POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Ingegneria Informatica (Computer Engineering)

Master Degree Thesis

Temporal Background Key-Value Reuse for Efficient Vision Transformer Inference

Supervisors
Prof. Alessio Sacco
Prof. Flavio Esposito

Candidate
Augusto Leogrande

Academic Year 2025-2026

Acknowledgements

Vorrei iniziare ringraziando la mia famiglia, perche' spesso, nei ringraziamenti, i familiari finiscono in secondo piano. Il loro supporto incondizionato in questi anni e' cio' che mi ha permesso di raggiungere tutti i miei obiettivi. Nel corso della mia carriera universitaria ho avuto persone fantastiche che chiamare "coinquilini" mi sembra riduttivo. Ringrazio Luigi, Maria Chiara, Alessandro, Giulia e Sara per essere stati i miei primi coinquilini, per loro fortuna o sfortuna, che mi hanno fatto sentire a casa ogni giorno, creando un vero e proprio ambiente familiare. Ringrazio anche la mia seconda famiglia Giulia, Ilaria, Giovanni e Lorenzo, di nuovo per vostra fortuna o sfortuna siete stati i miei coinquilini piu' duraturi. Non so se mai inventeranno la macchina del tempo, ma poter rivivere gli anni passati con voi da capo sarebbe il mio sogno piu' grande. Spiegare a parole cosa siete stati per me e quanto abbiate influenzato la mia vita e' impossibile, mi accontento di avere la fortuna di poter ricordare i momenti passati insieme. Ringrazio Luigi e Giulia per avermi fatto frequentare l'esame piu' difficile della mia vita, cioe' quello dei rapporti umani. In un mondo che spesso privilegia solo l'avanzamento tecnico, avete mostrato quanto gentilezza, carisma ed empatia possano fare la differenza. Avete avuto l'impatto piu' grande su di me, un impatto inaspettato, a volte difficile e pieno di soddisfazioni, che mi ha reso una persona migliore. Ringrazio il mio punto fisso, i ragazzi della mia citta' natale: Stefano, Ciccio, Dambo, Alessia, Luigi, Mirko, Marco, Cesar e Silvia. Avete sempre creduto in me, anche a distanza di mesi o chilometri, siete sempre rimasti il mio punto di riferimento su cui potevo fare affidamento quando avevo bisogno di staccare. Sapere che ci sarete sempre per me a prescindere da dove saro' mi rende una persona fortunata. Ringrazio Luca per essere in primis un grande amico ed anche la persona piu' disponibile che io abbia mai conosciuto. Averti nella mia vita e' uno dei piu' grandi privilegi che possa desiderare, e spero che la distanza non ci separi in futuro. Infine, vorrei ringraziare Simone ed Ayman per il loro costante supporto, per le lunghe chiacchierate e riflessioni, sia professionali che personali. A questo punto dei ringraziamenti solitamente ci si ringrazia da soli, la verita' e' che la mia vita, i miei obiettivi raggiunti e tutto quello che ho non appartengono a me. Sono il frutto del contributo che queste persone straordinarie che ho conosciuto nella mia vita hanno avuto su di me nel corso di questi anni. Spero di riuscire a restituire almeno una parte di cio' che mi avete dato. Vi voglio bene.

Abstract

Vision Transformers (ViTs) are compelling for edge deployment because they can operate on compact token representations instead of full images, along with impressive capabilities in video understanding. However, the combination of high computational cost and large amounts of continuous video data poses a major challenge for real-time deployment on resource-constrained edge devices within the Internet of Things (IoT) environments. Most efficiency methods for ViTs target single images or per-frame optimization. Token reduction techniques reduce intra-frame computation but leave substantial temporal redundancy untouched in videos, where large regions are static across consecutive frames. Recent video/token-reduction works highlight this redundancy by skipping redundant operations while preserving full-resolution tokens and quadratic attention complexity, which limits scalability in long-context and high-resolution settings. While these approaches are sound, they primarily target inference-time efficiency and do not address the underlying representation complexity, which remains a major bottleneck in large-scale video modeling. In this thesis, we fundamentally alter the computational scaling behavior of video transformers by compressing redundant spatial-temporal information into persistent background tokens. This enables long-term reuse, a bounded memory footprint, and superior scalability to large architectures, while avoiding token explosion and heavy memory operations. To this end, we propose a training-free method that provides a more scalable, semantically meaningful, and memory-efficient mechanism for temporal reuse compared to prior redundancy reduction approaches. The key idea is to reuse computations for background regions that remain unchanged while dynamically allocating compute to informative and evolving areas, significantly reducing redundant processing without sacrificing accuracy. We introduce a temporal KV caching mechanism for ViT inference on video that exploits background persistence across frames. In an initial warm-up phase, the algorithm is able to create a compact and meaningful representation of background tokens across the first P frames of inference by applying a spatial-temporal merging technique on both tokens and their respective keys and values. On subsequent frames, a matching algorithm selects background tokens that match the ones contained in the cache in order to effectively reduce the token count and reuse the cached KVs of the matched tokens in the cache. On action recognition benchmarks, our method achieves 2-3x FLOPs reduction with negligible accuracy loss, substantially outperforming prior temporal efficiency methods while maintaining semantic coherence. Since foreground tokens remain untouched, this method provides flexibility to apply any state-of-the-art techniques on them. In fact, when combined with orthogonal token reduction techniques, computational savings amplify to 6-7x FLOPs reduction, demonstrating that KV-level caching and token-level sparsity address complementary redundancy axes. Furthermore, this algorithm proves to be scalable to larger architectures; in fact, results on ViT-L as a backbone show a 6x FLOPs reduction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern computer vision systems increasingly rely on transformer-based architectures, which provide strong performance across a wide range of visual tasks but incur a significant computational cost. This challenge becomes particularly evident in video processing scenarios, where models must process large sequences of frames and reason about spatial and temporal relationships simultaneously. While transformer architectures such as Vision Transformers (ViTs) have achieved remarkable results in image and video understanding, their computational requirements make them difficult to deploy in real-world applications where efficiency is critical. The main reason behind this computational burden lies in the self-attention mechanism that forms the core of transformer architectures. Self-attention enables the model to capture long-range dependencies between visual tokens by computing pairwise interactions between all tokens in a sequence. However, this mechanism scales quadratically with respect to the number of tokens. As a consequence, when high-resolution images or video sequences are processed, the number of tokens quickly becomes very large, resulting in substantial computational and memory requirements. In the context of video understanding, this problem becomes even more pronounced because each frame contributes additional tokens to the sequence, and the model must recompute key and value representations for every token at every frame. At the same time, many practical applications of computer vision increasingly rely on continuous video streams rather than isolated images. Systems such as autonomous driving platforms, smart surveillance cameras, drones, and augmented reality devices must process visual input in real time while operating under strict computational and energy constraints. In these environments, deploying large transformer models directly on edge hardware can be impractical due to limitations in processing power, memory capacity, and energy consumption. As a result, improving the efficiency of transformer-based video models has become an important research problem. A large body of research has therefore focused on developing techniques that reduce the computational complexity of Vision Transformers during inference. One widely explored direction is token reduction, where the number of tokens processed by the transformer is decreased through pruning or merging operations. These methods attempt to identify redundant or less informative tokens and either remove them entirely or combine them into a smaller set of representative tokens. By reducing the token count, the computational cost of the attention mechanism

can be significantly lowered. Most existing token reduction techniques, however, focus primarily on spatial redundancy within individual frames. In these approaches, tokens corresponding to visually similar regions of an image are merged together or pruned based on learned importance scores. While these strategies can effectively reduce the number of tokens processed by the transformer, they generally treat each frame independently and therefore do not fully exploit the temporal structure of video data. In practice, video sequences exhibit a substantial amount of temporal redundancy. Consecutive frames often contain very similar visual content, especially in background regions where little motion occurs. For example, in many action recognition scenarios the background environment remains largely unchanged while only a small subset of objects or people move within the scene. Despite this redundancy, standard video transformer architectures recompute token representations for every frame, including those corresponding to static regions. This leads to a significant amount of redundant computation during inference. Recent works have started to explore strategies that exploit temporal redundancy in order to improve efficiency in video transformers. Some methods attempt to detect which tokens change between frames and selectively recompute only those representations that correspond to moving regions. Others introduce temporal caching mechanisms that store intermediate representations from previous frames. Although these approaches demonstrate promising results, they often suffer from limitations related to scalability, memory overhead, or the additional computation required to identify reusable tokens. In particular, storing raw token representations across frames can quickly lead to large memory footprints, especially when processing long video sequences. Furthermore, several existing methods rely on complex heuristics or additional modules to determine which tokens can be reused, which may introduce additional computational overhead that partially offsets the expected efficiency gains. To address these challenges, this thesis proposes a new method called *Temporal Background Key-Value Reuse (TBKV)*, designed to improve the efficiency of Vision Transformer inference on video data. The key observation behind TBKV is that background regions in videos tend to remain stable across consecutive frames. Instead of recomputing token representations for these regions repeatedly, the proposed approach stores a compact representation of background tokens in a cache and reuses their corresponding key and value representations during subsequent frames. More specifically, TBKV introduces a caching mechanism that stores merged representations of background tokens extracted from an initial set of frames. These cached tokens are then matched with tokens from later frames, allowing the model to reuse previously computed key-value pairs whenever visual similarity is detected. By compressing redundant information and selectively reusing computations, TBKV significantly reduces the number of operations performed during inference while maintaining the ability of the model to process informative foreground tokens normally. An important design goal of TBKV is to ensure that the caching mechanism remains efficient both in terms of computational overhead and memory usage. Rather than storing raw tokens directly, the method applies a token merging strategy that aggregates similar background tokens into a compact representation. This approach not only reduces the size of the cache but also preserves meaningful semantic information that can be reused across frames. As a result, the system is able to exploit temporal redundancy without introducing excessive memory

requirements or additional computational complexity. The effectiveness of the proposed method is evaluated on standard video understanding benchmarks using multiple Vision Transformer backbones. Through extensive experiments, the thesis demonstrates that TBKV can significantly reduce the number of floating-point operations required during inference while maintaining competitive accuracy compared to baseline models. These results suggest that temporal token reuse combined with spatial token merging provides a promising direction for improving the efficiency of transformer-based video architectures.

Thesis Contributions

The main contributions of this thesis can be summarized as follows:

- **Temporal KV caching for Vision Transformers.** We introduce a novel temporal key-value caching mechanism designed for Vision Transformer inference on video data. The method exploits the persistence of background regions across frames to avoid redundant computation of attention representations.
- **Background token merging strategy.** We propose a token merging mechanism that compresses redundant spatial-temporal information into a compact cache representation. By merging similar tokens before storing them, the approach significantly reduces memory footprint while preserving meaningful semantic information.
- **Dynamic matching and reuse mechanism.** We design a matching procedure that dynamically compares tokens from new frames with cached representations and selectively reuses previously computed key-value pairs. This mechanism allows the model to reuse computations for stable background regions while preserving full processing for foreground tokens that contain important motion cues.
- **Comprehensive empirical evaluation.** We evaluate the proposed method on standard video benchmarks using multiple transformer backbones. Experimental results demonstrate that TBKV achieves substantial reductions in computational cost (FLOPs) while maintaining competitive accuracy compared to baseline models and existing token reduction techniques.

Thesis Organization

The remainder of this thesis is organized as follows. Chapter 3 reviews existing research on efficient transformer architectures, token reduction methods, and temporal redundancy exploitation in video models. Chapter 4 introduces the proposed TBKV framework and describes the methodological design of the caching and token matching mechanisms. Chapter 5 presents the experimental setup, datasets, and evaluation metrics used to assess the proposed approach. The experimental results and ablation studies are then discussed in detail, highlighting the efficiency gains achieved by TBKV. Finally, the thesis concludes with a discussion of limitations and potential directions for future work.

Chapter 2

Background

2.1 Early Edge AI and CNNs

One of the earliest applications of computer vision models for edge devices was driven by convolutional neural networks (CNNs), which were also the first to be used for visual recognition tasks such as image classification, object detection, and video understanding. CNNs are neural networks tailored for extracting patterns from images, where each layer consists of filters (kernels) with their own parameters that slide across the image to extract patterns. The first layers extracted common patterns, such as edges, while more complex patterns emerged as the image passed through deeper layers. Pattern extraction was made possible by a mathematical operation called convolution. Thanks to their compact architecture, they were inherently optimized for resource-constrained platforms such as mobile devices. A larger number of works sought to design lightweight structures, since edge devices were still less powerful than today's devices. Pioneering works like MobileNet, ShuffleNet, and SqueezeNet tried to modify either the neural network architecture or the convolution operation itself to reduce quadratic complexity and, consequently, the number of floating-point operations (FLOPs) and parameters. As a result, CNN-based models became the backbone of early edge AI applications, enabling real-time visual perception across a wide range of tasks.

2.2 Early efficiency techniques for neural networks

Now we discuss in more depth the efficiency techniques for computer vision models developed in recent years. Beyond smarter architectural designs, a growing number of model compression and optimization techniques have been developed to deploy increasingly larger models on resource-constrained devices. These techniques aim to reduce computation, memory consumption, and energy usage while preserving model performance on specific tasks. Widely adopted strategies include quantization, which reduces the numerical precision of the model's weights; network pruning, which removes redundant parameters based on some metric and, in some cases, even entire computational blocks; and knowledge distillation, where a lighter student model is trained to mimic the behavior of a teacher model(original model). Despite their success and implementation

in all state-of-the-art models, these efficiency techniques often involve an unfavorable trade-off between accuracy when the number of data and their dimensionality increase, particularly in video understanding scenarios.

2.3 Another approach: dynamic inference efficiency

Previously mentioned approaches focus on developing efficiency techniques before deployment; another widely used approach consists of algorithms that dynamically apply efficiency techniques during inference. These newer approaches focus on computational reduction approaches tailored to the type of input in real-time, therefore adapting computational resources. Earlier works focused on exploiting the spatial and temporal redundancy present in video data by skipping or reusing computation for regions with limited variation. In video processing, change-based inference, frame differencing, and motion-aware gating mechanisms have demonstrated substantial efficiency gains by avoiding redundant computation across temporally adjacent frames. Techniques such as conditional computation, early exiting, spatial gating, and temporal skipping enable CNNs to dynamically allocate resources to salient regions while suppressing background processing. Although highly effective, these methods fundamentally rely on convolutional inductive biases, including locality and translation equivariance, which limit their direct applicability to emerging transformer-based architectures.

2.4 Transformers

Originally introduced for Natural Language Processing (NLP), transformers have rapidly emerged as a dominant paradigm across a wide range of machine learning domains, including computer vision. Their success is largely attributed to the self-attention mechanism, which enables explicit modeling of long-range dependencies and global contextual interactions within input sequences. Compared to the convolutional operations employed by CNNs, which rely on fixed local receptive fields and hierarchical feature aggregation, the self-attention mechanism offered by transformers is able to dynamically establish relationships between all elements in the input, regardless of their spatial or temporal distance. This allows each token to adaptively gather contextual information from the entire input, resulting in flexible receptive fields and enhanced representational capacity. Such properties are particularly advantageous in vision tasks, where semantic understanding often depends on interactions among distant regions and across multiple frames. From an architectural perspective, a transformer is composed of a stack of identical processing blocks, each consisting of two primary components: a multi-head self-attention (MHSA) module and a position-wise feed-forward network (FFN). These components are interleaved with residual connections and layer normalization, which facilitate optimization and stabilize training. The MHSA module captures global contextual relationships, while the FFN independently transforms each token to enhance feature expressiveness. By stacking multiple transformer layers, the model progressively refines token representations, enabling hierarchical abstraction and robust modeling of complex dependencies.

2.4.1 Self-Attention

As discussed in the context of Vision Transformers, images are decomposed into small patches that are embedded into a high-dimensional latent space, forming a sequence of tokens. Self-attention operates on this token sequence by computing pairwise similarities between all tokens, producing an attention map that effectively captures correlations among visual features. This mechanism enables each token to integrate contextual information from the entire image, resulting in representations that are highly adaptive to scene content. Beyond capturing correlations, self-attention dynamically updates each token based on the global context, allowing features to be refined according to their semantic relevance within the image. This leads to richer and more discriminative representations compared to the strictly local feature extraction performed by convolutional operators.

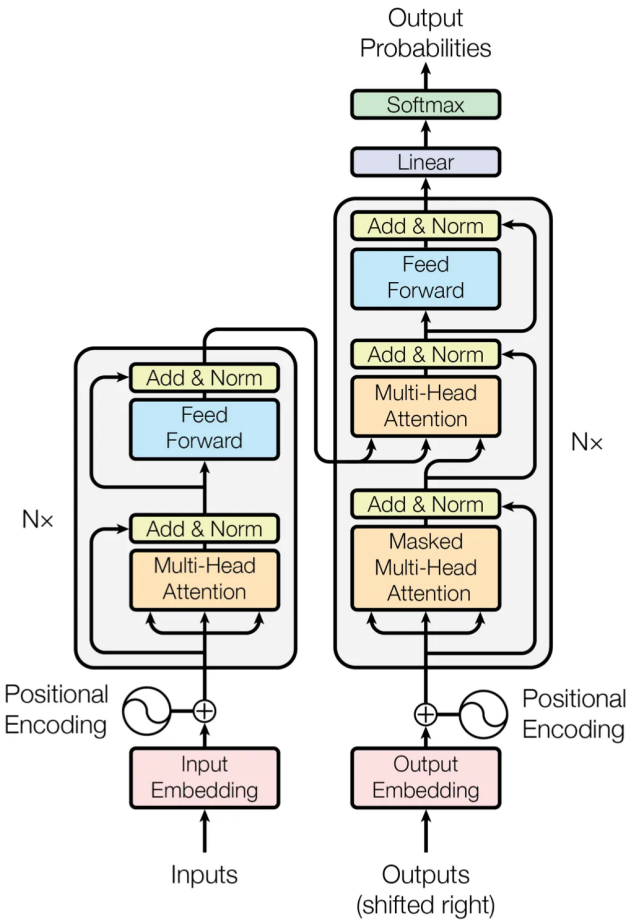

Figure 2.1. Transformer architecture used as the foundation for Vision Transformers. The multi-head self-attention mechanism illustrated here highlights the quadratic computation cost that motivates token reduction and KV reuse strategies, such as the proposed TBKV method. (Reproduced from [41])

This process can be described in three phases:

- **Input Representation:** the input tokens are linearly projected into three distinct sets of vectors: Queries (Q), Keys (K), and Values (V). These representations encode, respectively, the current contextual state, the reference features used for similarity computation, and the information content associated with each token.
- **Scoring:** attention scores measure query-key similarity, typically via scaled dot-product. These quantify each token's relevance to others.
- **Aggregation:** the value vectors are combined using normalized attention scores to produce context-aware token representations. A softmax function is applied to the scores to obtain weights that sum to one, yielding a weighted aggregation of the values (Eq. 2.1).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.1}$$

Here, Q, K, and V denote the query, key, and value matrices, respectively, and d_k represents the dimensionality of the key vectors, which serves as a normalization factor to stabilize gradients.

2.5 Vision Transformers (ViTs)

Transformers have long represented the dominant paradigm in natural language processing; however, their adoption in computer vision was initially limited, as early attempts largely relied on hybrid architectures combining convolutional neural networks (CNNs) with attention-based modules. A major shift occurred with the introduction of the *"An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale"* [12] paper, which demonstrated that a pure transformer-based architecture could effectively operate on visual data. The proposed Vision Transformer (ViT) departs from conventional convolutional designs by employing only the encoder component of the transformer architecture, resulting in a simpler and more modular structure. This design choice reduces architectural complexity and makes the model more suitable for deployment in scenarios with limited computational resources. In ViTs, an image is decomposed into a sequence of fixed-size patches, which can be interpreted as small visual regions. Each patch is then flattened and linearly projected into a latent embedding space, forming a set of tokens analogous to word embeddings in NLP. More formally, let the input image be represented as a tensor $\mathbf{x} \in \mathbf{R}^{H\times W\times C}$, where H and W denote the spatial resolution and C the number of channels. The image is partitioned into a grid of non-overlapping patches of size $P \times P$. This operation produces a sequence of N patches, where

$$N = \frac{H \cdot W}{P^2}. \tag{2.2}$$

Each patch $\mathbf{x}_p^i \in \mathbf{R}^{P\times P\times C}$ is flattened into a vector of dimension P^2C. To map these vectors into a representation suitable for transformer processing, a linear projection is applied using a learnable embedding matrix $\mathbf{E} \in \mathbf{R}^{(P^2C)\times D}$, where D denotes the embedding

dimension. The embedding of the i-th patch can therefore be written as

$$\mathbf{z}_i = \mathbf{x}_p^i \mathbf{E}. \tag{2.3}$$

After this transformation, the image is represented as a sequence of N tokens $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$, each encoding the visual information contained in a local region of the image. This tokenization process enables the transformer to process visual data using the same sequence modeling paradigm originally developed for language. Since transformers do not inherently encode positional information, they are permutation invariant with respect to the order of the input tokens. However, spatial structure is fundamental for visual understanding tasks. To incorporate this information, positional embeddings are added to the token sequence. In addition, ViT introduces a learnable *classification token*, denoted as $\mathbf{z}_{cls}$, which is prepended to the sequence of patch embeddings. This token acts as a global representation that aggregates information from all image patches during the attention process. The input to the transformer encoder is therefore constructed as

$$\mathbf{Z}_0 = [\mathbf{z}_{cls}; \mathbf{z}_1; \mathbf{z}_2; \ldots; \mathbf{z}_N] + \mathbf{E}_{pos}, \tag{2.4}$$

where $\mathbf{E}_{pos} \in \mathbf{R}^{(N+1)\times D}$ denotes a learnable positional embedding matrix. By adding these positional encodings, the model becomes capable of distinguishing tokens based on their spatial location in the original image grid. The resulting sequence of tokens is processed by a stack of transformer encoder layers. Each encoder layer consists of two main components: a Multi-Head Self-Attention (MHSA) module and a position-wise feed-forward network (FFN). Residual connections and layer normalization are applied around both components to stabilize training and improve gradient propagation. Let $\mathbf{Z}_{l-1}$ denote the input token sequence to the l-th encoder layer. The operations within the layer can be expressed as

$$\hat{\mathbf{Z}}_l = \mathbf{Z}_{l-1} + \mathrm{MHSA}(\mathrm{LN}(\mathbf{Z}_{l-1})) \tag{2.5}$$

$$\mathbf{Z}_l = \hat{\mathbf{Z}}_l + \mathrm{FFN}(\mathrm{LN}(\hat{\mathbf{Z}}_l)), \tag{2.6}$$

where $\mathrm{LN}(\cdot)$ denotes layer normalization. The self-attention mechanism represents the core component of the transformer architecture, allowing each token to dynamically aggregate contextual information from all other tokens in the sequence. Inside the attention module, the token sequence is linearly projected into three distinct representations: queries, keys, and values. These are computed as

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z}\mathbf{W}_V, \tag{2.7}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable projection matrices. The attention weights are then obtained through the scaled dot-product operation

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.8}$$

where d_k denotes the dimensionality of the key vectors. This formulation enables each token to attend to all other tokens in the sequence, thereby capturing both short-range

and long-range dependencies across the entire image. The use of multiple attention heads further improves the expressive capacity of the model. In multi-head attention, several independent attention operations are performed in parallel, each operating on a different projection of the input tokens. The outputs of these attention heads are then concatenated and linearly projected to produce the final attention representation. This mechanism allows the model to simultaneously capture different types of relationships among tokens. By stacking multiple transformer layers, the model progressively refines the token representations and learns increasingly abstract visual features. Importantly, unlike convolutional networks, where the receptive field grows gradually with depth, the self-attention mechanism provides each token with a global receptive field from the very first layer. This property enables the model to capture long-range spatial interactions that are difficult to model using purely convolutional operations. After the final encoder layer, the representation associated with the classification token $\mathbf{z}_{cls}$ is extracted and passed through a multilayer perceptron (MLP) head to produce the final prediction. During training, the model learns to aggregate relevant information from all patch tokens into this single vector, which effectively summarizes the entire image. Following this pioneering work, several studies have adopted ViT-style tokenization as a front-end for more complex architectures, leading to the development of multimodal systems such as Vision-Language Models. Moreover, ViTs enable a compelling edge-computing paradigm in which images can be tokenized directly on edge devices, and only the resulting compact representations are transmitted to remote servers. This strategy significantly reduces communication bandwidth while preserving semantic content, making ViTs particularly attractive for distributed and resource-constrained environments.

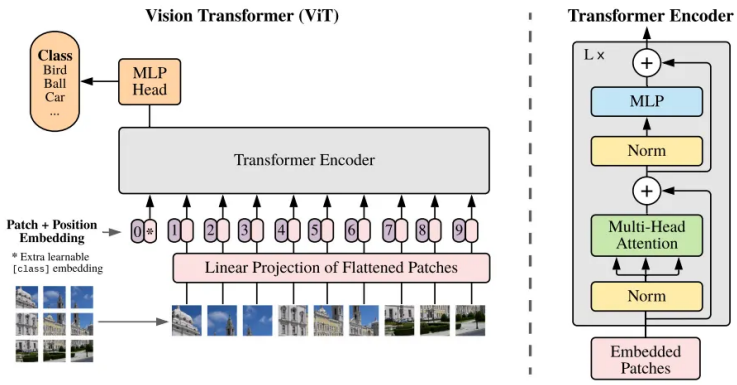

Figure 2.2. Vision Transformer (ViT) architecture showing the patch-based tokenization and transformer encoder stack. This representation illustrates how images are converted into sequences of tokens, emphasizing the high computational cost that video-based extensions face and motivating the development of KV caching strategies to exploit temporal redundancy(Reproduced from [12])

2.6 Video Vision Transformers

While Vision Transformers were originally designed for static image understanding, many real-world perception tasks involve sequential visual data such as videos. Video understanding introduces additional complexity compared to image-based tasks, as models must capture not only spatial relationships within individual frames but also temporal dependencies across frames. This requirement significantly increases both the dimensionality of the input data and the computational cost of processing it. A video sequence can be represented as a tensor

$$\mathbf{X} \in \mathbf{R}^{T \times H \times W \times C},$$

where T denotes the number of frames, H and W represent the spatial resolution of each frame, and C is the number of channels. Extending the Vision Transformer paradigm to video data typically involves converting this spatio-temporal tensor into a sequence of tokens that can be processed by transformer layers. A straightforward approach consists of applying the same patch tokenization procedure used in image-based ViTs independently to each frame. Each frame is divided into non-overlapping patches of size $P \times P$, resulting in

$$N = \frac{H \cdot W}{P^2} \tag{2.9}$$

tokens per frame. For a video containing T frames, the total number of tokens becomes

$$N_{video} = T \cdot N. \tag{2.10}$$

This formulation highlights one of the main challenges in video transformers: the number of tokens grows linearly with the number of frames, while the computational complexity of the self-attention mechanism grows quadratically with respect to the sequence length. Consequently, the cost of attention in video transformers scales approximately as

$$\mathcal{O}((T \cdot N)^2), \tag{2.11}$$

which quickly becomes prohibitive when processing long video sequences or high-resolution inputs. After patch extraction, each patch is flattened and projected into an embedding space in the same way as in Vision Transformers. Let $\mathbf{x}_p^{t,i}$ denote the i-th patch of frame t. The embedding of the patch is obtained through a linear projection

$$\mathbf{z}_{t,i} = \mathbf{x}_p^{t,i}\mathbf{E}, \tag{2.12}$$

where $\mathbf{E} \in \mathbf{R}^{(P^2C) \times D}$ is a learnable embedding matrix and D represents the embedding dimension. The video is therefore represented as a sequence of spatio-temporal tokens

$$\{\mathbf{z}_{1,1}, \mathbf{z}_{1,2}, \ldots, \mathbf{z}_{T,N}\}.$$

To preserve both spatial and temporal structure, positional encodings are extended to incorporate temporal information. In addition to spatial positions within each frame, tokens must encode their location in the temporal sequence. A common formulation introduces separate spatial and temporal positional embeddings:

$$\mathbf{z}_{t,i}^0 = \mathbf{z}_{t,i} + \mathbf{e}_i^{space} + \mathbf{e}_t^{time}, \tag{2.13}$$

where $\mathbf{e}_i^{space}$ represents the spatial positional embedding associated with patch i, and $\mathbf{e}_t^{time}$ denotes the temporal embedding corresponding to frame t. This formulation allows the model to distinguish tokens not only based on their spatial location but also according to their position in the temporal sequence. Once the token sequence is constructed, it is processed by a stack of transformer encoder layers similar to those used in Vision Transformers. Each layer consists of a Multi-Head Self-Attention (MHSA) module followed by a feed-forward network (FFN). Given a sequence of tokens $\mathbf{Z} \in \mathbf{R}^{N_{video} \times D}$, the self-attention mechanism computes query, key, and value representations:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z}\mathbf{W}_V, \tag{2.14}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable projection matrices. The attention operation then aggregates contextual information from all tokens in the sequence according to

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \tag{2.15}$$

Through this mechanism, each token can interact with all other tokens across both spatial and temporal dimensions. This enables the model to capture long-range dependencies, such as relationships between objects appearing in different frames or interactions that unfold over time. However, applying full self-attention over the entire spatio-temporal token sequence introduces significant computational challenges. In particular, the attention matrix has size $N_{video} \times N_{video}$, which can become extremely large when processing videos with many frames. As a result, several architectural variants have been proposed to reduce the computational cost of video transformers. One common strategy consists of factorizing attention into separate spatial and temporal components. In this formulation, spatial attention is applied within individual frames, while temporal attention is applied across tokens at corresponding spatial locations in different frames. This factorization reduces the computational complexity while still enabling the model to capture both spatial and temporal relationships. Another important property of video data is the presence of strong temporal redundancy. Consecutive frames often share highly similar visual content, particularly in static regions of the scene. For example, background elements such as buildings, landscapes, or stationary objects may remain nearly unchanged across several frames. Despite this redundancy, standard video transformers recompute token representations and attention interactions independently for each frame. This observation has motivated recent research into more efficient processing strategies that exploit temporal coherence in video streams. By identifying tokens that exhibit minimal variation across frames, it becomes possible to reuse previously computed representations instead of recomputing them from scratch. Such approaches aim to reduce redundant computation while preserving the ability of the model to capture meaningful temporal dynamics. In particular, the reuse of intermediate representations across frames can significantly reduce the computational burden associated with the attention mechanism. Since key and value tensors encode contextual information used during attention computation, storing and reusing these representations across frames provides a natural mechanism for exploiting temporal redundancy. This idea forms the basis for several recent approaches that extend

key-value caching techniques, originally developed for language models, to video transformer architectures. By leveraging the temporal structure of video data, these methods aim to maintain high representational capacity while reducing unnecessary computation. This principle is especially relevant for edge computing scenarios and real-time video analysis, where computational resources and memory bandwidth are often limited.

2.7 KV Cache

The computational cost of attention mechanisms in transformers is widely recognized as one of the main bottlenecks in large-scale models. As a result, a significant portion of recent research has focused on designing more efficient algorithms and architectural strategies to mitigate this overhead. This issue becomes particularly critical in large language models (LLMs), where extremely long input sequences and extended contextual dependencies are common. Since the complexity of self-attention grows quadratically with respect to the number of processed tokens, even moderate increases in sequence length can lead to substantial computational and memory burdens. To understand the motivation behind caching mechanisms, it is useful to briefly revisit the formulation of the self-attention operation. Given an input token sequence represented by a matrix $\mathbf{Z} \in \mathbf{R}^{N \times D}$, where N denotes the sequence length and D the embedding dimension, the transformer computes three projected representations known as queries, keys, and values:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{Z}\mathbf{W}_V, \tag{2.16}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable projection matrices. The attention output is then obtained through the scaled dot-product attention mechanism

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.17}$$

where d_k represents the dimensionality of the key vectors. This formulation requires computing the similarity between every pair of tokens in the sequence, producing an attention matrix of size $N \times N$. Consequently, the computational complexity of self-attention grows proportionally to $\mathcal{O}(N^2)$, while the memory requirements also increase with the square of the sequence length. In autoregressive transformer models, such as those used in text generation, tokens are generated sequentially. At each decoding step, the model predicts a new token based on the sequence of previously generated tokens. Formally, if the current sequence length at time step t is t, the model must compute attention over all tokens $\{1, \ldots, t\}$. Without any optimization, this implies that the key and value representations for all previous tokens must be recomputed at every step. As the sequence grows, this repeated computation leads to significant inefficiencies. One influential line of work in this area investigates the reuse of intermediate attention computations across successive decoding steps. In autoregressive generation, each newly predicted token must attend to all previously generated tokens, leading to repeated recomputation of attention scores and projections. To address this inefficiency, [22] proposed caching previously computed key and value tensors and reusing them during subsequent decoding steps. Their experiments demonstrated that this strategy can dramatically reduce computation while

introducing only negligible performance degradation. This technique, commonly referred to as *key-value caching* (KV cache), consists of storing the key and value tensors produced at each decoding step and reusing them when processing new tokens. Instead of recomputing keys and values for the entire sequence, the model only computes the query representation for the new token while retrieving the cached keys and values corresponding to previous tokens. Let $\mathbf{K}_{cache}$ and $\mathbf{V}_{cache}$ denote the stored key and value tensors accumulated during previous steps. When a new token is processed, its query vector $\mathbf{q}_t$ interacts with the cached tensors according to

$$\text{Attention}(\mathbf{q}_t, \mathbf{K}_{cache}, \mathbf{V}_{cache}) = \text{softmax}\left(\frac{\mathbf{q}_t \mathbf{K}_{cache}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}_{cache}. \tag{2.18}$$

By reusing previously computed representations, KV caching effectively avoids redundant projection operations and significantly reduces the computational overhead of autoregressive inference. In practice, this mechanism transforms the attention computation from repeatedly processing the entire token sequence to incrementally extending the cached representations as new tokens are generated. This strategy has become a standard component in modern transformer inference pipelines, particularly in large language models where long-context generation is common. The main trade-off introduced by KV caching concerns memory usage, since the cached tensors must be stored for the entire sequence. Nevertheless, in most practical scenarios, the reduction in computation outweighs the additional memory requirements, resulting in substantially faster inference. More recently, similar ideas have been explored in the context of vision transformers and video understanding tasks. Unlike static images, video data consists of a sequence of frames that often exhibit significant temporal redundancy. Consecutive frames frequently contain highly similar visual content, especially in regions corresponding to static background elements. As a result, recomputing attention representations for every frame can lead to unnecessary computation. To exploit this redundancy, recent works have investigated extending KV caching mechanisms to vision transformers operating on video sequences. In this setting, the keys and values extracted from previous frames can be stored and reused when processing subsequent frames. By maintaining a cache of previously computed attention representations, the model can avoid recomputing features for regions that remain largely unchanged over time. More formally, let $\mathbf{K}^{(t)}$ and $\mathbf{V}^{(t)}$ denote the key and value tensors extracted from frame t. A temporal cache can be constructed by accumulating these tensors across multiple frames,

$$\mathbf{K}_{cache} = [\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \ldots, \mathbf{K}^{(t-1)}], \tag{2.19}$$

$$\mathbf{V}_{cache} = [\mathbf{V}^{(1)}, \mathbf{V}^{(2)}, \ldots, \mathbf{V}^{(t-1)}]. \tag{2.20}$$

When processing frame t, the query representations extracted from the current frame interact with this cached information, enabling the model to reuse contextual representations from earlier frames. This mechanism allows the transformer to leverage temporal coherence in video streams while reducing redundant attention computations. Consequently, KV caching provides a natural framework for improving the efficiency of transformer-based architectures operating on sequential data. By reusing previously computed key and value representations, these methods reduce redundant computation while

preserving the expressive power of the attention mechanism. This idea is particularly relevant in video understanding scenarios, where temporal redundancy is pervasive and can be exploited to significantly reduce computational cost.

2.8 AI on Edge Devices and Split Execution

The rapid growth of artificial intelligence applications has led to an increasing demand for deploying machine learning models directly on edge devices. Edge devices include smartphones, embedded systems, Internet-of-Things (IoT) sensors, and other resource-constrained platforms that operate close to the data source. Performing inference on edge hardware enables several important advantages, including reduced latency, improved privacy, and lower communication costs compared to fully cloud-based solutions. In traditional cloud-centric AI pipelines, raw input data collected by sensors is transmitted to remote servers where models perform inference. Although this paradigm provides access to large computational resources, it introduces significant communication overhead and latency. For applications involving real-time perception, such as video analytics, autonomous systems, and augmented reality, transmitting high-dimensional data such as images or video streams to the cloud may become impractical. To address these limitations, modern AI systems increasingly adopt edge-centric architectures in which part of the inference pipeline is executed directly on the device. Let $\mathbf{x}$ denote the input data collected by an edge sensor, such as an image or a video frame. A typical neural network inference process can be viewed as a sequence of transformations

$$\mathbf{y} = f_L(f_{L-1}(\dots f_1(\mathbf{x}))), \tag{2.21}$$

where each function $f_l(\cdot)$ represents the operation performed by the l-th layer of the neural network. Executing the entire pipeline locally requires sufficient computational resources and memory to process all layers of the model. However, many edge devices have limited processing capability, restricted memory, and strict energy constraints. As a result, running large deep learning models entirely on-device may be infeasible. One widely adopted strategy to address this challenge is *split execution*, also referred to as *split computing*. In this paradigm, the neural network is partitioned into two components: an initial set of layers executed on the edge device and the remaining layers executed on a remote server. Formally, the model can be divided at layer k into two sub-functions:

$$\mathbf{z} = f_k(f_{k-1}(\dots f_1(\mathbf{x}))) \tag{2.22}$$

$$\mathbf{y} = f_L(f_{L-1}(\dots f_{k+1}(\mathbf{z}))). \tag{2.23}$$

The intermediate representation $\mathbf{z}$ is transmitted from the edge device to the cloud, where the remaining layers complete the inference process. This approach reduces the amount of data transmitted compared to sending the original input, since $\mathbf{z}$ often has a lower dimensionality than the raw data. In addition, the computational load is distributed between the edge device and the server, allowing complex models to be used even when local resources are limited. Split execution is particularly beneficial for applications involving high-dimensional inputs such as video streams. Raw video data typically consists

of a sequence of frames with high spatial resolution, leading to large data volumes that must be transmitted if processing occurs entirely in the cloud. By performing early feature extraction directly on the device, the system can convert raw frames into compact feature representations before transmission. This paradigm can be further extended to transformer-based vision models. In Vision Transformers, the tokenization stage converts image patches into compact embeddings that capture the essential visual content of each region. These tokens can be interpreted as a compressed representation of the input image. Consequently, edge devices can perform the patch extraction and embedding stages locally, while the computationally intensive transformer layers are executed remotely. Let $\mathbf{x}$ denote an image and $\mathbf{Z}_0$ the sequence of tokens generated by the patch embedding stage. The tokenization process can be expressed as

$$\mathbf{Z}_0 = \text{Tokenize}(\mathbf{x}), \tag{2.24}$$

where $\mathbf{Z}_0 \in \mathbf{R}^{N \times D}$ contains N tokens of dimension D. Instead of transmitting the original image $\mathbf{x}$, the system can transmit the token sequence $\mathbf{Z}_0$, which represents a more compact intermediate representation. This design enables flexible deployment strategies where different portions of the model are executed at different points in the computational pipeline. For instance, lightweight preprocessing and tokenization can be performed on edge devices, while deeper transformer layers responsible for complex reasoning operate on more powerful servers. Another important consideration in edge AI systems is energy efficiency. Edge devices are often battery-powered and must operate under strict energy budgets. Reducing the number of operations performed during inference can significantly extend device lifetime and improve system scalability. Consequently, a large body of research focuses on designing efficient inference mechanisms that minimize redundant computation while preserving model accuracy. These considerations become even more critical when dealing with video data. Processing multiple frames sequentially dramatically increases the number of tokens and attention computations required by transformer models. As a result, techniques that reduce redundant operations across frames or reuse previously computed representations can provide substantial efficiency improvements. In this context, mechanisms that exploit temporal redundancy in video sequences are particularly attractive for edge deployment scenarios. By identifying portions of the visual scene that remain stable across frames, it becomes possible to reuse previously computed features rather than recomputing them from scratch. Such strategies reduce both computational cost and memory bandwidth requirements, making them well suited for real-time video processing on resource-constrained devices.

Chapter 3

Related Work

Transformer-based architectures have become the dominant paradigm for many computer vision tasks. However, their strong performance often comes at the cost of significant computational complexity, particularly when processing high-resolution images or long video sequences. As a result, a large body of research has focused on improving the efficiency of Vision Transformers (ViTs), especially in scenarios where models must operate under strict computational or memory constraints such as edge devices or real-time video processing systems. Existing approaches for improving the efficiency of Vision Transformers can be broadly divided into three main categories: (i) efficient transformer architectures that reduce the computational complexity of attention, (ii) spatial token reduction methods that decrease the number of tokens processed within a frame, and (iii) temporal redundancy reduction techniques designed specifically for video data. In the following sections we review these directions and discuss how they relate to the approach proposed in this thesis.

3.1 Efficient Transformer Architectures

A first line of research focuses on designing more efficient transformer architectures by modifying the attention mechanism itself. The quadratic complexity of self-attention with respect to the number of tokens represents one of the main computational bottlenecks of Vision Transformers. Several works attempt to mitigate this limitation through sparse attention mechanisms or low-rank approximations that reduce the effective computational cost of the attention operation. Examples include sparse attention patterns and routing mechanisms [7,17,35], kernel-based approximations of self-attention such as Performer [8], and linearized attention formulations such as Linformer [42] and the efficient transformer model proposed by Katharopoulos et al. [18]. Other works explore alternative attention formulations and hybrid approaches that combine different attention approximations to balance efficiency and expressiveness [15, 20, 28, 34, 39, 51]. Another direction aims to design efficient Vision Transformer architectures directly at the model level. These methods introduce architectural modifications such as hierarchical feature representations, localized attention windows, or convolutional inductive biases in order to reduce computational complexity while maintaining strong performance. Notable examples include LeViT [14],

MobileViT [30], Convolutional Vision Transformers (CvT) [45], Swin Transformer [26], Multiscale Vision Transformers [13], Pyramid Vision Transformers (PVT) [43], CSWin Transformer [11], and Twins Transformer [9]. These architectures typically rely on hierarchical processing stages or localized attention windows to reduce the number of token interactions. In the context of video understanding, several transformer-based architectures have also been proposed specifically for spatio-temporal modeling. Models such as TimeSformer [2], ViViT [1], VideoMAE [40], and MemViT [46] extend Vision Transformers to process video sequences by incorporating temporal attention mechanisms, memory modules, or specialized spatio-temporal tokenization strategies. Although these architectural approaches successfully reduce computational complexity at the model level, they typically process each frame independently during inference and therefore do not explicitly exploit the large amount of redundancy present in video streams. In particular, many tokens corresponding to background regions remain nearly unchanged across consecutive frames, leading to redundant computation that could potentially be avoided.

3.2 Spatial Token Reduction

Another widely explored direction focuses on reducing the number of tokens processed by Vision Transformers. Since the computational cost of attention grows with the number of tokens, reducing token count can significantly improve efficiency. Token pruning methods remove tokens that are deemed less informative for the final prediction. DynamicViT [33], for instance, introduces a learnable importance score that allows the network to progressively discard unimportant tokens during the forward pass. Similarly, A-ViT [48] formulates token pruning as a reinforcement learning problem, enabling the model to dynamically determine how many tokens should be retained at each transformer layer. An alternative strategy consists of merging similar tokens instead of removing them entirely. Token Merging (ToMe) [3] proposes a simple but effective technique that groups visually similar tokens through bipartite matching and merges them into a smaller set of representative tokens. This approach reduces token count while preserving the most relevant information for downstream tasks. More recent works combine pruning and merging strategies in hybrid frameworks. For example, ToFu [25] and Prune & Merge [49] integrate importance-based token selection with similarity-based token fusion, enabling models to both discard redundant tokens and aggregate related visual features. These hybrid methods often provide better efficiency-accuracy trade-offs compared to approaches relying on a single token reduction mechanism. While these techniques have proven effective for image-based Vision Transformers, their benefits are limited when applied to video data. The number of tokens grows proportionally with the number of frames, which can quickly lead to extremely large token sequences. Consequently, several recent works extend token reduction strategies to the spatio-temporal domain by leveraging redundancy across frames [6,10,37]. Approaches such as VidToMe [23], AdaFocusV2 [47], and TOME-V [24] apply merging or pruning strategies jointly across spatial and temporal dimensions, while TokenLearner [36] learns to select a compact subset of representative tokens that summarize the visual content of each frame. Despite their effectiveness, these approaches primarily adapt spatial token reduction techniques to video inputs and still recompute

most intermediate representations at every frame. As a result, they do not explicitly exploit the possibility of reusing previously computed transformer features across time.

3.3 Temporal Redundancy Exploitation in Video Transformers

A different line of research focuses on reducing redundant computation across frames by exploiting temporal redundancy. Early work in this direction was originally developed for convolutional neural networks, where methods such as CBinfer [5], DeltaCNN [31], and related techniques attempt to reuse intermediate activations for pixels that remain unchanged between frames. Other approaches leverage attention mechanisms or feature skipping strategies to avoid recomputing stable regions [16, 44]. However, directly applying these techniques to Vision Transformers is challenging due to fundamental architectural differences. Unlike convolutional networks, transformer tokens have global receptive fields and lack the translation equivariance properties that make spatial reuse strategies easier to implement. More recent works therefore explore temporal redundancy reduction directly within transformer architectures. STGT [50] introduces a gating mechanism that classifies tokens as either foreground or background based on their temporal variation. Tokens that remain stable across frames can reuse previously computed representations, while dynamic tokens are recomputed. Eventful Transformers [32] extend this idea by identifying and skipping redundant computations not only in linear projection layers but also within the self-attention mechanism itself. This allows the model to avoid recomputing both token projections and attention interactions when visual content remains unchanged. EP-ViT [29] proposes an alternative strategy that leverages motion information already available in the video codec. Instead of explicitly computing token differences to detect changes, the method uses motion vectors extracted during video decoding to identify regions that have moved between frames, thereby avoiding additional computation for change detection. CST-ViT [21] further expands this idea by introducing a cascaded redundancy reduction pipeline. The method sequentially applies three different gating mechanisms to exploit multiple forms of redundancy. A Direct Temporal Gate (DTG) first identifies tokens that differ most from a reference frame and skips computation for static background tokens. An Offset Temporal Gate (OTG) then attempts to match changed tokens with nearby tokens in the previous frame, allowing moving objects to reuse cached representations. Finally, a Spatial Gate (SG) identifies spatially redundant tokens within the same frame and shares computation across visually similar regions. Although these methods represent important progress toward efficient video transformers, they often rely on storing raw token representations and their associated key-value projections in memory in order to enable temporal reuse. As video sequences grow longer, this strategy leads to increasingly large caches that scale with the number of frames and tokens, making memory management challenging in practical deployments. In addition, raw token representations are often highly redundant, particularly for static background regions that remain visually similar across time. Storing these uncompressed representations can therefore lead to inefficient memory usage and unstable cache behavior, especially when processing long videos. Furthermore, many existing approaches

require additional gating mechanisms, motion estimation modules, or complex matching strategies that increase system complexity. While these techniques successfully reduce some redundant computation, they do not explicitly address how cached representations themselves could be made more compact and efficient.

3.4 Research Gap

From the analysis above, it becomes clear that existing efficiency strategies for Vision Transformers typically focus on either architectural simplifications, spatial token reduction, or temporal redundancy detection. However, relatively few works explicitly address the problem of efficiently reusing transformer key-value representations across frames while maintaining a controlled and scalable memory footprint. In particular, many temporal redundancy methods rely on storing raw token representations and their associated key-value projections in cache in order to reuse computations across frames. While this strategy enables temporal reuse, it can lead to large and inefficient caches because highly redundant background tokens are stored in their original form. As video length increases, these raw caches may become memory-intensive and difficult to manage, limiting the scalability of such approaches. The method proposed in this thesis addresses these limitations by introducing a Temporal Background Key-Value Reuse (TBKV) mechanism that combines spatial token merging with temporal key-value caching. Instead of storing raw background token representations, the proposed approach compresses redundant tokens through a merging strategy before inserting them into the cache. This design reduces redundancy within the cached representations themselves, allowing the system to maintain a compact and more stable memory structure while still enabling effective key-value reuse during inference. As a result, the proposed method achieves significant computational savings while controlling both compute cost and cache memory growth.

Chapter 4

Methodology

In the previous chapters, we discussed the limitations of standard video transformer architectures when processing long video sequences, particularly the repeated computation of key and value representations for tokens that exhibit strong temporal redundancy. In this chapter, we introduce the proposed Temporal Background Key-Value (TBKV) caching mechanism, a method designed to exploit temporal consistency in video data in order to reduce the computational cost of transformer inference. The central idea is to identify redundant background tokens across frames and reuse their corresponding key-value representations instead of recomputing them. The proposed framework operates in two stages. First, during a warm-up phase, tokens from the initial frames are merged to construct a compact cache of representative background tokens. Second, during inference, tokens from incoming frames are matched against this cache in order to reuse previously computed keys and values whenever strong similarity is detected. This design enables the model to significantly reduce both the number of token projections and the attention computation while maintaining compatibility with existing Vision Transformer architectures. In the following sections, we first formalize the problem, then describe the pipeline and the algorithms used for token merging and matching.

4.1 Problem Definition

The problem we want to address is how to develop an efficient KV caching mechanism that doesn't scale too much in complexity as the streaming data increases, and that can be used to not only reuse tokens but also reduce them. The procedure has to be simple and efficient so that it can be applied on edge devices without worrying about loss in performance or heavy resource consumption as the model runs for a long period of time. Moreover, the algorithm has to scale with different backbone sizes, enabling manufacturers to be flexible depending on edge devices' capabilities in terms of resources. Finally, we also want to exploit the characteristic of only working with background tokens in order to be able to apply safely any token reduction technique of our choice to foreground tokens.

4.2 Pipeline Overview

The TBKV pipeline is designed to exploit the temporal redundancy of background regions in video sequences. Instead of recomputing token representations for each frame independently, the algorithm constructs a compact cache of representative background tokens and reuses them across subsequent frames. During the initial warm-up phase, the algorithm uses tokens of the first P frames of the data in order to populate a KV cache with a compact and more meaningful representation of tokens across the selected frames. The KV caching mechanism leverages a mathematical function called bipartite soft matching in order to merge similar background tokens of the P frames together with their corresponding keys and values. Experiments showed that background tokens tend to be less informative for the computer vision task and therefore present higher redundancy. On inference background tokens, a matching procedure is applied with the aim of substituting matched tokens with cached tokens and avoiding the computation of keys and values on those ones. Results show that due to the merging algorithm's effectiveness, a single token can match a large number of tokens on the inference frame, leading to a substantial token reduction. The merging mechanism presents an r_{merge} parameter that controls how aggressively we merge tokens on warp-up frames, leading to a controlled size of the cache together with memory operations. At the same time, the matching algorithm presents a specular r_{match} parameter that controls how many background tokens we want to match. This is particularly crucial for tasks where background is informative, for example, for Vision LLM. Periodically, the caching mechanism begins again in order to avoid a heavy change in the scenario; however, if before the *refreshing* phase there's a change in the context, the algorithm will still perform since only background tokens will be affected, which have, in general, a controlled impact on model performance.

4.3 Design Assumptions

The proposed method relies on two key observations about video data. First, consecutive frames in natural videos exhibit a high degree of temporal redundancy, meaning that large portions of the visual content remain unchanged across frames. In particular, background regions typically evolve slowly compared to foreground objects, leading to highly similar token representations over time. Second, transformer-based vision models compute keys and values independently for each token, resulting in repeated computation for tokens that represent nearly identical visual content across frames. Motivated by these observations, TBKV focuses on identifying and compressing redundant background tokens during a warm-up phase and subsequently reusing their corresponding key and value representations during inference. By doing so, the method avoids recomputation for tokens that are likely to produce similar attention contributions, while preserving the full representational capacity for foreground tokens that are more informative for the downstream task.

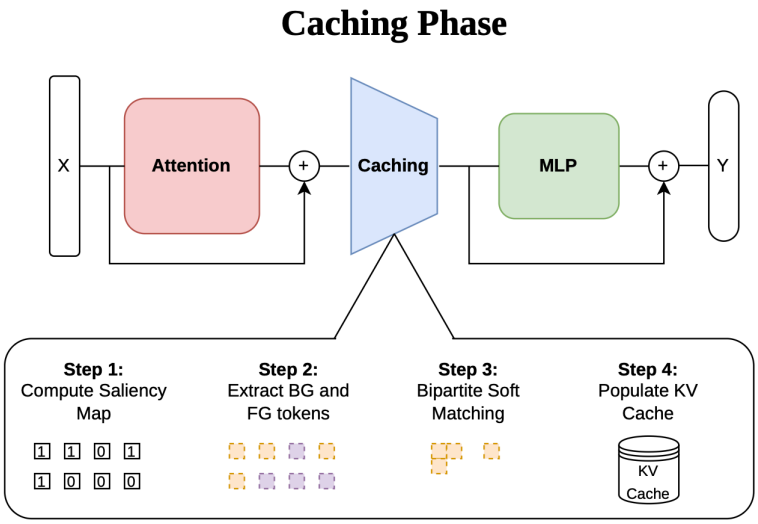

Figure 4.1. TBKV Caching Phase: Building the Background KV Cache from Initial Frames

4.4 Merging Algorithm

Let $\mathbf{X} \in \mathbf{R}^{B \times T \times N \times D}$ denote the input token tensor, where: - B is the batch size (number of videos processed simultaneously), - T is the number of frames per video, - N is the number of tokens extracted from each frame after patch embedding, - D is the dimensionality of the token embedding. Therefore, each frame is represented as a set of N tokens in a D-dimensional embedding space, and a video sample corresponds to a sequence of T such token sets. We first perform background extraction to decompose $\mathbf{X}$ into background tokens $\mathbf{X}_{\text{BG}}$ and foreground tokens $\mathbf{X}_{\text{FG}}$. Subsequently, a bipartite soft matching algorithm is applied to $\mathbf{X}_{\text{BG}}$ in order to obtain the merged background representation $\mathbf{X}_{\text{BG}}^{\text{merged}}$. Local merge ratio is one of our parameters that controls the fraction of tokens that have to be merged in the bipartite soft matching function. It is worth noting that due to the function construction, the maximum ratio is 0.5.

4.4.1 Background Extraction

The background extraction procedure leverages a saliency metric to assign a score to each token, quantifying the likelihood of the token belonging to the foreground or background. Let S_i denote the sharpness score of token i. Tokens satisfying $S_i = 0$ are classified as background, while tokens with $S_i \neq 0$ are classified as foreground. Since GPU efficiency benefits from fixed-size tensors, we compute the maximum number of background and

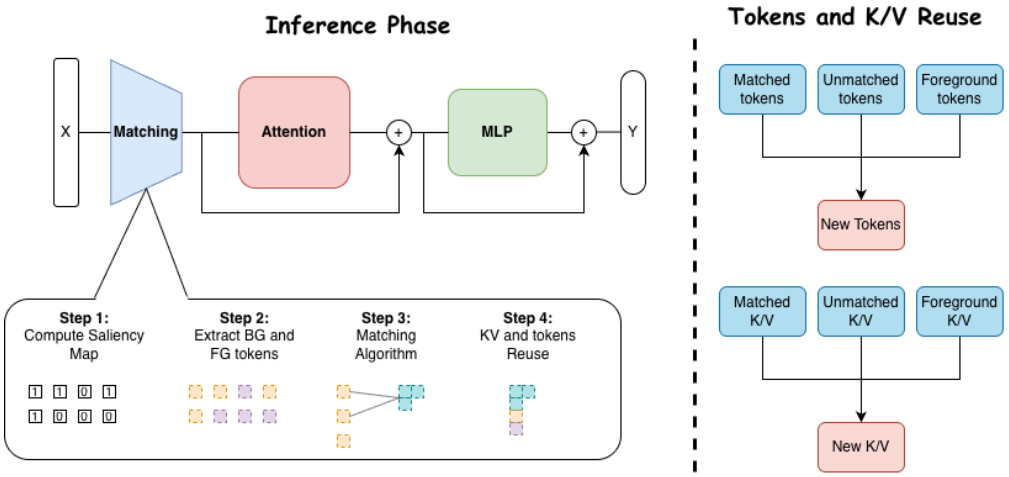

Figure 4.2. TBKV Inference Phase: Matching Incoming Background Tokens to Cached Representations

foreground tokens across all frames and batch elements, which enables padding to a uniform size and avoids explicit looping. These maxima are defined as:

$$N_{\text{BG}}^{\max} = \max_{b,t} \sum_{i=1}^{N} \mathbf{I}(S_i^{(b,t)} = 0), \tag{4.1}$$

$$N_{\text{FG}}^{\max} = \max_{b,t} \sum_{i=1}^{N} \mathbf{I}(S_i^{(b,t)} \neq 0), \tag{4.2}$$

where $\mathbf{I}(\cdot)$ denotes the indicator function. Using a top-k selection strategy with $k = N_{\text{BG}}^{\max}$ and $k = N_{\text{FG}}^{\max}$, we extract the background and foreground token indices, respectively. Employing a Top-K strategy instead of simply defining background tokens as the ones with saliency equal to 0 is an implementation choice that allows some low score foreground tokens to be considered background. This decision yield not only to pure GPU efficiency but also to a better robustness of the background extraction algorithms to errors such as the classification of an important foreground token as background that would result in a significant accuracy loss for that sample. Finally, we apply a gather operation to obtain the tensors $\mathbf{X}_{\text{BG}}$ and $\mathbf{X}_{\text{FG}}$.

4.4.2 Saliency Metric

To compute token saliency, we leverage the self-attention mechanism. For each token i, the attention distribution over all tokens is given by:

$$\mathbf{A}_i = \text{softmax}\left(\frac{\mathbf{q}_i \mathbf{K}^\top}{\sqrt{D}}\right), \tag{4.3}$$

where $\mathbf{q}_i$ is the query vector of token i, and $\mathbf{K}$ denotes the matrix of key vectors. Background tokens tend to exhibit uniformly distributed attention, whereas foreground tokens demonstrate sharper attention distributions concentrated on a few relevant tokens. Motivated by this observation, we exploit a sharpness function that follows the formulation in [27], based on the negative entropy of the attention distribution. Specifically, the entropy of token i is computed as:

$$H_i = -\sum_{j=1}^{N} A_{ij} \log A_{ij}, \tag{4.4}$$

where A_{ij} denotes the attention weight from token i to token j. We then normalize the entropy values across all tokens to obtain the sharpness score:

$$S_i = \frac{H_i - \min(\mathbf{H})}{\max(\mathbf{H}) - \min(\mathbf{H})}, \tag{4.5}$$

where $\mathbf{H} = [H_1, \ldots, H_N]$. Since informative foreground tokens typically exhibit lower entropy, their corresponding sharpness scores S_i tend to be higher, facilitating an effective separation between foreground and background tokens.

4.4.3 Bipartite Soft Matching

Bipartite soft matching, introduced by [4], is the core operation underlying token merging in Vision Transformers. Let $X = \{x_1, x_2, \ldots, x_N\}$ be the set of token embeddings with $x_i \in \mathbf{R}^d$. The goal is to reduce the number of tokens N to a smaller set $\hat{X} \in \mathbf{R}^{\hat{N} \times d}$ while preserving the majority of information. First, the token set X is partitioned into two disjoint subsets:

$$\mathcal{A} = \{a_1, \ldots, a_{N/2}\}, \quad \mathcal{B} = \{b_1, \ldots, b_{N/2}\}, \quad \mathcal{A} \cap \mathcal{B} = \emptyset. \tag{4.6}$$

A similarity matrix $S \in \mathbf{R}^{|\mathcal{A}| \times |\mathcal{B}|}$ is then computed between the two subsets using cosine similarity:

$$S_{ij} = \frac{\langle a_i, b_j \rangle}{\|a_i\|_2 \, \|b_j\|_2}, \quad i = 1, \ldots, |\mathcal{A}|, \; j = 1, \ldots, |\mathcal{B}|. \tag{4.7}$$

From S, a set of candidate edges $\mathcal{E}$ is selected by taking the top-r similarity scores:

$$\mathcal{E} = \{(i, j) \mid S_{ij} \in \text{Top-}r(S)\}, \tag{4.8}$$

where each edge (i, j) represents a pair of tokens to be merged. The merged embedding for each selected pair is computed as a convex combination:

$$\hat{x}_{ij} = \alpha a_i + (1 - \alpha) b_j, \quad \alpha \in [0,1], \tag{4.9}$$

where α can be fixed (commonly 0.5) or learned, allowing adaptive blending. Tokens not selected in $\mathcal{E}$ remain unchanged:

$$\hat{x}_k = x_k \quad \forall x_k \notin \{a_i, b_j \mid (i, j) \in \mathcal{E}\}. \tag{4.10}$$

Finally, the reduced token set is obtained by concatenating merged and untouched tokens:

$$\hat{X} = \left\{\hat{x}_{ij} \mid (i, j) \in \mathcal{E}\right\} \cup \left\{\hat{x}_k \mid x_k \notin \{a_i, b_j \mid (i, j) \in \mathcal{E}\}\right\}. \tag{4.11}$$

This formulation is highly parallelizable. Pairwise similarities can be computed as a single matrix operation, and top-r selection can be efficiently implemented using batched top-k routines on GPUs. By constraining each token in $\mathcal{A}$ to merge with at most one token in $\mathcal{B}$, the method preserves most tokens and introduces gradual perturbations, avoiding excessive information loss. Bipartite soft matching can also be extended across frames in video transformers. Let $X^{(t)}$ be the tokens of the current frame and $X^{(t-1)}$ be cached tokens from the previous frame. One can define $\mathcal{A} = X^{(t)}$ and $\mathcal{B} = X^{(t-1)}$, computing S to match and merge tokens across time. This enables reusing previously computed representations, exploiting temporal redundancy and further improving computational efficiency [27].

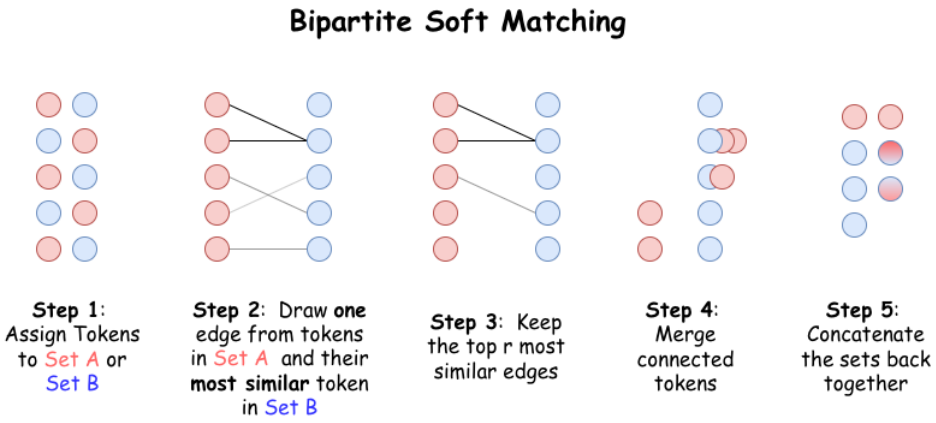

Figure 4.3. Bipartite Soft Matching: Merging Similar Tokens to Reduce Redundancy

4.5 Matching algorithm

In our framework, the first transformer layer is always computed normally for all tokens. This is necessary because no prior attention information exists to guide token selection or to determine which background tokens are redundant. Starting from the second layer onward, we exploit the attention map computed in the previous layer to generate a *saliency map*, which identifies the most informative tokens. This saliency-guided matching allows the algorithm to focus on foreground or changing regions while reusing cached representations for largely static background tokens, thereby reducing redundant computation. Given the background tokens extracted from an inference image, denoted by $\mathbf{X}_{\text{BG}}^{\text{inf}}$, and all tokens stored in the cache, we compute similarity scores using the normalized inner product (cosine similarity):

$$\text{scores}_{i,j} = \frac{\mathbf{x}_i^{\text{inf}} \cdot \mathbf{x}_j^{\text{cached}}}{\|\mathbf{x}_i^{\text{inf}}\|_2 \, \|\mathbf{x}_j^{\text{cached}}\|_2}, \quad \mathbf{x}_i^{\text{inf}} \in \mathbf{X}_{\text{BG}}^{\text{inf}}, \ \mathbf{x}_j^{\text{cached}} \in \mathbf{X}_{\text{BG}}^{\text{cached}}. \tag{4.12}$$

The similarity scores are then sorted to select the top r_{match} matches, where r_{match} is a hyperparameter controlling the number of cached tokens to retrieve:

$$\mathcal{I}_{\text{matched}} = \text{top\_k}(\text{scores}, k = r_{\text{match}}). \tag{4.13}$$

Using these indices, the corresponding background tokens from the cache are gathered:

$$\mathbf{X}_{\text{BG}}^{\text{matched}} = \text{gather}(\mathbf{X}_{\text{BG}}^{\text{cached}}, \mathcal{I}_{\text{matched}}). \tag{4.14}$$

To reuse keys and values of matched tokens efficiently, we define the set of tokens that require forward propagation as:

$$\mathbf{X}_{\text{BG}}^{\text{forward}} = \mathbf{X}_{\text{FG}} \parallel \mathbf{X}_{\text{BG}}^{\text{unmatched}}, \tag{4.15}$$

where $\mathbf{X}_{\text{FG}}$ are foreground tokens extracted from the input, and $\mathbf{X}_{\text{BG}}^{\text{unmatched}}$ are background tokens that did not find a high-similarity match in the cache. Only these tokens require fresh computation of keys and values:

$$\mathbf{K}_{\text{forward}}, \quad \mathbf{V}_{\text{forward}}. \tag{4.16}$$

The final keys and values for attention are obtained by concatenating newly computed and cached tokens:

$$\mathbf{K}_{\text{new}} = \mathbf{K}_{\text{forward}} \parallel \mathbf{K}_{\text{matched}}, \quad \mathbf{V}_{\text{new}} = \mathbf{V}_{\text{forward}} \parallel \mathbf{V}_{\text{matched}}. \tag{4.17}$$

Query tokens attend to all new keys and values:

$$\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{FG}} \parallel \mathbf{X}_{\text{BG}}^{\text{unmatched}} \parallel \mathbf{X}_{\text{BG}}^{\text{matched}}. \tag{4.18}$$

This approach achieves substantial computational savings because only a subset of tokens undergoes full attention computation, while redundant background tokens reuse cached keys and values. Using the attention map from the previous layer as a saliency guide ensures that the network focuses on informative regions, maintaining accuracy while reducing computation for static areas. Over multiple layers, this progressive saliency-based matching allows a transformer to leverage temporal and spatial redundancy without additional overhead in the first layer.

Two-Phase Processing. The TBKV pipeline can be conceptually divided into two phases. During the warm-up phase, a small set of initial frames is processed to construct a compact cache of merged background tokens together with their associated key and value representations. During the inference phase, new frames are processed by matching their background tokens against the cached representations, allowing the model to reuse previously computed keys and values whenever high similarity is detected. This separation enables the model to amortize the cost of token processing across multiple frames while maintaining compatibility with the standard transformer attention mechanism.

4.6 Computational Complexity Analysis

In this section, we analyze the computational cost of our token merging and key-value caching strategy. In a standard Vision Transformer applied to video data, each frame is processed independently, and keys and values are recomputed for all tokens. Given N tokens per frame, the computation of the attention mechanism scales as $\mathcal{O}(N^2D)$, while the linear projections required to compute queries, keys, and values scale as $\mathcal{O}(ND^2)$. When processing a video sequence of T frames, the total computational cost therefore scales linearly with the number of frames, resulting in repeated computation for tokens that represent nearly identical visual content across consecutive frames. Compared to naive ViT inference, the proposed TBKV method reduces both the number of token projections and the effective size of the attention matrix. By reusing cached key-value representations and merging redundant background tokens, the method significantly lowers computational complexity for long video sequences while preserving the full processing capacity for informative foreground tokens.

4.6.1 Background Extraction and Matching

Let the input tensor have dimensions $\mathbf{X} \in \mathbf{R}^{B\times T\times N\times D}$, where B is the batch size, T is the number of frames, N is the number of tokens per frame, and D is the embedding dimension.

- **Background extraction:** Computing the sharpness scores S_i for all tokens requires attention entropy calculations. For each token, computing the entropy over N tokens is $\mathcal{O}(N)$, so the total cost is

$$\mathcal{O}(BTN^2).$$

- **Top-k matching:** Selecting the top r_{match} most similar cached tokens for each background token requires computing cosine similarity between $\mathbf{X}_{\text{BG}}^{\text{inf}}$ and $\mathbf{X}_{\text{BG}}^{\text{cached}}$, resulting in

$$\mathcal{O}(N_{\text{BG}}^{\text{inf}}\, N_{\text{BG}}^{\text{cached}}\, D),$$

  where $N_{\text{BG}}^{\text{inf}}$ and $N_{\text{BG}}^{\text{cached}}$ are the number of background tokens in the inference image and cache, respectively.

- **Top-r selection and gather:** Sorting and selecting r_{match} matches is $\mathcal{O}(N_{\text{BG}}^{\text{inf}} \log r_{\text{match}})$, which is negligible compared to the similarity computation.

4.6.2 Key-Value Computation

After matching, only unmatched background tokens and foreground tokens require fresh key and value computation. Let the total number of tokens requiring new computation be $N_{\text{forward}} = |\mathbf{X}_{\text{FG}}| + |\mathbf{X}_{\text{BG}}^{\text{unmatched}}|$. The cost of computing queries, keys, and values is:

$$\mathcal{O}(N_{\text{forward}} D^2),$$

assuming standard linear projections for Q, K, V. The matched tokens reuse cached keys and values, resulting in a **significant reduction** in computation proportional to the number of matched tokens:

$$\text{Savings} \sim \frac{|\mathbf{X}_{\text{BG}}^{\text{matched}}|}{N}.$$

4.6.3 Attention Computation

Finally, the attention matrix is computed over all new tokens:

$$\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{FG}} \parallel \mathbf{X}_{\text{BG}}^{\text{unmatched}} \parallel \mathbf{X}_{\text{BG}}^{\text{matched}}.$$

For $N_{\text{new}} = |\mathbf{X}_{\text{new}}|$, the self-attention cost is

$$\mathcal{O}(N_{\text{new}}^2 D),$$

which is reduced compared to the naive full-token attention due to the reduced number of background tokens after merging.

Table 4.1. Computational complexity of the main components of the TBKV pipeline.

Operation	Complexity
Background saliency computation	$\mathcal{O}(BTN^2)$
Token matching with cache	$\mathcal{O}(N_{\text{BG}}^{inf} N_{\text{BG}}^{cached} D)$
Q/K/V projections (forward tokens)	$\mathcal{O}(N_{\text{forward}} D^2)$
Self-attention computation	$\mathcal{O}(N_{\text{new}}^2 D)$

Chapter 5

Evaluation

This chapter presents the experimental setup and results obtained by applying the proposed Temporal Background Key Value Reuse mechanism to the task of video action detection. The experiments present a wide range of evaluations on well-known datasets and Video Vision Transformers. Additionally, for completeness, experiments were conducted using different sizes of Vision Transformer backbones. The experiments are designed to evaluate the effectiveness of applying a temporal token reduction technique in conjunction with a KV cache that is able to store a meaningful representation of tokens. The objectives of this experimental evaluation are the following:

- assess a consistent FLOPs reduction of the algorithm on video data while maintaining acceptable accuracy
- analyze advantages, in terms of memory footprint, of caching mechanisms with merged representation against raw representation
- prove model scalability to different sizes of vision transformer backbones
- establish algorithm flexibility for the application of other state–of–the–art token reduction techniques on foreground tokens

We first describe the datasets and model on which the algorithm was tested, together with an extensive description of model implementation, configuration, and preprocessing applied to the input. Follows a description of the experimental setup and evaluation metrics together with a reproducibility section. We then report results on video action detection datasets using proposed models against baseline and with the integration of other tokens reduction procedure. Finally, in order to prove our objective, we present ablation studies where a wide number of plots are used to prove each characteristic of the proposed algorithm. The remainder of the chapter is organized as follows: ...

5.1 Evaluation Settings

5.1.1 Datasets

Video action detection is a challenging task in computer vision that requires identifying both *what* action is happening in a video and *where* it occurs over time. In contrast to simpler video classification tasks, where a single label is assigned to an entire clip, action detection requires the model to reason about the temporal evolution of a scene and to localize the start and end of actions within the video sequence. This problem is particularly demanding because videos contain a large number of frames, and each frame is typically represented by hundreds of visual tokens when processed by modern Vision Transformers. As a result, models must process a significant amount of redundant information, especially in scenes where large portions of the background remain unchanged across consecutive frames. While this redundancy does not usually contribute meaningful information for recognizing the action itself, it still leads to a substantial computational cost during inference. In recent years, transformer–based architectures have shown strong performance on video understanding tasks thanks to their ability to model long–range dependencies between tokens. However, this benefit comes with a considerable increase in computational complexity, since attention operations and key–value projections must be computed for every token at every frame. When dealing with long videos or high–resolution inputs, this quickly becomes a bottleneck in terms of both computation and memory consumption. The proposed Temporal Background Key–Value Reuse (TBKV) mechanism is motivated by the observation that background regions in videos tend to change slowly over time compared to foreground elements involved in the action. For example, in many action detection scenarios, such as sports or daily activities, the environment often remains largely static while only a small subset of objects or people exhibit meaningful motion. This suggests that repeatedly recomputing representations for background tokens across frames may be unnecessary. Our approach, therefore, focuses on identifying and compressing redundant background tokens and reusing their corresponding key and value representations through a caching mechanism. By doing so, the model can avoid recomputing expensive operations for tokens that carry similar visual information across frames. At the same time, foreground tokens, which are typically more informative for the action recognition task, are preserved and processed normally. This design allows the model to maintain strong detection performance while significantly reducing the computational burden associated with processing long video sequences. In the following sections, we evaluate the effectiveness of this strategy on standard video action detection datasets using multiple Vision Transformer models.

Kinetics–400

Kinetics–400 is one of the most widely used large–scale datasets for video understanding tasks. It contains 400 different human action classes, with at least 400 video clips available for each class. The videos were collected from YouTube and trimmed so that each clip focuses on a specific action instance. On average, the duration of each clip is around 10 seconds, which makes the dataset suitable for training and evaluating models that need to capture short–term temporal dynamics. The dataset covers a very broad set

of everyday activities and interactions. Many classes involve human–object interactions, such as playing musical instruments or using tools, while others involve interactions between people, such as hugging, shaking hands, or playing team sports. Because the videos originate from different online sources, they exhibit a high degree of variability in terms of camera motion, lighting conditions, viewpoints, and background scenes. This diversity makes the dataset particularly useful for evaluating how well models generalize to real–world scenarios. Another reason why Kinetics–400 is commonly adopted in the literature is that it provides a good balance between dataset size and class diversity. Compared to smaller action recognition datasets, the larger number of classes and training samples allows modern deep learning models to learn more robust visual and temporal representations. As a result, Kinetics–400 has become a standard benchmark for many works in video classification and video transformer architectures. In our experiments, the dataset serves as a strong evaluation benchmark to assess whether the proposed token reduction and KV reuse strategy can maintain competitive accuracy while reducing computational cost.

UCF101

UCF101 is another widely used benchmark dataset for human action recognition in videos. It consists of 13,320 video clips belonging to 101 different action categories. Similar to Kinetics–400, the videos were collected from YouTube and therefore contain realistic variations in camera motion, background scenes, and recording quality. One distinctive aspect of UCF101 is the way the videos are organized into 25 groups. Each group contains several clips (typically between four and seven) that correspond to the same action class and often share common characteristics, such as similar backgrounds, viewpoints, or performers. This grouping structure is particularly useful when evaluating generalization, since models can be tested on videos that differ from those seen during training but still belong to the same action category. The action classes included in UCF101 cover a diverse set of human activities that can be broadly divided into five categories. These categories include human–object interactions (for example using tools or interacting with everyday objects), actions involving only body motion, interactions between multiple people, activities related to playing musical instruments, and various sports actions. Because of this diversity, the dataset has been extensively used as a benchmark to evaluate the performance of video recognition models. Despite being smaller than more recent large–scale datasets, UCF101 remains an important benchmark for assessing model efficiency and generalization. In particular, its relatively shorter videos and clear action labels make it suitable for evaluating whether computational optimizations, such as the proposed temporal token reduction and key–value reuse mechanism, can reduce the processing cost without significantly affecting recognition performance.

5.1.2 Models

In recent years, Vision Transformers (ViTs) have emerged as a strong alternative to convolutional neural networks for many computer vision tasks. Instead of relying on convolutional filters, these models represent an image as a sequence of visual tokens and process

them using the self–attention mechanism originally introduced in natural language processing. This formulation allows the model to capture long–range relationships between different regions of an image, which can be beneficial for tasks that require understanding global context. Video Vision Transformers extend this idea to video data. Rather than processing a single image, the model receives a sequence of frames and converts them into spatio–temporal tokens. In practice, the video is divided into small 3D patches, often referred to as tubelets, which encode both spatial information and temporal information across consecutive frames. These tokens are then processed by a transformer encoder that applies attention across the entire sequence. Compared to standard image–based Vision Transformers, video transformers are able to model how visual content evolves over time. This capability is particularly important for tasks such as action recognition or activity detection, where motion and temporal context play a key role. For instance, recognizing an action like "drinking from a cup" usually requires observing several frames in sequence in order to understand the movement being performed. At the same time, introducing the temporal dimension significantly increases the computational cost of the model. While a single image may generate a few hundred tokens, a short video clip can easily produce several thousand tokens once all frames are tokenized. Since the attention mechanism has quadratic complexity with respect to the number of tokens, the computational requirements grow rapidly as the number of frames increases. Despite this additional cost, video transformers generally outperform approaches that process frames independently and aggregate the results afterwards. By modeling spatial and temporal information jointly, the model can better capture complex actions and interactions that unfold over time. However, this improved representation also highlights an important inefficiency. In many videos, large portions of the scene remain relatively static across consecutive frames, especially in background regions. As a result, the model repeatedly processes tokens that contain very similar visual information. This observation motivates the development of techniques that reduce redundancy in the token representation. The TBKV method proposed in this work addresses this issue by identifying redundant background tokens and reusing their key–value representations across frames, thereby reducing unnecessary computation while maintaining the model's ability to capture meaningful temporal dynamics.

VideoMAE

In our experiments, we primarily rely on VideoMAE as the backbone architecture for evaluating the proposed method. VideoMAE is a recent Video Vision Transformer that has demonstrated strong performance across several video understanding benchmarks. One of the main reasons for its effectiveness is its masked autoencoding pretraining strategy, which enables the model to learn robust spatio–temporal representations from large amounts of unlabeled video data. From an architectural perspective, VideoMAE processes videos by dividing them into small spatio–temporal patches, commonly referred to as tubelets. Each tubelet corresponds to a small spatial region of the video observed over a few consecutive frames. These tubelets are then embedded into tokens that can be processed by a transformer encoder. This representation produces a relatively dense sequence of tokens that captures both spatial and temporal information. While such a

rich representation is useful for modeling fine–grained visual patterns, it also introduces a significant amount of redundancy. In particular, background regions that remain stable across frames tend to generate tokens with highly similar features. This characteristic makes VideoMAE a suitable architecture for studying token reduction techniques. Another practical advantage of VideoMAE is the availability of publicly released pretrained models that have been fine–tuned on standard video benchmarks such as Kinetics–400. These checkpoints provide a strong baseline and allow us to evaluate the impact of TBKV under consistent experimental conditions. By integrating our token merging and key–value reuse mechanism into this architecture, we can directly analyze the computational savings and the potential trade–offs between efficiency and accuracy.

VideoMAE Architecture VideoMAE is a self–supervised video pretraining framework that extends the Masked Autoencoder paradigm to spatio–temporal data. The core idea is to divide a video into small 3D patches and train the model to reconstruct the missing parts of the input. The input video is first divided into non–overlapping space–time cubes of size $T \times S \times S$, where T denotes the number of frames per cube and $S \times S$ represents the spatial resolution of the patch. Each cube is flattened and projected into a token embedding through a linear layer, producing a sequence of spatio–temporal tokens. During pretraining, VideoMAE applies an extremely high masking ratio, typically masking around 90—-95% of the tokens. This design choice is motivated by the high temporal redundancy present in videos. Since consecutive frames often contain similar visual information, the model can still infer the missing content even when most tokens are hidden. To avoid trivial solutions, VideoMAE uses a tube masking strategy, where the same spatial position is masked across all frames. Only the visible tokens are processed by the transformer encoder, which follows a standard Vision Transformer architecture. A lightweight decoder is then used to reconstruct the masked patches from the encoded representations. The training objective is to minimize the reconstruction error between the predicted and original pixel values of the masked cubes. This asymmetric encoder–decoder design significantly reduces the computational cost during pretraining, since the encoder processes only a small subset of the original tokens. At the same time, the model is forced to learn meaningful spatio–temporal representations in order to solve the reconstruction task.

ViViT

To further evaluate the generality of our approach, we also conduct additional experiments using ViViT (Video Vision Transformer), one of the earliest transformer–based architectures specifically designed for video understanding. Similar to other video transformers, ViViT represents a video as a sequence of tokens and processes them through self–attention layers. One of the main contributions of the model is the exploration of different strategies for handling spatial and temporal information within the transformer architecture. In particular, ViViT investigates several ways to factorize attention across space and time. Instead of always computing attention jointly over all tokens, some variants separate spatial attention (within each frame) from temporal attention (across frames). This design can reduce computational complexity while still allowing

the model to capture temporal relationships. Because of these architectural variations, ViViT provides an interesting alternative setting in which to evaluate the TBKV mechanism. Testing the proposed method on both VideoMAE and ViViT allows us to better understand whether the benefits of token merging and key–value reuse generalize across different transformer architectures. Overall, including multiple Video Vision Transformer backbones in our experiments helps demonstrate that the proposed approach is not limited to a single model. Instead, it can be applied more broadly to transformer–based video architectures that operate on large sequences of spatio–temporal tokens.

ViViT Architecture ViViT adapts the Vision Transformer paradigm to the video domain by converting a video into a sequence of tokens and applying multi–head self–attention. A key aspect of the model design is how the video is tokenized and how attention is applied across space and time. One possible tokenization strategy consists of sampling a fixed number of frames from the video and dividing each frame into non–overlapping 2D patches, as done in standard Vision Transformers. The patches from all frames are then concatenated, producing a sequence of $N \times T$ tokens, where N is the number of patches per frame and T is the number of frames. An alternative approach uses tubelet embeddings, where tokens correspond to small 3D cubes that span both spatial and temporal dimensions. This representation allows the model to directly capture spatio–temporal patterns within each token. ViViT also explores several variants for applying attention across these tokens. In the simplest version, attention is computed jointly across all tokens from all frames, allowing the model to capture global spatio–temporal relationships. However, this approach can be computationally expensive when the number of tokens becomes large. To address this limitation, other variants factorize attention into spatial and temporal components. For example, one design processes each frame independently using a spatial transformer encoder and then models interactions across frames using a separate temporal transformer. Another variant splits attention operations within each transformer block, first attending across spatial tokens and then across temporal tokens. These factorization strategies reduce the computational complexity while still allowing the model to capture both spatial structure and temporal dynamics. As a result, ViViT provides a flexible framework for exploring different trade–offs between computational cost and representational power in video transformers.

5.1.3 Experimental Setup

All experiments were conducted on a GPU node from the Chameleon Cloud infrastructure, equipped with an **NVIDIA RTX 6000 GPU featuring 24** GB GDDR6 VRAM, 4608 CUDA cores, Tensor Cores, and RT Cores for mixed–precision acceleration, and full CUDA/cuDNN support in the Chameleon OpenStack environment. Video action detection tasks are performed using a final average pooling layer to aggregate features before propagating them in an MLP layer, which outputs logits whose size depends, of course, on the number of classes. For completeness, results reported in the tables were obtained using test split of both datasets since they provide a larger number of samples. While for ablation studies, since usually the number of configurations was significant, experiments

were conducted on evaluation splits, which still provide informative results while tapering off the amount of time needed for each experiment. In all experiments, the first P frames of each video were used exclusively for caching, while matching operations were performed on the remaining last $N - P$ frames (where N is the total number of frames per video). Since caching is applied only to the initial frames and its computational cost is typically negligible, all reported metrics were computed solely on the matching frames. For the fairness of comparison, baseline evaluations were conducted on the same $N - P$ frames of the matching part in order to have reliable metrics. Further details on model and dataset configuration will be explained in the following sections.

Evaluation Metrics

Following most of the other works on token reduction techniques in the literature, the evaluation of the proposed method is based on two main metrics: the average number of floating–point operations (FLOPs) required to process a video and the classification accuracy. FLOPs are commonly used as a proxy for measuring the computational cost of deep learning models. In transformer–based architectures, a large portion of the computation comes from matrix multiplications inside the attention layers and the feed–forward networks. Since token reduction methods aim to decrease the number of tokens processed by the model, their main effect is a reduction in the amount of computation required during inference. Measuring FLOPs, therefore, provides a direct way to quantify how much computation is saved when applying the proposed technique compared to the baseline model. In this work, we report the *average FLOPs per video*. Given a dataset containing N videos, the metric is computed by averaging the number of operations required to process each individual sample:

$$\text{Avg FLOPs} = \frac{1}{N} \sum_{i=1}^{N} \text{FLOPs}(v_i) \tag{5.1}$$

where v_i denotes the i–th video in the evaluation set. This metric allows us to estimate the expected computational cost of the model during inference on a typical dataset. While reducing computational cost is important, it should not come at the expense of significantly degrading model performance. For this reason, the second evaluation metric considered in this work is classification accuracy. In video classification benchmarks, it is common to report both *Top–1* and *Top–5* accuracy. Top–1 accuracy measures the percentage of samples for which the model's most confident prediction matches the ground–truth label. Formally, if $\hat{y}_i$ is the predicted label for the i–th sample and y_i is the true label, Top–1 accuracy can be computed as

$$\text{Top–1 Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(\hat{y}_i = y_i) \tag{5.2}$$

where $\mathbf{1}(\cdot)$ is the indicator function that equals 1 when the condition is satisfied and 0 otherwise. Top–5 accuracy is a slightly more relaxed metric. Instead of considering only the most confident prediction, it checks whether the correct label appears among the five most probable predictions produced by the model. This metric is particularly useful for

large–scale classification datasets where several classes may be visually similar. More formally, if $\text{Top}_5(\hat{y}_i)$ denotes the set of the five highest scoring predictions for sample i, Top–5 accuracy can be expressed as

$$\text{Top–5 Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(y_i \in \text{Top}_5(\hat{y}_i)) \tag{5.3}$$

By reporting both Top–1 and Top–5 accuracy together with the average number of FLOPs per video, it is possible to evaluate the trade–off between computational efficiency and predictive performance. This comparison is particularly relevant for token reduction techniques, whose goal is to significantly decrease computational cost while preserving as much accuracy as possible.

FLOPs computation To obtain an accurate estimate, we implemented a counting mechanism directly within the transformer architecture. Specifically, a base class was introduced for the transformer modules, where common operations such as linear projections and matrix multiplications were replaced with custom implementations that track the number of floating–point operations performed during inference. In particular, custom versions of the linear layers and matrix multiplication operations were defined (e.g., `CountedLinear`), allowing us to accumulate the number of FLOPs produced by each operation while the model processes the input video. The collected values are then aggregated to obtain the total computational cost for each forward pass. For clarity, the total number of FLOPs is decomposed into several components corresponding to the main operations performed inside the transformer blocks:

- **Linear FLOPs**: operations generated by linear projections, including the query, key, and value projections as well as the output projection in the attention layers.
- **Matmul FLOPs**: operations produced by matrix multiplications inside the attention mechanism, in particular the computation of the attention scores and the subsequent weighted aggregation of the value vectors.
- **Bias FLOPs**: operations associated with bias additions in the linear layers.

This breakdown allows us to better understand which parts of the transformer architecture contribute the most to the overall computational cost and how the proposed token reduction strategy affects these components. This strategy follows other works' FLOPs computation strategy since most of the libraries present in the literature do not adapt efficiently to a custom module inserted in the transformer class.

Kinetics–400 Dataset and Preprocessing

All experiments are conducted on the Kinetics–400 dataset [19], a widely used benchmark for large–scale video action recognition. The dataset contains approximately 240,000 training videos and around 20,000 validation videos spanning 400 human action classes. Each clip is typically around ten seconds long and trimmed to contain a single action. Videos are decoded using the Decord library, which allows efficient frame extraction

directly from compressed files. For each video, a fixed number of frames is sampled uniformly across the temporal duration of the clip. In our setup, we sample $T = 16$ frames with a temporal sampling rate of 4, meaning that every fourth frame is selected from the original sequence. This strategy provides a reasonable temporal coverage while keeping the computational cost manageable. Each sampled frame undergoes a standard sequence of spatial preprocessing steps. First, frames are resized so that the shorter side equals 224 pixels. A center crop of size 224×224 is then applied. Finally, pixel values are converted to tensors and normalized channel–wise using the ImageNet statistics:

$$\mu = [0.485, 0.456, 0.406], \qquad \sigma = [0.229, 0.224, 0.225].$$

After preprocessing, each clip is represented as a tensor of shape.

$$(3, T, 224, 224),$$

where 3 denotes the RGB channels and $T = 16$ is the number of sampled frames. During evaluation, we follow a multi–view inference protocol commonly adopted in video recognition works. Each video is evaluated using two temporal segments and three spatial crops, resulting in six clips per video. The predictions from all clips are averaged to obtain the final video–level classification.

ViViT Model Configuration

For part of our experiments, we employ the Video Vision Transformer (ViViT) architecture [1]. ViViT extends the Vision Transformer to video inputs by modeling both spatial and temporal relationships between patches.

Tubelet Embedding The input clip is partitioned into non–overlapping spatio–temporal patches, commonly referred to as tubelets. In our configuration, the spatial patch size is 16×16 pixels while the temporal tubelet size is 2 frames. Given an input clip of $T = 16$ frames with spatial resolution 224×224, the number of spatial tokens per frame is

$$N_s = \frac{224}{16} \times \frac{224}{16} = 14 \times 14 = 196.$$

Since the temporal tubelet size is 2, the number of temporal tokens becomes

$$N_t = \frac{16}{2} = 8.$$

Each tubelet is flattened and projected through a linear embedding layer into a feature vector of dimension $d = 768$.

Transformer Encoder The token sequence is processed by a stack of transformer blocks. Each block contains a multi–head self–attention (MHSA) layer followed by a feed–forward network (MLP). The model uses 12 attention heads, each with dimension 64, resulting in a total embedding dimension of 768. The feed–forward network uses an expansion ratio of 4, leading to a hidden dimension of 3072. A classification token is prepended to the token sequence, and its final representation is passed through a linear layer to produce predictions over the 400 action classes.

VideoMAE Model Configuration

In addition to ViViT, we also evaluate our method using a VideoMAE backbone [40]. VideoMAE is a transformer–based architecture designed for video representation learning, which follows a structure similar to Vision Transformers but incorporates spatio–temporal patch embeddings tailored for video data.

Tubelet Embedding Similar to ViViT, VideoMAE divides the input clip into spatio–temporal patches. The spatial patch size is 16×16, while the temporal tubelet size is set to 2 frames. Therefore, the input video of size (3,16,224,224) is decomposed into a sequence of tokens representing localized regions across space and time. Each tubelet is flattened and linearly projected into a 768–dimensional embedding space. Positional embeddings are added to retain information about the spatial and temporal ordering of tokens.

Transformer Backbone The embedded tokens are processed by a ViT–Base style transformer encoder composed of 12 transformer layers. Each layer contains multi–head self–attention with 12 heads and a feed–forward network with hidden dimension 3072. A classification token is appended to the sequence, and its final representation is used by a linear classification head to produce logits for the 400 Kinetics classes. The architecture follows the standard configuration commonly adopted in VideoMAE–based video classification pipelines, allowing a fair comparison between the baseline model and our proposed modifications.

UCF101 Dataset and Preprocessing

In addition to Kinetics–400, we also evaluate our method on the UCF101 dataset [38]. UCF101 is a widely used benchmark for action recognition consisting of 13,320 videos spanning 101 human action categories. The videos are collected from YouTube and cover a wide range of activities such as sports, musical performances, and everyday actions.

Frame Sampling and Preprocessing For each video, we sample $T = 16$ frames using a temporal sampling rate of 4, meaning that every fourth frame from the original sequence is selected. This allows the model to observe motion across a longer temporal span while maintaining a manageable number of tokens for the transformer encoder. Each sampled frame is resized so that the shorter side is 224 pixels, followed by a center crop of size 224×224. The frames are then converted into tensors and normalized using the standard ImageNet statistics:

$$\mu = [0.485, 0.456, 0.406], \qquad \sigma = [0.229, 0.224, 0.225].$$

After preprocessing, each video clip is represented as a tensor of shape (3,16,224,224).

Inference Protocol During evaluation, we adopt a multi–view testing strategy similar to the one commonly used in video recognition literature. Each video is evaluated using

two temporal segments and three spatial crops, resulting in a total of six clips per video. Predictions from all clips belonging to the same video are averaged to obtain the final video–level classification score.

Model Configuration Experiments on UCF101 use the same VideoMAE backbone described in the previous section, based on a ViT–Base architecture with an embedding dimension of 768, 12 transformer layers, and 12 attention heads. The spatial patch size is 16×16 pixels, and the temporal tubelet size is set to 2 frames. The classification head is adapted to output predictions over the 101 action classes of the UCF101 dataset. This setup allows us to evaluate the proposed method on a smaller but widely adopted benchmark while maintaining consistency with the configuration used for larger datasets.

5.1.4 Baselines

ToMe

As previously mentioned, ToMe is the most renowned token merging algorithm in the literature. Other works, such as PRANCE [25], have utilized ToMe for various purposes, including enabling a reinforcement learning (RL) agent to prune or merge tokens. A comparison of my technique, which is a merging technique for background tokens that exploits temporal reduction, with one of the best state–of–the–art merging methods seemed more relevant. It is important to note that ToMe implements token merging without accounting for temporal dependencies between frames. This process is primarily applied to individual images, though there have been experimental endeavors in video. Furthermore, the present algorithm merges tokens exclusively in the background, whereas ToMe merges them on all tokens. In the course of the experiments, a fused version of the algorithm for merging background tokens and ToMe for merging foreground tokens was also included. The present study will demonstrate the adaptability of the proposed method to augment existing techniques, thereby underscoring its position as a viable alternative to existing methods. This method represents a complementary strategy, and none explicitly implements training–free background–aware KV reuse across generic video inference, which is the focus of this thesis

Naive Temporal Caching Baseline

Caching mechanism memory footprint and performance represent a central focus of this thesis, as temporal key–value caching is one of the main advantages of the proposed approach compared to the few existing works that explore temporal token reduction strategies in video transformers. Unfortunately, at the time of writing, implementations and detailed experimental materials for most of these methods are not publicly available. This makes a direct empirical comparison difficult, particularly with respect to memory usage and runtime behavior. To partially address this limitation, we introduce a controlled baseline designed to isolate the effect of the caching mechanism itself. In this baseline configuration, the TBKV framework remains unchanged in terms of architecture and execution flow. However, instead of performing token merging between the current tokens and the cached key–value representations, the system simply stores the

key and value tensors without applying any reduction operation. This setup allows us to approximate the behavior of a naive temporal KV caching strategy in which previously computed representations are reused, but no token compression is applied. As a result, the comparison highlights the contribution of the proposed token merging mechanism in terms of both computational efficiency and memory footprint, while keeping the rest of the pipeline identical.

5.2 Evaluation Results

5.2.1 Efficiency–Accuracy Trade–off Compared to Baseline Methods

In this first plot, we analyze the overall performance of TBKV compared to the baseline token merging method, ToMe, as well as a hybrid configuration where TBKV is applied on background tokens while ToMe operates on foreground tokens. Results show a clear advantage of TBKV in terms of the accuracy/efficiency trade–off. In particular, TBKV consistently achieves lower computational cost while maintaining comparable or higher accuracy than the baseline approach. For completeness, ToMe was evaluated under different values of the parameter r, which controls the token merge ratio. Larger values of r correspond to more aggressive token reduction and therefore higher computational savings, but typically lead to a larger degradation in accuracy. The results indicate that TBKV provides a more stable reduction strategy, as it selectively targets background tokens that contain redundant information across frames. In contrast, ToMe performs merging globally across all tokens, which may affect foreground regions that are more relevant for the final prediction. Finally, the hybrid configuration combining TBKV with ToMe further highlights the flexibility of the proposed method. While TBKV already achieves strong reductions on its own, applying ToMe on the remaining foreground tokens can lead to additional FLOPs savings. This demonstrates that TBKV can serve not only as a standalone token reduction strategy but also as a complementary mechanism that can be integrated with existing token pruning or merging techniques. On plot 5.1, we keep the matching ratio fixed for TBKV application and only change the merging ratio during caching. We can show, for completeness, on 5.2 the performance of tbkv as we change the matching ratio. We can notice how having a low matching ratio tends to have close performance to baseline and ToMe.

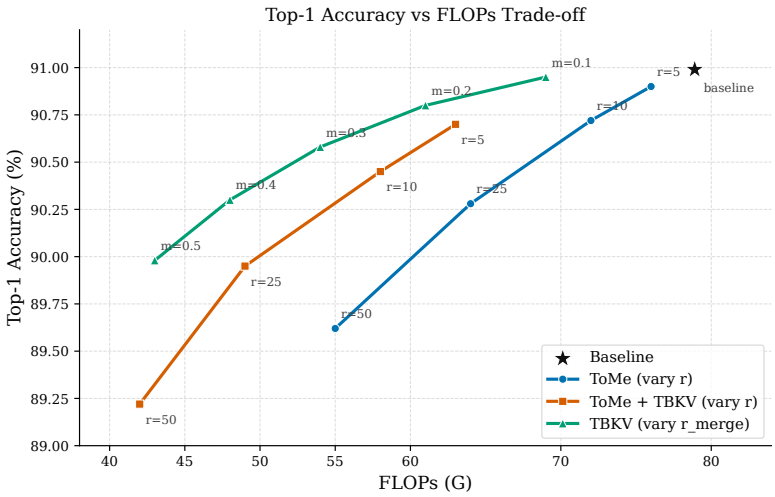

Figure 5.1. Accuracy versus FLOPs trade–off for TBKV, ToMe, and the hybrid TBKV+ToMe configuration. The plot shows that TBKV achieves a more favorable efficiency–accuracy balance by selectively reducing background tokens while preserving informative foreground tokens.

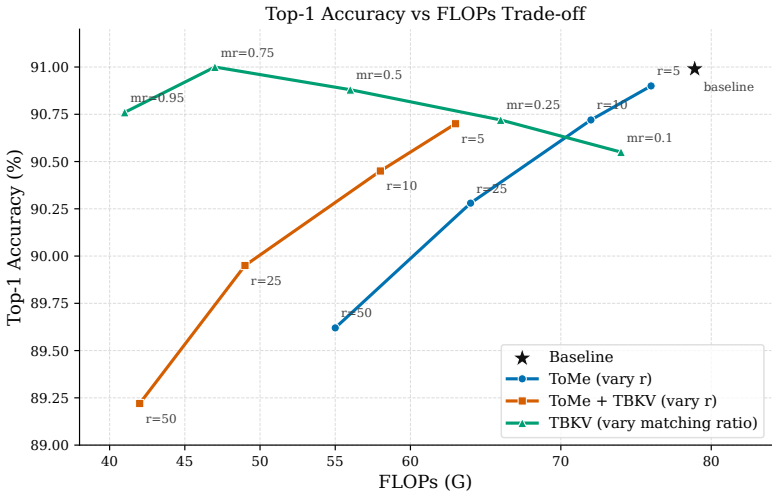

Figure 5.2. Impact of the TBKV matching ratio on the accuracy–FLOPs trade-off. Lower matching ratios approach baseline performance, while higher ratios provide greater computational savings by increasing the reuse of cached token representations.

5.2.2 Scalability of TBKV Across Vision Transformer Backbone Sizes

Due to the increasing computational capabilities of modern edge devices, larger Vision Transformer backbones are increasingly deployed in applications where accuracy is critical. However, the computational and memory requirements of such architectures grow substantially with model size, making efficient token handling strategies essential. In this

plot, we compare the performance of a standard VideoMAE baseline, VideoMAE augmented with a representative token reduction technique, and VideoMAE equipped with our proposed TBKV mechanism, across different backbone sizes ranging from small to huge. Our results show that as the ViT backbone scales, the performance gains achieved by TBKV consistently increase. In particular, the gap between TBKV and competing approaches widens for larger architectures, indicating that the benefits of structured token merging and reuse become more significant at scale. Overall, these findings demonstrate that TBKV is not only effective for compact models but particularly well–suited for large–scale Vision Transformers. As model capacity increases, TBKV leverages the growing redundancy in representations to achieve greater efficiency gains without sacrificing accuracy. This scalability property highlights TBKV as a robust and future–proof solution for high–capacity video models deployed in compute–constrained environments.

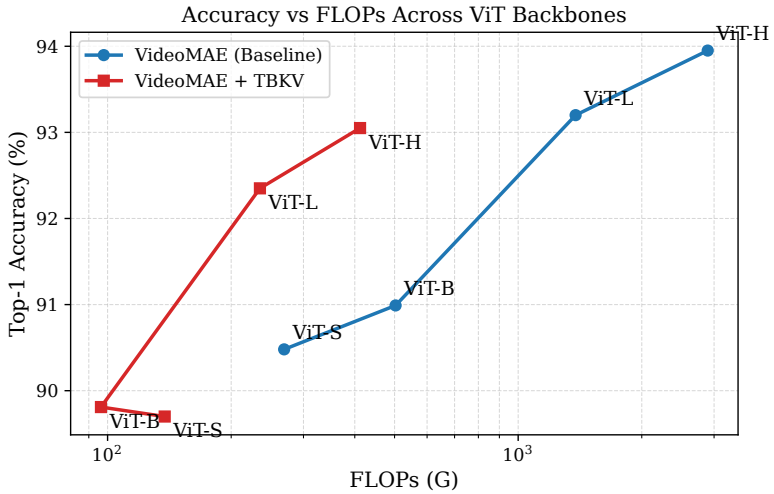

Figure 5.3. Scalability analysis of TBKV across Vision Transformer backbone sizes. The figure shows that computational savings increase as model size grows, demonstrating that TBKV becomes more effective for large–scale video transformers.

5.2.3 Impact of KV Reuse on Transformer Computation

In this experiment, we compare the computational cost of key–value (KV) operations between a standard Video Vision Transformer baseline and our TBKV method across 50 videos. The plot reports cumulative KV FLOPs on the y–axis and video index on the x–axis. The baseline exhibits a linear increase in KV computations as more videos are processed, reflecting the repeated recomputation of key and value projections for each frame. In contrast, TBKV leverages KV reusing across frames, resulting in a cumulative KV FLOPs curve that remains consistently below the baseline line. This demonstrates the efficiency of the caching strategy, where previously computed KV pairs are reused rather than recomputed, reducing overall computation without sacrificing accuracy. The

plot clearly illustrates that KV reuse in TBKV enables nearly linear scaling in practice while maintaining a lower absolute computational cost compared to the naive baseline. This reduction becomes more pronounced as more videos are processed, highlighting the compounding benefits of temporal token caching.

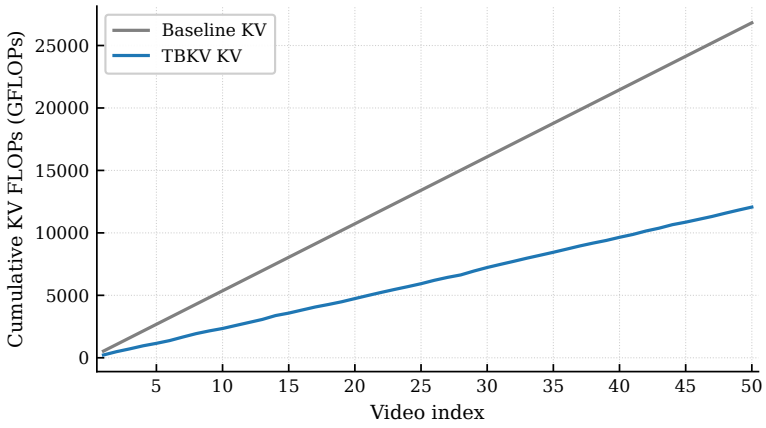

Figure 5.4. Cumulative KV computation for the baseline model and TBKV across multiple videos. The baseline recomputes key and value projections for every frame, while TBKV reuses cached KV representations, significantly reducing the total computational cost.

5.2.4 Effect of Token Merging on Cached Representation Quality

Prior works on temporal token reduction typically store raw tokens along with their corresponding keys and values in the cache. In contrast, our approach stores a merged representation of tokens, aiming to reduce the memory footprint while maintaining semantic fidelity. To isolate the effect of merging, we implement a variant of TBKV that performs key–value reuse without token merging, effectively caching raw features. This allows for a direct comparison between storing raw tokens and storing merged representations. Our results show that, as the merging ratio varies, caching merged tokens consistently outperforms caching raw features. Despite the reduced representation size, the merged cache preserves and in several cases improves downstream performance. These findings suggest that token merging is not merely a compression mechanism, but a crucial component for maintaining informative and stable representations in the cache.

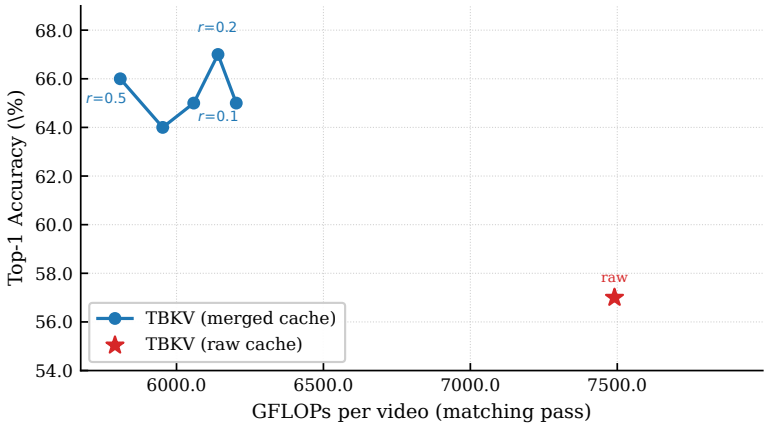

Figure 5.5. Comparison between TBKV caching using merged token representations and a naive cache storing raw tokens. The figure shows that merged representations reduce computation while preserving or improving accuracy, demonstrating the effectiveness of token merging within the cache.

5.2.5 Influence of Cache Size on Accuracy and Computational Cost

In this experiment, we analyze the impact of caching and merging tokens across multiple frames. We vary the parameter P, which represents the number of frames stored in the cache; the number of matching frames is therefore $n_{\text{frames}} - -P$. The plot reports accuracy on the y–axis and FLOPs on the x–axis. Two curves are shown in the plot: one corresponding to the TBKV cache, which stores merged token representations across P frames, and the other corresponding to a raw cache that stores tokens without any merging. The results demonstrate that merged tokens provide higher accuracy per FLOP by consolidating redundant information across frames while maintaining essential semantic content. They also reduce the computational cost by decreasing the number of token–level operations. In addition, the TBKV curve is broader than the raw cache curve, indicating that merging allows stable performance across a wider range of cache sizes. Storing merged tokens requires less memory compared to caching raw tokens, which is particularly relevant for edge devices and streaming applications. Finally, merged tokens enable a smoother trade–off between cache size and computation, avoiding spikes in computation that can occur when using raw tokens. Overall, this experiment highlights how TBKV's token merging mechanism effectively balances efficiency and accuracy, making it a superior strategy for temporal token caching compared to naive raw token storage.

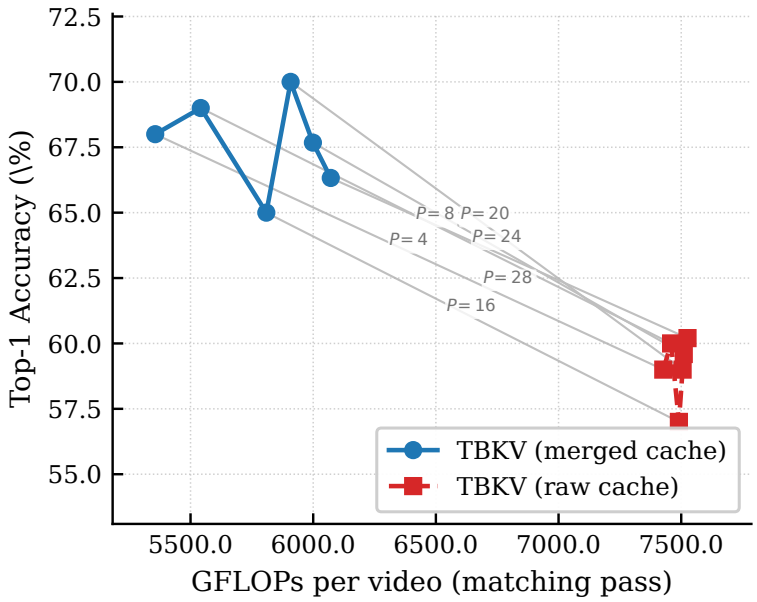

Figure 5.6. Overview of the TBKV caching pipeline during the warm–up phase. Background tokens extracted from the first P frames are merged using bipartite soft matching to build a compact KV cache, enabling computation reuse during subsequent inference frames.

5.2.6 Memory Efficiency of TBKV Caching

To analyze the memory efficiency of TBKV, we study the relationship between cache size and memory operations under different merging ratios. Specifically, we report the cache footprint (in MB) on the y–axis and the number of memory operations (reads and writes associated with KV storage and retrieval) on the x–axis. Each point on the curve corresponds to a different merging ratio, making the curve parametric in the degree of token merging. As the merging ratio decreases, fewer tokens are stored in the cache, leading to a proportional reduction in both memory footprint and memory traffic. This results in a monotonic curve shifting toward the lower–left region of the plot, indicating simultaneous reductions in storage and memory requests. Importantly, the curve demonstrates that TBKV does not introduce excessive memory traffic despite operating on merged representations. On the contrary, reducing the number of stored tokens directly lowers memory accesses, preventing the cache mechanism from becoming a bottleneck. Even at moderate merging ratios, TBKV achieves substantial reductions in cache size while maintaining controlled memory request growth. Overall, this analysis confirms that TBKV scales favorably from a memory systems perspective: The reduction in cache footprint translates into a proportional decrease in memory operations, ensuring that efficiency gains are not offset by increased bandwidth demands.

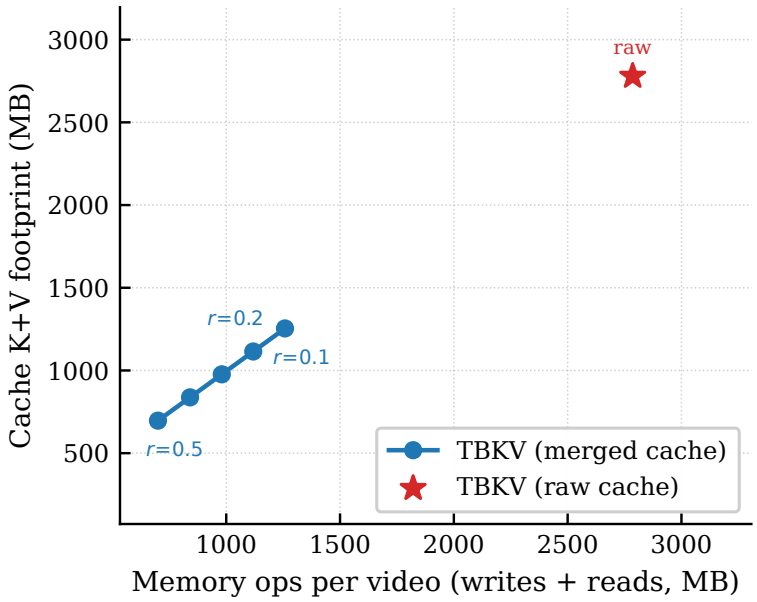

Figure 5.7. Memory footprint and memory operations for TBKV caching compared to a raw token cache. Token merging reduces the number of stored tokens, leading to smaller cache sizes and fewer memory accesses during inference.

5.2.7 VideoMAE

Results on UCF–101 dataset

Method	Backbone	FLOPs	Top–1 (%)	Top–5 (%)
VideoMAE (Baseline)	ViT–B	156.31	93.38	98.54
VideoMAE + TBKV	ViT–B	**86.03 (–44.96%)**	93.47(–0.09 %)	98.52

Table 5.1. Performance comparison on the UCF101 dataset. TBKV significantly reduces FLOPs while maintaining comparable classification accuracy to the baseline VideoMAE model.

Results on Kinetics400 dataset

As shown in table 5.2.7, TBKV achieves substantial FLOPs against while having a negligible accuracy loss. Kinetics400 is our biggest dataset, being our de facto main reference when it comes to testing the efficiency of TBKV. It is worth noticing that while TOP–1 accuracy drops slightly, the TOP–5 accuracy is many times greater with respect to baseline ones since representation–level tokens in the cache provide a way to stabilize the output of logits of the model. This is one of the biggest differences with respect to other kv–caching works applied to ViTs. Moreover, testing with different backbones of different dimensions shows another important characteristic of TBKV, which is its scalability to larger architectures. For a large backbone like ViT–L, we achieve the same FLOPs as a

ViT–S without our algorithm applied, which allows the application of larger architectures on edge devices at the same cost as basic ones with no significant accuracy loss.

Method	Backbone	FLOPs	Top–1 (%)	Top–5 (%)
VideoMAE (Baseline)	ViT–S	269.16	90.48	97.98
VideoMAE + TBKV	ViT–S	**137.76 (–48.82%)**	89.70 (–0.78%)	98.01
VideoMAE (Baseline)	ViT–B	502.86	90.99	98.44
VideoMAE + TBKV	ViT–B	**96.36(–80.82%)**	89.81(–1.17 %)	98.06
VideoMAE (Baseline)	ViT–L	1459.34	93.34	99.04
VideoMAE + TBKV	ViT–L	**229.33(–84.29%)**	92.31(–1.03%)	98.93
VideoMAE (Baseline)	ViT–H	2895.00	93.95	99.22
VideoMAE + TBKV	ViT–H	**412.00(–85.78%)**	93.05(–3.21%)	99.05

Table 5.2. Performance of TBKV on the Kinetics–400 dataset across different backbone sizes. The results show substantial computational reductions while maintaining high classification accuracy.

5.2.8 Comparison with Existing Token Reduction Techniques

VideoMAE results on Kinetics400

Method	Backbone	FLOPs	Top–1 (%)	Top–5 (%)
VideoMAE (Baseline)	ViT–B	502.86	90.99	98.44
VideoMAE + TBKV	ViT–B	**96.36(–80.82%)**	89.81(–1.17 %)	98.06
VideoMAE + TBKV + ToMe	ViT–B	**76.1028(–84.87%)**	89.82% (–1.17%)	98.08

Table 5.3. Evaluation of TBKV combined with ToMe applied to foreground tokens on Kinetics–400. The results highlight how TBKV can serve as a complementary mechanism to existing token reduction methods.

ViViT results on Kinetics400

Method	Backbone	FLOPs	Top–1 (%)	Top–5 (%)
ViViT (Baseline)	ViT–B	537.6	71	97
ViVit + TBKV	ViT–B	**392(–24%)**	71	87
ViViT + TBKV + ToMe	ViT–B	**155.36(–71.1%)**	63.7(–10.242%)	78

Table 5.4. Performance of TBKV on the Kinetics–400 dataset using the ViViT backbone. The results demonstrate that the proposed temporal KV reuse mechanism can be integrated with different video transformer architectures while still providing meaningful reductions in computational cost.

In order to further prove the flexibility of TBKV with other state–of–the–art token reduction techniques applied to foreground tokens, we applied ToMe to foreground tokens. Results show a significant reduction of FLOPs on both models, highlighting the possibility

of using TBKV as a backbone for additional algorithms. At the same time, we can notice how unpredictable the behaviour of token reduction can be when applied to foreground tokens, since we see a –10% reduction in accuracy for ViViT model tested on Kinetics400 dataset. We demonstrate once more our decision to apply token efficiency methods on the background, since foreground tokens tend to have a much higher sensitivity to complex procedures. However, it still provides a design choice depending on the trade–off between accuracy and efficiency we want to apply to our specific implementation.

5.2.9 Visualizations

Figure 5.8 presents a qualitative example of the TBKV pipeline applied to a short video clip from the Kinetics–400 dataset labeled *training dog*. In this sequence, a dog runs and jumps across a training field, while the surrounding environment remains largely static. This type of scenario is representative of many real–world videos where a small dynamic foreground object moves within a mostly stable background. The first row of the figure shows four consecutive frames extracted from the video. The action of interest is centered around the dog, which changes position significantly across frames as it runs and jumps over a training obstacle. Despite this motion, the majority of the scene remains unchanged, including the grass field, distant vegetation, and the black borders at the top and bottom of the image. The second row visualizes the foreground/background separation produced by the saliency extraction module. Regions highlighted in red correspond to foreground tokens that exhibit strong temporal variation or receive higher attention from the transformer model. These regions primarily correspond to the dog and the nearby area involved in the action. Regions highlighted in green correspond instead to background tokens that remain relatively stable across frames. Several observations can be made from this visualization. First, the moving dog is consistently identified as foreground across all frames, indicating that the saliency mechanism correctly captures the dynamic regions associated with the action class. Second, large static areas such as the grass field are correctly classified as background. Even regions that contain some visual structure, such as the fence visible in the third frame, are still classified as background because they remain stationary relative to the camera. This demonstrates that the method does not rely purely on texture complexity but rather on temporal stability. Additionally, the black borders present at the top and bottom of the frames are correctly detected as background tokens, highlighting the robustness of the saliency estimation even in low–information regions. The third row illustrates the merging process applied to background tokens during the cache warm-up phase. Each colored rectangle represents a group of tokens that have been merged together through the bipartite soft matching algorithm. Since background regions tend to exhibit strong spatial and temporal redundancy, many tokens can be safely aggregated into a smaller set of representative tokens. This effect is particularly visible in the grass field, where large areas are merged into relatively few token groups. In contrast, areas closer to the moving dog show a finer token partitioning, reflecting higher visual variability. This example highlights a key motivation behind the TBKV approach. In many video understanding tasks, the semantic information required to recognize the action is concentrated in a relatively small foreground region, while the majority of the frame contains redundant background content. By identifying these stable

regions and compressing their representations through token merging, the algorithm can construct a compact cache of background key–value representations that can be reused across subsequent frames. As a result, the model avoids recomputing large portions of the transformer representation while still preserving full fidelity for the dynamic foreground tokens that are most relevant to the action being recognized.

Figure 5.8. Qualitative visualization of the TBKV pipeline on a video from the Kinetics-400 dataset labeled *training dog*. The first row shows consecutive frames from the video sequence. The second row illustrates the saliency–based foreground/background separation, where foreground tokens are highlighted in red and background tokens in green. Dynamic regions corresponding to the dog and nearby interaction areas are correctly identified as foreground, while static regions such as the grass field, image borders, and distant scene elements are classified as background. The third row visualizes the merging of background tokens during the cache warm-up phase. Colored rectangles represent groups of tokens that are merged using bipartite soft matching, forming compact background representations that are stored in the cache. Large uniform regions are aggressively merged due to their high redundancy, while areas near the moving subject preserve finer token granularity.

5.3 Discussion

The results presented in this thesis show that the proposed Temporal Background Key–Value Reuse (TBKV) mechanism effectively reduces computational costs in Video Vision Transformers while maintaining competitive action recognition accuracy. Across both Kinetics–400 and UCF101 datasets, TBKV demonstrates consistent FLOPs reduction,

often between 40% and 85%, depending on the backbone size without causing significant drops in Top–1 or Top–5 accuracy. These findings confirm the initial hypothesis that selectively merging redundant background tokens and reusing their key–value representations can improve efficiency without degrading performance. A major advantage of TBKV lies in its focus on background tokens. Unlike ToMe, which merges tokens indiscriminately, TBKV preserves foreground tokens, which are typically the most informative for recognizing actions. This selective approach results in a more favorable trade–off between accuracy and computational cost, as shown in the hybrid experiments where TBKV was combined with ToMe on foreground tokens. The hybrid configuration further highlights TBKV's flexibility: it can complement existing token reduction methods, offering a pathway for more advanced, combined strategies. Scalability analysis also supports the method's effectiveness for larger architectures. As transformer backbones grow in size, the amount of redundant background information naturally increases. TBKV leverages this redundancy, producing more significant efficiency gains for high–capacity models. This property makes the method particularly relevant for practical applications, such as deploying large video models on GPUs with limited memory or in edge devices where computational resources are constrained. Finally, memory analysis demonstrates that TBKV reduces the cache footprint while maintaining semantic fidelity in the merged key–value representations. Compared to naive caching, merged tokens allow stable performance across a range of cache sizes, highlighting the approach's suitability for both offline and near real–time video processing scenarios.

5.4 Limitations

While TBKV shows clear advantages, there are some limitations that future work should address. First, the method currently relies on a periodic warm–up mechanism where the first P frames of a video are used to populate the cache, and this warm–up is repeated at fixed intervals for longer sequences. The optimal frequency and length of these refresh periods remain unexplored. Future research could investigate adaptive or content–aware refresh strategies, where the model dynamically decides when to update the cache based on changes in the video, potentially further improving computational efficiency without sacrificing accuracy. Second, TBKV assumes a clear distinction between background and foreground tokens based on motion or semantic cues. In videos with highly dynamic backgrounds, subtle actions, or low–contrast movement, this separation may not perfectly capture the most relevant information, potentially leading to small accuracy drops. More advanced token selection strategies, possibly guided by attention maps or temporal saliency, could help mitigate these issues. Third, the experiments in this thesis focus primarily on offline evaluation with fixed–length video clips. Extending TBKV to real–time streaming or variable–length sequences introduces additional challenges related to cache management, memory bandwidth, and latency, which were not addressed in this study. Handling these scenarios would require further experimentation and potentially algorithmic adjustments to the current approach.

Chapter 6

Conclusion

This thesis presents TBKV, a temporal token reduction mechanism that improves the efficiency of Video Vision Transformers by reusing the key-value representations of background tokens across frames. The method addresses a major inefficiency in video transformers: repeated computation of largely redundant representations in static background regions. Experimental results demonstrate that TBKV significantly reduces computational costs up to 85% in some configurations while maintaining high classification accuracy on Kinetics-400 and UCF101 datasets. TBKV scales effectively to larger transformer backbones, integrates seamlessly with other token-reduction methods such as ToMe, and reduces memory requirements through compact token merging. While the method shows strong efficiency and scalability, some limitations remain, including the fixed periodic warm-up mechanism, potential challenges in separating foreground and background tokens in complex videos, and applicability to real-time streaming scenarios. These areas provide clear directions for future work. Overall, TBKV offers a promising approach for improving the computational efficiency of video understanding models. By selectively reducing redundant computations, it enables high-capacity video transformers to operate more efficiently without compromising accuracy, providing a foundation for practical deployment in resource-constrained environments.

Bibliography

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Aurelien Lucchi, Thomas Cohen, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 813–824, 2021.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, and Judy Hoffman. Token merging: Your vit but faster. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[4] Daniel Bolya, Cheng-Yang Zhou, Fanyi Xiao, and Bharath Hariharan. Token merging: Your vit but faster. In *Advances in Neural Information Processing Systems*, 2022.

[5] Lukas Cavigelli, Patrick Degen, and Luca Benini. Cbinfer: Change-based inference for convolutional neural networks on video data. In *Proceedings of the 11th International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–8, 2017.

[6] Pengtao Chen, Xianfang Zeng, Maosen Zhao, Peng Ye, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers. *arXiv preprint arXiv:2506.03065*, 2025.

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] Krzysztof M. Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlós, Peter Hawkins, Jared Q. Davis, Afroz Mohiuddin, Lukasz Kaiser, David B. Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[9] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxin Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[10] Hangliang Ding, Dacheng Li, Runlong Su, Peiyuan Zhang, Zhijie Deng, Ion Stoica, and Hao Zhang. Efficient-vdit: Efficient video diffusion transformers with attention tile. *arXiv preprint arXiv:2502.06155*, 2025.

[11] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, and Baining Chen. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 12124–12134, 2022.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6824–6835, 2021.

[14] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: A vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12259–12269, 2021.

[15] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. *arXiv preprint arXiv:1902.09113*, 2019.

[16] Amir Habibian, Davide Abati, Taco S. Cohen, and Babak E. Bejnordi. Skip-convolutions for efficient video processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2695–2704, 2021.

[17] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. In *Advances in Neural Information Processing Systems*, volume 34, pages 9895–9907, 2021.

[18] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165, 2020.

[19] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Apostol Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[20] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[21] Chen Li, Rui Zhao, Marco Fernandez, Harsh Gupta, and Chen Tang. Cst-vit: Cascaded spatio-temporal gating for efficient video transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15032–15042, 2025.

[22] Liyuan Li, Bowen Yu, Shihao Ji, and Wei Chu. Fast transformer decoding: One write-head is all you need. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4842–4847. Association for Computational Linguistics, 2018.

[23] Yifan Li, Tianyi Chen, Qingnan Fan, Yu Wang, and Yu Qiao. Vidtome: Video token merging for efficient video transformers. *arXiv preprint arXiv:2501.01916*, 2025.

[24] Yifan Li, Qingnan Fan, Tianyi Chen, Yu Wang, and Yu Qiao. Tome-v: Efficient

video transformers via token merging. *arXiv preprint arXiv:2401.03444*, 2024.

[25] Yuhui Li et al. Tofu: Token fusion for efficient vision transformers. *arXiv preprint arXiv:2403.07944*, 2024.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[27] Zicheng Liu, Wenhai Wang, Weisheng Dong, Xiangyang Zhou, and Yu Qiao. Vidtldr: Spatio-temporal token pruning for efficient video transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1734–1745, 2023.

[28] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer with linear complexity. In *Advances in Neural Information Processing Systems*, volume 34, pages 21297–21309, 2021.

[29] Elena Martinez, Trevor Smith, Yifan Huang, and Wei Zhou. Ep-vit: Video efficiency through codec-assisted token prediction. *arXiv preprint arXiv:2507.04567*, 2025.

[30] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5137–5147, 2022.

[31] Michael Parger, Chen Tang, Christopher D. Twigg, Cem Keskin, Ruofei Wang, and Markus Steinberger. Deltacnn: End-to-end cnn inference of sparse frame differences in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12497–12506, 2022.

[32] Sungho Park, Jessica Wu, and R. Venkatesh Babu. Eventful transformers: Exploiting temporal redundancy in video via attention sparsity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11340–11349, 2025.

[33] Yongming Rao, Wenliang Zhao, Bo Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[34] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[35] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

[36] Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[37] Mattia Soldan, Fabian Caba Heilbron, Bernard Ghanem, Josef Sivic, and Bryan Russell. Residualvit for efficient temporally dense video encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22305–22315, October 2025.

[38] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[39] X. Sun et al. Drvit: A dynamic redundancy-aware vision transformer accelerator via algorithm and architecture co-design on fpga. *Journal of Parallel and Distributed Computing*, 199:105042, 2025.

[40] Xiaolong Tong, Xiyang Gong, Di Xu, Jianfei Li, Chunhua Sun, and Cewu Lu. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15990, 2022.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[42] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[43] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, 2021.

[44] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[45] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31, 2021.

[46] Yuxin Wu, Xin Chen, Yi Li, Ze Liu, and Lu Yuan. Memvit: Video transformer with memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11765–11774, 2022.

[47] Mang Ye, Shuyuan Zhang, Yifan Liu, Yufei Wang, and Ming-Hsuan Yang. Adafocusv2: End-to-end training of spatial dynamic networks for video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[48] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, and Jan Kautz. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10809–10818, 2022.

[49] Qiang Zhang et al. Prune and merge: Efficient token reduction for vision transformers. *arXiv preprint arXiv:2402.05164*, 2024.

[50] Yi Zhang, Anil Kumar, Joon Lee, Luca Rossi, and Minh Nguyen. Stgt: Spatio-temporal gate transformer for efficient video inference. *arXiv preprint arXiv:2503.01234*, 2025.

[51] X. Zou et al. Revit: Vision transformer accelerator with reconfigurable semantic-aware differential attention. *IEEE Transactions on Computers*, 74(3):1079–1093,

2025.