shown in Result $(t+n)$ in Figure 1. The proposed system can function properly in an unexpectedly long processing time environment and provide the best detection at the time. In this study, we construct a model that provides insight into the required time-step frame. The feature queue module saves multi-time-step image features. It prevents the need for additional feature extraction processes, which incur additional computational costs. The dynamic feature select module constantly monitors the processing delay and chooses the best feature for the estimated delay. We use StreamYOLO's dual flow perception (DFP) module, which fuses the previous and current frames to make objects moving trend. The model can make moving trend of the target time step by fusing features from the dynamic feature select module. Experiments are conducted on Argoverse-HD (Chang et al., 2019) dataset. We tested delays in various settings. Our method showed significant improvement when compared to the baseline (StreamYOLO). As the mean processing delay increased, the difference between the baseline and ours widened.

This research makes three main contributions as follows:

- We introduce delay-adaptive detector (DaDe), which can produce future results that are tailored to the output environment. Processing delay cannot be stable in a real-time environment and must be considered. We improved the baseline so that it could handle unexpected delays. This process has no computational cost or accuracy trade-off.
- We create a simple feature control module that can choose the best image feature based on the current delay trend. The feature queue module stores previous features to avoid additional computation. The feature select module monitors pipeline delays and chooses the best feature to create the moving trend. We can achieve accurate target time results by using the DFP module from StreamYOLO.
- Our system outperformed the state-of-the-art (StreamYOLO) model in delay-varying environments. We changed the mean delay time to stimulate delayed environments during the evaluation. Our method achieved $0.7 \sim 1.5\%$ higher sAP compared to the state-of-the-art method. This work also shows considerable room for a delay-critical system.

2 RELATED WORK

2.1 Image Object Detection

Both latency and accuracy are critical factors in object detection. Two-stage detectors (Girshick et al., 2013)(Girshick, 2015)(He et al., 2017)(Lin et al., 2016) use a regional proposal system to focus on the accuracy of offline applications. However, in the real world, latency is more important than accuracy because the environment changes throughout processing. One-stage detectors (Lin et al., 2017)(Ma et al., 2021) have an advantage in processing time compared to regional proposed methods. YOLO series (Redmon and Farhadi, 2018)(Ge et al., 2021) is one of the mainstream performances in the one-stage detection method. Our research is based on the YOLOX (Ge et al., 2021) detector. YOLOX added many advanced detection technologies (anchor free, decoupled head, etc.) from YOLOv3 (Redmon and Farhadi, 2018) to achieve powerful performance.

2.2 Video Detection and Prediction

There have been attempts to improve the detection performance by using previous images from the video stream. Recent methods, such as (Bergmann et al., 2019)(Chen et al., 2020)(Deng et al., 2019)(Zhu et al., 2017) use attention, optical flow, and a tracking method to aggregate image features and achieve high detection accuracy. The video future frame predictor (Bhattacharyya et al., 2019)(Luc et al., 2017) creates unobserved future images from previous images. The ConvLSTM-based autoencoder (Chang et al., 2021)(Lee et al., 2021)(Chaabane et al., 2020)(Jin et al., 2020)(Wang et al., 2020) generates representations of previous frames, and then the decoder generates future frame pixels based on those representations. However, they cannot be used in real-time applications because they are designed for an offline environment and do not account for latency.

2.3 Progressive Anytime Algorithm

There are previous studies on planning systems under resource constraints (Boddy and Dean, 1994) and flexible computation. The anytime algorithm can return results at any point in time. The quality of the results gradually increases as the processing time increases (Zilberstein, 1996). However, the preceding studies do not consider environmental changes that occur during processing. It bridges the gap between the actual environment and the results. Our efforts are aimed at making the model more robust, even in de-

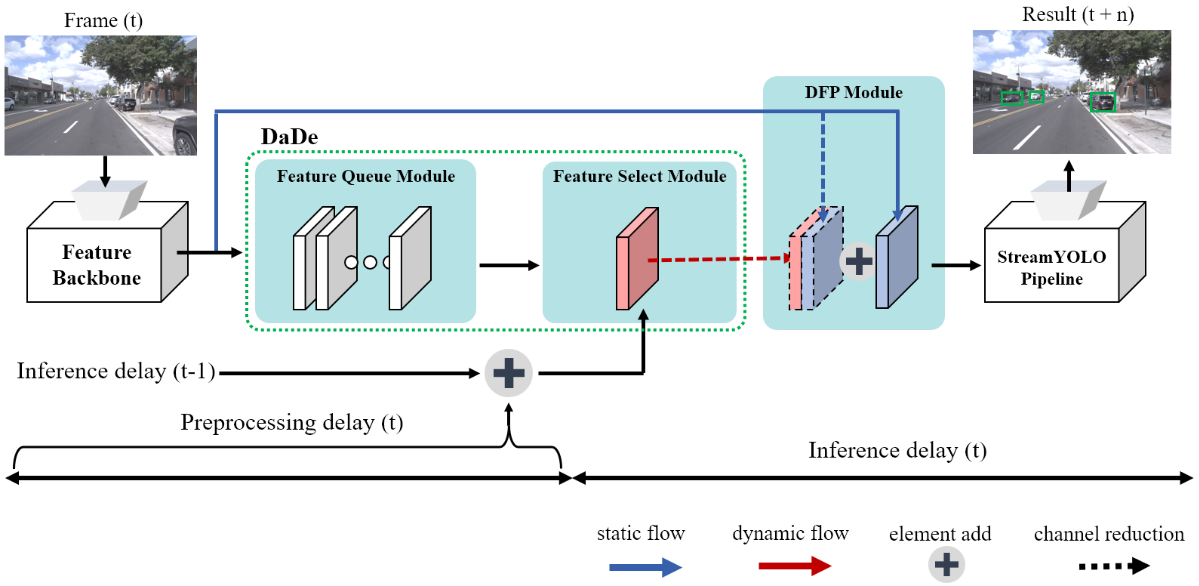

Figure 2: Delay-adaptive Detector is illustrated for delay adaptive stream perception. The image feature is extracted by CSPDarknet-53 and PANet, and stored in the fixed-length Feature Queue Module. The Feature Select Module investigates delay using the currently performed preprocessing delay and the most recent inference delay. The Feature Select Module chooses appropriate features from a list of stored features. Selected features are used to perform detection in the StreamYOLO pipeline.

layed situations. It can produce appropriate results to the current time even when processing time becomes unexpectedly long.

2.4 Streaming Perception

There are two types of detectors in a real-time system: real-time and non-real-time. A real-time detector can complete all processing steps before the next frame arrives. There was no metric to evaluate considering both latency and accuracy. As latency becomes more important in a real-world application, (Li et al., 2020) integrates latency and accuracy into a single metric. Streaming AP (sAP) was proposed to evaluate accuracy while considering time delay. It also suggests methods for recovering sAP performance drops, such as decision-theoretic scheduling, asynchronous tracking, and future forecasting. StreamYOLO (Yang et al., 2022) reduced streaming perception to the task of creating a real-time detector that predicts the next frame. It achieved state-of-the-art performance, but when real-time processing is violated, the system can collapse because it can only produce the next time-step result.

2.5 Delay Variance

If an additional delay occurs while processing, the gap between the results and the surrounding environment grows larger. This leads to a bad decision, which can lead to serious safety issues. Delays can occur for various reasons while running real-world applications.

2.5.1 Temperature

Temperature is a well-known issue in a real-time system. Performance may be reduced to avoid permanent damage to the processing chips. In our hardware environment, processing speed drops to 70% at 90°C, with a performance drop of up to 50% at 100°C. In autonomous driving, the temperature control gets harder since the device operates in various outdoor environments.

2.5.2 Multiple Sensor Configuration

It is critical to use multiple sensors to perceive the surrounding environment to obtain diverse data or a multi-view of the environment. It is becoming more common to use lidar-camera fusion (Liu et al., 2022)(Chen et al., 2016) or multi-camera system (Li et al., 2022)(Wang et al., 2021) to make surrounding recognition. Each sensor has a unique frequency and time delay. Here, fast sensor data must wait until all sensor data arrives, which can add to the delay.

2.5.3 Multiple Model Deploy

To build an autonomous driving system, multiple tasks (for example, lane segmentation, traffic light detection, 3D object detection, multi-object tracking, and so on) must be performed parallel and dynamically. (Lobanov and Sholomov, 2021) proposes a multiple detection model with a shared backbone to reduce computational costs. In any case, running multiple models can cause an unexpected delay in the system's sequential task. If the processing delay exceeds the inter-frame spacing, the real-time system fails and the detector results become out-of-date. We changed the mean delay time to stimulate delayed environments during the evaluation.

2.6 Preprocessing and Inference Delay

There are preprocessing and inference stages in the processing sensor system. The main processing unit receives raw sensor data and refines it to create a proper shape for the model during the preprocessing stage. Following preprocessing, input data flows into a neural processing unit, which produces results through neural networks. Since it runs sequentially, preprocessing is fast but the delay can be easily changed by a processing bottleneck. As the task becomes more demanding, the number of sequential operations in the main processing unit increases, which can directly affect the delay.

3 METHODOLOGY

In this study, we present a simple and effective method for achieving the desired future results. As in Figure 2, the Delay-adaptive Detector (DaDe) can return the desired time-step by equipping the feature queue module, feature select module, and Dual-Flow Perception (DFP) module. We avoid adding extra computation in processing to maintain real-time performance.

3.1 Dual Flow Perception Module

StreamYOLO (Yang et al., 2022) suggested a DFP module for tracking moving trends. DFP uses the previous frame's feature map. Dynamic flow in DFP fuses the features of two adjacent frames to create an object moving trend, whereas static flow is created using a standard feature extraction based on the current frame only. 1x1 convolution layer is used for dynamic flow, which is followed by the batchnorm and SiLU (Ramachandran et al., 2017) to reduce the channel for previous and current features in half. Then, DFP concatenates these two reduced features to generate the dynamic features. The detector can sense the object's moving trends and detect future frames without additional latency by combining dynamic flow and static flow. To advance this method, we use a multi-time-step feature fusing system to create a delay adaptive detector. The model can make a multi-time-step prediction by following the feature queue and feature select module.

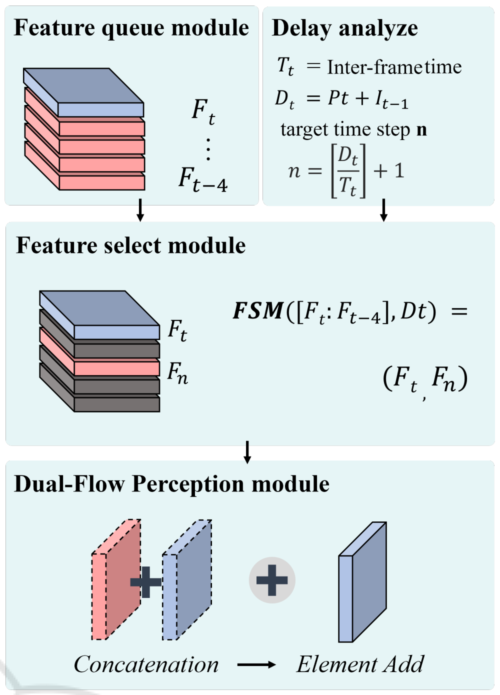

Figure 3: Pipeline of DaDe. The feature queue of the module stores previous image features to avoid additional computation. The feature select module selects features from queues based on the current delay time. The dual flow perception module generates object moving trends based on input features.

3.2 Feature Queue Module

The feature queue module stores extracted image features in a featured queue of a fixed length. Features are represented as F_t, whereas t is the corresponding time-step. The computational cost of saving old image features is as same as StreamYOLO. The feature queue module continues to save features from the four previous frames. Storing the previous feature prevents additional processing while occupying little additional memory (100MB for each frame feature). This enables the model to use older time-step features.

3.3 Delay Analyze

Preprocessing and inference time in time step t are denoted as P_t and I_t respectively. The inter-frame time between input frames is denoted by the symbol T_t. The delay trend can be calculated by adding the most recent inference delay and the current preprocessing delay. The final delay trend D_t is as follows:

$$D_t = P_t + I_{(t-1)} \tag{1}$$

3.4 Feature Select Module

The feature select module chooses two time-step features that best match the current delay trend. When the delay trend is entered into the feature select module, the target time-step n can be calculated using Equation 2. The expected time-step is the delay trend quotient over inter-frame time. The target time-step n is the next time-step of the division result.

$$n = \left[\frac{D_t}{T_t} \right] + 1 \tag{2}$$

In the feature select module, features for the required time step are extracted as the equation below.

$$\mathbf{FSM}([F_t : F_{(t-4)}], D_t) = (F_t, F_{(t-n)}) \tag{3}$$

If there is no last inference delay, the result of FSM should be (F_t, F_{t-1}). If no suitable feature exists for n, the nearest F should be chosen. In summary, the proposed model calculates the time delay by combining the current preprocessing delay and the last inference delay. After determining the desired time-step using Equation 2, the module chooses the stored features using Equation 3. The DFP creates a dynamic time-step object moving trend using the selected feature. The entire feature flow process is explained in Figure 3.

4 EXPERIMENTS

4.1 Environment Settings

4.1.1 Dataset

Argoverse-HD (Chang et al., 2019) provides high-resolution (1920x1080) and high-frame-rate driving video image data (30FPS). The Argoverse-HD validation set contains 24 videos of $15 \sim 30$ seconds each, totaling 15k image frames. Streaming perception (Li et al., 2020) adds dense 2D annotations in MS COCO (Lin et al., 2014) format.

Table 1: Delay analysis in various delay environments. All delays are measured in milliseconds.

Environment	Mean delay	Standard deviation	Minimum delay	Maximum delay
Low				
StreamYOLO	23.5	3.2	21.8	69.1
DADE(Ours)	24.1	3.66	21.9	66
Medium				
StreamYOLO	40.1	9.35	22.3	86.8
DADE(Ours)	39.3	9.22	22.3	88
High				
StreamYOLO	63	12.5	41.7	121
DADE(Ours)	63.1	12.7	41.3	124

4.1.2 Streaming AP

Streaming AP (Li et al., 2020) is used in the evaluation of experiments. During evaluation, the model updates the output buffer with its latest prediction of the current state of the world. At the same time, the benchmark constantly queries the output buffer for estimates of world states. So, output will be evaluated with the current ground truth frame in world state. To AP metric from COCO (Lin et al., 2014), sAP evaluates the average mAP over intersection-over-union (IoU) thresholds at 0.5 and 0.75. sAP_S, sAP_M, and sAP_L denote sAP for object size.

4.1.3 Platforms

Each experiment in this study was run on an RTX 3070Ti GPU (8GB VRAM, TDP 80W), an Intel 12700 H CPU, and 16 GB RAM. Pytorch 1.7.1 with CUDA 11.4 was used in the software environment.

4.1.4 Delay

To change mean delay time, we deployed multiple DaDe model on single environment. Table 1 shows various delay environments (Low, Medium, and High) by running up to 3 models in parallel. Inter-frame space is $33ms$ at 30FPS, so processing time should be less than $33ms$ to maintain real-time performance. In a low delay environment, the average delay is $24ms$, so StreamYOLO and DaDe can process each frame before the next frame arrives. In a medium delay environment, the average delay is $40ms$, after which some frames fail to meet real-time performance. As a result, their evaluation is based on frame$(t+2)$ rather than frame$(t+1)$. The mean delay time in a high delay environment is more than $60ms$, so the difference between the assumed future frame and the real environment increases in the baseline. Figure 4 shows the processing delay distribution in each delay environment. A vertical dotted

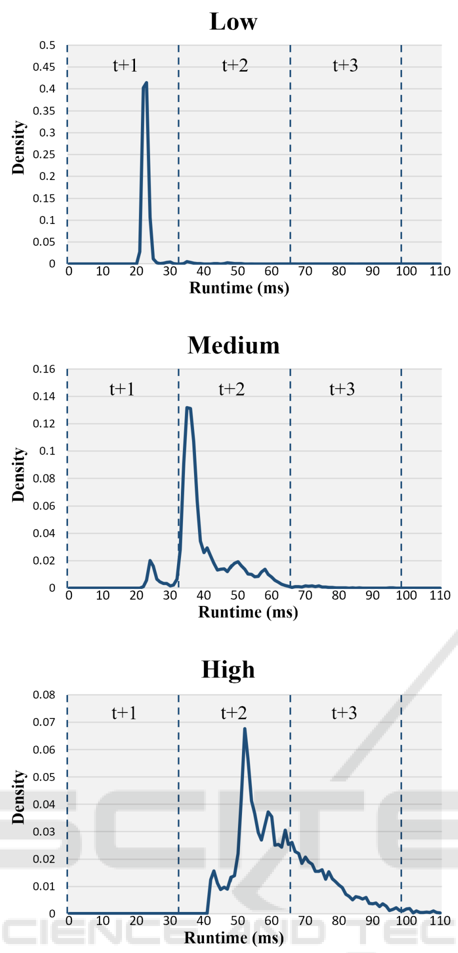

Figure 4: Visualization of delay distribution for various delay environments. Distribution gradually spreads as the mean delay lengthens.

line shows inter-frame spaces at 30 frames per second (FPS). Frames in each inter-frame space should produce corresponding time-step results.

4.2 Results

Table 2 shows the results of streaming perception methods. The performance is the same as in a low-delay environment between baseline (StreamYOLO) and ours (DaDe). Standard YOLOX suffers from a significant performance drop in the absence of future prediction. The baseline fails to follow the trend of future frames as the mean delay increases and the frame begins to fail to finish processing before the next frame arrives. In high-delay conditions, all frames fail to detect in real-time, increasing the gap between the baseline and the suggested method. As shown in Table 2, a 1.5% improvement in sAP is achieved compared to the baseline in a high delay environment.

Table 2: Performance of each method in Argoverse-HD dataset. YOLOX is the standard real-time detector.

Method	sAP	sAP_L	sAP_M	sAP_S	sAP_{50}	sAP_{75}
			Low (mean delay: 24 ± 1ms)			
YOLOX	31.7	52.3	29.7	12.5	54.8	30.1
StreamYOLO	**36.7**	63.8	**37.0**	**14.6**	**57.9**	37.2
DADE(Ours)	**36.7**	**63.9**	36.9	**14.6**	**57.9**	**37.3**
			Medium (mean delay: 39 ± 1ms)			
YOLOX	23.0	40.0	21.3	6.1	45.5	21.1
StreamYOLO	29.1	48.1	27.4	9.5	51.6	27.7
DADE(Ours)	**29.8**	**50.1**	**28.3**	**10.8**	**52.5**	**28.4**
			High (mean delay: 63 ± 1ms)			
YOLOX	19.2	31.2	15.5	5.1	38.8	16.3
StreamYOLO	25.7	40.2	23.9	8.6	47.3	23.7
DADE(Ours)	**27.2**	**41.5**	**26.1**	**9.5**	**48.2**	**25.9**

Figure 5 shows the visualization of each method in a high delay environment. Ours outperforms others in various conditions and scenarios.

5 CONCLUSIONS

This study proposes a simple but effective method for detecting objects in real time. Multi-time-step prediction can be performed using the feature queue module and feature select module without any additional computation. It achieved state-of-the-art performance in a delayed environment. Furthermore, real-time feature operation in DaDe can significantly improve the performance of other real-world object detection tasks. Predicting the appropriate time step can significantly improve their localization accuracy. For further research, only one time-step feature is considered in our method to generate the object's moving trend. Accuracy can be improved if each time-step feature is considered within a reasonable computation cost.

REFERENCES

Bergmann, P., Meinhardt, T., and Leal-Taixé, L. (2019). Tracking without bells and whistles. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 941–951. IEEE.

Bhattacharyya, A., Fritz, M., and Schiele, B. (2019). Bayesian prediction of future street scenes using synthetic likelihoods. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Boddy, M. S. and Dean, T. L. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artif. Intell.*, 67(2):245–285.

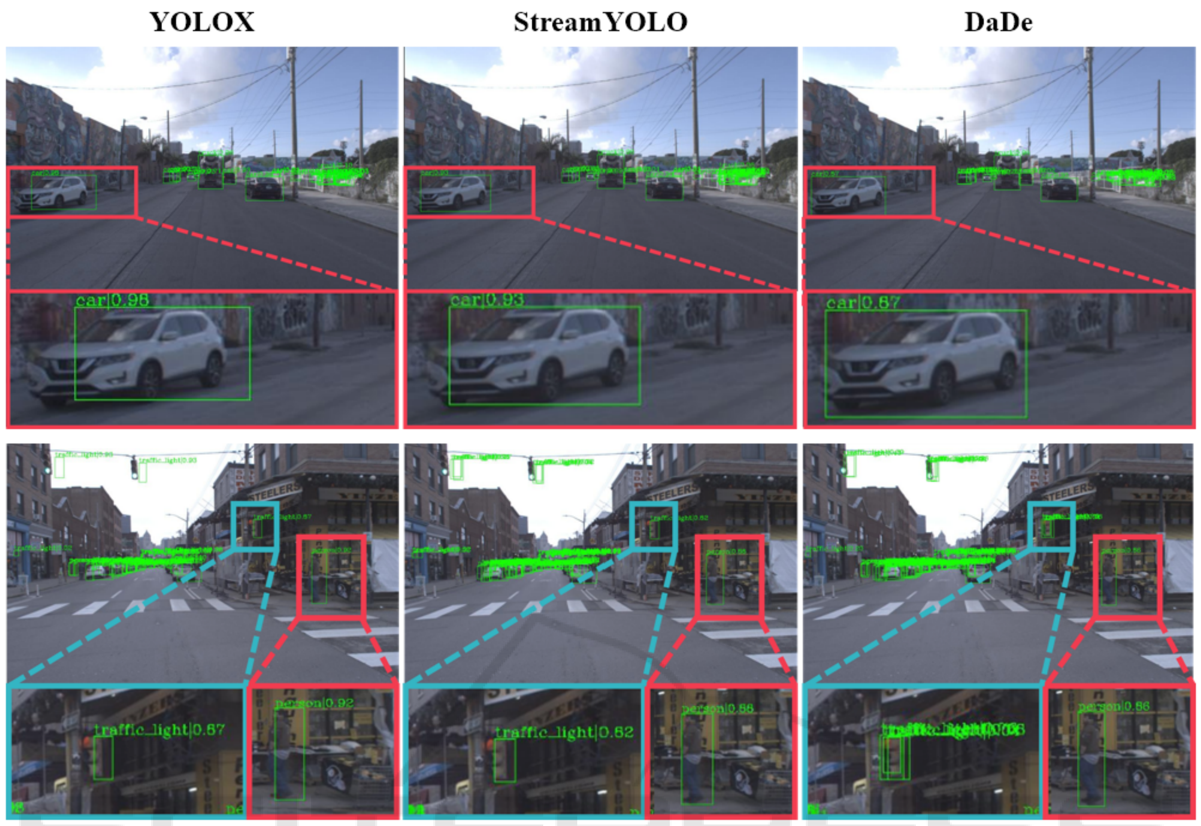

Figure 5: Qualitative analysis of results. Detection is conducted in a streaming environment with a high delay. DaDe can successfully predict the future environment by forming object-moving trends. It achieves better results not only for moving objects (top row) but also for detecting objects during ego-vehicle turns (bottom row).

Cai, Z. and Vasconcelos, N. (2018). Cascade R-CNN: delving into high quality object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6154–6162. Computer Vision Foundation / IEEE Computer Society.

Chaabane, M., Trabelsi, A., Blanchard, N., and Beveridge, J. R. (2020). Looking ahead: Anticipating pedestrians crossing with future frames prediction. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 2286–2295. IEEE.

Chang, M., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 8748–8757. Computer Vision Foundation / IEEE.

Chang, Z., Zhang, X., Wang, S., Ma, S., Ye, Y., Xinguang, X., and Gao, W. (2021). MAU: A motion-aware unit for video prediction and beyond. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 26950–26962.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2016). Multi-view 3d object detection network for autonomous driving.

Chen, Y., Cao, Y., Hu, H., and Wang, L. (2020). Memory enhanced global-local aggregation for video object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10334–10343. Computer Vision Foundation / IEEE.

Deng, J., Pan, Y., Yao, T., Zhou, W., Li, H., and Mei, T. (2019). Relation distillation networks for video object detection. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7022–7031. IEEE.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988. IEEE Computer Society.

Jin, B., Hu, Y., Tang, Q., Niu, J., Shi, Z., Han, Y., and Li, X. (2020). Exploring spatial-temporal multi-frequency analysis for high-fidelity and temporal-consistency video prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 4553–4562. Computer Vision Foundation / IEEE.

Lee, S., Kim, H. G., Choi, D. H., Kim, H., and Ro, Y. M. (2021). Video prediction recalling long-term motion context via memory alignment learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3054–3063. Computer Vision Foundation / IEEE.

Li, M., Wang, Y., and Ramanan, D. (2020). Towards streaming perception. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J., editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, volume 12347 of *Lecture Notes in Computer Science*, pages 473–488. Springer.

Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., and Dai, J. (2022). Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*.

Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., and Belongie, S. J. (2016). Feature pyramid networks for object detection. *CoRR*, abs/1612.03144.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., and Han, S. (2022). Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv*.

Lobanov, M. and Sholomov, D. (2021). Application of shared backbone dnns in adas perception systems. *Proceedings of SPIE - The International Society for Optical Engineering*, 11605.

Luc, P., Neverova, N., Couprie, C., Verbeek, J., and LeCun, Y. (2017). Predicting deeper into the future of semantic segmentation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 648–657. IEEE Computer Society.

Ma, Y., Liu, S., Li, Z., and Sun, J. (2021). Iqdet: Instance-wise quality distribution sampling for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 1717–1725. Computer Vision Foundation / IEEE.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *CoRR*, abs/1710.05941.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Wang, Y., Guizilini, V., Zhang, T., Wang, Y., Zhao, H., and Solomon, J. (2021). DETR3D: 3d object detection from multi-view images via 3d-to-2d queries. In Faust, A., Hsu, D., and Neumann, G., editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 180–191. PMLR.

Wang, Y., Wu, J., Long, M., and Tenenbaum, J. B. (2020). Probabilistic video prediction from noisy data with a posterior confidence. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10827–10836. Computer Vision Foundation / IEEE.

Yang, J., Liu, S., Li, Z., Li, X., and Sun, J. (2022). Real-time object detection for streaming perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5395.

Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. (2017). Flow-guided feature aggregation for video object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 408–417. IEEE Computer Society.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *AI Mag.*, 17(3):73–83.