# Transtreaming: Adaptive Delay-aware Transformer for Real-time Streaming Perception

**Xiang Zhang[1*], Yufei Cui[2*], Chenchen Fu[1†], Zihao Wang[1], Yuyang Sun[1], Xue Liu[2], Weiwei Wu[1]**

[1]School of Computer Science and Engineering, Southeast University
[2]School of Computer Science, McGill University
{230238536, chenchen_fu, 213213628, yysun, weiweiwu}@seu.edu.cn, yufei.cui@mail.mcgill.ca, xueliu@cs.mcgill.ca

## Abstract

Real-time object detection is critical for the decision-making process for many real-world applications, such as collision avoidance and path planning in autonomous driving. This work presents an innovative real-time streaming perception method, Transtreaming, which addresses the challenge of real-time object detection with dynamic computational delays. The core innovation of Transtreaming lies in its adaptive delay-aware transformer, which can concurrently predict multiple future frames and select the output that best matches the real-world present time, compensating for any system-induced computational delays.

The proposed model outperforms existing state-of-the-art methods, even in single-frame detection scenarios, by leveraging a transformer-based methodology. It demonstrates robust performance across a range of devices, from powerful V100 to modest 2080Ti, achieving the highest level of perceptual accuracy on all platforms. Unlike most state-of-the-art methods that struggle to complete computation within a single frame on less powerful devices, Transtreaming meets the stringent real-time processing requirements on all kinds of devices. The experimental results emphasize the system's adaptability and its potential to significantly improve the safety and reliability of many real-world systems, such as autonomous driving.

**Code** — https://github.com/DecAngel/Transtreaming

## Introduction

Existing object detection methods have now achieved high performance with low latency. Most of these studies have either focused on 1) improving detection accuracy (Parmar et al. 2018; Liu et al. 2021), or 2) striking a balance between performance and latency (Redmon et al. 2016; Ge et al. 2021). More recently, while real-world applications such as autonomous driving systems require real-time detection results for accurate decision making, researchers began to pay much attention on real-time object detection. This area, also known as streaming perception, is gaining prominence for its potential to revolutionize the way we interact with and understand our environment in real time.

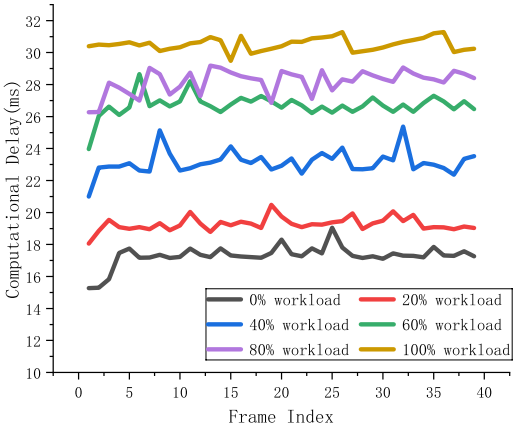



Figure 1: Fluctuation of the computational delay using the same detector when processing different frames with varying workloads (simulated by adopting a similar approach to GPU contention generation in (Xu et al. 2020)) on a server equipped with NVIDIA GeForce RTX 4080.

For real-time object detection, researchers initially aimed at enhancing the speed of non-real-time detectors to meet stringent real-time requirements, such as achieving frame rates of 60 frames per second (FPS). However, as noted by (Yang et al. 2022), no matter how fast the detector is, in real-world online scenarios, when the detector handles the latest observation, the state of the world around the moving vehicle has already changed. Thus the detection result will always be at least one frame delay from the latest observation.

To this end, some recent studies (Yang et al. 2022; Li et al. 2023; He et al. 2023; Jo et al. 2022; Huang and Chen 2023) have improved the detector models to include forecasting capabilities. These models are trained to anticipate the future positions of objects (e.g. one frame into the future), such that the detector can finish the computation within one frame (Yang et al. 2022). For example, assuming a constant video streaming rate of 30FPS, this implies that the detection process needs to be completed within $\frac{1}{30}$ second. To emulate varying vehicular speeds, these models were trained across a spectrum of video streaming rates. For example, if a vehicle is moving at $30km/h$ (or $60km/h$), the video streaming rate is set at $1\times$ (or $2\times$) the original rate, respectively. Despite

*These authors contributed equally.
†This author is the corresponding author.

these advancements, the aforementioned literature presents several limitations:

1. The detectors are constrained to complete computations within a limited time, necessitating the majority of the models (especially those with high performance) to operate on devices with powerful GPUs such as V100 (Yang et al. 2022; Li et al. 2023; He et al. 2023). This requirement restricts the application of real-time detection systems to devices with substantial computational resources.
2. Existing models are typically restricted to predicting a fixed single frame in the future, such as one or two frames ahead. However, we observe that computational delays can fluctuate significantly over time due to varying workloads. As shown in Figure 1, the same detector has different computational delays in dealing with different frames, ranging from $15ms$ to $31ms$ due to various workloads of the system. This variability implies that models may not be able to meet fixed real-time constraints during periods of burst workload, potentially leading to significant failures in accurate real-time detection.

To address the above limitations, this work presents a novel real-time streaming perception method, Transtreaming, which addresses the challenge of real-time object detection with dynamic computational delay. The contributions of this work are summarized as follows.

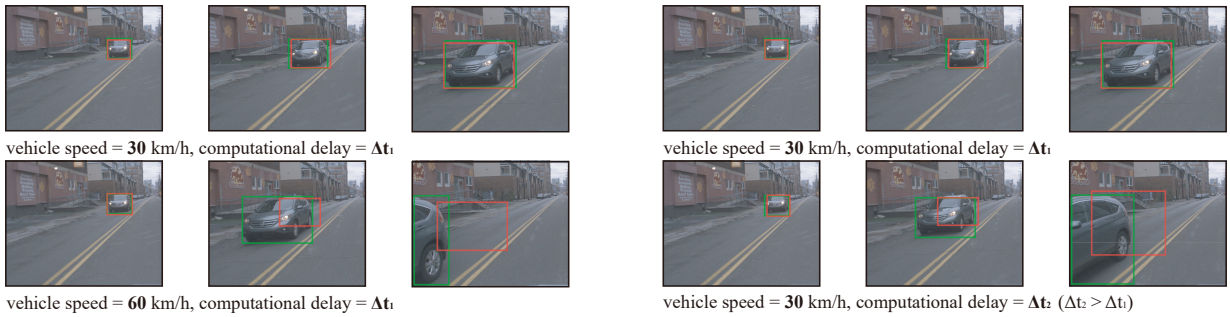
- This work facilitates adaptive delay-aware streaming perception by integrating runtime information within detection models, concurrently predicting **multiple frames** and selecting the most appropriate output aligned with the real-world present time.
- Even in the realm of **single-frame** real-time detection, the proposed approach, which leverages a transformer-based methodology, outperforms all existing state-of-the-art (SOTA) techniques.
- The proposed model demonstrates robust performance across a spectrum of devices, ranging from the high-performance V100 to the modest 2080Ti. It achieves the highest level of perceptual accuracy on all platforms, whereas most SOTA methods struggle to complete computation within a single frame when applied on less powerful devices, thereby failing to meet the stringent real-time processing requirement.

## Related Work

**Image Object Detection:** Deep learning's evolution has dramatically transformed object detection, with CNNs outpacing traditional methods. Image object detection methods are primarily divided into two-stage (such as R-CNN (Girshick et al. 2014) and Fast R-CNN (Girshick 2015)) and one-stage methods (such as SSD (Liu et al. 2016) and YOLO (Redmon et al. 2016)). Two-stage methods employ region proposals to pinpoint potential objects, breaking the detection process into two distinct phases. Improvements in the two-stage approach, seen in Fast R-CNN (Girshick 2015) and Faster R-CNN (Ren et al. 2015), streamlined the process by merging region proposal networks with CNN architectures, thereby reducing latency. Yet, these techniques still face delays due to proposal refinement. One-stage detectors offer a balance of speed and precision for real-time tasks but lack the temporal context necessary for streaming detection, as they focus solely on the current frame.

**Video Object Detection:** The objective of Video Object Detection (VOD) is to identify objects within video data, as opposed to static images. In order to utilize the unique characteristics of video data, recent progress is made in four aspects: 1) tracking-based methods (Feichtenhofer, Pinz, and Zisserman 2017): enhance detection by learning feature similarities across frames; 2) optical-flow-based methods (Zhu et al. 2017a; Wang et al. 2018; Zhu et al. 2018, 2017b): utilize flow information to enhance key frames; 3) attention-based methods (Deng et al. 2019; Wu et al. 2019; Shvets, Liu, and Berg 2019; Han et al. 2020): apply attention mechanisms to relate abstract features for improved detection; and 4) other methods: aggregate features through networks like DCN (Bertasius, Torresani, and Shi 2018), RNN (Xiao and Lee 2018), and LSTM (Liu and Zhu 2018). Compared to streaming perception, VOD generally focuses on the offline setting, overlooking the runtime delay of detection algorithms. In contrast, this work takes into account both runtime delay and real-time requirements to ensure high-performance detection.

**Streaming Perception:** Streaming Perception tackles the drift in real-time detection due to computational latency by predicting entity locations after computational delays using temporal information from historical results (Li, Wang, and Ramanan 2020). The streaming average precision ($sAP$) metric was introduced to evaluate object detection algorithms in streaming scenarios, factoring in both latency and accuracy. Chanakya (Ghosh et al. 2024) proposed a method to enhance $sAP$ performance by learning a policy to adjust input resolution and model size, effectively balancing latency and accuracy. However, Chanakya's approach did not utilize a temporal-predictive detector. Consequently, subsequent studies introduced various models designed to predict object locations. For example, StreamYOLO (Yang et al. 2022) used a dual-flow perception module for next-frame prediction, combining both previous and current frames' features. Dade (Jo et al. 2022) and MTD (Huang and Chen 2023) introduced mechanisms to dynamically select features from past or future timestamps, taking the runtime delay of the algorithm into consideration. DAMO-StreamNet (He et al. 2023) and Longshortnet (Li et al. 2023) used dual-path architectures to capture long-term motion and calibrate with short-term semantics. They extended dynamic flow path from 1 past frame to 3 frames, successfully capturing the long-term motion of moving objects, and thus achieved the state-of-the-art performance in streaming perception. Existing research primarily focuses on detection within a single subsequent frame, which necessitates the completion of computations within a single frame. This requirement becomes challenging under conditions of limited computational resources or high system workloads. In this study, we introduce a temporal attention module designed to predict multiple future outcomes from given past time series. The module selects the outcome that most closely aligns with real-world present time, thereby accommodating various computational delays.




(a) Given fixed $\Delta t_1$, varying speed reduces the accuracy.

(b) Given fixed speed, dynamic delay reduces the accuracy.

Figure 2: Observation example: when a detector is trained with fixed vehicle speed ($30km/h$) and fixed computational delay ($\Delta t_1$), the detection is always accurate when both of the speed and delay remain the same. But the detection accuracy will significantly decrease when either (a) the vehicle speed changes (from $30km/h$ to $60km/h$) or (b) the computational delay varies (from $\Delta t_1$ to $\Delta t_2$).

# Methodology

## Observation and Motivation

The motivation for our proposed scheme is rooted in the limitations observed in current streaming perception studies (Yang et al. 2022; Li et al. 2023; He et al. 2023), particularly in handling the varying velocities of vehicles and the dynamic computational delays inherent in real-time systems.

The above SOTA methods in streaming perception have noticed that the different velocities of vehicles significantly affect the real-time detection. If the detector is only trained at a fixed frame rate (e.g., 60 FPS), it can only be equipped to simulate vehicles moving at a corresponding constant speed (e.g., $30km/h$). As depicted in Figure 2.(a), when the speed of a vehicle increases to $60km/s$, the detector fails to detect it accurately. Even though mix-velocity training was applied in these works, they need to pre-set the velocity (Yang et al. 2022), i.e. pre-determine whether they do the detection for frame $t+1$ for slow or $t+3$ for fast velocities based on prior frames. These approaches are inherently restrictive because they did not allow the dynamic velocity of vehicles.

Furthermore, as shown in Figure 1, random computational delays can be involved due to dynamic workloads in time. Therefore, even if the vehicle drives at the same speed, different computational delays can prevent accurate detection, as shown in Figure 2.(b). Notably, both varying vehicle velocities and computational delays introduce a similar element of randomness to the detection process, complicating the prediction task.

To address these challenges, we introduce Transtreaming, a novel approach that captures the inherent randomness in both vehicle velocity and computational delay. Our solution is designed to provide real-time detection forecasts across a range of temporal conditions while being dynamically adaptive to the environment. Specifically, Transtreaming is tailored to accommodate diverse runtime scenarios, thereby enhancing the safety and reliability of streaming perception systems in various operational contexts.

## Problem Formation

Given a RGB video sequence $\{\mathrm{I}_i\} \in \mathbb{R}^{L\times 3\times H\times W}$ and the ground truth object bounding boxes $\{\mathrm{O}_i\} \in \mathbb{R}^{L\times Objects\times 5}$, each frame $\mathrm{I}_i$ and bounding box $\mathrm{O}_i$ at index $i$ is firstly associated with a timestamp $t = \frac{i}{k}$ with frame rate $k$ ($k = 30$, typically) FPS under streaming perception settings. The task of streaming perception then runs an online procedure that delivers the frames to the detector at their corresponding timestamp $t$. Then the detector's predictions $\{\hat{\mathrm{O}}_{t'}\}$ is collected at timestamp $t'$. Note that the detector has computational delay $\Delta t^I$ and start-up delay $\Delta t^S$ (delay caused by not inferring $\mathrm{I}_i$ at exactly $t$ because of delay from the previous loop). Consequently $t' = t + \Delta t^I + \Delta t^S$, making detector output timestamps $t'$ misaligned with $t$. As a final metric, streaming perception uses Mean Average Precision but pairs prediction and ground truth differently. Each prediction $\hat{\mathrm{O}}_{t'}$ is paired with the ground truths after its output timestamp $t'_1$ before the subsequent prediction's output timestamp $t'_2$, which is $\{\mathrm{O}_i\}, i \in [\lceil t'_1 \times k\rceil, \lfloor t'_2 \times k\rfloor]$. Unpaired ground truths are considered detection misses. Therefore, the detector is expected to generate bounding box prediction via forecasting into the future, in order to align $\hat{\mathrm{O}}_{t'}$ with $\mathrm{O}_{\lceil t'\times k\rceil}$. Every prediction of the detector should manage to compensate the object displacement error cause by computational delay $\Delta t^I$, and subsequent start-up delay $\Delta t^S$. Without loss of generality, we simplify notations by using relative indices $\Delta i$ against current frame index. In this way, current frames is denoted as $\mathrm{I}_0$, past frames are denoted as $\mathrm{I}_{-1}, \mathrm{I}_{-2}, \ldots$, while future objects are denoted as $\mathrm{O}_1, \mathrm{O}_2, \ldots$.

The overall architecture of Transtreaming is shown in Figure 3. We split the design of Transtreaming to a detection model named Transtreamer (TS) and a scheduling algorithm named Adaptive Strategy (AS). At every timestamp, given previous input frames $\{\mathrm{I}_i\}, i \in [0, \lfloor t * k\rfloor]$, Transtreaming utilizes AS to estimate runtime statistics $\Delta\hat{t}^I$ and schedule the execution of TS. To provide the detection model with temporal cues, AS generates 2 temporal proposals $\{\mathbf{P^P}, \mathbf{P^F}\}, \mathbf{P^P} \subseteq \mathbb{N}^-, \mathbf{P^F} \subseteq \mathbb{N}^+$ from estimated $\Delta\hat{t}^I$ and observed start-up delay $\Delta t^S$. Proposal $\mathbf{P^P}$ indicates the

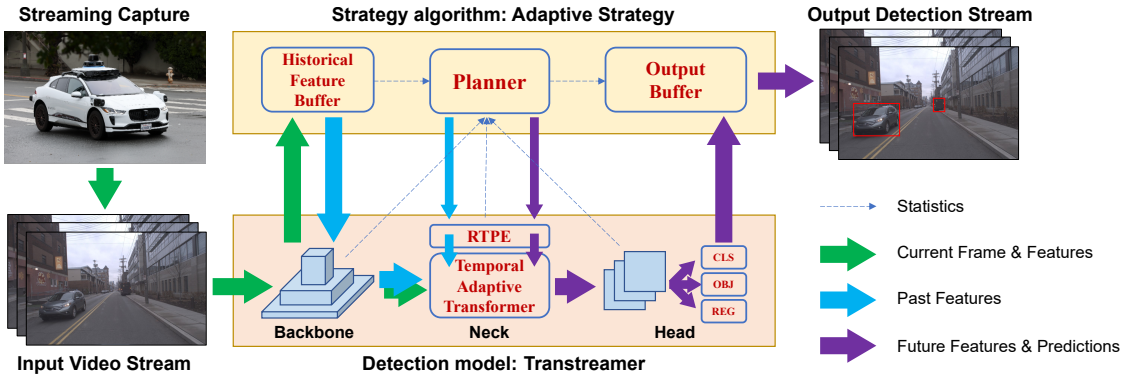



Figure 3: Overall architecture of Transtreaming. Transtreaming composes of a detection model Transtreamer and strategy algorithm Adaptive Strategy. The detection process is mainly done by Transtreamer in the lower part of the figure. Adaptive Strategy provides support by producing temporal proposals that encoded by RTPE (Relative Temporal Positional Embedding) and by using 2 buffers to store historical features and dispatch detection results.

indices of available past frames' features, while $\mathbf{P^F}$ indicates the temporal indices of forecasting targets. AS also uses buffering techniques to store previously computed image features $\{\mathrm{F}_i\}, i \in \mathbf{P^P}$, in order to accelerate model inference speed. Furthermore, the multi-frame output of TS $\{\hat{\mathrm{O}}_j\}, j \in \mathbf{P^F}$ is stored in output buffer and dispatched at appropriate timestamp. This approach avoids producing $\hat{\mathrm{O}}_j$ before its actual timestamp. Inside the detection model, TS encodes the temporal proposals $\{\mathbf{P^P}, \mathbf{P^F}\}$ as Relative Temporal Positional Embedding (RTPE) and merge features buffers with current frame's features. TS then uses a temporal transformer module to query the features of future objects from current and past features with RTPE. Additionally, Transtreamer is capable of forecasting object on multiple future timestamps with dynamic length (i.e., $\mathbf{P^F}$ can have any number of elements), thus achieving flexibility in diverse runtime environments. The details of Transtreaming's components are elaborated in the following subsections.

## Transtreamer

Following previous work's (He et al. 2023) design of detection models, we split TS into three modules: backbone for feature extraction, neck for temporal forecasting and head for bounding box decoding. To develop a neck module that maximally enhances temporal reasoning in TS, we employ the transformer paradigm to establish relationship between past features and future features. Transformers can effectively utilize the temporal proposals provided by AS. To evaluate against the current SOTA methods, we leverage the same backbone (DRFPN) and head (TALHead) module as in (He et al. 2023) and (Yang et al. 2022). Assume the temporal proposals $\{\mathbf{P^P}, \mathbf{P^F}\}$, and buffered features $\{\mathrm{F}_i\}, i \in \mathbf{P^P}$ is known beforehand. The features of current frame $\mathrm{I}_0$ is extracted by the backbone module $\mathbf{W^B}(\cdot)$. The resulting features $\mathrm{F}_0$ is then concatenated to the buffered features in a chronological order. Given the concatenated features and $\mathbf{P^P}, \mathbf{P^F}$, TS encodes $\mathbf{P^P}$ and $\mathbf{P^F}$ as Relative Temporal Positional Embedding (denoted as RTPE). RTPE carries temporal information for the succeeding Temporal Adaptive Transformer (TAT, denoted as $\mathbf{W^T}(\cdot)$). TAT makes use of cross attention modules that decodes future objects' features $\{\hat{\mathrm{F}}_j\}, j \in \mathbf{P^F}$ from past frames' features. Finally, $\{\hat{\mathrm{F}}_j\}$ is fed into the head module $\mathbf{W^H}(\cdot)$ to generate the estimated bounding boxes for future objects $\{\hat{\mathrm{O}}_j\}$. Formally, Transtreamer can be represented as

$$\begin{aligned}\{\hat{\mathrm{O}}_j\} &= \mathrm{TS}(\mathrm{I}_0, \{\mathrm{F}_i\}, \mathrm{RTPE}) \\ &= \mathbf{W^H}(\mathbf{W^T}(\mathrm{Concat}(\mathbf{W^B}(\mathrm{I}_0), \{\mathrm{F}_i\}), \mathrm{RTPE}))\end{aligned} \tag{1}$$

where $i \in \mathbf{P^P}$, $j \in \mathbf{P^F}$. TS denotes Transtreamer and Concat denotes concatenation along temporal dimension. The whole architecture of Transtreamer is shown in Figure 4.

**3D Window Partitioning:** The extracted past and current features $\{\mathrm{F}_i\}, i \in \mathbf{P}^P + \{0\}$ consist of 2 spatial dimensions (height $H$ and width $W$), 1 temporal dimension (time $T$) and a feature dimension (channel $C$). Common tokenizing approach flattens the input into a $THW \times C$ token sequence. But this approach is computational expensive with a computing complexity of $O(T^2H^2W^2C)$. To reduce the complexity of attention computation, we leverage the windowed attention to break input features to 3d spatial-temporal windows $(win^T, win^H, win^W)$. We assume the objects' trajectories can be captured within the window size, so shrinking the transformer's perception range can have negligible effect. In practice, we apply window partitioning before every layer of TAT and reverse window partitioning after that.

**Relative Temporal Positional Embedding:** The order of input sequence does not affect the computation of attention weight in attention modules. Therefore, the spatial and temporal locality of 3d features cannot be preserved. To remedy this problem, we employ positional embedding to integrate locality information into aforementioned Temporal Transformer by directly adding to attention weights. Since the concatenated features $\{F_i\}$ involves relative spatial and temporal position, we extend 2d learnable relative positional embedding used in (Liu et al. 2021) to 3d, adding an additional temporal dimension. Different from spatial dimensions, our RTPE only offers embedding for positive temporal dif-

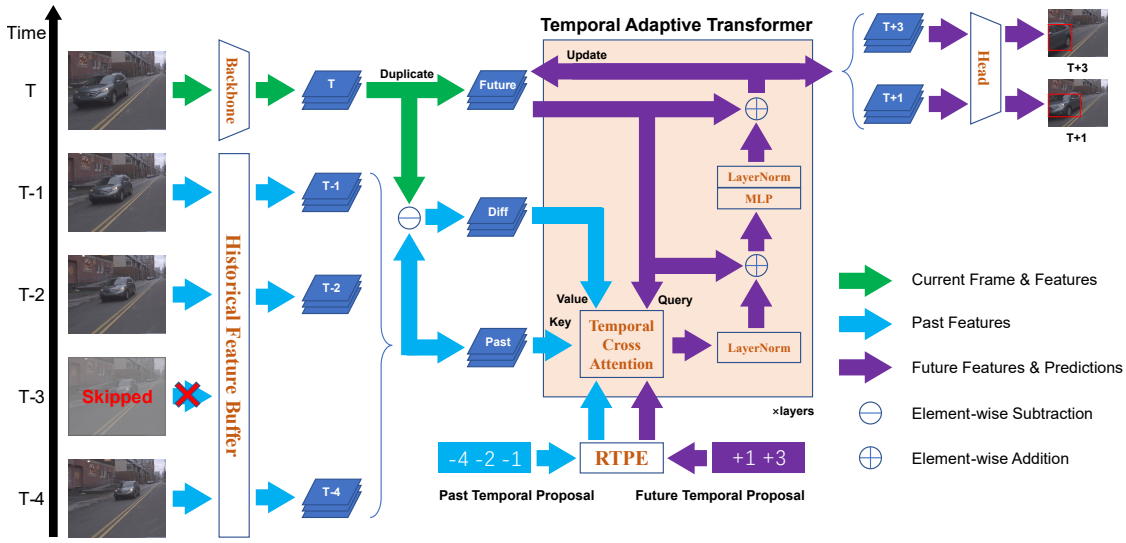



Figure 4: Detailed architecture of the detection model: Transtreamer. The figure illustrates a scenario where, due to computational delay, the frame at $T-3$ is skipped. Adaptive Strategy instructs Transtreamer to generate the detection results at $T+1, T+3$ from the image at $T$ and the buffered features at $T-1, T-2, T-4$.

ference (i.e. future tokens attend to past tokens, but not vice versa). Formally, the embedding created by RTPE is denoted as $E_{dt,dh,dw}$. $E_{dt,dh,dw}$ is generated by MLP projections from cuboid 3d coordinates $(t,h,w), t \in [0, max(\mathbf{P^F}) - min(\mathbf{P^F})], h \in [-win^H, win^H], w \in [-win^W, win^W]$ to ensure its continuity in 3 dimensions. Here $max(\mathbf{P^F}) - min(\mathbf{P^F})$ is the maximum temporal difference given $\mathbf{P^P}$ and $\mathbf{P^F}$. For each query token $Q_{t_1,h_1,w_1}$, its temporal embedding value against key token $K_{t_2,h_2,w_2}$ is $E_{t_1-t_2,h_1-h_2,w_1-w_2}$, effectively reflecting their relative positions.

**Temporal Adaptive Transformer:** The neck of Transtreamer should model the temporal relationship between past frames' features $\{\mathrm{F}_i\}$ and future objects' features $\{\mathrm{F}_j\}$. Directly adopting convolution or linear layers as in (He et al. 2023; Li et al. 2023) will result in a fix-lengthed $\mathbf{P^P}$ and $\mathbf{P^F}$, which fails to reflect the dynamic nature of computational delays. Inspired from the decoder layers of the transformer, we introduce a novel temporal cross-attention module TAT, which encodes future objects' features as queries while past frames' features as keys and values. Since attention modules often utilize residual connection to help mitigate the vanishing gradient problem, we argue that such approach is still valid in this situation. This is due to the fact that the features of current frame $\mathrm{F}_0$ is a reasonable initial guess of the feature of future frames $\{\mathrm{F}_j\}$. Specifically, we firstly replicate current features $\mathrm{F}_0$ along the temporal dimension by $|\mathrm{P^F}|$ as the initial queries. Past frames' features $\{\mathrm{F}_i\}$ are used as keys while the difference of $\mathrm{F}_0$ and $\{\mathrm{F}_i\}$ are used as values. Secondly, in every layer of TAT, 3D Window Partitioning is utilized to create corresponding input tokens $\mathrm{F}^Q$, $\mathrm{F}^K$ and $\mathrm{F}^V$. We adopt a cross attention with RTPE as gain to attention weight, serving as a source of temporal information. The tokens are transformed back to their original form by reverse window partitioning, followed by normalization layers and MLP layers as in original transformer architecture. Finally, we update the future queries with layer output and repeat the actions within every layer of TAT.

## Adaptive Strategy

Adaptive Strategy, which consists of a planner module and two buffers, defines the way Transtreamer interacts with streaming perception environments.

**Planner:** The main function of AS is planning, in other words, is generating a proper $\mathbf{P^P}$ and $\mathbf{P^F}$ that maximize the detection model's performance in Streaming Perception. During streaming inference, the past loops' inference time of Transtreamer's backbone, neck and head module, as well as other processing delay is firstly recorded as $\{\Delta t_i^B, \Delta t_i^N, \Delta t_i^H, \Delta t_i^O\}, i \in \mathbb{N}^-$, respectively. As a reasonable guess, we estimate these delay in current loop $\{\Delta \hat{t}_0^B, \Delta \hat{t}_0^N, \Delta \hat{t}_0^H, \Delta \hat{t}_0^O\}$ with exponential moving average (EMA) of $\Delta t_i$ with a decay of 0.5. Additionally, the start-up delay can be directly measured as $\Delta t_0^S$. To fully utilize past buffers and to give a rational estimate of future timestamps, We select at most $M^P$ latest available buffers' index as $\mathbf{P_0^P}$. For $\mathbf{P_0^F}$, we produce at most $M^F$ indices, with each prediction evenly spaced on temporal axis. Finally, both $\mathbf{P^P}$ and $\mathbf{P^F}$ are clipped within the range of $[\Delta t_{min} \times k, \Delta t_{max} \times k]$ and remove duplicates to keep $\mathbf{P}$ in reasonable range and avoid redundant computation. This operation is denoted as $\mathrm{Clip}(\cdot)$. The value of $\Delta t_{min}$ and $\Delta t_{max}$ are empirically assigned as $-29$ and $+19$, respectively. This solution manages to maintain the balance between model latency and forecasting accuracy with respect to high and fluctuating delays.

**Historical Feature Buffer:** Though the features of past frames $\{\mathrm{F}_i\}, i \in \mathbf{P^P}$ can be acquired by computing $\{\mathbf{W^B}(\mathrm{I}_i)\}$, such method is not applicable in streaming evaluations, where the computation load clearly affects the final performance. Therefore, we cache historical frame's feature

| Methods | $sAP$ | $sAP_{50}$ | $sAP_{75}$ | $sAP_S$ | $sAP_M$ | $sAP_L$ |
|---|---|---|---|---|---|---|
| Non Real-time Methods | | | | | | |
| Streamer (S=900) | 18.2 | 35.3 | 16.8 | 4.7 | 14.4 | 34.6 |
| Streamer (S=600) | 20.4 | 35.6 | 20.8 | 3.6 | 18.0 | 47.2 |
| Streamer + AdaS | 13.8 | 23.4 | 14.2 | 0.2 | 9.0 | 39.9 |
| Adaptive Streamer | 21.3 | 37.3 | 21.1 | 4.4 | 18.7 | 47.1 |
| YOLOX-S | 25.8 | 47.0 | 24.3 | 8.8 | 25.7 | 44.5 |
| YOLOX-M | 29.4 | 51.6 | 28.1 | 10.3 | 29.9 | 50.4 |
| YOLOX-L | 32.5 | 55.9 | 31.2 | 12.0 | 31.3 | 57.1 |
| Real-time Methods | | | | | | |
| StreamYOLO-S | 28.8 | 50.3 | 27.6 | 9.7 | 30.7 | 53.1 |
| StreamYOLO-M | 32.9 | 54.0 | 32.5 | 12.4 | 34.8 | 58.1 |
| StreamYOLO-L | 36.1 | 57.6 | 35.6 | 13.8 | 37.1 | 63.3 |
| DADE-L | 36.7 | 63.9 | 36.9 | 14.6 | 57.9 | 37.3 |
| LongShortNet-S | 29.8 | 50.4 | 29.5 | 11.0 | 30.6 | 52.8 |
| LongShortNet-M | 34.1 | 54.8 | 34.6 | 13.3 | 35.3 | 58.1 |
| LongShortNet-L | 37.1 | 57.8 | 37.7 | 15.2 | 37.3 | 63.8 |
| DAMO-StreamNet-S | 31.8 | 52.3 | 31.0 | 11.4 | 32.9 | 58.7 |
| DAMO-StreamNet-M | 35.7 | 56.7 | 35.9 | 14.5 | 36.3 | 63.3 |
| DAMO-StreamNet-L | 37.8 | 59.1 | 38.6 | 16.1 | 39.0 | 64.6 |
| Transtreaming-S(Ours) | 32.5 | 53.9 | 32.2 | 10.9 | 33.8 | 60.9 |
| Transtreaming-M(Ours) | 36.0 | 57.0 | 36.6 | 14.2 | 36.9 | 63.6 |
| Transtreaming-L(Ours) | **38.2** | **59.7** | **39.0** | **16.3** | **40.0** | **65.8** |

Table 1: Main result of $sAP$ comparison with real-time and non real-time SOTA detectors on the Argoverse-HD dataset. Best $sAP$ scores are indicated in bold.

to avoid re-computation. We also set a limit to the length of the buffer and choose to discard the earliest buffered features after the buffer is full. Since early frames have a negligible influence on the current frame, the limitation ease the computation burden of TAT.

**Output Buffer:** Although multiple future object predictions $\{\hat{\mathrm{O}}_j\}, j \in \mathbf{P^F}$ are produced by Transtreamer, they should not be merely emitted in a sequential manner. This approach might lead to outdated results if $\Delta\hat{t}^I$ is underestimated. To address this, we introduce an additional buffer to the output of Transtreamer. At each timestamp, the most proximate prediction within the buffer is dispatched to ensure optimal use of all predictions at their relevant timestamps. In instances where a more recent prediction becomes available due to an overestimation of $\Delta\hat{t}^I$, it can replace the prediction with the same timestamp.

## Training and Inference

We mainly replicate the training scheme used by baseline methods (He et al. 2023; Li et al. 2023), with the exception of Asymmetric Knowledge Distillation proposed by (He et al. 2023). We believe this contribution is orthogonal with our work.

**Mixed Speed Training:** To enhance model's delay-awareness, we also employ mixed speed training to extend the temporal perception range of Transtreamer. The model will be ineffective on other temporal intervals if only a fixed relative indices $\mathbf{P^P}$ and $\mathbf{P^F}$ is sampled for training. Therefore, we adopt mixed speed training scheme that samples $\mathbf{P^P}$ from $[-24, -1]$ and $\mathbf{P^F}$ from $[1, 16]$. The loss for each predicted future object is given equal weights.

# Experiment

## Implementation Details

**Dataset:** In our experiment, we trained and tested our method on Argoverse-HD, a common urban driving dataset composed of the front camera video sequence and bounding-box annotations for common road objects (e.g. cars, pedestrians, traffic lights). This dataset contains high-frequency annotation of 30 FPS that simulates real-world situations, which is suitable for streaming evaluation. We believe other datasets (e.g. nuScenes, Waymo) are not suitable for streaming evaluation as they are annotated at a lower frequency. We follow the train and validation split as in (He et al. 2023).

**Model:** The base backbone of our proposed model is pre-trained on the COCO dataset, consistent with the approach of (He et al. 2023). Other parameters are initialized following Lecun weight initialization. The model is then fine-tuned on the Argoverse-HD dataset for 8 epochs using a single Nvidia RTX4080 GPU with a batch size of 4 and half-resolution input ($600 \times 960$). To ensure a fair comparison with other methods, we provide 3 configurations of Transtreaming as Transtreaming-S (small), Transtreaming-M (medium) and Transtreaming-L (large) that differs in the number of parameters. This is in accordance with previous methods' approaches (He et al. 2023; Li et al. 2023).

**Main Metric:** We follow the streaming evaluation methods proposed by (Li, Wang, and Ramanan 2020) as the main test metric. The streaming average precision ($sAP$) is used to evaluate the performance of the whole pipeline under a

| Devices | Methods | $sAP^2$ | $sAP^4$ | $sAP^8$ | $sAP^{16}$ |
|---|---|---|---|---|---|
| v100 cluster | LongShortNet-S | 25.5 | 21.6 | 16.7 | 11.0 |
| | DAMO-StreamNet-S | 25.0 | 19.9 | 14.2 | 9.4 |
| | Transtreaming-S*(Ours) | **26.4** | **22.4** | **16.8** | **11.8** |
| 4080 server | LongShortNet-S | 24.5 | 20.3 | 14.8 | 9.8 |
| | DAMO-StreamNet-S | 23.9 | 18.5 | 12.7 | 8.8 |
| | Transtreaming-S*(Ours) | **25.3** | **20.9** | **15.5** | **10.8** |
| 3090 server | LongShortNet-S | 24.0 | 20.0 | 14.4 | 9.5 |
| | DAMO-StreamNet-S | 23.7 | 18.2 | 12.6 | 8.8 |
| | Transtreaming-S*(Ours) | **24.9** | **20.8** | **15.1** | **10.4** |
| 2080Ti server | LongShortNet-S | 23.0 | 18.0 | 12.7 | 8.6 |
| | DAMO-StreamNet-S | 21.9 | 16.4 | 11.3 | 8.0 |
| | Transtreaming-S*(Ours) | **23.5** | **19.0** | **13.2** | **9.1** |

Table 2: $sAP$ comparison for Computational Delay Adaptation with different real-world devices. * means the model is trained using mixed speed training technique. Best $sAP$ scores are indicated in bold.

| Sizes | Methods | $mAP^2$ | $mAP^4$ | $mAP^8$ | $mAP^{16}$ |
|---|---|---|---|---|---|
| S | LongShortNet-S | 26.4 | 20.6 | 13.4 | 8.9 |
| | DAMO-StreamNet-S | 28.9 | 22.0 | 14.6 | 9.4 |
| | Transtreaming-S(Ours) | **29.5** | 23.1 | 15.2 | 9.6 |
| | Transtreaming-S*(Ours) | 28.9 | **24.5** | **17.9** | **12.3** |
| M | LongShortNet-M | 30.7 | 23.8 | 15.8 | 9.7 |
| | DAMO-StreamNet-M | 31.7 | 24.4 | 16.0 | 10.2 |
| | Transtreaming-M(Ours) | **33.1** | 26.5 | 17.9 | 11.2 |
| | Transtreaming-M*(Ours) | 32.3 | **27.3** | **19.6** | **13.5** |
| L | LongShortNet-L | 33.2 | 25.8 | 17.2 | 10.7 |
| | DAMO-StreamNet-L | 33.8 | 26.1 | 17.1 | 10.8 |
| | Transtreaming-L(Ours) | **34.7** | 27.1 | 17.9 | 11.3 |
| | Transtreaming-L*(Ours) | 33.9 | **28.4** | **20.4** | **14.0** |

Table 3: Vehicle Acceleration Adaptation Comparison with different simulated speed variation. $2x, 4x, 8x, 16x$ of original vehicle speed are simulated by temporally downsampling frames, labeled as $mAP^2, mAP^4, mAP^8, mAP^{16}$, respectively. Best $mAP$ scores are indicated in bold.

simulated real-time situation. The $sAP$ metric compares the model output with ground truth in the output timestamp, rather than the input timestamp used in offline evaluations. $sAP$ use the same calculation formula as mean average precision ($mAP$), computing the average precision scores of matched objects with IOU thresholds ranging from 0.5 to 0.95. The $sAP$ scores for small, medium, and large objects (denoted as $sAP_S, sAP_M, sAP_L$, respectively) are also reported. To ensure a fair comparison, we did not employ mixed speed training in our model, in alignment with the training scheme of baseline methods.

**Computational Delay Adaptation Metric:** To simulate streaming perception on devices with diverse computation capabilities, we test the framework on 1 online GPU computing cluster (denoted as v100 cluster) and 3 real-world servers (denoted as 4080 server, 3090 server, and 2080Ti server) with different hardware specifications. The detailed information about these devices are listed in supplemental materials. Additionally, we adopt a delay factor $d$ to simulate high-delay situations, where all computational delays are multiplied by $d$. We assign $d \in \{2, 4, 8, 16\}$ and denote corresponding $sAP$ value as $sAP^d$.

| RTPE | TAT | $sAP$ | $sAP_{50}$ | $sAP_{75}$ |
|---|---|---|---|---|
| × | × | 29.0 | 50.1 | 29.1 |
| ✓ | PF & RQI | 29.8 | 49.7 | 29.4 |
| ✓ | PF | 31.6 | 52.4 | 31.1 |
| ✓ | RQI | 31.0 | 51.0 | 30.8 |
| × | ✓ | 32.1 | 53.2 | 31.6 |
| ✓ | ✓ | **32.5** | **53.9** | **32.2** |
| Planner | Buffer | $sAP$ | $sAP_{50}$ | $sAP_{75}$ |
| × | × | 3.9 | 5.8 | 4.1 |
| × | ✓ | 32.1 | 53.1 | 31.5 |
| ✓ | ✓ | **32.5** | **53.9** | **32.2** |

Table 4: Ablation study on Transtreaming-S. All the four components presented in this table are helpful in our framework. We split the ablation study into two parts: model components (RTPE & TAT) and strategy components (Planner & Buffer). Variations of TAT (PF: use past feature for the TAT attention value instead of difference of features, RQI: use random query initialization instead of duplication) are also reported. Note the Buffer module is crucial in streaming perception settings by avoiding huge computation cost, both for Transtreaming and baseline methods. Best $sAP$ scores are indicated in bold.

**Vehicle Acceleration Adaptation Metric:** To offer a more comprehensive comparison, we also evaluate the models (w/o strategy algorithm) under an offline setting using $mAP$. This approach ignores the impact of computational delay and only simulates different amplitude of vehicle accelerations, testing the model's ability to handle various objects' displacement between adjacent frames. Different from common offline evaluation, the model is given past frames $\{\mathrm{I}_i\}$ and evaluated against future objects $\{\mathrm{O}_j\}, j \in \{2, 4, 8, 16\}$. The resulting $mAP$ scores are denoted as $mAP^2, mAP^4, mAP^8, mAP^{16}$, respectively.

## Quantitative Results

**Overall Performance Comparison:** As the main result, our framework is evaluated against SOTA methods to demonstrate its strengths. Our proposed method achieves 38.2% in $sAP$, surpassing current SOTA method by 0.4%. Under S and M configurations, our pipeline also achieves the first place of most metrics, comparing to (Yang et al. 2022), (Li et al. 2023) and (He et al. 2023). The results clearly demonstrate the power of Transtreamer. Note that $sAP_L$ of our models is high, indicating that our proposed Transtreamer has recognized the temporal movement of large objects.

**$sAP$ Comparison for Computational Delay Adaptation:** To demonstrate the robustness of the Transtreaming framework, we employ the Computational Delay Adaptation Metric and evaluate both our framework and baseline

| Device Name | CPU | GPU | Memory |
|---|---|---|---|
| 2080Ti server | Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz × 48 | NVIDIA GeForce RTX 2080 Ti × 4 | 257547MB |
| 3090 server | Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz × 36 | NVIDIA GeForce RTX 3090 × 2 | 128527MB |
| 4080 server | Intel(R) Core(TM) i9-10900X CPU @ 3.70GHz × 20 | NVIDIA GeForce RTX 4080 × 2 | 257420MB |
| v100 cluster | Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz × 24 | Tesla V100-SXM2-32GB × 8 | 256235MB |

Table 5: The hardware specifications of four devices used in the experiment.

models across 4 devices with 4 different delay factor $d$. As illustrated in Table 2, our method surpasses current SOTA DAMO-StreamNet (He et al. 2023) by 1.1%-2.8% $sAP$ on numerous real-world devices, demonstrating its strength on computation-bound environments. Notably, in high-latency scenarios, LongShortNet surpasses DAMO-StreamNet despite using a lighter backbone, indicating that large models do not necessarily scale effectively with increased latency. The result indicates that our method maintains its performance superiority even on devices with low computing power.

$sAP$ **Comparison for Vehicle Acceleration Adaptation:** We also compare our framework for simulated circumstances with drastic vehicle accelerations. We evaluate our framework with and without mixed speed training, as to investigate the influence of the strategy. As shown in Table 3, our proposed framework achieves an absolute improvement of 0.6%, 1.1%, 0.6%, 0.2% under $mAP^2$, $mAP^4$, $mAP^8$ and $mAP^{16}$ respectively w/o mixed speed training compared to DAMO-StreamNet-S. Interestingly, though our models trained with mixed speed training only have a small advantage over other baseline models under $mAP^2$, a maximum 3.2% $mAP$ gain is observed in larger temporal intervals. The experimental results demonstrate the temporal flexibility of our framework.

**Device Hardware Specifications** As illustrated in Table 5, there are four devices with different computational capabilities used for our experiment. The Delay Adaptation Metric experiment is conducted on all four devices, while other experiments are only performed on the 4080 server. One thing to mention is that although we used different devices from those used in StreamYOLO (Yang et al. 2022) and DAMO-StreamNet (He et al. 2023) (RTX 4080 vs Tesla V100), it does not affect the validity of our experiments. Generally, the V100 has similiar or even slightly higher deep learning capacity (112 TFlops for V100 vs. 97.42 TFlops for 4080), so we believe the comparison to the SOTAs did not overestimate our method. Additionally, we also evaluated our S model on the V100 server, which achieved an sAP of 32.5%, the same as our 4080 results, further validating the fairness of the comparison.

## Ablation Study

The results of the ablation study are listed in Table 4. We verify the effectiveness of four proposed components: RTPE, TAT, Planner, and Buffer. For RTPE, we remove it from our model and observe a 0.4% decrease in the test $sAP$. We also utilize sinusoidal positional encoding and learnable positional encoding as substitutions, but both are inferior to our current approach. Variations of TAT are explored as well. The variations use past features for the TAT attention value (short for PF) and random query initialization (short for RQI), but exhibit inferior performance compared to our original approach. As for Planner and Buffer, we remove them from our framework and observe a 0.4% and an astonishing 28.2% decrease in the metric. The absence of Buffer will cause the detection model to compute the features of 4 past frames at each time, unrealistically extending the computational delay of the detector.

## Discussion

**Conclusion:** We introduce Transtreaming, a novel streaming perception framework that utilizes temporal cues to produce multiple prediction that aligns with real-world time, effectively producing real-time detection results. Transtreaming is the pioneer framework in streaming perception that makes use of temporal proposals and multi-frame output. Inspired by cross-attention, we design the Transtreamer detection model that handles input and output with dynamic temporal range. Adaptive Strategy is also proposed to provide buffer techniques and temporal cues to Transtreamer. Our framework not only outperforms current SOTA methods under ordinary streaming perception settings, but also surpasses SOTA methods by a large margin under high/dynamic computational delay and drastic vehicle acceleration environments.

**Limitation:** Despite the demonstrated strengths, Transtreaming still has limitations. First, though proposals $\mathbf{P^P}$ and $\mathbf{P^F}$ are produced to provide temporal information, Transtreamer only utilizes it in the computation of attention weights. A stronger integration could be achieved if it directly interacts with the input tokens. Second, our Computational Delay Adaptation Metric will sometimes not reflect the true ability of the model due to temporal aliasing effect. In other words, the model will perform poorly if computational time slightly misaligns with multiples of frame rate. Therefore, we should sample more values of $d$ to concretely evaluate the model. We leave these limitations for future work.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant No. 62232004, 62272099 and 61972086, the Natural Science Foundation of Jiangsu Province under Grant No. BK20231543 and BK20230024, the Fundamental Research Funds for the Central Universities, and the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education.

## References


Bertasius, G.; Torresani, L.; and Shi, J. 2018. Object Detection in Video with Spatiotemporal Sampling Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Deng, J.; Pan, Y.; Yao, T.; Zhou, W.; Li, H.; and Mei, T. 2019. Relation Distillation Networks for Video Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Feichtenhofer, C.; Pinz, A.; and Zisserman, A. 2017. Detect to Track and Track to Detect. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Ge, Z.; Liu, S.; Wang, F.; Li, Z.; and Sun, J. 2021. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430*.

Ghosh, A.; Balloli, V.; Nambi, A.; Singh, A.; and Ganu, T. 2024. Chanakya: learning runtime decisions for adaptive real-time perception. *Advances in Neural Information Processing Systems*, 36.

Girshick, R. 2015. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. ArXiv:1311.2524 [cs].

Han, M.; Wang, Y.; Chang, X.; and Qiao, Y. 2020. Mining Inter-Video Proposal Relations for Video Object Detection. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J.-M., eds., *Computer Vision – ECCV 2020*, 431–446. Cham: Springer International Publishing. ISBN 978-3-030-58589-1.

He, J.-Y.; Cheng, Z.-Q.; Li, C.; Xiang, W.; Chen, B.; Luo, B.; Geng, Y.; and Xie, X. 2023. DAMO-StreamNet: Optimizing Streaming Perception in Autonomous Driving. *arXiv preprint arXiv:2303.17144*.

Huang, Y.; and Chen, N. 2023. MTD: Multi-Timestep Detector for Delayed Streaming Perception. *arXiv preprint arXiv:2309.06742*.

Jo, W.; Lee, K.; Baik, J.; Lee, S.; Choi, D.; and Park, H. 2022. DaDe: Delay-adoptive Detector for Streaming Perception. *arXiv preprint arXiv:2212.11558*.

Li, C.; Cheng, Z.-Q.; He, J.-Y.; Li, P.; Luo, B.; Chen, H.; Geng, Y.; Lan, J.-P.; and Xie, X. 2023. Longshortnet: Exploring temporal and semantic features fusion in streaming perception. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Li, M.; Wang, Y.-X.; and Ramanan, D. 2020. Towards streaming perception. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, 473–488. Springer.

Liu, M.; and Zhu, M. 2018. Mobile Video Object Detection With Temporally-Aware Feature Maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. 2016. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*, volume 9905, 21–37. ISBN 978-3-319-46447-3.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; and Tran, D. 2018. Image Transformer. ArXiv:1802.05751 [cs].

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Shvets, M.; Liu, W.; and Berg, A. C. 2019. Leveraging Long-Range Temporal Relationships Between Proposals for Video Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Wang, S.; Zhou, Y.; Yan, J.; and Deng, Z. 2018. Fully Motion-Aware Network for Video Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Wu, H.; Chen, Y.; Wang, N.; and Zhang, Z. 2019. Sequence Level Semantics Aggregation for Video Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Xiao, F.; and Lee, Y. J. 2018. Video Object Detection with an Aligned Spatial-Temporal Memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Xu, R.; Zhang, C.-l.; Wang, P.; Lee, J.; Mitra, S.; Chaterji, S.; Li, Y.; and Bagchi, S. 2020. ApproxDet: content and contention-aware approximate object detection for mobiles. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 449–462.

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*, 5385–5395.

Zhu, X.; Dai, J.; Yuan, L.; and Wei, Y. 2018. Towards High Performance Video Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhu, X.; Wang, Y.; Dai, J.; Yuan, L.; and Wei, Y. 2017a. Flow-Guided Feature Aggregation for Video Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Zhu, X.; Xiong, Y.; Dai, J.; Yuan, L.; and Wei, Y. 2017b. Deep Feature Flow for Video Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.