ViPE: Visual Perception in Parameter Space for Efficient Video-Language Understanding

Shichen Lu[1*] **Tongtian Yue**[2,3*] **Longteng Guo**[2,3†]
Handong Li[2,3] **Xingjian He**[2,3] **Si Liu**[4] **Jing Liu**[2,3]
[1] School of Computer Science and Engineering, Beihang University
[2] Institute of Automation, CASIA [3] University of Chinese Academy of Sciences
[4] Institute of Artificial Intelligence, Beihang University
{sclu2020,liusi}@buaa.edu.cn {yuetongtian2022,lihandong2023}@ia.ac.cn
{longteng.guo, xingjian.he, jliu}@nlpr.ia.ac.cn

Abstract

Existing video-language models (Video-LLMs) typically rely on concatenating visual tokens with textual inputs for joint modeling. However, this token-level alignment leads to significant inefficiency, especially when scaling to long videos with dense visual inputs. In this work, we propose a video-to-parameter efficiency paradigm named ViPE that eliminates redundant visual tokens by transforming video content into visual perceptual weights, which are directly injected into the LLM's parameters. ViPE consists of a visual injection module that compresses video features into a small set of perceptual queries using a hierarchical merge strategy, and a visual perception module that integrates the resulting representations into the LLM through a lightweight LoRA-like mechanism. ViPE achieves performance comparable to token-based baselines such as LLaVA, while reducing FLOPs by 85% and inference time by up to 65%, demonstrating a highly efficient and scalable solution for video understanding.

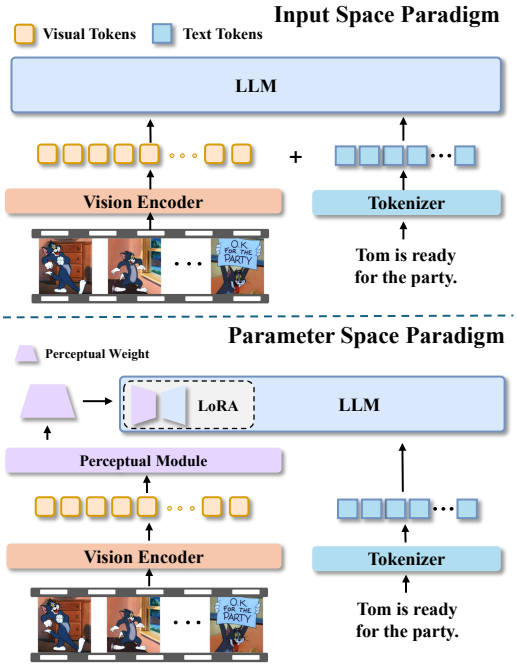

Figure 1: Comparison of input space and parameter space alignment paradigms.

1 Introduction

By integrating image and text inputs, Vision Language Models (VLMs) (Dai et al., 2023; Alayrac et al., 2022; Achiam et al., 2023) have achieved impressive performance in tasks such as image captioning and visual question answering. As the demand for richer and more dynamic visual understanding grows, the focus of VLM research has naturally progressed from static images to video data, giving rise to video-language models (Video-LLMs) (Jin et al., 2024; Li et al., 2024b; Zhang et al., 2024a). Most existing approaches (Liu et al., 2023; Team et al., 2023; Zhang et al., 2023b) follow a vision-language spatial alignment paradigm, where visual features extracted by an image encoder are converted into visual tokens and concatenated with text tokens. While effective for short clips or static images, this strategy suffers from severe efficiency bottlenecks when scaling to long videos, due to the rapid growth in visual token length, which leads to high computational cost, memory usage, and inference latency.

Recently, several approaches(Li et al., 2023b; Zhang et al., 2023a; Maaz et al., 2023; Luo et al., 2023) have been proposed to extend VLMs beyond static images to video understanding. These methods typically aim to mitigate the token burden by compressing visual inputs or incorporating memory mechanisms. However, the challenge of long video modeling remains largely unresolved. A key bottleneck lies in the excessive number of tokens required to represent frame-wise visual content. For instance, LLaVA(Liu et al., 2023) encodes a single

*These authors contributed equally to this work.
†Corresponding author: lognteng.guo@nlpr.ia.ac.cn.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 17776–17787
November 4-9, 2025 ©2025 Association for Computational Linguistics

image into over 576 visual tokens. A video containing 10K frames would thus necessitate over 5760K visual tokens, exceeding the capacity of current VLMs. Moreover, simple temporal compression strategies often result in severe degradation of long-range video representations, limiting the model's ability to retain temporal coherence and semantic richness.

To overcome these limitations, we explore a fundamentally different direction: instead of aligning at the token level, we align video content at the parameter level by injecting visual information directly into the LLM's weights. Building on this idea, we propose a novel video-to-parameter paradigm (ViPE) that replaces token-level alignment with a lightweight perceptual injection mechanism. ViPE transforms video features into learnable perceptual weights, which are then modulated into LLM parameters through a LoRA-like strategy, enabling efficient video-language understanding without relying on visual tokens during inference. To realize this paradigm, ViPE is designed with two key components: the Visual Injection Module and the Visual Perception Module. First, a visual encoder extracts features from the input video, which are then injected into a set of n learnable perceptual queries. To enhance the efficiency of this injection, we introduce a hierarchical context merge strategy that progressively filters redundant visual information, allowing the perceptual queries to capture more abstract and compact video representations. Second, the visual perception module generates visual perceptual weights that are integrated into the LLM layers using a low-rank adaptation(LoRA) style lightweight injection mechanism, enabling the LLM to perceive visual context without incurring additional computational cost. In summary, our method eliminates redundant visual token inputs, significantly enhancing computational efficiency. With 7M publicly available video-text pairs, ViPE achieves comparable video performance to the LLaVA paradigm while reducing FLOPs by 85% and inference time by a factor of 65%. Our contribution can be concluded as:

- We introduce the ViPE, a novel Video-to-Parameter paradigm that eliminates the need for redundant visual token inputs and enables efficient visual-language alignment.
- We design two lightweight modules: the Visual Injection module and the Visual Perception module that jointly enable compact yet expressive visual integration via hierarchical context merging and LoRA-style injection.
- ViPE achieves comparable performance to LLaVA while reducing FLOPs by 85% and inference time by 65%, demonstrating the paradigm's practicality and efficiency.

2 Related Works

2.1 Video Large Language Models.

Video-LLMs typically process videos by extracting and encoding frame-level features, which are then aggregated to form the final video representation. Several methods(Cheng et al., 2024; Li et al., 2024a; Ataallah et al., 2024; Luo et al., 2023) concatenate frame features directly to construct video inputs. For instance, Video-LLaMA(Zhang et al., 2023a) incorporates both visual and auditory signals via a Q-Former(Li et al., 2023a) to model temporal dependencies, while Video-LLaVA(Lin et al., 2023) introduces a unified framework that aligns visual representations before projecting them into the language space. When processing lengthy videos, longer visual tokens need to be properly handled on the LLM side. Video-ChatGPT(Maaz et al., 2023) and PLLaVA(Xu et al., 2024) adopt pooling strategies to compress visual tokens, but at the cost of information loss. LLaMA-VID(Li et al., 2024b)introduces a text-guided cross-attention mechanism to aggregate per-frame features into content tokens. While effective to some extent, these methods still follow the input space alignment paradigm, which constrains scalability and efficiency. In contrast, ViPE shifts alignment to the parameter space by injecting visual perception weights directly into the LLM. This token-free design eliminates the dependency on long visual sequences, enabling efficient video-LLM.

2.2 Parameter-Efficient Fine-Tuning.

Fine-tuning multi-billion-parameter large language models (LLMs)(Touvron et al., 2023a,b; Brown et al., 2020) is computationally expensive, motivating the development of parameter-efficient fine-tuning (PEFT) methods that freeze the backbone and train lightweight, task-specific modules. Representative PEFT strategies include Adapters(Houlsby et al., 2019; Pfeiffer et al., 2020; Sung et al., 2022), Prefix-tuning(Lester et al., 2021; Li and Liang, 2021), and Low-Rank Adaptation (LoRA)(Hu et al., 2022; Dong et al., 2024; Ma

et al., 2024). Adapters(Houlsby et al., 2019) insert small bottleneck layers between Transformer blocks, enabling task-specific adaptation with minimal training overhead. Prefix-tuning(Li and Liang, 2021) prepends trainable vectors to key and value matrices in attention, while LoRA(Hu et al., 2022) injects low-rank updates into weight matrices, effectively learning additive deltas without modifying the original weights. Despite their success, these methods have trade-offs: Adapters introduce additional inference latency, and prefix-tuning can be sensitive to initialization and optimization. In contrast, ViPE aligns video and language at the parameter level, inspired by the PEFT philosophy. It generates visual perception weights from video features and injects them directly into the LLM's parameters using a LoRA-style mechanism. This design enables efficient video-language modeling without introducing extra inference overhead, making ViPE highly suitable for efficient Video-LLM.

3 Method

3.1 LLaVA Paradigm

LLaVA (Large Language and Vision Assistant)(Liu et al., 2023) is an advanced multimodal architecture that combines vision and language processing capabilities. LLaVA leverages a vision encoder, such as ViT (Vision Transformer), and a large language model (LLM) to generate visual features F_v and textual features F_t. The main idea is to use a projector layer to transform the visual features into visual tokens, which are then mapped into the LLM's input space for joint training with textual tokens. Specifically, a pre-trained CLIP ViT-L/14 and a projector layer are employed to encode the input image I into visual tokens T_v. Then the visual tokens T_v and text token T_t are fed into an LLM, such as Vicuna or Qwen to generate the response X. In practice, visual tokens are often connected with text tokens, so the inputs of LLM can be formally represented as:

$$X = LLM(T_v; T_t) \in \mathbb{R}^{(N_v+N_t)\times d} \quad (1)$$

When N_v and N_t denote the lengths of the visual tokens and text tokens, respectively. For instance, in LLaVA-1.5, N_v is 576, and N_t typically ranges from 20 to 80. In video processing scenarios, each frame will be process independently encoded as visual tokens T_v. When processing F frames, the total number of visual tokens increases to $F \times T_v$. This leads to a significant computational overhead in the LLM, particularly in the self-attention mechanism, where the complexity scales quadratically with the input length:

$$\mathcal{O}\left((N_t + F \cdot N_v)^2 \cdot d\right) \quad (2)$$

When N_v and N_t denote the lengths of the visual tokens and text tokens, d is the hidden dimension. It can be observed that the computational complexity $\mathcal{O}$ is positively correlated with the square of the input token length. The excessively long visual token inputs have become a major limitation for the practicality of video large language models.

3.2 ViPE

Inspired by Parameter-Efficient Fine-Tuning (PEFT), we consider whether the alignment paradigm can be shifted from the input space to the parameter space, eliminating the substantial computational overhead introduced by video tokens $F \cdot N_v$. To address this limitation, we propose ViPE, a novel Video-to-Parameter alignment paradigm that eliminates the need for additional visual token inputs. Instead of injecting visual information through tokens, ViPE converts video features into perceptual weights, which are then integrated directly into the LLM's parameters for alignment. ViPE has two key components: 1) Visual Injection Module, which injects video information into a set of visual perceptual queries Q_v, which serve as compact and semantically rich representations for downstream parameter generation. 2) Visual Perception Module, which transformed the visual perceptual queries Q_v into visual perceptual weights and injected them into the parameters of the LLM, enabling the model to acquire the ability to understand video information.

Visual Perceptual Query Generation. To enable video information to be effectively injected into the model parameters, we first employ a visual injection module to transform video features into a set of learnable perceptual queries. This module serves as a bridge between raw video features and parameter-level adaptation. The Visual Injection Module is composed of three core components: self-attention, cross-attention, and a feed-forward.

We begin by initializing a set of N learnable visual perceptual queries $Q \in \mathbb{R}^{N\times D_v}$. Let $\mathbf{V} = F \cdot N_v$ and $\mathbf{V} \in \mathbb{R}^{B\times N\times P\times D_v}$ be the video features extracted by vision encoder, where B is the batch size, N is the number of frames sampled, P is the video patches, and D_v is the visual embedding dimension. The visual perceptual query after

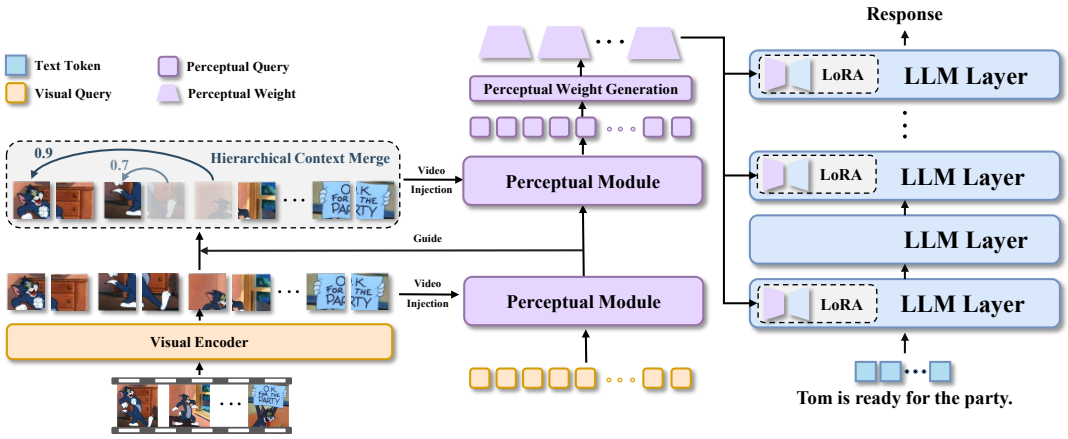

Figure 2: Overview of the ViPE.ViPE injects video information into perceptual queries via a visual injection module and generates visual perceptual weights through a perception module. These weights are then integrated into the LLM using a LoRA-based mechanism. Additionally, a Hierarchical Context Merging strategy is introduced to merge redundant video information.

self-attention is computed as a weighted sum of the values:

$$Q' = \text{SelfAttn}(Q) = \text{softmax}\left(\frac{Q_q Q_k^\top}{\sqrt{d}}\right) Q_v \quad (3)$$

This allows visual perceptual queries Q to interact with one another and model intra-query relationships. Then updated visual perceptual queries attend to the visual features via cross-attention, integrating spatial-temporal cues from the video:

$$\tilde{Q}_v = \text{Cross Attn}(Q', V, V) \quad (4)$$

This enables each visual perceptual query to focus on relevant visual content across frames. Finally, a feed-forward network is applied to further enhance the representation capacity:

$$Q_v = \text{FFN}\left(\tilde{Q}_v\right) \quad (5)$$

These visual perceptual queries Q_v are then projected and generate visual perceptual weights.

Visual Perceptual Weight Generation. To convert visual perceptual queries to perceptual weight, we propose the visual perception generator module. The visual perception queries are first projected into the LLM's hidden space via a learnable linear projector:

$$\Delta W_v' = \mathcal{F}(Q_v), \mathcal{F} \in \mathbb{R}^{D_v \times D} \quad (6)$$

In an LLM, different parameter types at different layers exhibit varying levels of information sensitivity. For example, lower layers tend to capture fine-grained visual patterns such as textures and edges, while higher layers are more responsive to semantic-level concepts. Additionally, each weight type plays a distinct role, like query, key, value, output, and mlp. To effectively inject visual information across the model, we assign layer-specific adaptation weights for each parameter type. However, directly learning a full-rank matrix for every layer and weight type would introduce a prohibitive number of additional parameters, which contradicts our goal of efficiency. To address this, we adopt a low-rank decomposition strategy to generate the visual-aware adaptation weights. Specifically, for each layer, the visual perception weight is formulated as:

$$\Delta W_v = W_v W_l \quad (7)$$

Where $W_v \in \mathbb{R}^{D \times r}, W_l \in \mathbb{R}^{r \times D}$, $r \ll D_{lmm}$ is the decomposition rank. Only W_v will introduce visual perceptual. This formulation allows us to maintain expressive capacity while significantly reducing parameter overhead, enabling scalable and efficient video-to-parameter alignment across layers and weight types. Then, we can inject the visual perception weights ΔW_v into the LLM's weights for alignment training. The overall model weights are as follows:

$$W_{total} = W_{llm} + \Delta W_v \quad (8)$$

Unlike the alignment approach used by LLaVA, ViPE inject video information into the LLM by converting visual features into perceptual weights. The advantage of this approach is that it eliminates

the need for long visual token sequences, significantly enhancing the model's efficiency. Additionally, by leveraging weight fusion, we avoid introducing additional inference burdens during the inference phase, thereby improving the practicality of video large language models.

3.3 Hierarchical Context Merging

To improve the efficiency of cross-attention operations between the visual perceptual queries and high-dimensional video features, we propose a Hierarchical Context Merging (HCM) strategy. This approach is designed to progressively filter redundant visual tokens across layers, allowing each layer within the Visual Injection Module to attend only to the most informative subsets of video features. Specifically, to identify redundant visual tokens at each layer ℓ, we compute a pairwise cosine similarity matrix $S \in \mathbb{R}^{N_q \times N_v}$ between each query in the set of visual perceptual queries $Q^{(\ell)} \in \mathbb{R}^{N_q \times D}$ and each visual token in visual features V:

$$S_{i,j} = \frac{Q_i^{(\ell)} \cdot V_j^{(\ell)}}{|Q^{(\ell)}i||V^{(\ell)}j|}, \quad \forall i \in [1, N_q], j \in [1, N_v] \tag{9}$$

For each visual token j, we then define its relevance score as the maximum similarity across all queries:

$$r_j = \max_{i \in [1, N_q]} S_{i,j} \tag{10}$$

A higher r_j indicates that the token's information is already highly aggregated within at least one query. Hence, a top-k filtering is applied to discard the k most relevant visual tokens (*i.e.*, those with highest r_j values), denoted as $\tilde{V}^{(\ell)} \in \mathbb{R}^{k \times D}$. The remained tokens are subsequently used in the cross-attention operation detailed in Equation (4).

This hierarchical merging strategy is applied across layers, where each layer filters out redundant visual tokens based on their cosine similarity to the perceptual queries. By progressively discarding already-attended tokens, deeper layers attend to increasingly abstract and informative visual content. This not only reduces the computational cost of cross-attention, but also encourages the queries to capture complementary semantics at each stage. The top-k parameter can be layer-adaptive, enabling efficient and expressive visual integration that further enhances the overall effectiveness of our token-free ViPE paradigm.

3.4 Analysis of Effiency

In this section, we compare the computational cost of ViPE and LLaVA-v1.5-7B to validate the efficiency of our proposed method. For convenience, we only consider the computational cost of the self-attention and feed-forward network in LLM. When the hidden dimension of the LLM is denoted as D, D_{ffn} denotes the hidden dimension of the feed-forward network, and the input visual and textual tokens are of length $L_v and L_t$. The floating-point operations (FLOPsϕ) can be estimated as follows:$\phi_{attn} = 4D^2$, $\phi_{ffn} = 2 \times D \times D_{ffn}$, and $\phi_{llm} = \phi_{attn} + \phi_{ffn}$. Specifically, the flops of LLava can be described as:

$$\phi_{llava} = (L_v + L_t) \times \phi_{llm} \tag{11}$$

Compared to previous token-based approaches, ViPE eliminates the need for visual token input and instead introduces visual perception weights ΔW_v, which are injected into the model's parameters. The corresponding computational cost of ViPE is calculated as:

$$\phi_{ViPE} = L_t \times (\phi_{llm} + \phi_{\Delta W}) \tag{12}$$

Here, $\phi_{\Delta W} = n \times (16r \times D + 4r \times (D + D_{fnn}))$ and r is the rank used in the low-rank decomposition of the injected visual perceptual weights, n denotes the number of target weight types(e.g query, key, value, output, and mlp). For inference, the perceptual weights can be merged into the LLM:

$$\phi_{inference} = L \times \phi_{llm} \tag{13}$$

4 Experiments

Implementation Details. We use Vicuna-7B-v1.5(Zheng et al., 2023) as the foundational LLM and CLIP ViT/L-14(Radford et al., 2021) as the vision encoder. For the visual perceptual weight, we set the hidden size $D_v = 512$, and both the query number, rank, and alpha are set to 64. To inject the visual perceptual weights into the LLM layers, we adopt a sparse approach. Specifically, for Vicuna-7B, which contains 32 transformer layers, we inject a visual perceptual weight ΔW_v very 4 layers, resulting in a total of 8 instances. For ΔW_l, we initialize it to zero to ensure the stability of the training process. For video inputs, we utilize the full outputs of the vision encoder as vision conditioning. For video inputs, we uniformly sample 32 frames. During the pre-training phase, only the

Table 1: Performance on zero-shot video-language benchmarks. We evaluate ViPE on 3 short-video benchmarks and 5 long-video benchmarks. VC denotes Video-ChatGPT. Vision Tokens is the number of vision tokens fed to LLM per frame.

Model	**Visual Tokens**	**Short Video Benchmark**			**Long Video Benchmark**				
		MSVD	ActivityNet	VC	EgoSchema	MVBench	VideoMME	MLVU	CinePile
VideoLLaMA	256	51.6/2.5	12.4/1.1	1.98	-	34.1	-	-	-
VideoChat	64	56.3/2.8	26.5/2.2	2.29	-	35.5	-	-	-
Video-ChatGPT	576	64.9/3.3	35.2/2.7	2.42	-	32.7	-	31.3	14.6
Chat-UniVi	112	65.0/3.6	46.1/3.3	2.99	-	-	-	-	-
Video-LLaVA	576	70.7/3.9	45.3/3.3	2.84	38.4	41.0	39.9	47.3	22.5
LLAMA-VID	2	69.7/3.7	47.4/3.3	2.89	38.5	41.9	25.9	33.2	-
LLaVA-NeXT-Video	576	67.8/3.5	53.5/3.2	**3.26**	43.9	33.7	-	46.5	-
VideoChat2	32	70.0/3.9	49.1/3.3	2.95	**54.4**	**60.4**	42.3	47.9	29.3
VideoLLaMA2	576	70.9/3.8	50.2/3.3	3.13	51.7	54.6	46.6	48.5	44.6
LLaVA-Mini	1	70.9/4.0	53.5/3.5	-	51.2	44.5	-	42.8	-
ViPE	0	**71.6/4.0**	**53.6/3.5**	3.07	51.2	51.3	**47.2**	**51.3**	**45.3**

visual perceptual weights are trainable. We use the AdamW optimizer with a learning rate of 2e-5, employing a linear warm-up followed by cosine decay. The global batch size is set to 256. During fine-tuning, the entire LLM is made trainable. The learning rate is again set to 2e-5, and the learning rate schedule remains the same as in pre-training. All experiments are conducted on 8 NVIDIA A800 GPUs, with 48 hours for pre-training and 30 hours for fine-tuning.

Training Data. For pre-training, we utilized a dataset including 4M image-text pairs and 3M video-text pairs. The supervised fine-tuning (SFT) dataset contains LLaVA665K(Liu et al., 2023)and LLaVA-Video178K(Zhang et al., 2024b). Specifically, the image data includes 1M original samples from DenseFusion and 3M re-annotated samples derived from the CC3M and COCO datasets. The video data consists of 2M samples from WebVid(Bain et al., 2021) and 1M samples from VALOR(Chen et al., 2023), with all captions re-annotated for consistency and quality.

Benchmarks for Evaluation. We evaluate ViPE on a series of comprehensive visual language benchmarks:1) Video-based benchmarks include MSVD-QA(Xu et al., 2017), ActivityNet-QA(Yu et al., 2019), Video generative benchmark(Maaz et al., 2023) and 2) Long Video Understanding encompass Egoschema(Mangalam et al., 2023), CinePile(Rawal et al., 2024), VideoMME(Fu et al., 2024), MVBench(Li et al., 2024a), MLVU(Zhou et al., 2024).

4.1 Main Results

We evaluate ViPE's performance on video-related tasks across three short video benchmarks and five long video benchmarks. In Table1, ViPE achieves competitive or superior results across these benchmarks. In terms of computational efficiency, ViPE reduces the overall training and inference cost by approximately 85%, highlighting the scalability and practicality of our token-free video-to-parameter paradigm. In addition, ViPE surpasses existing video understanding models that rely on visual-text token alignment, such as Video-LLaVA(Lin et al., 2023), LLaMA-VID(Li et al., 2024b), and LLaVA-Mini(Zhang et al., 2025), in both short and long video tasks. This performance improvement highlights the efficacy of our visual-to-parameter alignment paradigm. By replacing the need for long, cumbersome visual token inputs with perceptual weights, we enable the LLM to process video content more effectively, extracting essential visual information while minimizing computational overhead. The shift from token-level alignment to parameter-level integration ensures that ViPE retains high performance, offering an efficient solution for video understanding.

4.2 Efficiency Analysis

To evaluate the efficiency of our method, we report visual token, max memory allocated, average inference time per sample, and flops. As shown in Table1, LLaVA-v1.5-Video is our reimplementation of the LLaVA-1.5 framework using the same pre-training and fine-tuning datasets as our method. The results demonstrate that ViPE achieves comparable performance to LLaVA-v1.5-Video on standard video benchmarks. Notably, LLaVA-v1.5-Video relies on 4,608 visual tokens per sample, leading to approximately twice the training time compared to ViPE. All models are deployed on NVIDIA A100

Table 2: Efficiency analysis under different models. We report the maximum visual token, max memory allocated, and inference time, respectively.

Model	Flops(T)	Men(G)	Inference Time(ms)	EgoSchema	MVBench	VideoMME	MLVU	CinePile
Video-LLaVA	7.5	20.4	391	38.4	41.0	39.9	47.3	22.5
LLAMA-VID	4.3	19.8	273	38.5	41.9	25.9	33.2	-
LLaVA-V1.5-Video	8.4	20.7	365	**51.4**	50.1	46.6	**51.3**	43.9
ViPE	1.3	16.7	132	51.2	**51.3**	**47.2**	**51.3**	**45.3**

Table 3: Effect of video frames sample.

Frames	EgoSchema	MVBench	VideoMME	MLVU	CinePile
4	43.2	43.5	38.6	44.2	41.8
8	46.3	48.3	41.2	46.3	42.3
16	48.7	50.0	43.4	49.3	44.8
32	**51.4**	**51.3**	**47.2**	**51.3**	**45.3**

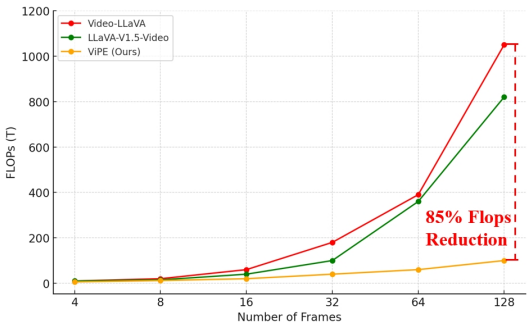

Figure 3: FLOPs comparison across frame counts.

GPUs. For the LLaVA-based baselines, visual tokens are directly concatenated with text tokens and fed into the model without any reduction or optimization, which leads to significantly higher peak memory usage and FLOPs due to the large number of visual tokens. In contrast, ViPE eliminates the need for explicit visual token inputs by directly injecting visual perceptual weights during inference. This approach reduces the inference time by 3× compared to Video-LLaVA, 2× compared to LLaMA-VID, and 2.8× compared to LLaVA-V1.5-Video, highlighting ViPE's superior efficiency in video-language understanding tasks.

In Fig3, we compare FLOPs ViPE and baseline models as frame counts grow. ViPE achieves significant computational overhead reduction, saving 85.0% in FLOPs, at 128 frames compared to LLaVA-V1.5-Video, while maintaining superior performance on video understanding benchmarks.

4.3 Ablation Study

Effect of Video Frames. To evaluate the impact of temporal granularity, we vary the number of sampled frames from 4 to 32 and report performance across five long video benchmarks, as shown in Table3. To assess the impact of temporal granularity, we vary the number of sampled frames from 4 to 32 and evaluate performance across five long-video benchmarks, as summarized in Table 3. The results reveal a consistent trend: increasing the number of frames substantially improves model performance, suggesting that denser temporal sampling enhances the model's ability to capture long-range temporal context. For example, on EgoSchema, performance improves from 43.2 to 51.4 when the number of frames increases from 4 to 32. Similar improvements are observed on MVBench and VideoMME. However, this performance gain comes at a significant computational cost in token-based models, as each additional frame introduces hundreds of visual tokens. This leads to a rapid increase in memory usage and inference time, limiting practicability for long videos. ViPE addresses this challenge by eliminating redundant visual tokens and directly injecting video information into the LLM's parameters. As a result, ViPE is able to benefit from richer temporal input without suffering from the prohibitive computational overhead associated with traditional token-based approaches.

Effect of Hierarchical Context Merging. To evaluate the effectiveness of our proposed hierarchical context merge, we examine how different merge method and ratios impact performance across long video benchmarks. This strategy aims to reduce the number of visual perceptual queries by hierarchically merging less informative ones, while preserving key semantic content.

We conduct ablations by replacing it with two common strategies: ViPE + Frame Summarization (FS): Uniformly sample fewer frames to match HCM's token budget. ViPE + Token Pooling (TP): Keep all frames, but apply mean pooling for token pruning under the same compression ratio. All variants are configured to output the same number of visual tokens. HCM consistently outperforms FS and TP across all five long video benchmarks. As shown in Tabel4, HCM is a more effective method for pro-

Table 4: Comparison of Hierarchical Context Merging and Previous Methods.

Methdo	EgoSchema	MVBench	VideoMME	MLVU	CinePile
VIPE with HCM	51.2	**51.3**	**47.2**	**51.3**	**45.3**
VIPE with FS	**51.5**	50.9	46.8	51.1	45.1
VIPE with TP	49.8	48.5	45.6	49.3	42.4

Table 5: Effect of hierarchical context merging. The Merge Ratio represents the ratio merged to the total tokens.

Merge Ratio	Flops	EgoSchema	MVBench	VideoMME	MLVU	CinePile
80	24.3	**51.3**	**51.0**	**47.5**	51.1	**45.6**
60	20.7	51.2	50.7	47.2	**51.3**	45.3
40	16.8	49.7	49.6	44.4	49.9	42.4
20	13.5	48.2	47.7	42.7	48.1	39.7

cessing long videos, as it better retains key spatio-temporal information compared to traditional frame sampling or token pooling techniques.

As shown in Table5, when the merge ratio is set to 60% or higher, ViPE maintains strong and stable performance across all benchmarks. Performance degradation only becomes noticeable when the merge ratio drops below 40%. At 20%, significant drops appear across datasets, particularly on visually complex benchmarks such as CinePile (from 45.6 to 39.7), suggesting that excessive merging begins to erase critical video information. These results validate the effectiveness of the Hierarchical Merge approach in balancing efficiency and representation quality. With a merge ratio of 60%, ViPE achieves a 1.2× speedup in training time and a 30% reduction in FLOPs, demonstrating the practical advantages of this strategy for scalable and efficient video-language modeling.

Effect of Injection Interval. To evaluate the contribution of visual perceptual weights to enhancing video-language comprehension, we experiment with different integration intervals by injecting them into the LLM every 2, 4, or 8 layers. As shown in Table 6, the scale of visual perceptual weight injection has a notable impact on model performance across all benchmarks. Injecting visual perceptual weights every 4 transformer layers yields the best overall results, demonstrating an effective trade-off between incorporating visual context and preserving the language modeling capability of the LLM. When visual information is injected too sparsely, such as every 8 layers, the performance drops significantly, particularly on datasets like EgoSchema and CinePile, which require strong visual grounding in first-person and long-form video contexts. This suggests that insufficient visual integration limits the model's ability to retain and reason over long-term visual information. Conversely, injecting visual perceptual weights too frequently (e.g., every 2 layers) offers no clear advantage and may even introduce unnecessary visual interference, potentially disrupting the model's language reasoning.

Table 6: Effect of visual perceptual weight injection interval.

Interval	EgoSchema	MVBench	VideoMME	MLVU	CinePile
2	**51.4**	50.7	46.7	50.8	**45.5**
4	51.2	**51.3**	**47.2**	**51.3**	45.3
8	50.1	49.6	45.3	49.8	43.5

Table 7: Effect of perceptual weights equipped in the model.q,k,v,o, and m, denote the query, key, output, and mlp, respectively.

Weights Equipped	EgoSchema	MVBench	VideoMME	MLVU	CinePile
qkvom	**51.2**	**51.3**	**47.2**	**51.3**	**45.3**
qkvm	48.1	49.2	44.9	48.7	43.6
qkv	48.3	49.6	45.8	49.7	43.5
qko	47.9	49.4	44.8	49.0	42.9
qk	40.3	43.2	39.7	43.5	38.6

Effect of Perceptual Weights Type. To examine how visual perceptual weights affect video understanding when integrated into different types of model parameters, we conduct a series of ablation experiments across five key weight categories: Query, Key, Value, Output, and Mlp. For each type, we evaluate the model's performance by selectively incorporating visual perception through visual queries. Our results show that integrating visual perceptual weights into all five parameter types achieves the best overall performance across benchmarks. While query and key are vital for computing attention scores, and thus determine whether visual content is attended to. As such, they are retained across all variants to ensure the integrity of the attention mechanism. Among these, the value projection proves to be particularly crucial. In transformer-based models, the value projection is directly responsible for carrying content information to be aggregated during attention. Injecting visual perception into V enables the model to effectively ground language tokens in semantically rich visual content. Conversely, omitting this integration (as seen in the qk or qko variants) results in a significant performance drop.

Effect of Perceptual Rank. To better understand the impact of visual perceptual weights on video-language modeling, we evaluate different weight ranks, which control the capacity of the injected

Table 8: Effect of perceptual weights' rank. The rank controls the degree of visual information compression.

Rank	EgoSchema	MVBench	VideoMME	MLVU	CinePile
16	48.5	48.3	45.6	48.7	43.5
32	50.4	50.6	46.1	50.8	44.1
64	51.2	**51.3**	47.2	**51.3**	**45.3**
128	**51.3**	50.7	**47.6**	50.9	45.1

visual perceptual weights. A higher rank corresponds to lower compression and stronger representational power for visual features. As shown in Table 8, increasing the rank from 16 to 64 consistently improves performance across all five benchmarks, indicating that stronger visual representation leads to better video understanding. However, this improvement plateaus at rank 128, with some tasks (e.g., MVBench and MLVU) even showing marginal drops. This suggests that overly large ranks may introduce redundant or noisy visual features, which can interfere with the LLM's language modeling capabilities. Importantly, rank 64 offers a favorable trade-off between performance and efficiency, achieving near-peak results with moderate computational cost.

5 Conclusion

In this paper, we propose ViPE, a parameter-level alignment paradigm tailored for video-LLMs. Instead of relying on a large number of visual tokens, ViPE efficiently aligns visual and language modalities by generating perceptual weights from compactly merged video features and integrating them directly into the LLM. Specifically, ViPE first employs a visual injection module to aggregate and inject video information into a set of visual queries. Then, a perceptual modulation module generates the perceptual weights ΔW, which are sparsely integrated into the LLM using a LoRA-style approach for improved efficiency. We evaluate ViPE 8 video benchmarks. The results show that ViPE achieves comparable performance to LLaVA-style models, while reducing FLOPs by 85% and achieving 2.8 $\times$ faster inference speed, demonstrating the effectiveness and efficiency of our approach.

6 Acknowledgement

This research is supported by Artificial Intelligence-National Science and Technology Major Project 2023ZD0121200 and the National Natural Science Foundation of China (6243000159 62102416), and the Key Research and Development Program of Jiangsu Province under Grant BE2023016-3.

7 Limitations

1) Dependence on Pretrained Visual Encoders: ViPE relies on pretrained vision encoders to extract frame-level features. Existing visual encoders may not be suitable for performing parameter-level alignment. 2) Different types of weights may require distinct learning strategies: In ViPE, the same learning approach is applied to all weight types (K, Q, V, O, M), which leaves room for further exploration in future work. 3) The video information integration method: The Hierarchical Context Merging proposed in ViPE, still operates at the token level. This approach could be further enhanced by incorporating higher-level semantic merging strategies.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Kirolos Ataallah, Xiaoqian Shen, Eslam Abdelrahman, Essam Sleiman, Deyao Zhu, Jian Ding, and Mohamed Elhoseiny. 2024. Minigpt4-video: Advancing multimodal llms for video understanding with interleaved visual-textual tokens. *arXiv preprint arXiv:2404.03413*.

Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. 2021. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1728–1738.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sihan Chen, Xingjian He, Longteng Guo, Xinxin Zhu, Weining Wang, Jinhui Tang, and Jing Liu. 2023. Valor: Vision-audio-language omni-perception pretraining model and dataset. *arXiv preprint arXiv:2304.08345*.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang,

Ziyang Luo, Deli Zhao, and 1 others. 2024. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint arXiv:2305.06500*.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, and 1 others. 2024. Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *arXiv preprint arXiv:2401.16420*.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, and 1 others. 2024. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. 2024. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13700–13710.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023b. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, and 1 others. 2024a. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yanwei Li, Chengyao Wang, and Jiaya Jia. 2024b. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, pages 323–340. Springer.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Da Li, Pengcheng Lu, Tao Wang, Linmei Hu, Minghui Qiu, and Zhongyu Wei. 2023. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*.

Feipeng Ma, Hongwei Xue, Yizhou Zhou, Guangting Wang, Fengyun Rao, Shilin Yan, Yueyi Zhang, Siying Wu, Mike Zheng Shou, and Xiaoyan Sun. 2024. Visual perception by large language model's weights. *Advances in Neural Information Processing Systems*, 37:28615–28635.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. *Advances in Neural Information Processing Systems*, 36:46212–46244.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR.

Ruchit Rawal, Khalid Saifullah, Miquel Farré, Ronen Basri, David Jacobs, Gowthami Somepalli, and Tom Goldstein. 2024. Cinepile: A long video question answering dataset and benchmark. *arXiv preprint arXiv:2405.08813*.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 5227–5237.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. 2017. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1645–1653.

Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See Kiong Ng, and Jiashi Feng. 2024. Pllava: Parameter-free llava extension from images to videos for video dense captioning. *arXiv preprint arXiv:2404.16994*.

Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. 2019. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9127–9134.

Hang Zhang, Xin Li, and Lidong Bing. 2023a. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*.

Pan Zhang, Xiaoyi Dong, Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Haodong Duan, Songyang Zhang, Shuangrui Ding, and 1 others. 2023b. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*.

Shaolei Zhang, Qingkai Fang, Zhe Yang, and Yang Feng. 2025. Llava-mini: Efficient image and video large multimodal models with one vision token. *arXiv preprint arXiv:2501.03895*.

Y Zhang, B Li, H Liu, Y Lee, L Gui, D Fu, J Feng, Z Liu, and C Li. 2024a. Llava-next: A strong zero-shot video understanding model.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2024b. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. 2024. Mlvu: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*.

A Appendix

A.1 Details of PiVE

Training Details. Tab 9 summarizes the hyperparameters used across different training stages. During the spatial pretraining stage, we adopt a low number of frames, increasing to 32 frames for supervised fine-tuning (SFT) stages.

Table 9: Hyper-parameter Settings for Training Details.

Hyperparameter	Pretraining	Supervised Fine-tuing
Data Scale	4M	3M
Batch Size	256	256
Video Frame	1	32
Hierarchical Merge	✗	✓
Learning Rate (lr)	2e-5	2e-5
LR Schedule	cosine decay	cosine decay decay
LR Warmup Ratio	0.03	0.01
Epoch	1	1
Weight Decay		0
Optimizer		AdamW
DeepSpeed stage		2

We utilize a total of 4M image samples, comprising 1M from Densefusion and 3M from re-annotated CC3M and COCO in pretraining. For SFT, we employ 3M re-annotated samples, including 2M from WebVid (Bain et al., 2021) and 1M from VALOR (Chen et al., 2023). Tab 10 summarizes the hyperparameters of Visual Perceptual Weight. See Table 11 for a detailed breakdown of data sources.

Table 10: Hyper-parameter Settings for Visual Perceptual Weight.

Hyperparamter	
Dimension	512
Skip Layer	4
Rank	64
Alpha	64
Type	kqvom
Perceptual Query	64

Table 11: Data used in pre-training and multimodal supervised fine-tuning stages.

Stage	Dataset	Scale	Source
Pretraining	Image	3M	CC-3M, COCO
	DenseFusion	1M	LAION-2B
	Video	3M	Webvid-2.5M, VALOR-1M
Supervised Fine-tuing	LLaVA-Video	178K	NeXT-QA, ActivityNetQA, PerceptionTest, LLaVA-Hound
	LLaVA-665K	665K	COCO, VG, OCR-VQA, GQA, TextVQA

A.2 Visualization

For the video understanding task, we present the visualization results of ViPE in the figure4. For enhanced visualization, different colors are used to highlight distinct types of information within the descriptions.

In a quiet, rural area, a woman walks alone toward a group of old, run-down houses from her past. The memories overwhelm her, and in a burst of emotion, she throws something to the ground in frustration and anger. Unable to hold back her feelings, she collapses to her knees, consumed by pain and sorrow. A man nearby sees her from a distance. With a calm and concerned expression, he walks toward her. Gently, he kneels beside her, offering silent comfort and support.

A well-dressed man in a white tuxedo stands confidently in what appears to be a luxurious concert hall. He holds a pocket watch in one hand, exuding composure and control, while a crowd watches him closely. He gestures firmly toward the man seated at the piano, asking him to step aside. A photographer rushes in to document the moment as the man takes his seat at the grand piano, his face focused and composed. Then, surrounded by a silent, anticipating audience, he begins to play his fingers dancing across the keys with precision and purpose.

Figure 4: Visualizations of the descriptions for ViPE.