2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

SToRM: Supervised Token Reduction for Multi-modal LLMs toward efficient end-to-end autonomous driving

Seo Hyun Kim, Jin Bok Park, Do Yeon Koo, Hogun Park†, Il Yong Chun†

***Abstract*— In autonomous driving, end-to-end (E2E) driving systems that predict control commands directly from sensor data achieved significant advancements. For safe autonomous driving in unexpected scenarios, one may additionally rely on human interventions such as natural language instructions. Using a multi-modal large language model (MLLM) in autonomous driving facilitates human–vehicle interactions, and may improve driving performances in unexpected scenarios. However, this approach requires substantial computational resources due to its reliance on an LLM and many visual tokens from sensor inputs, that are inherently limited in autonomous vehicles. Many MLLM studies have explored reducing the number of visual tokens, and many approaches tend to exhibit some end-task performance degradation compared to using all tokens. For efficient E2E driving while maintaining driving performance comparable to using all tokens, this paper proposes the first *S*upervised *To*ken *R*eduction framework for *M*ulti-modal LLMs (*SToRM*). The proposed SToRM framework consists of three key elements. First, we propose a lightweight *importance predictor* with short-term sliding windows that predicts the importance scores of visual tokens. Second, we propose a supervised learning approach for the importance predictor, that uses an auxiliary path to obtain *pseudo-supervision signals* from an all-token pass through the LLM. Third, guided by predicted importance scores, we propose an *anchor–context merging* module that partitions tokens into "anchors" and "context" tokens, then merges the latter into their most relevant anchors to reduce redundancy while minimizing information loss. Experiments with the LangAuto benchmark dataset show that the proposed SToRM outperforms state-of-the-art E2E driving MLLM under an equal reduced-token budget and maintains all-token performance while substantially reducing computational cost, by up to $30\times$, and enabling real-time E2E driving on a standard GPU.**

I. Introduction

The end-to-end (E2E) driving approach that directly transforms sensor data to control signals has achieved significant advancements in autonomous driving [1]. However, E2E driving methods often face challenges in complex and unforeseen scenarios that need high-level reasoning and contextual understanding. For safe autonomous driving, human intervention through natural language instructions is critical, by facilitating interactions between human decisions and vehicles [2]. For example, in scenarios where a pedestrian suddenly appears, an autonomous vehicle can promptly adapt to such scenario by additionally understanding the language instructions from a driver. By leveraging language instructions, one may mitigate the inherent generalization limitation in E2E driving, ultimately achieving safer driving in unforeseen scenarios [3]–[5].

With the success of large language models (LLMs) in language understanding, multi-modal LLMs (MLLMs) have been proposed to compensate for modality-specific limitations and to harness complementary information across modalities. MLLMs integrate information from diverse inputs (e.g., text, images, audio) by projecting modality-specific features into a shared representation space and processing aggregated tokens with an LLM backbone. This architecture enables joint reasoning across modalities and facilitates alignment between visual and textual information.

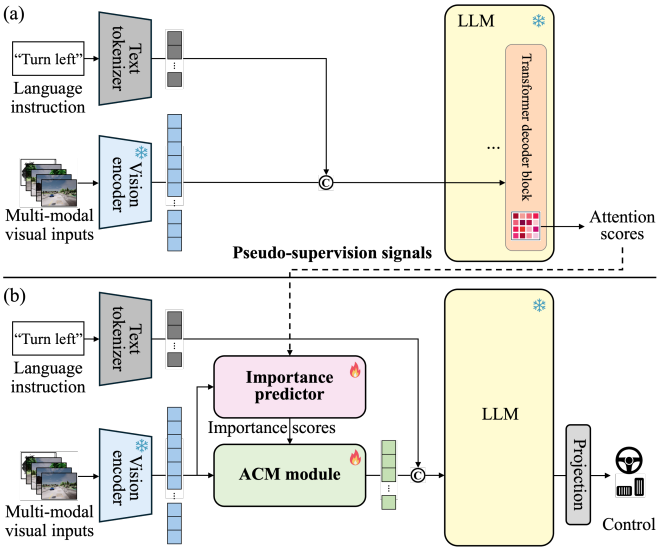

Fig. 1. Overview of the proposed SToRM framework. (a) SToRM is built on the central idea of leveraging intermediate results from an MLLM – specifically, attention scores derived from all tokens – as **pseudo-supervision signals** for training an importance predictor to reduce visual tokens. (b) In addition, we propose *i)* a new lightweight **importance predictor** learned from pseudo-supervision signals; and *ii)* an **anchor-context token merging (ACM)** module that reduces visual tokens while preserving essential information. We train SToRM in an E2E manner.

†Corresponding authors. The work was supported in part by NRF Grant RS-2023-00213455 funded by MSIT, the Digital Therapeutics Development and Demonstration Support Program funded by MSIT and NIPA, the BK21 FOUR Project, IITP Grant RS-2019-II190421 (AI Graduate School Support Program (Sungkyunkwan University)) funded by MSIT, KIAT Grant RS-2024-00418086 (HRD Program for Industrial Innovation) funded by MOTIE, and IBS-R015-D1.

Seo Hyun Kim, Jin Bok Park and Do Yeon Koo are with the Department of Electrical and Computer Engineering (ECE), Sungkyunkwan University (SKKU), Suwon 16419, South Korea (email: rlatjgus0608@g.skku.edu, bjb663@g.skku.edu, kdy1021@g.skku.edu).

Hogun Park is with the Department of Computer Science Engineering, Artificial Intelligence (AI), and Intelligent Software, SKKU, Suwon 16419, South Korea (email: hogunpark@skku.edu).

Il Yong Chun is with the Departments of ECE, AI, Advanced Display Engineering, and Semiconductor Convergence Engineering, Display Convergence Engineering, SKKU, Suwon 16419, South Korea, and also with the Center for Neuroscience Imaging Research, Institute for Basic Science, Suwon 16419, South Korea (email: iychun@skku.edu).

In autonomous driving, MLLMs can leverage complementary information from external sensors together with language inputs such as navigation instructions.

However, applying MLLMs to E2E autonomous driving systems has significant limitations. In E2E autonomous driving, temporal reasoning over historical information can improve driving performance by fusing features across multiple frames [6]. Yet, processing several past frames through a vision encoder produces a substantially larger number of visual tokens than text tokens. Leveraging many visual tokens may improve driving performance, but it incurs a considerable computational burden, since LLMs consist of numerous attention layers whose computational complexity grows quadratically with input length [7]. This severely degrades inference speed, which is critical for autonomous vehicles that demand real-time operation. To mitigate this issue, the prior work [3] reduced the number of visual tokens by Q-Former, a query-based transformer module [8]. Although this approach lowers computational cost, we observed that it often results in degraded E2E driving performance compared to using all visual tokens.

To reduce computational costs in E2E driving while maintaining performance comparable to using all tokens, this paper proposes the first **S**upervised **To**ken **R**eduction framework for **M**ulti-modal LLMs (**SToRM**). The central idea behind SToRM is to leverage intermediate results from an MLLM, particularly attention scores derived from all tokens, as pseudo-supervision signals for training an importance predictor to reduce visual tokens. This proposed mechanism is based on the assumption that tokens receiving higher attention in an LLM indicate greater importance. The proposed SToRM framework consists of three key elements: *i)* constructing pseudo-supervision signals (see above); *ii)* importance predictor for all visual tokens, and *iii)* anchor-context token merging (ACM) module. See the overview of SToRM in Fig. 1. To predict importance scores of visual tokens with low computational costs, we propose a new lightweight importance predictor that captures short-term spatio-temporal relations among visual tokens and intra-token cross-channel dependencies, designed via a new architecture of Multi-Layer Perceptron-Mixer (MLP-Mixer) [9] block with a temporal sliding window. To reduce visual tokens while minimizing information loss, we propose an ACM module that partitions visual tokens into "anchor" and "context" groups based on predicted importance, and then merges each context token into its most relevant anchor.

Our main contributions can be summarized as follows:

- We propose **SToRM**, the *first* supervised token-reduction framework for MLLMs in E2E driving, that leverages pseudo-supervision signals to guide importance-aware token reduction.
- We propose a lightweight importance predictor that captures *short-term* spatio-temporal relations among visual tokens rather than long-range dependencies spanning the entire token sequence; it also models intra-token cross-channel dependencies.
- We propose an ACM module that reduces the number of visual tokens by merging each context token into its most relevant anchor, with anchors selected based on high predicted importance scores.
- Our experiments with the LangAuto benchmark [3] show that the proposed SToRM outperforms state-of-the-art (SOTA) E2E driving MLLM under an equal token budget and maintains all-token performance, while substantially reducing computational resources.

II. Related Works

To handle challenging scenarios in E2E autonomous driving, one may use an MLLM with both visual and language information, promoting interactions between human decisions and information from multiple sensors. In general, such MLLMs suffer from a large number of visual tokens generated from raw sensor inputs that leads to quadratic growth in computation cost due to the numerous attention layers in an LLM backbone. To accelerate the inference speed of E2E driving MLLM, it is crucial to reduce the number of visual tokens.

Unlike Vision Transformers [10], where the number of visual tokens is fixed by patch size and image resolution, MLLMs feed multi-modal tokens into an LLM backbone that is inherently capable of processing variable-length sequences. Several studies have leveraged this property to reduce visual tokens in MLLMs.

For example, LMDrive, the first E2E driving MLLM, used Q-Former to reduce the number of visual tokens from several sensors [3]. Q-Former uses the cross-attention mechanism between learnable queries and visual tokens, and only these queries are passed to an LLM [8]. Beyond autonomous driving, many studies have also explored reducing visual tokens in MLLMs. For example, HICom is a hybrid-level instruction-guided token compression method that injects instructions into both local and global visual tokens using learnable queries [11]. HiRED is a plug-and-play token-dropping method that select the most informative tokens based on a special classification token ([CLS]) from vision encoder [12]. Token Merging (ToMe) gradually reduces the number of visual tokens in each transformer block by selecting the most representative tokens via bipartite matching [13]. LLaVA-PruMerge adaptively reduces visual tokens by pruning and merging based on similarities between the most representative token and others [14]. VisionZip selects informative visual tokens based on its attention score and merges remaining tokens based on similarity [15]. DivPrune is a token pruning method by selecting a subset that the diversity among the selected visual tokens is maximized [16].

However, the aforementioned methods reduce visual tokens *without* task supervision, relying on heuristic signals like similarity that led to a trade-off between inference efficiency and task performance. In contrast, the proposed supervised token reduction framework reduces computational cost *without* sacrificing performance, by supervising token importance with pseudo-importance scores.

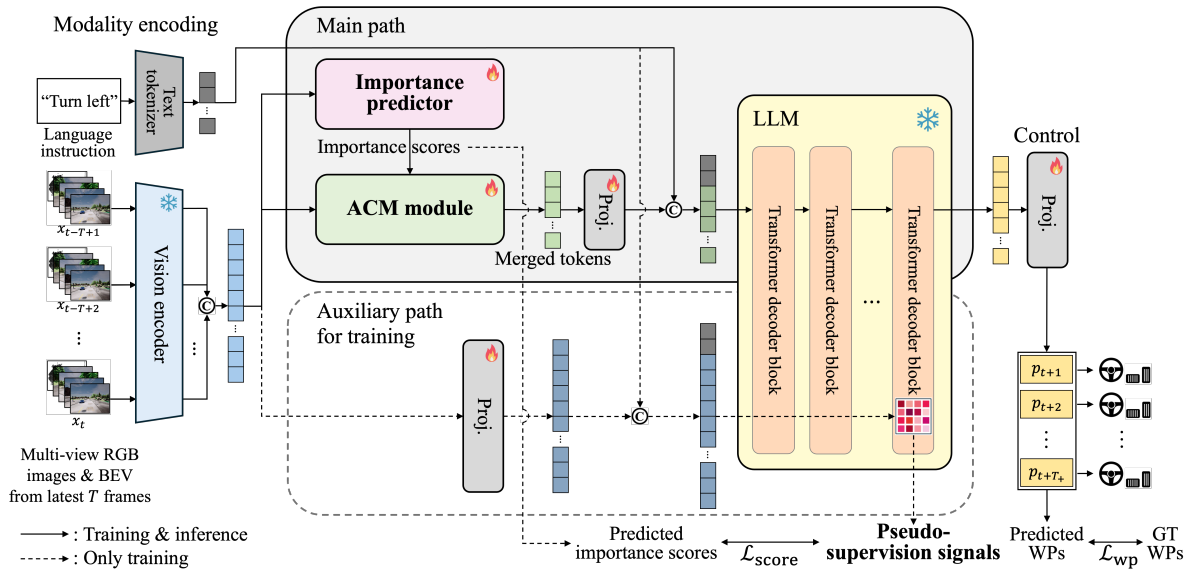

Fig. 2. The overall architecture of the proposed SToRM framework (at each current time step t). The symbol ⓒ, WPs, and GT denote a concatenation operator, waypoints over the next T_+ frames, and ground truth, respectively. The overall architecture is described at the beginning of §III.

III. Methods

The proposed SToRM framework takes red-green-blue (RGB) images from multi-view cameras, a bird's-eye-view (BEV) map derived from point clouds obtained by a light detection and ranging (LiDAR) sensor and language instructions as input, and predicts control commands (e.g., steer, throttle, and brake) as output through an MLLM, while reducing visual tokens. To capture temporal relations, we use a set of multi-modal visual inputs from T consecutive frames, paired with a language instruction. Fig. 2 illustrates the overall architecture of SToRM for E2E driving:

- **Modality encoding:** We employ a frozen vision encoder backbone to extract visual tokens from multi-view RGB images and a BEV map at each frame, and a text tokenizer to extract text tokens from a language instruction. We concatenate the visual tokens from T frames to capture temporal driving context.
- **Main path:** The main path consists of two core modules. First, we predict importance scores of all visual tokens from a vision encoder by the proposed lightweight **importance predictor**. We then reduce the number of visual tokens by the proposed **ACM module** using predicted importance scores. We project the merged tokens, concatenate them with text tokens, and feed them into a frozen LLM to generate output tokens for estimating control commands.
- **Auxiliary path for training:** The purpose of this path is to generate **pseudo-supervision signals** for training the importance predictor, by using *all* visual tokens *without* token reduction. We feed all visual and text tokens into the frozen LLM without token reduction. We then use its last decoder's attention scores as pseudo-supervision signals to train the importance predictor.
- **Control:** At each time step, we predict multiple future waypoints from the output tokens of the frozen LLM using reduced visual tokens, and convert them into control commands.
- **End-to-end training:** We train the entire SToRM in an E2E way by using two losses: *i)* $\mathcal{L}_{\text{score}}$ for training the importance predictor via the auxiliary path for training; and *ii)* $\mathcal{L}_{\text{wp}}$ for training the entire model using predicted waypoints.

A. Modality encoding

At each time step $t \in \mathbb{N}_{\geq T}$, we generate a set of visual tokens $\{\mathbf{Z}_{t'} \in \mathbb{R}^{N \times D} : t' = t - T + 1, \ldots, t\}$ from multi-view RGB images and a BEV map over the most recent T frames, using the vision encoder pretrained by [3], where N and D denote the number of visual tokens for each frame and the embedding dimension of each visual token, respectively. The vision encoder backbone [3] consists of two convolutional neural network (CNN) encoders and a fusion transformer. A single CNN encoder is shared across all views to extract features from RGB images, while the other CNN encoder processes the BEV map. A fusion transformer then generates visual tokens by fusing the features extracted from the two encoders. We transform a language instruction into text tokens using the SentencePiece tokenizer.

B. Main path

The main path consists of the following three core modules: *i)* proposed importance predictor, *ii)* proposed ACM module, and *iii)* an LLM backbone.

1) Proposed importance predictor: In this section, we propose a lightweight importance predictor that estimates the importance scores of visual tokens from a vision encoder backbone with low computational costs. Built upon the MLP-Mixer architecture [9], we propose a new mixing mechanism using sliding windows that captures spatial relations with local temporal context among visual tokens; and we also capture intra-token, cross-channel dependencies. See the overall architecture of the proposed module in Fig. 3(a).

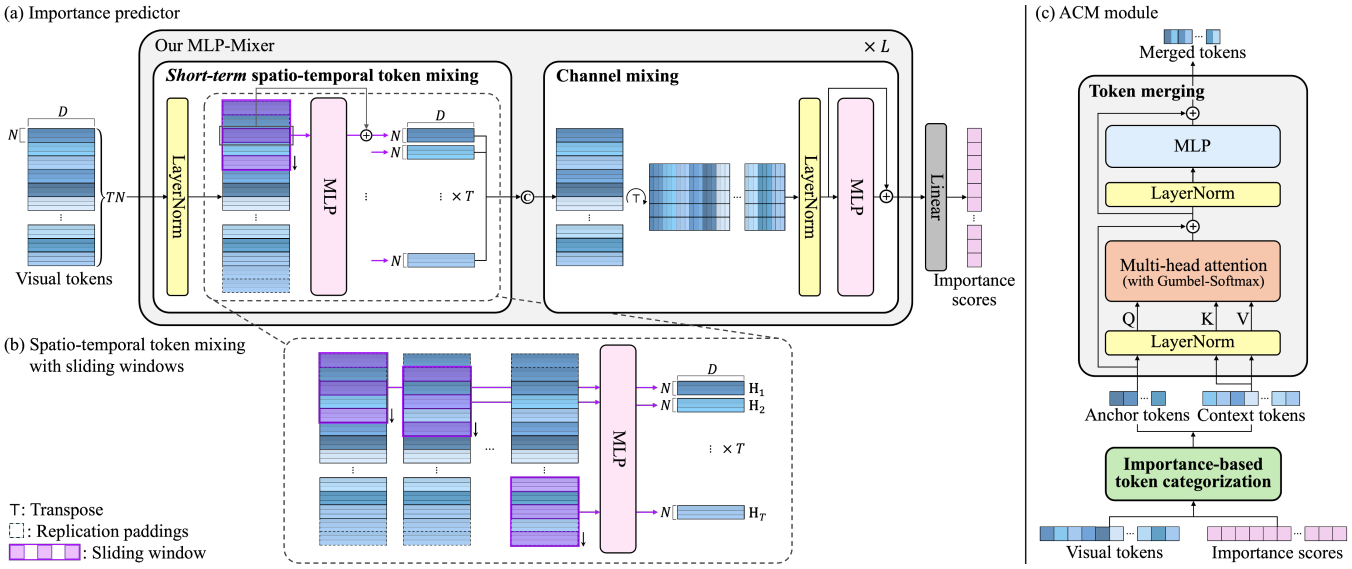

Fig. 3. The overall architectures of the proposed lightweight importance predictor and ACM module. (a) The proposed importance predictor consists of *i) short-term* spatio-temporal visual token mixing, *ii)* channel mixing, and *iii)* importance score computation. (b) The proposed *short-term* spatio-temporal visual token mixing mechanism with sliding windows. The input is the entire visual token matrix $\widetilde{\mathbf{Z}}$ in (1), where the τth block of N rows indicates the token-embedding matrix $\widetilde{\mathbf{Z}}_\tau$ at time step τ. A purple shaded block denotes a set of short-term spatio-temporal visual tokens selected by a sliding window, $\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)}$ in (2); the one-dimensional checkerboard indicates a dilated sliding window. The output of MLP, $\mathbf{H}_\tau$ in (3), represents both spatial structure and short-time temporal evolution at τ. (c) The ACM module comprises *i)* importance-based token categorization and *ii)* token merging: we first categorize visual tokens by predicted importance scores from (a), then merge "context" tokens into their most relevant "anchors" via cross-attention.

At each time step, we aggregate visual tokens across the most recent T frames by concatenating them in a frame-wise manner, followed by layer normalization:

$$\begin{aligned}\widetilde{\mathbf{Z}} &= [\widetilde{\mathbf{Z}}_1^\top, \cdots, \widetilde{\mathbf{Z}}_T^\top]^\top \\ &= \mathrm{LayerNorm}\big([\mathbf{Z}_{t-T+1}^\top, \cdots, \mathbf{Z}_t^\top]^\top\big) \in \mathbb{R}^{TN\times D} \end{aligned} \quad (1)$$

where we omit the time index t in $\widetilde{\mathbf{Z}}$ for simplicity. Here, we apply layer normalization, $\mathrm{LayerNorm}(\cdot)$, across the embedding dimension, normalizing each token's embedding vector to have zero mean and unit variance. If the token mixing module of the conventional MLP-Mixer [9] is naïvely applied to (1), it incurs substantial computational and memory overhead particularly due to the large TN (e.g., $T = 30$ and $N = 100$). To overcome this limitation, we propose a new spatio-temporal visual token mixing mechanism with sliding windows.

***Short-term* spatio-temporal visual token mixing.** Rather than considering all visual tokens in $\widetilde{\mathbf{Z}}$ of (1), we instead focus on a subset defined within a sliding window:

$$\begin{aligned}\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)} &= [\widetilde{\mathbf{Z}}_{\tau-\ell\cdot\kappa}^\top, \cdots, \widetilde{\mathbf{Z}}_{\tau+\ell\cdot\kappa}^\top]^\top \in \mathbb{R}^{|\mathcal{W}(\tau)|N\times D}, \\ \mathcal{W}(\tau) &= (\tau-\ell\cdot\kappa,\ \ldots,\ \tau,\ \ldots,\ \tau+\ell\cdot\kappa),\end{aligned} \quad (2)$$

for $\tau = 1, \ldots, T$, $\ell \in \mathbb{N}_{>0}$ denotes the window radius, such that the window size is given by $2\ell+1$ (e.g., $\ell = 1$ for a window size of 3 and $\ell = 2$ for a window size of 5), and $\kappa \in \mathbb{N}_{[1,\ell+1]}$ denotes the dilation factor. Since the window may exceed the valid range at sequence boundaries, we adopt *replication padding*: out-of-bound indices are set equal to the nearest valid frame, i.e., $\widetilde{\mathbf{Z}}_k = \widetilde{\mathbf{Z}}_1$ if $k < 1$ and $\widetilde{\mathbf{Z}}_k = \widetilde{\mathbf{Z}}_T$ if $k > T$. This ensures that the window size remains fixed at $2\ell+1$. We remark that $|\mathcal{W}(\tau)| = |\mathcal{W}|$, $\forall\tau$, i.e., the input dimension of a subsequent MLP is identical for $\{\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)} : \forall\tau\}$, where $|\cdot|$ denotes the length of a sequence.

TABLE I

COMPUTATIONAL COMPLEXITY COMPARISON BETWEEN NAÏVE APPL. OF EXISTING TOKEN MIXING AND PROPOSED TOKEN MIXING

Methods	Computation complexity
Existing token mixing w/ (1)	$\mathcal{O}(D\cdot(TN)^2)$
Proposed token mixing w/ (2)	$\mathcal{O}\big(D\cdot(2\ell+1)^2\cdot(T/\kappa)\cdot N^2\big)$

Now, we design an MLP to mix visual tokens within each sliding window, effectively capturing *short-term* spatio-temporal dependencies:

$$\mathbf{H}_\tau = \widetilde{\mathbf{Z}}_\tau + \mathbf{W}_2\cdot\sigma\big(\mathbf{W}_1\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)}\big) \in \mathbb{R}^{N\times D}, \quad (3)$$

for $\tau = 1, \ldots, T$, where $\widetilde{\mathbf{Z}}_\tau$ is given in (1), $\sigma(\cdot)$ denotes the GeLU activation, $\mathbf{W}_2 \in \mathbb{R}^{N\times 2|\mathcal{W}|\cdot N}$, $\mathbf{W}_1 \in \mathbb{R}^{2|\mathcal{W}|\cdot N\times|\mathcal{W}|\cdot N}$, and $\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)}$ is given in (2). Note that $\{\mathbf{W}_1, \mathbf{W}_2\}$ is shared for all τ. See an illustration of the proposed processes (2)–(3) in Fig. 3(b).

Here, $\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)}$ concatenates visual tokens from multiple frames within the sliding window, so the MLP captures both spatial relations (within each frame) and temporally local relations (across frames), enabling $\mathbf{H}_\tau$ to encode short-term spatio-temporal context. For example, the term $\mathbf{W}_1\widetilde{\mathbf{Z}}_{\mathcal{W}(\tau)}$ corresponds to a linear projection of all visual tokens from different spatial locations and time points within a sliding window (to a higher-dimensional space), where each token is modulated by a learnable vector in $\mathbf{W}_1$. In effect, the TN spatio-temporal tokens $\{\mathbf{H}_\tau : \tau \in [1, T]\}$ in (3) are visual tokens that represent both spatial structure and short-time temporal evolution.

The proposed model in (3) is lightweight, as it performs

spatio-temporal mixing only within a sliding window rather than across all tokens. By restricting interactions to a local temporal context, the model significantly reduces computational and memory overhead while still capturing essential spatio-temporal relations. This design choice enables efficient processing of long sequences, since the complexity grows with the window size rather than the entire sequence length. See Table I for computational complexity comparison between a naïve application of existing token mixing [9] using (2) and the proposed token mixing mechanism using slide windows via (3). Consequently, the sliding-window token mixing offers a favorable trade-off between efficiency and performance, making it a practical choice for latency-sensitive autonomous driving settings. We will later demonstrate that our approach (3) offers computational advantages and favorable driving performance trade-offs over an naïve application of existing token mixing in [9].

Mixing across the embedding dimension, a.k.a., channel mixing. Next, we introduce channel mixing to model how individual feature dimensions (channels) interact across temporally and spatially connected token sequences, complementing the spatio-temporal token mixing mechanism (3). This operation can enrich the representation of each token in $\{\mathbf{H}_\tau : \forall \tau\}$ in (3), by capturing dependencies that span along the embedding dimension, which are not addressed by token-level mixing (3) alone.

We aggregate transposed spatio-temporal token representations from T recent frames by concatenating them along the token dimension, followed by layer normalization in (1):

$$\widetilde{\mathbf{H}} = \mathrm{LayerNorm}\big([\mathbf{H}_1^\top, \cdots, \mathbf{H}_T^\top]\big) \in \mathbb{R}^{D \times TN}, \quad (4)$$

with $\{\mathbf{H}_\tau : \tau \in [1, T]\}$ from (3). Each row of the constructed matrix $\widetilde{\mathbf{H}}$ by (4) corresponds to a single embedding channel and spans the entire token sequence (all spatial locations across all T frames). For a given channel $d \in \mathbb{N}_{[1,D]}$, the row vector $\tilde{\mathbf{h}}_d^\top \in \mathbb{R}^{TN}$ traces how that feature dimension responds as the viewpoint changes and time progresses. Channel mixing learns a function over this per-channel sequence to capture dependencies along the token axis.

We model an MLP for channel mixing, similar to [9]:

$$\mathbf{U} = \widetilde{\mathbf{H}} + \mathbf{W}_4 \cdot \sigma\big(\mathbf{W}_3 \widetilde{\mathbf{H}}\big) \in \mathbb{R}^{D \times TN}, \quad (5)$$

where $\widetilde{\mathbf{H}}$ is given in (4), $\mathbf{W}_4 \in \mathbb{R}^{D \times 2D}$, and $\mathbf{W}_3 \in \mathbb{R}^{2D \times D}$. Concretely, we apply the two-layer MLP with a residual connection in (5) independently to each token vector (i.e., each column of $\widetilde{\mathbf{H}}$ in (4)), thereby mixing embedding channels within the token. This operation captures intra-token, cross-channel dependencies, whereas inter-token (sequence-wise) dependencies are captured by the preceding spatio-temporal token mixing module (3). The representation $\mathbf{U}$ in (5) thus can encode not only spatial-temporal information but also cross-channel dependencies, enriching the original visual token representations $\{\mathbf{Z}_\tau : \forall \tau\}$ in (1). We repeat the sequence of (1)–(5) for L blocks.

We will later show, in the context of predicting importance scores, that the proposed sliding window–based MLP mixer blocks (3) and (5) effectively avoid the quadratic computational complexity of conventional Transformer blocks.

Importance score computation. We finally compute the importance score of each token that quantifies its contribution to downstream decisions, by applying a linear projection:

$$\mathbf{s}^\top = \mathbf{W}_5 \mathbf{U} \in \mathbb{R}^{1 \times TN}, \quad (6)$$

where $\mathbf{U}$ is given in (5) and $\mathbf{W}_5 \in \mathbb{R}^{1 \times D}$.

2) Proposed ACM module: In this section, we propose an ACM module that, at each time step $t' \in [t-T+1, t]$, merges "context" tokens into a group of "anchor" tokens that are selected based on high predicted importance scores. Given visual tokens $\widetilde{\mathbf{Z}}$ in (1) and the corresponding importance $\mathbf{s}$ in (6) over recent T frames, proposed ACM operates independently at each frame by defining the followings: $\widetilde{\mathbf{Z}}_\tau \in \mathbb{R}^{N \times D}$ and $\mathbf{s}_\tau \in \mathbb{R}^N$, for $\tau = 1, \ldots, T$. This frame-wise design enforces a fixed per-frame token budget, reducing visual tokens to a constant count for each frame and preventing anchor selection from collapsing onto a few frames. The ACM modules comprises two sub-modules; see its overall architecture in Fig. 3(c).

Importance-based token categorization. Via the importance-based token categorization block in Fig. 3(c), at each frame τ, we rank the visual tokens in $\widetilde{\mathbf{Z}}_\tau$ with their importance scores $\mathbf{s}_\tau$, and categorize the top-K tokens as "anchor" tokens $\mathbf{A}_\tau \in \mathbb{R}^{K \times D}$ and the remaining $N-K$ tokens as "context" tokens $\mathbf{C}_\tau \in \mathbb{R}^{(N-K) \times D}$, where K is the number of anchors. Anchor tokens represent critical visual evidence on which a driving model relies, whereas context tokens provide complementary but less salient information. This scheme concentrates salient visual evidence in anchors, while context tokens carry complementary details that can be merged into anchors, thereby reducing the number of visual tokens. For example, in a crosswalk scene, tokens for the pedestrian, lane markings, and lead vehicle are treated as anchors. Tokens for road texture, shadows, and background patterns are context; merging them into the corresponding anchors preserves decision-critical cues while reducing the overall token count.

Token merging. To reduce visual tokens by minimizing redundancy while preserving critical information, we propose a new module that merges context tokens into their most relevant anchors. See the overall architecture of the proposed token merging sub-module in Fig. 3(c). At the τth frame, we first compute the similarity between the anchor tokens $\mathbf{A}_\tau$ and context tokens $\mathbf{C}_\tau$. To this end, we embed anchor tokens as queries (Q) and context tokens as keys (K) and values (V):

$$\mathbf{Q}_\tau = \widehat{\mathbf{A}}_\tau \mathbf{W}_\mathrm{Q}, \quad \mathbf{K}_\tau = \widehat{\mathbf{C}}_\tau \mathbf{W}_\mathrm{K}, \quad \mathbf{V}_\tau = \widehat{\mathbf{C}}_\tau \mathbf{W}_\mathrm{V}, \quad \forall \tau, \quad (7)$$

where $\widehat{\mathbf{A}}_\tau = \mathrm{LayerNorm}(\mathbf{A}_\tau)$ and $\widehat{\mathbf{C}}_\tau = \mathrm{LayerNorm}(\mathbf{C}_\tau)$, $\mathbf{W}_\mathrm{Q}, \mathbf{W}_\mathrm{K}, \mathbf{W}_\mathrm{V} \in \mathbb{R}^{D \times D_\mathrm{head}}$ are learnable projections (shared across frames), and D_head denotes the per-head dimension in the multi-head attention mechanism [10], $\tau = 1, \ldots, T$. Consequently, $\mathbf{Q}_\tau \in \mathbb{R}^{K \times D_\mathrm{head}}$ and $\mathbf{K}_\tau, \mathbf{V}_\tau \in \mathbb{R}^{(N-K) \times D_\mathrm{head}}$.

To encode anchor–context relationships as (near) hard assignments, we construct an assignment matrix $\mathbf{M}_\tau \in \mathbb{R}^{K \times (N-K)}$, where each column corresponds to a context

token and its entries give the (approximately one-hot) assignment probabilities over anchors. We adopt the Gumbel–Softmax operator, inspired by [17]:

$$\mathbf{M}_\tau = \text{Gumbel-Softmax}\big(D_{\text{head}}^{-1/2} \cdot \mathbf{Q}_\tau \mathbf{K}_\tau^\top\big), \quad \forall \tau, \tag{8}$$

where $\mathbf{Q}_\tau$ and $\mathbf{K}_\tau$ are the projected queries (anchors) and keys (context), respectively. The similarity matrix $\mathbf{Q}_\tau \mathbf{K}_\tau^\top$ is scaled by $\sqrt{D_{\text{head}}}$ for numerical stability. Unlike standard softmax – that yields soft, distributed assignments over anchors – Gumbel–Softmax provides a differentiable approximation to categorical sampling [18], producing columns of $\mathbf{M}_\tau$ that are nearly one-hot so that each context token is assigned to a single anchor.

To assign each context token to a single anchor, we convert each column of $\mathbf{M}_\tau$ into a one-hot vector by taking argmax over anchors. Because this mapping is non-differentiable, we adopt the straight-through estimation (STE) approach [19]: we use the one-hot matrix in the forward pass, but in backpropagation we pass gradients as if the soft scores $\mathbf{M}_\tau$ had been used. This enables end-to-end training while enforcing (near) one-hot anchor assignments for each context token. We denote the resulting STE-approximated hard-assignment matrix by $\widehat{\mathbf{M}}_\tau$.

Finally, we update the anchor representations so that the resulting tokens, $\widetilde{\mathbf{A}}_\tau \in \mathbb{R}^{K \times D}$, encode both their original salient signal and the complementary cues contributed by assigned context tokens. We realize this with the following residual merge-and-project model:

$$\widetilde{\mathbf{A}}_\tau = \mathbf{A}_\tau + (\widehat{\mathbf{M}}_\tau \, \mathbf{V}_\tau)\mathbf{W}_{\text{O}}, \quad \forall \tau, \tag{9}$$

where $\widehat{\mathbf{M}}_\tau$ is the STE-approximated hard-assignment matrix given above, $\mathbf{V}_\tau$ are context values given in (7), and $\mathbf{W}_{\text{O}} \in \mathbb{R}^{D_{\text{head}} \times D}$ is a learnable projection. The product $\widehat{\mathbf{M}}_\tau \mathbf{V}_\tau$ aggregates, for each anchor (row), all context tokens assigned to it, and the projection $\mathbf{W}_{\text{O}}$ maps the merged features back to the embedding dimension D. The resulting $\widetilde{\mathbf{A}}_\tau$ in (9) constitutes the reduced set of visual tokens for each frame.

3) LLM: We aggregate the reduced visual tokens in (9) from T recent frames – specifically, $\widetilde{\mathbf{A}} = [\widetilde{\mathbf{A}}_1^\top, \cdots, \widetilde{\mathbf{A}}_T^\top]^\top \in \mathbb{R}^{TK \times D}$ – and further concatenate $\widetilde{\mathbf{A}}$ with text tokens. Finally, we feed the reduced visual-text tokens to a frozen LLM to produce the output tokens for predicting waypoints.

C. Proposed auxiliary path for training

To train the importance predictor in §III-B.1, we proposes an auxiliary path that generates pseudo-supervision signals by using all visual tokens *without* reduction. We first project all visual tokens from a vision backbone, concatenate them with text tokens along the token dimension, and pass the result into the frozen LLM backbone (§III-B.3). From the last transformer decoder block in the LLM, we extract the attention score matrix, where rows correspond to query tokens and columns correspond to key tokens. To quantify how much attention each token receives, we take the column-wise mean of this matrix, resulting in a vector of pseudo importance scores. Each element of this vector indicates the average attention a token receives across all query positions (i.e., all text and visual tokens in the last self-attention layer), and tokens with higher pseudo-importance scores are considered more important for the downstream task. We use the pseudo-importance scores for visual tokens as supervision signals to train the importance predictor.

D. Control

From the output tokens of the frozen LLM (§III-B.3), we predict future waypoints at each time step t. Specifically, at (current) time t, given past T frames $t' = t - T + 1, \ldots, t$, the proposed model predicts T_+ future waypoints $\{p_{t+1}, \ldots, p_{t+T_+}\}$ via a two-layer MLP [3]. Following [3], we use two proportional–integral–derivative controllers to map predicted waypoints to low-level control: a lateral controller that tracks the trajectory heading to produce steering commands, and a longitudinal controller that regulates speed along the waypoints to produce throttle/brake commands.

E. E2E training loss

The proposed training objective comprises two components. First, we define the score loss $\mathcal{L}_{\text{score}}$ as the ℓ_1 distance between pseudo-supervision signals from the auxiliary path in §III-C and importance scores predicted by the importance predictor in §III-B.1. Second, the waypoint loss $\mathcal{L}_{\text{wp}}$ is defined as the ℓ_1 discrepancy between predicted future waypoints in §III-D and ground-truth waypoints. The final objective is $\mathcal{L} = \mathcal{L}_{\text{wp}} + \lambda \mathcal{L}_{\text{score}}$, where λ balances the two losses. The proposed design enables end-to-end training, jointly learning to predict waypoints and to estimate visual token importance.

IV. Experimental Results and Discussion

This section describes our experimental setups and presents the results with some discussion. §IV-B compares driving performances and inference efficiencies of three E2E driving MLLMs, each with two LLM backbones of different scales, LLaVA [20] and TinyLLaVA [21]: *i)* an E2E driving MLLM using all visual tokens, hereafter referred to as "all-token LMDrive", *ii)* SOTA E2E driving MLLM, LMDrive [3],[1] and *iii)* proposed SToRM. §IV-C compares driving performances of proposed SToRM with E2E driving MLLMs with the seven representative SOTA visual token reduction methods [8], [11]–[16]. For fair comparisons, we reduced the number of visual tokens to 120 across all the methods. §IV-D–§IV-E investigate variants of proposed importance prediction in §III-B.1 and ACM in §III-B.2.

A. Experimental setups

1) Dataset: We used the Language-guided Autonomous Driving (LangAuto) benchmark dataset [3] constructed by the CARLA simulator. The LangAuto dataset consists of challenging driving scenarios (e.g., highways, intersections, and roundabouts) and various environmental conditions (e.g., weathers and daylight). The LangAuto benchmark dataset

[1]The SOTA LMDrive setup incorporates Q-Former-based visual token reduction and LLaVA [3].

TABLE II

COMPARISONS BETWEEN STORM AND SOTA E2E DRIVING MLLM WITH TWO LLM BACKBONES OF DIFFERENT SCALES (THE SYMBOLS $^{\uparrow}$ AND $^{\downarrow}$ DENOTE THAT HIGHER AND LOWER VALUES ARE BETTER, RESPECTIVELY; LANGAUTO-LONG DATASET).

LLM backbones	MLLMs	# of visual tokens	Driving performances			Inference efficiencies		
			DS$^{\uparrow}$	RC$^{\uparrow}$	IS$^{\uparrow}$	TFLOPs$^{\downarrow}$	Memory$^{\downarrow}$	FPS$^{\uparrow}$
LLaVA (7B)	All-token LMDrive	3,000	44.0 ± 2.2	56.5 ± 3.8	$\mathbf{0.82} \pm 0.02$	30.80	16.20	4
	LMDrive	120	36.2 ± 2.3	46.5 ± 4.3	0.81 ± 0.03	1.04	**14.10**	24
	SToRM (ours)	120	$\mathbf{44.2} \pm 2.5$	$\mathbf{56.8} \pm 3.3$	$\mathbf{0.82} \pm 0.02$	**1.02**	**14.10**	**25**
TinyLLaVA (1.5B)	All-token LMDrive	1,800	39.6 ± 2.1	48.7 ± 4.0	0.82 ± 0.03	2.65	3.70	6
	LMDrive	120	30.9 ± 2.6	39.5 ± 4.1	0.82 ± 0.01	0.18	**3.30**	34
	SToRM (ours)	120	$\mathbf{40.8} \pm 1.9$	$\mathbf{49.3} \pm 3.5$	$\mathbf{0.84} \pm 0.01$	**0.16**	**3.30**	**36**

TABLE III

COMPARISONS BETWEEN STORM AND SOTA E2E DRIVING MLLM (LANGAUTO-SHORT AND LANGAUTO-TINY DATASETS)

LLMs	MLLMs	LangAuto-Short			LangAuto-Tiny		
		DS$^{\uparrow}$	RC$^{\uparrow}$	IS$^{\uparrow}$	DS$^{\uparrow}$	RC$^{\uparrow}$	IS$^{\uparrow}$
LLaVA	LMDrive	50.6	60.0	0.84	66.5	77.9	0.85
	SToRM	**64.5**	**74.7**	**0.88**	**78.8**	**86.9**	**0.92**
TinyLLaVA	LMDrive	43.2	56.1	0.83	60.5	74.9	0.86
	SToRM	**55.4**	**65.4**	**0.86**	**75.0**	**82.1**	**0.94**

TABLE IV

COMPARISONS B/W DIFFERENT TOKEN RED. METHODS IN E2E DRIVING

Reduction methods (120 visual tokens)	DS$^{\uparrow}$	RC$^{\uparrow}$	IS$^{\uparrow}$
Random	24.3 ± 1.8	36.2 ± 2.2	0.75 ± 0.02
ToMe [13]	28.4 ± 1.3	38.0 ± 1.9	0.75 ± 0.01
LLaVA-PruMerge [14]	30.7 ± 2.6	39.6 ± 2.9	0.80 ± 0.03
DivPrune [16]	31.5 ± 2.0	39.8 ± 2.7	0.79 ± 0.01
VisionZip [15]	34.7 ± 2.2	44.3 ± 2.5	0.78 ± 0.02
Q-Former [8]	36.2 ± 2.3	46.5 ± 4.3	0.81 ± 0.03
HiRED [12]	36.6 ± 2.3	46.7 ± 4.0	0.80 ± 0.01
HiCom [11]	37.5 ± 2.5	49.5 ± 3.8	0.78 ± 0.02
SToRM (ours)	$\mathbf{44.2} \pm 2.5$	$\mathbf{56.8} \pm 3.3$	$\mathbf{0.82} \pm 0.02$

includes *i)* RGB images from front-, left-, right-, and rear-facing cameras *ii)* point clouds from a center LiDAR, *iii)* navigation instructions (e.g., lane changing, turning), and *iv)* control commands. In constructing training, validation, test splits, we followed the setup [3]. For training, we used 51k chunks, totaling 2.5M frames. For validation, we used 13k chunks, totaling 0.6M frames. For testing, we used three sub-tracks with different route lengths: LangAuto-Long, Short, and Tiny. LangAuto-Long, -Short, and -Tiny contain routes of $> 500\,\mathrm{m}$, 150–500 m, and $< 150\,\mathrm{m}$, respectively. Unless otherwise specified, we report results with LangAuto-Long.

2) Evaluation metrics: We evaluated driving performance using route completion (RC), infraction score (IS), and driving score (DS). RC is the fraction of the route completed; IS $\in [0,1]$ quantifies agent-triggered infractions; and DS captures both progress and safety as RC $\times$ IS, reported as a percentage. We report inference efficiency in terms of floating-point operations (FLOPs; $1\,\mathrm{GFLOP} = 10^9$ FLOPs; $1\,\mathrm{TFLOP} = 10^{12}$ FLOPs), peak memory usage in gigabytes (GB) and frames per second (FPS), measured on a single NVIDIA RTX 4090 GPU. We report averages over three runs for all metrics.

3) Implementation details: Following [3], we used $\{T = 30, N = 100, D = 256, T_+ = 10\}$. For sliding windows of importance predictor in §III-B.1, we set $\{\ell = 1, \kappa = 2, L = 4\}$. For ACM module in §III-B.2, we set $\{K = 4, D_{\text{head}} = 64\}$. For training, we followed the hyperparameters and optimizer chosen in [3]. For the proposed loss in §III-E, we set λ as 50. We ran all experiments with four NVIDIA A100 GPUs and the CARLA simulator (ver. 0.9.10).

B. Comparisons between SToRM and SOTA E2E driving MLLM with different LLM scales

Across LLM backbones of different scales, Table II shows that the proposed SToRM achieves very comparable driving performances with SOTA E2E driving MLLM [3] using all tokens, i.e., all-token LMDrive, with far lower computational cost. In particular, compared with all-token LMDrive, SToRM reduces FLOPs by approximately $30\times$ with the large LLM backbone and $16.6\times$ with the tiny backbone. These reductions make real-time E2E driving inference feasible on a standard GPU, with over 25 FPS. Tables II-III show that under the same visual-token budget, SToRM outperforms SOTA model [3], indicating efficient token use and robust generalization across LLM sizes. Finally, Table II shows that compared with *the* SOTA model, LMDrive (using LLaVA + Q-Former token reduction) [3], SToRM with the tiny LLM backbone, TinyLLaVA, attains significantly better driving performance while substantially improving inference efficiency, specifically, $6.5\times$ reduction in FLOPs, $4.27\times$ less memory, and $1.5\times$ higher FPS.

C. Comparisons between different visual token reduction methods in MLLM-based E2E driving

Table IV demonstrates that with the E2E driving MLLM architecture held fixed, the proposed SToRM outperforms the representative SOTA visual token reduction baselines for E2E driving. This implies that the proposed task-relevant supervision for token reduction is more effective than heuristic, e.g., similarity-based criteria, particularly in E2E driving.

D. Comparisons b/w different importance predictor designs

To evaluate architectural choices for importance prediction, this section compares our lightweight MLP-Mixer-based predictor in §III-B.1 with a Transformer-based alternative (using four encoding blocks [7]), with and without the proposed sliding window mechanism in (2). Table V shows that the proposed lightweight MLP-Mixer-based predictor

TABLE V

COMPARISONS B/W DIFFERENT IMPORTANCE PREDICTOR DESIGNS

Block architectures	Our sliding window mech.	DS↑	RC↑	IS↑	GFLOPs↓
Transformer	×	**46.1**	**59.8**	**0.82**	11.3
Transformer	◯	45.0	58.1	0.80	5.4
MLP-Mixer	×	45.7	59.6	0.81	6.2
MLP-Mixer	◯	44.2	56.8	**0.82**	**2.1**

TABLE VI

COMPARISONS B/W DIFFERENT TOKEN REDUCTION SCHEMES IN ACM

Reduction schemes	DS↑	RC↑	IS↑
Only anchor tokens	41.8	53.6	**0.83**
Soft merging	40.1	51.2	0.80
Hard merging (ours)	**44.2**	**56.8**	0.82

achieves comparable driving performance to a Transformer-based alternative while substantially reducing computational cost, under both sliding-window and non-sliding configurations. Moreover, the proposed sliding-window mechanism in (2) significantly reduces computation without degrading driving performances, implying that short-term windows capture sufficient temporal context – obviating the need to process all visual tokens from the T most recent frames.

E. Comparisons between different visual token reduction approaches in ACM module

This section studies different token reduction schemes in the proposed ACM module in §III-B.2: *i)* only use top-K anchor tokens from our importance-based token categorization without merging, *ii)* a "soft" alternative that merges context tokens into *all* anchors with weighted contributions, and *iii)* the proposed "hard" scheme in (9), which merges each context token into its single most relevant anchor. Table VI shows that the proposed hard-assignment merging scheme outperforms the top-K-only selection scheme and the soft, naïve attention-based merging alternative. This suggests that hard merging preserves informative anchor representations, whereas soft merging introduces over-smoothing by merging contributions from all context tokens.

V. CONCLUSION

E2E autonomous driving models process multi-modal sensor inputs and can benefit from language guidance. However, MLLM-based designs are computationally heavy due to large LLM backbones and many multi-modal tokens. It is critical to reduce compute for resource-constrained vehicles.

We proposed SToRM, the first supervised token reduction framework for MLLMs that can significantly reduce computational cost *without* degrading E2E driving performance. The proposed SToRM outperformed the SOTA E2E driving MLLM [3] in both driving performances and inference efficiencies on the LangAuto benchmark. The key idea of SToRM is to leverage pseudo-supervision signals to guide importance-aware token reduction. SToRM employs a lightweight importance predictor and an ACM module that merges less important tokens into anchor tokens for efficient processing.

REFERENCES

[1] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Trans. on Intell. Veh.*, vol. 9, no. 1, pp. 103–118, 2024.

[2] X. Zhou, M. Liu, E. Yurtsever, B. L. Zagar, W. Zimmer, H. Cao, and A. C. Knoll, "Vision language models in autonomous driving: A survey and outlook," *IEEE Trans. on Intell. Veh*, pp. 1–20, 2024.

[3] H. Shao, Y. Hu, L. Wang, G. Song, S. L. Waslander, Y. Liu, and H. Li, "LMDrive: Closed-loop end-to-end driving with large language models," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2024, pp. 15 120–15 130.

[4] Z. Dong, Y. Zhu, Y. Li, K. Mahon, and Y. Sun, "Generalizing end-to-end autonomous driving in real-world environments using zero-shot LLMs," in *Proc. Conf. on Robot Learn.*, 2025, pp. 1231–1249.

[5] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, J. Beißwenger, P. Luo, A. Geiger, and H. Li, "DriveLM: Driving with graph visual question answering," in *Proc. Eur. Conf. Comput. Vis.*, 2025, pp. 256–274.

[6] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu, "ReasonNet: End-to-end driving with temporal and global reasoning," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2023, pp. 13 723–13 733.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6000–6010.

[8] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 19 730–19 742.

[9] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-Mixer: An all-MLP architecture for vision," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24 261–24 272.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[11] Z. Liu, C.-W. Xie, P. Li, L. Zhao, L. Tang, Y. Zheng, C. Liu, and H. Xie, "Hybrid-level instruction injection for video token compression in multi-modal large language models," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2025, pp. 8568–8578.

[12] K. H. I. Arif, J. Yoon, D. S. Nikolopoulos, H. Vandierendonck, D. John, and B. Ji, "HiRED: Attention-guided token dropping for efficient inference of high-resolution vision-language models," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 2, 2025, pp. 1773–1781.

[13] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token Merging: Your ViT but faster," in *Proc. Int. Conf. Learn. Represent.*, 2023.

[14] Y. Shang, M. Cai, B. Xu, Y. J. Lee, and Y. Yan, "LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models," *(Preprint) arXiv:2403.15388*, 2024.

[15] S. Yang, Y. Chen, Z. Tian, C. Wang, J. Li, B. Yu, and J. Jia, "VisionZip: Longer is better but not necessary in vision language models," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2025, pp. 19 792–19 802.

[16] S. R. Alvar, G. Singh, M. Akbari, and Y. Zhang, "DivPrune: Diversity-based visual token pruning for large multimodal models," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2025, pp. 9392–9401.

[17] J. Xu, S. De Mello, S. Liu, W. Byeon, T. Breuel, J. Kautz, and X. Wang, "GroupViT: Semantic segmentation emerges from text supervision," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2022, pp. 18 134–18 144.

[18] S. G. Eric Jang and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[19] A. van den Oord, O. Vinyals, and k. kavukcuoglu, "Neural discrete representation learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 6309–6318.

[20] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, vol. 36, 2023, pp. 34 892–34 916.

[21] B. Zhou, Y. Hu, X. Weng, J. Jia, J. Luo, X. Liu, J. Wu, and L. Huang, "TinyLLaVA: A framework of small-scale large multimodal models," *(Preprint) arXiv:2402.14289*, 2024.