This WACV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

DyRoNet: Dynamic Routing and Low-Rank Adapters for Autonomous Driving Streaming Perception

Xiang Huang[1]*, Zhi-Qi Cheng[2]†, Jun-Yan He[3], Chenyang Li[3], Wangmeng Xiang[3], Baigui Sun[3]
[1]Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu, China
[2]School of Engineering and Technology, University of Washington, Tacoma, WA, USA
[3]Institute for Intelligent Computing, Alibaba Group, China
xianghuang@my.swjtu.edu.cn, zhiqics@uw.edu, {junyanhe1989,marquezxm}@gmail.com,
lichenyang.scut@foxmail.com, sunbaigui85@126.com

Abstract

The advancement of autonomous driving systems hinges on the ability to achieve low-latency and high-accuracy perception. To address this critical need, this paper introduces ***Dy****namic* ***Ro****utering* ***Net****work (*DyRoNet*), a low-rank enhanced dynamic routing framework designed for streaming perception in autonomous driving systems.* DyRoNet *integrates a suite of pre-trained branch networks, each meticulously fine-tuned to function under distinct environmental conditions. At its core, the framework offers a speed router module, developed to assess and route input data to the most suitable branch for processing. This approach not only addresses the inherent limitations of conventional models in adapting to diverse driving conditions but also ensures the balance between performance and efficiency. Extensive experimental evaluations demonstrating the adaptability of* DyRoNet *to diverse branch selection strategies, resulting in significant performance enhancements across different scenarios. This work not only establishes a new benchmark for streaming perception but also provides valuable engineering insights for future work.*[1]

1. Introduction

In autonomous driving systems, it is crucial to achieve low-latency and high-precision perception. Traditional object detection algorithms [38], while effective in various contexts, often confront the challenge of latency due to inherent computational delays. This lag between algorithmic processing and real-world states can lead to notable discrepancies between predicted and actual object locations. Such latency issues have been extensively reported and are known to significantly impact the decision-making process in autonomous driving systems [3].

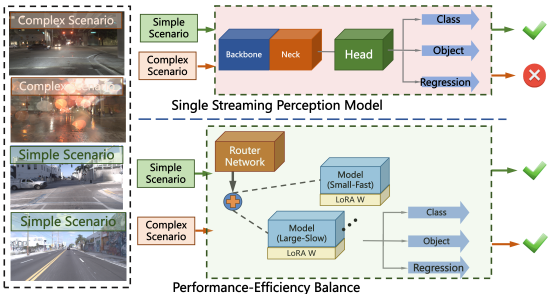

Figure 1. Illustration of DyRoNet's adaptive selection mechanism in streaming perception, contrasting with static traditional methods in complex environments [Best viewed in color and enlarged].

Addressing these challenges, the concept of streaming perception has been introduced as a response [19]. This perception task aims to predict "future" results by accounting for the delays incurred during the frame processing stage. Unlike traditional methods that primarily focus on detection at a given moment, streaming perception transcends this limitation by anticipating future environmental states, and aligning perceptual outputs closer to real-time dynamics. This new paradigm is key in addressing the critical gap between real-time processing and real-world changes, thereby enhancing the safety and reliability of autonomous driving systems [21].

Although the existing streaming approach seems promising, it still faces contradictions in real-world scenarios. These contradictions primarily stem from the diverse and unpredictable nature of driving environments. The factors such as *camera motion*, *weather conditions*, *lighting variations*, and the presence of *small objects* seriously impact the performance of perception measures, leading to fluctuations that challenge their robustness and reliability (see Sec. 3.1). This complexity in real-world scenarios under-

*This work was completed during visit to CMU and Alibaba.
†Corresponding author, also a Visiting Assistant Professor at CMU.
[1]Project: https://tastevision.github.io/DyRoNet/

scores the limitations of a single, uniform model, which often struggles to adapt to the varied demands of different driving conditions [7]. In general, the challenges of streaming perception mainly include:

(1) *Diverse Scenario Distribution*: Autonomous driving environments are inherently complex and dynamic, showing a myriad of scenarios that a single perception model may not adequately address (see Fig. 1). The need to customize perception algorithms to specific environmental conditions, while ensuring that these models operate cohesively, poses a significant challenge. As discussed in Sec. 3.1, adapting models to various scenarios without compromising their core functionality is a crucial aspect of streaming perception.

(2) *Performance-Efficiency Balance*: To our knowledge, the integration of both large and small-scale models is essential to handle the varying complexities encountered in different driving scenes. The large models, while potentially more accurate, may suffer from increased latency, whereas smaller models may offer faster inference at the cost of reduced accuracy. Balancing performance and efficiency, therefore, becomes a challenging task. In Sec. 3.1, we explore the strategies for optimizing this balance, exploring how different model architectures can be effectively utilized to enhance streaming perception.

Generally speaking, these challenges highlight the demand for streaming perception. As we study in Sec. 3.1, addressing the *diverse scenario distribution* and achieving an *optimal balance between performance and efficiency* are key to advancing the state-of-the-art in autonomous driving. To address the intricate challenges presented by real-world streaming perception, we introduce *DyRoNet*, a framework designed to enhance dynamic routing capabilities in autonomous driving systems. *DyRoNet* stands as a low-rank enhanced dynamic routing framework, specifically crafted to cater to the requirements of streaming perception. It encapsulates a suite of pre-trained branch networks, each meticulously fine-tuned to optimally function under distinct environmental conditions. A key component of *DyRoNet* is the speed router module, ingeniously developed to assess and efficiently route input to the optimal branch, as detailed in Sec. 3.2. To sum up, the contributions are listed as:

- We emphasize the impact of environmental speed as a key determinant of streaming perception. Through analysis of various environmental factors, our research highlights the imperative need for adaptive perception responsive to dynamic conditions.

- By utilizing a variety of streaming perception techniques, *DyRoNet* provides the speed router as a major invention. This component dynamically determines the best route for handling each input, ensuring efficiency and accuracy in perception. The ability to adapt and be versatile is demonstrated by this dynamic route-choosing mechanism.

- Extensive experimental evaluations have demonstrated that *DyRoNet* is capable of adapting to diverse branch selection strategies, resulting in a substantial enhancement of performance across various branch structures. This not only validates the framework's wide-ranging applicability but also confirms its effectiveness in handling different real-world scenarios.

2. Related Work

This section revisits developments in streaming perception and dynamic neural networks, highlighting differences from our proposed *DyRoNet* framework. While existing methods have made progress, limitations persist in addressing real-world autonomous driving complexity.

2.1. Streaming Perception

The existing streaming perception methods fall into three main categories. (1) The initial methods focused on single-frame, with models like YOLOv5 [14] and YOLOX [5] achieving real-time performance. However, lacking motion trend capture, they struggle in dynamic scenarios. (2) The recent approaches incorporated current and historical frames, like StreamYOLO [33] building on YOLOX with dual-flow fusion. LongShortNet [18] used longer histories and diverse fusion. DAMO-StreamNet [9] added asymmetric distillation and deformable convolutions to improve large object perception. (3) Recognizing the limitations of single models, current methods explore dynamic multi-model systems. One approach [6] adapts models to environments via reinforcement learning. DaDe [13] extends StreamYOLO by calculating delays to determine frame steps. A later version [12] added multi-branch prediction heads. Beyond 2D detection, streaming perception expands into optical flow, tracking, and 3D detection, with innovations in metrics and benchmarks [23, 28, 30]. Distinct from these existing approaches, our proposed method, *DyRoNet*, introduces a low-rank enhanced dynamic routing mechanism specifically designed for streaming perception. *DyRoNet* stands out by integrating a suite of advanced branch networks, each fine-tuned for specific environmental conditions. Its key innovation lies in the speed router module, which not only routes input data efficiently but also dynamically adapts to the diverse and unpredictable nature of real-world driving scenarios.

2.2. Dynamic Neural Networks

Dynamic Neural Networks (DNNs) feature adaptive network selection, outperforming static models in efficiency and performance [8, 16, 34]. The existing research primarily focuses on structural design for core deep learn-

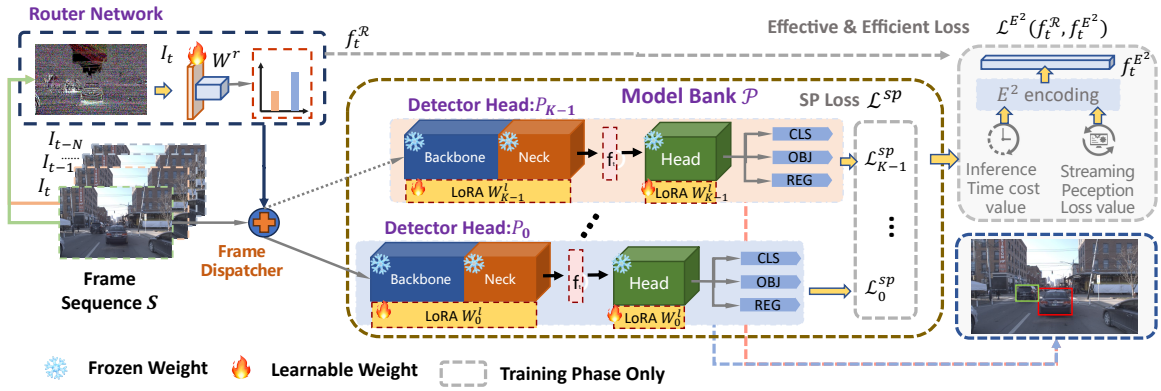

Figure 2. The *DyRoNet* Framework: This figure shows *DyRoNet*'s architecture with a multi-branch network. Two branches are illustrated, each as a streaming perception sub-network. The upper right details the core architecture. Each branch processes the current frame I_t and historical frames $I_{t-1}, I_{t-2}, \cdots, I_{t-n}$. Features are extracted by the backbone and neck, split into streams for current and historical frames, fused, then passed to the prediction head. The Speed Router selects the branch based on frame difference ΔI_t from I_t and I_{t-1}.

ing tasks like image classification [11, 27, 29]. DNNs follow two approaches: (1) Multi-branch models [1, 2, 17, 22, 24, 26, 31] rely on a lightweight router assessing inputs to direct them to appropriate branches, enabling tailored computation. (2) By generating new weights based on inputs [4, 25, 32, 37], these models dynamically alter computations to match diverse needs. DNN applications expand beyond conventional tasks. In object detection, DynamicDet [20] categorizes inputs and processes them through distinct branches. This illustrates DNNs' broader applicability and promising contributions for dynamic environments.

3. Proposed Method

This section outlines the framework of our proposed *DyRoNet*. Beginning with its underlying motivation and the critical factors driving its design, we subsequently provide an overview of its architecture and training process.

3.1. Motivation for DyRoNet

Autonomous driving faces variability from *weather*, *scene complexity*, and *vehicle velocity*. By strategically analyzing key factors and routing logic, this section details the rationale behind the proposed *DyRoNet*.

Analysis of Influential Factors. Statistical analysis of the Argoverse-HD dataset [19] underscores the profound influence of environmental dynamics on the effectiveness of streaming perception. While *weather* inconsistently impacts accuracy, suggesting the presence of other influential factors (see Appendix A.1), fluctuations in the *object count* show limited correlation with performance degradation (see Appendix A.2). Conversely, the presence of *small objects* across various scenes poses a significant challenge for detection, especially under varying motion states (see Appendix A.3). Notably, disparities in performance are most pronounced across different *environmental motion states* (see Appendix A.4), thereby motivating the need for a dynamic, velocity-aware routing mechanism in *DyRoNet*.

Rationale for Dynamic Routing. Analysis reveals that StreamYOLO's reliance on a single historical frame falters at high velocities, in contrast to multi-frame models, highlighting a connection between speed and detection performance (see Tab. 2). Dynamic adaptation of frame history, based on vehicular speed, enables *DyRoNet* to strike a balance between accuracy and latency (see Sec. 4.3). Through first-order differences, the system efficiently switches models to align with environmental motions. Specifically, the dynamic routing is designed to select the optimal architecture based on the vehicle's speed profile, ensuring precision at lower velocities for detailed perception and efficiency at higher speeds for swift response. These comprehensive analysis imforms *DyRoNet* as a robust solution for reliable perception across diverse autonomous driving scenarios.

3.2. Architecture of DyRoNet

Overview of *DyRoNet*. The structure of *DyRoNet*, as depicted in Fig. 2, proposes a multi-branch structure. Each branch within *DyRoNet* framework functions as an independent streaming perception model, capable of processing both the current and historical frames. This dual-frame processing is central to *DyRoNet*'s capability, facilitating a nuanced understanding of temporal dynamics. Such a design is key in achieving a delicate balance between latency and accuracy, aspects crucial for real-time autonomous driving.

Mathematically, the core of *DyRoNet* lies the processing of a frame sequence, $\mathcal{S} = \{I_t, \cdots, I_{t-N\delta t}\}$, where N indicates the number of frames and δt the interval between successive frames. The framework process is formalized as:

$$\mathcal{T} = \mathcal{F}(\mathcal{S}, \mathcal{P}, \mathcal{W}),$$

where $\mathcal{P} = \{P_0, \cdots, P_{K-1}\}$ denotes a collection of

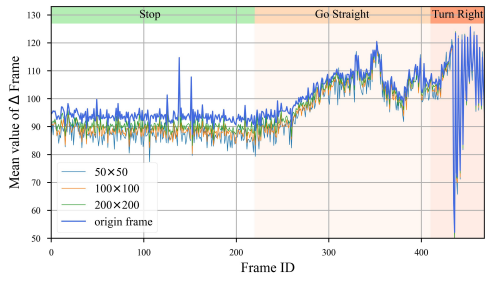

Figure 3. The mean curves of frame differences are depicted here. The four curves correspond to frame sizes of the original frame, 200×200, 100×100, and 50×50. Notably, these curves show distinct fluctuations across different vehicle motion cases.

streaming perception models, with each P_i denoting an individual model within this suite. The architecture is further enhanced by incorporating a feature extractor $\mathcal{G}_i$ and a perception head $\mathcal{H}_i$ for each model. The Router Network, $\mathcal{R}$, is instrumental in selecting the most suitable streaming perception model for each specific scenario.

Correspondingly, the weights of *DyRoNet* are denoted by $\mathcal{W} = \{W^d, W^l, W^r\}$, where W^d indicates the weights of the streaming perception model, W^l relates to the Low-Rank Adaptation (LoRA) weights within each model, and W^r pertains to the Router Network. The culmination of this process is the final output, $\mathcal{T}$, a compilation of feature maps. These maps can be further decoded through $Decode(\mathcal{T})$, revealing essential details like objects, categories, and locations.

Router Network. The *Router Network* in *DyRoNet* plays a crucial role in understanding and classifying the dynamics of the environment. This module is designed for both environmental classification and branch decision-making. To effectively capture environmental speed, frame differences are employed as the input to the Router Network. As shown in Fig. 3, frame differences exhibit a high discriminative advantage for different environmental speeds.

Specifically, for frames at times t and $t-1$, represented as I_t and I_{t-1} respectively, the frame difference is computed as $\Delta I_t = I_t - I_{t-1}$. The architecture of the Router Network, $\mathcal{R}$, is simple yet efficient. It consists of a single convolutional layer followed by a linear layer. The network's output, denoted as $f^r \in \mathbb{R}^K$, captures the essence of the environmental dynamics. Based on this output, the index σ of the optimal branch for processing the current input frame I_t is determined through the following equation:

$$\sigma = \arg\max_K(\mathcal{R}(\Delta I_t), W^r), \quad \sigma \in \{0, \cdots, K-1\}, \quad (1)$$

where σ is the index of the branch deemed most suitable for the current environmental context. Once σ is determined, the input frame I_t is automatically routed to the corresponding branch by a dispatcher.

In particular, this strategy of using frame differences to gauge environmental speed is efficient. It offers a faster alternative to traditional methods such as optical flow fields. Moreover, it focuses on frame-level variations rather than the speed of individual objects, providing a more generalized representation of environmental dynamics. The sparsity of ΔI_t also contributes to the robustness of this method, reducing computational complexity and making the Router Network's operations nearly negligible in the context of the overall model's performance.

Model Bank & Dispatcher. The core of the *DyRoNet* framework is its *model bank*, which consists of an array of streaming perceptual models, denoted as $\mathcal{P} = \{P_0, \cdots, P_{K-1}\}$. Typically, the selection of the most suitable model for processing a given input is intelligently managed by the *Router Network*. This process is formalized as $P_\sigma = \text{Disp}(\mathcal{R}, \mathcal{P})$, where Disp acts as a dispatcher, facilitating the dynamic selection of models from $\mathcal{P}$ based on the input. The operational flow of *DyRoNet* can be mathematically defined as:

$$\mathcal{T} = \mathcal{F}(\mathcal{S}, \mathcal{P}, W) = \text{Disp}(\mathcal{R}(\Delta I_t), \mathcal{P})(I_t; W^d_\sigma, W^l_\sigma),$$

where $\mathcal{R}$ symbolizes the *Router Network*, and ΔI_t refers to the frame difference, a key input for model selection. The weights W^d_σ and W^l_σ correspond to the selected streaming perception model and its LoRA parameters, respectively.

Note that the versatility of *DyRoNet* is further highlighted by its compatibility with a wide range of Streaming Perception models, even ones that rely solely on detectors [5]. To demonstrate the efficacy of *DyRoNet*, it has been evaluated using three contemporary streaming perception models: StreamYOLO [33], LongShortNet [18], and DAMO-StreamNet [9] (see Sec. 4.3). This *Model Bank & Dispatcher* strategy illustrates the adaptability and robustness of *DyRoNet* across different streaming perception scenarios.

Low-Rank Adaptation. A key challenge arises when fully fine-tuning individual branches, especially under the direction of *Router Network*. This strategy can lead to biases in the distribution of training data and inefficiencies in the learning process. Specifically, lighter branches may become predisposed to simpler cases, while more complex ones might be tailored to handle intricate scenarios, thereby heightening the risk of overfitting. Our experimental results, detailed in Sec. 4.3, support this observation.

To address these challenges, we have incorporated the LoRA [10] into *DyRoNet*. Within each model P_i, initially pre-trained on a dataset, the key components are the convolution kernel and bias matrices, symbolized as W^d_i. The rank of the LoRA module is defined as r, a value significantly smaller than the dimensionality of W^d_i, to ensure efficient adaptation. The update to the weight matrix adheres to a low-rank decomposition form, represented as

$W_d^i + \delta W = W_d^i + BA$.[2] This adaptation strategy allows for the original weights W_d^i to remain fixed, while the low-rank components BA are trained and adjusted. The adaptation process is executed through the following projection:

$$W_i^d x + \Delta W x = W_i^d x + W_i^l x, \tag{2}$$

where x represents the input image or feature map, and $\Delta W = W_i^l = BA$. The matrices A and B start from an initialized state and are fine-tuned during adaptation. This approach maintains the general applicability of the model by fixing W_d^i, while also enabling specialization within specific sub-domains, as determined by Router Network.

In *DyRoNet*, we employ $r = 32$ for the LoRA module, though this can be adjusted based on specific requirements of the scenarios in question. This low-rank adaptation mechanism not only enhances the flexibility of the *DyRoNet* framework but also mitigates the risk of overfitting, ensuring that each branch remains efficient and effective in its designated role.

3.3. Training Details of DyRoNet

The training process of *DyRoNet* focuses on two primary goals: (1) Improving the performance of individual branches. The backpropagation only updates the chosen model's weights in this step, enabling fine-tuning on segregated samples. (2) Achieving an optimal balance between accuracy and computational efficiency. This step only train the speed router while the remaining branches are frozen. This dual-objective framework is represented by the overall loss function:

$$L = \mathcal{L}^{sp} + \mathcal{L}^{E^2}, \tag{3}$$

where $\mathcal{L}^{sp}$ represents the streaming perception loss, and $\mathcal{L}^{E^2}$ denotes the effective and efficient (E^2) loss, which supervises branch selection.

Streaming Perception (SP) Loss. Each branch in *DyRoNet* is fine-tuned using its original loss function to maintain effectiveness. The router network is trained to select the optimal branch based on efficiency supervision. Let $\mathcal{T}_i = \{F_i^{cls}, F_i^{reg}, F_i^{obj}\}$ denote the logits produced by the i-th branch and $\mathcal{T}_{gt} = \{F_{gt}^{cls}, F_{gt}^{reg}, F_{gt}^{obj}\}$ represent the corresponding ground-truth, where $F_\cdot^{cls}$, $F_\cdot^{reg}$, and $F_\cdot^{obj}$ are the classification, objectness, and regression logits, respectively. The streaming perception loss for each branch, $\mathcal{L}_i^{sp}$, is defined as follows:

$$\begin{aligned}\mathcal{L}_i^{sp}(\mathcal{T}_i, \mathcal{T}_{gt}) = \mathcal{L}_{cls}(F_i^{cls}, F_{gt}^{cls}) + \mathcal{L}_{obj}(F_i^{obj}, F_{gt}^{obj}) \\ + \mathcal{L}_{reg}(F_i^{reg}, F_{gt}^{reg}),\end{aligned} \tag{4}$$

[2]Here, B is a matrix in $R^{d\times r}$, and A is in $R^{r\times k}$, ensuring that the rank r remains much smaller than d.

where $\mathcal{L}_{cls}(\cdot)$ and $\mathcal{L}_{obj}(\cdot)$ are defined as Mean Square Error (MSE) loss functions, while $\mathcal{L}_{reg}(\cdot)$ is represented by the Generalized Intersection over Union (GIoU) loss.

Effective and Efficient (E^2) Loss. During the training phase, streaming perception loss values from all branches are compiled into a vector $v^{sp} \in \mathbb{R}^K$, and inference time costs are aggregated into $v^{time} \in \mathbb{R}^K$, with K indicating the total number of branches in *DyRoNet*. To account for hardware variability, a normalized inference time vector $\hat{v}^{time} = \mathrm{softmax}(v^{time})$ is introduced. This vector is derived using the Softmax function to minimize the influence of hardware discrepancies. The representation for effective and efficient (E^2) decision-making is defined as:

$$f^{E^2} = \mathcal{O}_N(\arg\min_k(\mathrm{softmax}(v^{time}) \cdot v^{sp})), \tag{5}$$

where $\mathcal{O}$ denotes one-hot encoding, producing a boolean vector of length K, with the value of 1 at the index representing the estimated optimal branch at that moment. The E^2 Loss is then formulated as:

$$\mathcal{L}^{E^2} = \mathrm{KL}(f^{E^2}, f^r), \tag{6}$$

where $f_r = \mathcal{R}(\Delta I_t)$ and KL represents the Kullback-Leibler divergence, utilized to constrain the distribution.

***DyRoNet* and relevant techniques**. Although the structure of *DyRoNet* is somewhat similar to MoE [24, 35], in contrast, the gate network of MoE is not well-suited for streaming perception. This limitation has led to the development of speed router and the $\mathcal{L}^{E^2}$ loss function. *DyRoNet*, for efficiency, chooses one model over MoE's multiple experts, aligning with MoE in concept but differing in gate structure and selection strategy, making it unique for streaming contexts.

The speed router is inspired by Network Architecture Search (NAS). However, it uniquely addresses the challenge of streaming perception by transforming the search problem into an optimization of coded distances. The formulation of the loss function, $\mathcal{L}^{E^2}$, involves converting v^{time} into a distribution via softmax, which is then combined with v^{sp} to determine the optimal model through an one-hot vector f^{E^2}. This approach effectively simplifies the intricate problem of balancing accuracy and latency into a more tractable optimization task. Instead of employing NAS for loss search, our design is intricately linked to the specific needs of streaming perception, with KL divergence selected for its robustness in noisy situations [15]. This demonstrates the efficiency and innovation of our approach.

Overall, the process of training *DyRoNet* involves striking a meticulous balance between the SP loss, which ensures the efficacy of each branch, and the E^2 loss, which optimizes efficiency. The primary objective of this training is to develop a model that not only delivers high accuracy

in perception tasks but also operates within acceptable latency constraints, which is a critical requirement for real-time applications. This balanced approach enables *DyRoNet* to adapt dynamically to varying computational resources and environmental conditions, thereby maintaining optimal performance in diverse streaming perception scenarios.

4. Experiments

4.1. Dataset and Metric

Dataset. Argoverse-HD dataset [19] is utilized for our experiments, specifically designed for streaming perception in autonomous driving scenarios. It comprises high-resolution RGB images captured from urban city street, offering a realistic representation of diverse driving conditions. The dataset is structured into two main segments: a training set consisting of 65 video clips and a test set comprising 24 video clips. Each video clip in the dataset spans over 600 frames in average, contributing to a training set with approximately 39k frames and a test set containing around 15k frames. Notably, Argoverse-HD provides high-frame-rate (30fps) 2D object detection annotations, ensuring accuracy and reliability without relying on interpolated data.
Evaluation Metric. Streaming Average Precision (sAP) are adopted as the primary metric for performance evaluation, which is widely recognized for its effectiveness in streaming perception tasks [19]. It offers a comprehensive assessment by calculating the mean Average Precision (mAP) across various Intersection over Union (IoU) thresholds, ranging from 0.5 to 0.95. This metric allows us to evaluate detection performance across different object sizes, including large, medium, and small objects, providing a robust measurement in real-world streaming perception scenarios.

4.2. Implementation Details

We tested three state-of-the-art streaming perception models: StreamYOLO [33], LongShortNet [18], and DAMO-StreamNet [9]. These models, integral to the *DyRoNet* architecture, come with pre-trained parameters across three distinct scales: small (S), medium (M), and large (L), catering to a variety of processing requirements. In constructing the model bank $\mathcal{P}$ for *DyRoNet*, we strategically selected different model configurations to evaluate performance across diverse scenarios. For instance, the notation *DyRoNet* ($\text{DAMO}_{\text{S+M}}$) represents a configuration where *DyRoNet* employs the small (S) and medium (M) scales of DAMO-StreamNet as its two branches.[3] All experiments were conducted on a high-performance computing platform equipped with Nvidia 3090Ti GPUs (x4), ensuring robust and reliable computational power to handle the intensive processing demands of the streaming perception models. This setup provided a consistent and controlled environment for evaluating the efficacy of *DyRoNet* across different model configurations, contributing to the thoroughness and validity of our results. For more implementation details, please refer to Appendix C.

[3]Similar notations are used for other model combinations, allowing for a systematic exploration of the framework's adaptability and performance under varying computational constraints.

4.3. Comparision with SOTA Methods

We compared *DyRoNet* with state-of-the-art methods to evaluate its performance. In this subsection, we directly copied the reported sAP performance from their original papers as their results, for fair comparison, we evaluated the latency of each real-time model on 1x RTX 3090. The performance comparison was conducted on the Argoverse-HD dataset [19]. An overview of the results reveals that our proposed *DyRoNet* with a model bank of DAMO-StreamNet series achieves 37.8% sAP in 37.61 ms latency, outperforming the current state-of-the-art methods in latency by a significant margin. This demonstrates the effectiveness of the systematic improvements in *DyRoNet*.

4.4. Inference Time

In this subsection, we conducted detailed experiments analyzing the trade-offs between *DyRoNet*'s inference time and performance under different model bank selection. It is notable that the latency of *DyRoNet* presented in Tab. 2 do not accurately reflect the real outperform. Due to the varying hardware platforms for measuring latency across methods, a fair comparison cannot be achieved. For instance, DAMO-StreamNet is tested on 1x V100. To address these differences, we conducted additional tests on a 1x RTX 3090, which highlight DyRoNet's performance enhancements. Tab. 1 presents the latency comparison and highlights *DyRoNet*'s superior performance—maintaining competitive inference speed alongside accuracy gains versus the *random* and *MoE* approaches. Where the *MoE* predicts the weights of each branch via a gate module and then combines the output results accordingly. Specifically, *DyRoNet* achieves efficient speeds while preserving or enhancing performance. This balance enables meeting real-time needs without compromising perception quality, critical for autonomous driving where both factors are paramount. By validating effectiveness in inference time reductions and accuracy improvements, the results show the practicality and efficiency of *DyRoNet*'s dynamic model selection.

4.5. Ablation Study

Router Network. To validate the effectiveness of the *Router Network* based on frame difference, we conducted comparative experiments using frame difference ΔI_t, the current frame I_t, and the concatenation of the consecutive frames $[I_t + I_{t-1}]$ as input modality of the *Router Network*. For comparison, a naive method of input guidance

Model Bank	Random latency	Random sAP	MoE latency	MoE sAP	*DyRoNet* latency	*DyRoNet* sAP	*DyRoNet* sAP_{50}	*DyRoNet* sAP_{75}	*DyRoNet* sAP_s	*DyRoNet* sAP_m	*DyRoNet* sAP_l
$sYOLO_{S+M}$	39.16	31.5	66.16	29.5	**26.25** (-12.91)	**33.7** (+2.2)	53.9	34.1	13.0	35.1	59.3
$sYOLO_{S+L}$	24.04	33.2	70.19	29.5	29.35 (+5.31)	**36.9** (+3.7)	58.2	37.5	14.8	37.4	64.2
$sYOLO_{M+L}$	24.69	35.4	83.65	33.7	**23.51** (-1.18)	35.0 (-0.4)	55.7	35.5	13.7	36.2	61.1
LSN_{S+M}	24.79	31.8	128.74	29.8	**21.47** (-3.32)	30.5 (-1.3)	51.2	30.2	11.3	31.1	56.1
LSN_{S+L}	21.49	33.4	121.62	29.8	30.48 (+8.99)	**37.1** (+3.7)	58.3	37.6	15.1	37.6	63.7
LSN_{M+L}	24.75	35.6	136.66	34.1	29.05 (+4.30)	**36.9** (+1.3)	58.2	37.4	14.9	37.5	63.3
$DAMO_{S+M}$	36.61	33.5	188.42	31.8	**33.22** (-3.39)	**35.5** (+2.0)	56.9	36.2	14.4	36.8	63.2
$DAMO_{S+L}$	35.12	34.5	188.57	31.8	39.60 (+4.48)	**37.8** (+3.3)	59.1	38.7	16.1	39.0	64.2
$DAMO_{M+L}$	37.30	36.5	195.87	35.5	37.61 (+0.31)	**37.8** (+1.3)	58.8	38.8	16.1	39.0	64.0

Table 1. Comparison of latency (ms) and the corresponding sAP performance on 1x RTX 3090. The values are highlighted in **bold** font if the *DyRoNet* perform better than the corresponding *random* and *MoE* case. Due to the overall poorer performance of *MoE*, we only lists the relative latency and sAP differences between *DyRoNet* and the *random* approach.

Methods	Latency (ms)	sAP ↑
Non-real-time detector-based methods		
Adaptive Streamer [6]	-	21.3
Streamer (S=600) [19]	-	20.4
Streamer (S=900) [19]	-	18.2
Streamer+AdaScale [6]	-	13.8
Real-time detector-based methods		
DAMO-StreamNet-L [9]	39.6	37.8
LongShortNet-L [18]	29.9	37.1
StreamYOLO-L [33]	29.3	36.1
DAMO-StreamNet-M [9]	33.5	35.7
LongShortNet-M [18]	25.1	34.1
StreamYOLO-M [33]	24.8	32.9
DAMO-StreamNet-S [9]	30.1	31.8
LongShortNet-S [18]	20.3	29.8
StreamYOLO-S [33]	21.3	28.8
DyRoNet ($DAMO_{M+L}$)	<u>37.61</u> (-1.99)	**37.8** (same)
DyRoNet (LSN_{M+L})	<u>29.05</u> (-0.85)	36.9 (-0.2)
DyRoNet ($sYOLO_{M+L}$)	<u>23.51</u> (-5.79)	35.0 (-1.1)

Table 2. The comparison of *DyRoNet* and SOTA. The optimal values over its larger model are highlighted in **bold** font and the optimal values of online evaluation latency are shown in <u>underline</u> font. The latency of *DyRoNet* is evaluated on 1x RTX 3090 and compared with the latency of corresponding smaller model.

Model Bank	Full (sAP)	LoRA (sAP)
$StreamYOLO_{S+M}$	32.9	**33.7** (+0.8)
$StreamYOLO_{S+L}$	36.1	**36.9** (+0.8)
$StreamYOLO_{M+L}$	**36.2**	35.0 (-1.2)
$LongShortNet_{S+M}$	29.0	**30.5** (+1.5)
$LongShortNet_{S+L}$	36.2	**37.1** (+0.9)
$LongShortNet_{M+L}$	36.3	**36.9** (+0.6)
$DAMO\text{-}StreamNet_{S+M}$	34.8	**35.5** (+0.7)
$DAMO\text{-}StreamNet_{S+L}$	31.1	**37.8** (+6.7)
$DAMO\text{-}StreamNet_{M+L}$	37.4	**37.8** (+0.4)

Table 3. Comparison of LoRA finetune and Full finetune. The optimal values between *Full* and *LoRA* are shown in **bold** font.

also employed: By calculating $\mathbb{E}(\Delta I_t)$, the larger branch is selected if $\mathbb{E}(\Delta I_t) > 0$, otherwise the smaller branch is chosen. The results are presented in Tab. 6. For comparison, the *Router Network* only be trained in these experiments while the leftover parts be frozen.

	Model Bank	b_0	b_1	b_2	sAP
$K=2$ same model	$DAMO_{S+M}$	31.8	35.5	-	35.5
	$DAMO_{S+L}$	31.8	37.8	-	37.8
	$DAMO_{M+L}$	35.5	37.8	-	37.8
	LSN_{S+M}	29.8	34.1	-	30.5
	LSN_{S+L}	29.8	37.1	-	37.1
	LSN_{M+L}	34.1	37.1	-	36.9
	$sYOLO_{S+M}$	29.5	33.7	-	33.7
	$sYOLO_{S+L}$	29.5	36.9	-	36.9
	$sYOLO_{M+L}$	33.7	36.9	-	35.0
$K=2$ different model	$DAMO_S + LSN_S$	31.8	29.8	-	30.5
	$DAMO_S + LSN_M$	31.8	34.1	-	34.1
	$DAMO_S + LSN_L$	31.8	37.1	-	31.8
	$DAMO_M + LSN_S$	35.5	29.8	-	29.8
	$DAMO_L + LSN_S$	37.8	29.8	-	29.8
$K=3$ same model	$DAMO_{S+M+L}$	31.8	35.5	37.8	37.7
	LSN_{S+M+L}	29.8	34.1	37.1	36.1
	$sYOLO_{S+M+L}$	29.5	33.7	36.9	36.6

Table 4. The performance of *DyRoNet* under various model bank selection. K means the number of the model in bank $\mathcal{P}$.

It shows that using ΔI_t as input exhibits better performance than other methods (35.0 sAP of $sYOLO_{S+L}$ and 34.6 sAP of $sYOLO_{M+L}$). This indicates that utilizing ΔI_t offers significant advantages in comprehending and characterizing environmental speed. Conversely, it also underscores that employing single frames as input or using multiple frames as input renders the lightweight model bank selection model ineffective. Furthermore, the proportion of sample splits across branches can also illustrate the discriminative power with respect to environmental factors. For instance, the $\mathbb{E}(\Delta I_t)$ criterion resulting in a evenly spliting distribution (48.22% $\mathbb{E}(\Delta I_t) > 0$ over train set and 49.85% $\mathbb{E}(\Delta I_t) > 0$ over test set). Indicating the direct sample selection without router lacks estimation of environmental factors, thereby weakening its discriminative power.

In contrast, Tab.5 presents statistics indicating the router layer's effectiveness in allocating samples to specific models and showcase its ability to strike a balance between latency and performance. This balance is crucial for stream-

Model Combination	training time		inference time	
	Model 1	Model 2	Model 1	Model 2
SYOLO (M+L)	37.53%	62.47%	94.67%	5.33%
LSN (M+L)	30.86%	69.14%	19.87%	80.13%
DAMO (M+L)	84.61%	15.39%	0.02%	99.98%

Table 5. The statistics of model selection by *DyRoNet* under different model choices during both training and inference time.

Model Bank	Input Modality / Criterion			
	I_t	$[I_t + I_{t-1}]$	$\mathbb{E}(\Delta I_t)$	ΔI_t (*DyRoNet*)
$\text{sYOLO}_{\text{S+M}}$	**33.7**	**33.7**	31.5	32.6
$\text{sYOLO}_{\text{S+L}}$	34.1	30.2	32.9	**35.0**
$\text{sYOLO}_{\text{M+L}}$	33.7	33.7	34.2	**34.6**

Table 6. Ablation of router network input / criterion. The optimal results are marked in **bold** font under the same model bank setting.

ing perception and underscores our contribution.

Branch Selection. Our research on streaming perception models has shown that configuring these models across varying scales can optimize their performance. We found that combining L and S models strikes an optimal balance, resulting in significant speed improvements. This conclusion is supported by the empirical evidence presented in Tab. 4, which clearly shows that the L+S model pairing outperforms both the L+S and L+M cases. Our findings highlight the importance of strategic model scaling in streaming perception and provide a framework for future model optimization in similar domains.

Fine-tuning Scheme. We contrasted the performance of direct fine-tuning with the LoRA fine-tuning strategy [36] for streaming perception models. Tab. 3 shows that LoRA fine-tuning surpasses direct fine-tuning, with the DAMO-Streamnet-based model bank configuration realizing an absolute gain of over 1.6%. This substantiates LoRA's fine-tuning proficiency in circumventing the pitfalls of forgetting and data distribution bias inherent to direct fine-tuning. This result demonstrates that LoRA fine-tuning can effectively mitigate the overfitting while fine-tuning, leading to a stable performance improvement.

LoRA Rank. To assess the impact of LoRA ranks in *DyRoNet*, we conducted experiments with rank $r = 32, 16, 8$ respectively. All experiments were train for 5 epochs between *Router Network* training and model bank fine-tuning. The results are presented in Tab. 7. It can be observed that the performance is better with $r = 32$ compared to $r = 8$ and $r = 16$, and only occupy 10% of the total model parameters. Therefore, based on these experiments, $r = 32$ was selected as the default setting for our experiments. Although a smaller LoRA rank occupies fewer parameters, it leads to a rapid performance decay. The experimental results clearly demonstrate that with LoRA fine-tuning, it is possible to achieve superior performance than a single model while utilizing a smaller parameter footprint.

Model Bank	Rank	branch 0	branch 1	after train	Param.(%)
$\text{DAMO}_{\text{S+L}}$	8	31.8	37.8	35.9	4.02
$\text{DAMO}_{\text{S+L}}$	16	31.8	37.8	35.9	7.73
$\text{DAMO}_{\text{S+L}}$	32	31.8	37.8	**37.8**	14.35
$\text{LSN}_{\text{S+L}}$	8	29.8	37.1	30.6	5.48
$\text{LSN}_{\text{S+L}}$	16	29.8	37.1	30.6	5.48
$\text{LSN}_{\text{S+L}}$	32	29.8	37.1	**36.9**	10.39
$\text{sYOLO}_{\text{S+L}}$	8	29.5	36.9	35.0	2.7
$\text{sYOLO}_{\text{S+L}}$	16	29.5	36.9	35.0	5.38
$\text{sYOLO}_{\text{S+L}}$	32	29.5	36.9	**36.6**	10.21

Table 7. Ablation of LoRA rank: In the *Param.* column, we solely compare the proportion of parameters occupied by LoRA to the entire model. The best performance under the same model bank setting are highlighted in **bold** font.

Model Bank	b1	b2	Random	MoE	DyRoNet
$\text{sYOLO}_{\text{S+M}}$	6.9	9.2	8.1	6.9	**8.9**
$\text{sYOLO}_{\text{S+L}}$	7.3	9.9	8.6	7.3	**9.0**
$\text{sYOLO}_{\text{M+L}}$	8.9	9.6	9.2	8.9	**9.3**
$\text{sYOLO}_{\text{S}} + \text{LSN}_{\text{S}}$	6.6	6.2	6.4	6.2	**6.5**
$\text{sYOLO}_{\text{M}} + \text{LSN}_{\text{M}}$	9.1	9.3	9.2	9.1	**9.3**
$\text{sYOLO}_{\text{L}} + \text{LSN}_{\text{L}}$	9.0	9.6	9.3	9.0	**9.6**

Table 8. The sAP results are shown in the table. Where *b1* denotes the independent performance of the first branch and *b2* denotes the second one. The branch fusion method *Random* and *MoE* are similar with Tab. 1. The best method is highlighted in **bold** font.

4.6. Extra Experiment on NuScenes-H dataset

To validate the *DyRoNet* on other dataset, we converted the 3D streaming perception dataset nuScenes-H [28] into 2D format. The experiment details are provided in the Appendix D. As shown in Tab. 8, *DyRoNet* consistently achieves better results than other branch fusion methods on nuScenes-H 2D dataset. It demonstrates *DyRoNet*'s advantages in branch fusion selection and its versatility.

5. Conclusion

In conclusion, we present the **Dy**namic **Ro**utering **Net**work (*DyRoNet*), a system that dynamically selects specialized detectors for varied environmental conditions with minimal computational overhead. Our innovative increase-boosting fine-tuning, featuring a Low-Rank Adapter, mitigates distribution bias and overfitting, enhancing scene-specific performance. Experimental results validate *DyRoNet*'s state-of-the-art performance, offering a benchmark for streaming perception and insights for future research. In the future, *DyRoNet*'s principles will undoubtedly inform the development of more advanced, reliable systems.

Acknowledgments

Zhi-Qi Cheng's research in this project was supported by the US Department of Transportation, Office of the Assistant Secretary for Research and Technology, under the University Transportation Center Program (Federal Grant Number 69A3551747111), and Intel and IBM Fellowships.

References

[1] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. In *International Conference on Learning Representations*, 2019. 3

[2] Shaofeng Cai, Yao Shu, and Wei Wang. Dynamic routing networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3588–3597, January 2021. 3

[3] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers, 2023. 1

[4] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020. 3

[5] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2, 4

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

[7] Junyao Guo, Unmesh Kurup, and Mohak Shah. Is it safe to drive? an overview of factors, metrics, and datasets for driveability assessment in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(8):3135–3151, 2019. 2

[8] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021. 2

[9] Jun-Yan He, Zhi-Qi Cheng, Chenyang Li, Wangmeng Xiang, Binghui Chen, Bin Luo, Yifeng Geng, and Xuansong Xie. Damo-streamnet: Optimizing streaming perception in autonomous driving. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 810–818. International Joint Conferences on Artificial Intelligence Organization, 8 2023. 2, 4, 6, 7

[10] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 4

[11] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. 3

[12] Yihui Huang and Ningjiang Chen. Mtd: Multi-timestep detector for delayed streaming perception. In *Chinese Conference on Pattern Recognition and Computer Vision*, pages 337–349. Springer, 2023. 2

[13] Wonwoo Jo, Kyungshin Lee, Jaewon Baik, Sangsun Lee, Dongho Choi, and Hyunkyoo Park. Dade: Delay-adoptive detector for streaming perception. *arXiv preprint arXiv:2212.11558*, 2022. 2

[14] Glenn Jocher, Alex Stoken, Jirka Borovec, Ayush Chaurasia, Liu Changyu, Adam Hogan, Jan Hajek, Laurentiu Diaconu, Yonghye Kwon, Yann Defretin, et al. ultralytics/yolov5: v5. 0-yolov5-p6 1280 models, aws, supervise. ly and youtube integrations. *Zenodo*, 2021. 2

[15] Taehyeon Kim et al. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. In *IJCAI*, pages 2628–2635, 2021. 5

[16] Jin-Peng Lan, Zhi-Qi Cheng, Jun-Yan He, Chenyang Li, Bin Luo, Xu Bao, Wangmeng Xiang, Yifeng Geng, and Xuansong Xie. Procontext: Exploring progressive context transformer for tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023. 2

[17] JunKyu Lee, Blesson Varghese, and Hans Vandierendonck. Roma: Run-time object detection to maximize real-time accuracy. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6405–6414, 2023. 3

[18] Chenyang Li, Zhi-Qi Cheng, Jun-Yan He, Pengyu Li, Bin Luo, Hanyuan Chen, Yifeng Geng, Jin-Peng Lan, and Xuansong Xie. Longshortnet: Exploring temporal and semantic features fusion in streaming perception. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5. IEEE, 2023. 2, 4, 6, 7

[19] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *Proceedings of the European Conference on Computer Vision*, pages 473–488. Springer, 2020. 1, 3, 6, 7

[20] Zhihao Lin, Yongtao Wang, Jinhe Zhang, and Xiaojie Chu. Dynamicdet: A unified dynamic architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6282–6291, June 2023. 3

[21] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2020. 1

[22] Jian-Jun Qiao, Zhi-Qi Cheng, Xiao Wu, Wei Li, and Ji Zhang. Real-time semantic segmentation with parallel multiple views feature augmentation. In *ACM International Conference on Multimedia*, pages 6300–6308, 2022. 3

[23] Gur-Eyal Sela, Ionel Gog, Justin Wong, Kumar Krishna Agrawal, Xiangxi Mo, Sukrit Kalra, Peter Schafhalter, Eric Leong, Xin Wang, Bharathan Balaji, et al. Context-aware streaming perception in dynamic environments. In *Proceedings of the European Conference on Computer Vision*, pages 621–638. Springer, 2022. 2

[24] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017. 3, 5

[25] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional

neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019. 3

[26] Hao Wang, Zhi-Qi Cheng, Jingdong Sun, Xin Yang, Xiao Wu, Hongyang Chen, and Yan Yang. Debunking free fusion myth: Online multi-view anomaly detection with disentangled product-of-experts modeling. In *ACM International Conference on Multimedia*, pages 3277–3286, 2023. 3

[27] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision*, September 2018. 3

[28] Xiaofeng Wang, Zheng Zhu, Yunpeng Zhang, Guan Huang, Yun Ye, Wenbo Xu, Ziwei Chen, and Xingang Wang. Are we ready for vision-centric driving streaming perception? the asap benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9600–9610, 2023. 2, 8

[29] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2432–2444. Curran Associates, Inc., 2020. 3

[30] Zixiao Wang, Weiwei Zhang, and Bo Zhao. Estimating optical flow with streaming perception and changing trend aiming to complex scenarios. *Applied Sciences*, 13(6):3907, 2023. 2

[31] Ran Xu, Fangzhou Mu, Jayoung Lee, Preeti Mukherjee, Somali Chaterji, Saurabh Bagchi, and Yin Li. Smartadapt: Multi-branch object detection framework for videos on mobiles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2528–2538, 2022. 3

[32] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[33] Jinrong Yang, Songtao Liu, Zeming Li, Xiaoping Li, and Jian Sun. Real-time object detection for streaming perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5385–5395, June 2022. 2, 4, 6, 7

[34] Ji Zhang, Xiao Wu, Zhi-Qi Cheng, Qi He, and Wei Li. Improving anomaly segmentation with multi-granularity cross-domain alignment. In *ACM International Conference on Multimedia*, pages 8515–8524, 2023. 2

[35] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022. 5

[36] Jiawen Zhu, Zhi-Qi Cheng, Jun-Yan He, Chenyang Li, Bin Luo, Huchuan Lu, Yifeng Geng, and Xuansong Xie. Tracking with human-intent reasoning. *arXiv preprint arXiv:2312.17448*, 2023. 8

[37] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019. 3

[38] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023. 1