Eventful Transformers: Leveraging Temporal Redundancy in Vision Transformers

Matthew Dutson, Yin Li, and Mohit Gupta
University of Wisconsin–Madison
{dutson,yin.li,mgupta37}@wisc.edu

Abstract

Vision Transformers achieve impressive accuracy across a range of visual recognition tasks. Unfortunately, their accuracy frequently comes with high computational costs. This is a particular issue in video recognition, where models are often applied repeatedly across frames or temporal chunks. In this work, we exploit temporal redundancy between subsequent inputs to reduce the cost of Transformers for video processing. We describe a method for identifying and re-processing only those tokens that have changed significantly over time. Our proposed family of models, Eventful Transformers, can be converted from existing Transformers (often without any re-training) and give adaptive control over the compute cost at runtime. We evaluate our method on large-scale datasets for video object detection (ImageNet VID) and action recognition (EPIC-Kitchens 100). Our approach leads to significant computational savings (on the order of 2-4x) with only minor reductions in accuracy.

1. Introduction

Transformers, initially designed for language modeling [60], have been recently explored as an architecture for vision tasks. Vision Transformers [16] have achieved impressive accuracy across a range of visual recognition problems, attaining state-of-the-art performance in tasks including image classification [16], video classification [1, 2, 18], and object detection [8, 38, 42, 64].

One of the primary drawbacks of vision Transformers is their high computational cost. Whereas typical convolutional networks (CNNs) consume tens of GFlops per image [7], vision Transformers often require an order of magnitude more computation, up to hundreds of GFlops per image. In video processing, the large volume of data further amplifies these costs. High compute costs preclude vision Transformers from deployment on resource-constrained or latency-critical devices, limiting the scope of this otherwise

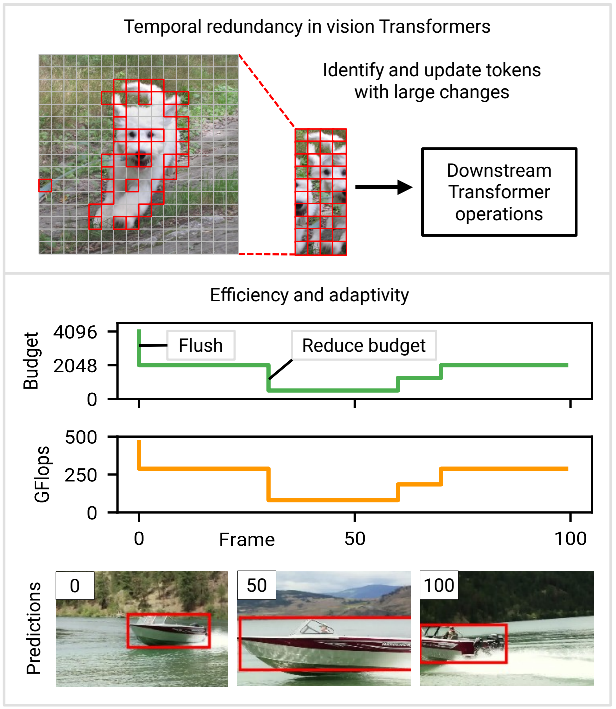
Temporal redundancy in vision Transformers
Identify and update tokens with large changes
Downstream Transformer operations
Efficiency and adaptivity
Budget
4096
2048
0
Flush
Reduce budget
GFlops
500
250
0
0
Frame
50
100
Predictions
0
50
100

Figure 1. **Eventful Transformers.** Our method exploits temporal redundancy between subsequent model inputs. **(Top)** Within each Transformer block, we identify and update only those tokens with significant changes over time. Image: [5]. **(Bottom)** In addition to improving efficiency, our method gives fine-grained control over the compute cost at runtime. "Budget" refers to parameter r as described in Section 4.3. "Flush" refers to the initialization of all tokens on the first time step. This example shows the ViTDet [38] object detection model on a video from the VID [57] dataset.

exciting technology. In this paper, we present one of the first methods to use *temporal redundancy between subsequent inputs* to reduce the cost of vision Transformers when applied to video data.

Temporal redundancy. Consider a vision Transformer that is applied frame-by-frame or clip-by-clip to a video sequence. This Transformer might be a simple frame-wise model (*e.g.*, an object detector) or an intermediate step in some spatiotemporal model (*e.g.*, the first stage of the factorized model from [1]). Unlike in language processing, where one Transformer input represents a complete sequence, we consider Transformers applied to several distinct inputs (frames or clips) over time.

Natural videos contain significant temporal redundancy, with only slight differences between subsequent frames. Despite this fact, deep networks (including Transformers) are commonly computed "from scratch" on each frame. This approach is wasteful, discarding all potentially relevant information from previous inferences. Our key intuition is that we can reuse intermediate computations from earlier time steps to improve efficiency on redundant sequences.

Adaptive inference. For vision Transformers (and deep networks in general), the inference cost is typically fixed by the architecture. However, in real-world applications, the available resources may vary over time (*e.g.*, due to competing processes or variations in power supply). As such, there is a need for models whose computational cost can be modified at runtime [48]. In this work, adaptivity is one of our primary design objectives; we design our method to allow real-time control over the compute cost. See Figure 1 (bottom portion) for an example where we vary the compute budget throughout a video.

Challenges and opportunities. There are past works exploring temporal redundancy [17, 23, 51] and adaptivity [47, 61, 69] for CNNs. However, these methods are generally incompatible with vision Transformers, owing to substantial architectural differences between Transformers and CNNs. Specifically, Transformers introduce a new primitive, self-attention, that does not conform to the assumptions of many CNN-based methods.

Despite this challenge, vision Transformers also represent a unique opportunity. In CNNs, it is difficult to translate sparsity improvements (*i.e.*, the sparsity gained by considering temporal redundancy) into concrete speedups. Doing so requires imposing significant restrictions on the sparsity structure [23] or using custom compute kernels [51]. In contrast, the structure of Transformer operations (centered on manipulating token vectors) makes it easier to translate sparsity into reduced runtime using standard operators.

Eventful Transformers. We propose Eventful Transformers, a new class of Transformer that leverages temporal redundancy between inputs to enable efficient, adaptive inference. The term "Eventful" is inspired by event cameras [4, 40], sensors that produce sparse outputs based on scene changes. Eventful Transformers track token-level changes over time, selectively updating the token representations and self-attention maps on each time step. Blocks in an Eventful Transformer include gating modules that allow controlling the number of updated tokens at runtime.

Our method can be applied to off-the-shelf models (generally without re-training) and is compatible with a wide range of video processing tasks. Our experiments demonstrate that Eventful Transformers, converted from existing state-of-the-art models, significantly reduce computational costs while largely preserving the original model's accuracy. We publicly release our code, which includes PyTorch modules for building Eventful Transformers. See our project page: wisionlab.com/project/eventful-transformers.

Limitations. We demonstrate wall-time speedups on both the CPU and GPU. However, our implementation (based on vanilla PyTorch operators) is likely sub-optimal from an engineering standpoint. With additional effort to reduce overhead (*e.g.*, implementing a fused CUDA kernel for our gating logic), we are confident that the speedup ratios could be further improved. Our method also involves some unavoidable memory overheads. Perhaps unsurprisingly, reusing computation from previous time steps requires maintaining some tensors in memory. These memory overheads are relatively modest; see Section 6 for further discussion.

2. Related Work

Efficient Transformers. Several past works improve the efficiency of Transformers. Many of these methods focus on reducing the quadratic complexity of self-attention, often using low-rank or sparse approximations [11, 13, 22, 31, 32, 34, 44, 55, 56, 63]. In this work, we consider standard self-attention (with windowing in some cases). Our approach is orthogonal to the above methods.

Selecting and summarizing tokens. Some recent works improve the efficiency of vision Transformers by exploiting spatial redundancy within each input. Many of these methods prune or fuse tokens based on a salience measure [19, 21, 39, 49, 52]. A notable example is the Adaptive Token Sampling (ATS) algorithm [19], which has an adaptive computation cost and does not require re-training. Other spatial redundancy methods include adaptive token pooling [3, 45], hierarchical pooling [50], learned tokenization [58], and progressive token sampling [72].

Unlike these works, which consider *spatial* redundancy, our method targets *temporal* redundancy. This makes our work complementary to these approaches. A single model can leverage both spatial and temporal redundancy by only updating tokens that are both salient *and* not temporally repetitive. We illustrate this compatibility in our experiments by building a simple proof-of-concept model that combines temporal and spatial redundancy.

Another related work is Spatiotemporal Token Selection (STTS) [62], which exploits spatiotemporal redundancy for video inference. STTS is intended for models with explicit temporal reasoning that take an entire video as input. In contrast, our method is designed for models that are repetitively applied to frames or clips. Compared to STTS, our method covers a wider range of architectures and tasks.

Temporal redundancy between inputs. There has been recent work on exploiting inter-frame temporal redundancy in CNNs [9, 17, 23, 51]. While we draw some inspiration from these methods, directly applying them to vision Transformers is not feasible due to significant architectural differences between CNNs and Transformers.

There is limited existing research on exploiting temporal redundancy between subsequent vision Transformers inputs. To our knowledge, the only past work in this area is the Spatiotemporal Gated Transformers (STGT) method [37]. There are two noteworthy differences between STGT and our work. Most notably, STGT only considers temporal redundancy within token-level operations (*e.g.*, token-wise linear transforms), and not within the self-attention operator. Our method accelerates all major Transformer components, including self-attention. Further, STGT uses lossy gating logic that leads to accuracy degradation on long sequences with gradual changes. Our method avoids this issue by employing an improved, reference-based gating mechanism.

Adaptive neural networks. Many existing methods add adaptivity to deep CNNs [12, 20, 28, 47, 59, 61, 65, 67, 69, 70]. However, due to architectural differences (*e.g.*, the use of relative position embeddings in Transformers), these methods (*e.g.*, those based on input resizing) often do not translate to vision Transformers.

There has been some recent work on adaptive vision Transformers [46, 66, 71]. These works leverage redundancy within a single input, whereas we consider redundancy between inputs. Unlike our method, these approaches generally require re-training or fine-tuning the model.

Efficient neural networks. There is a substantial body of work on improving the efficiency of deep networks. Some works propose efficient CNN architectures [27, 30, 73]. Others use reduced-precision arithmetic [14, 29, 53] or pruning [24, 25, 35, 36]. Our method is loosely connected to pruning; it can be viewed as adaptively pruning redundant tokens on each time step.

3. Background: Vision Transformers

In this section, we describe the basic elements of a vision Transformer (see [16] for more details) and define the notation we use throughout the rest of the paper.

A vision Transformer consists of a sequence of Transformer blocks. The input to each block is a list of N, D-dimensional token vectors; we denote this as $\boldsymbol{x} \in \mathbb{R}^{N \times D}$. Before the first Transformer block, a vision Transformer maps each image patch to a token vector using a linear transform. Positional embedding [60] can be injected before the first block [16] or at every block [38].

A Transformer block. A Transformer block maps input $\boldsymbol{x} \in \mathbb{R}^{N \times D}$ to output $\boldsymbol{z} \in \mathbb{R}^{N \times D}$, according to

$$\boldsymbol{y} = \text{MSA}(\text{LN}(\boldsymbol{x})) + \boldsymbol{x}, \quad (1)$$
$$\boldsymbol{z} = \text{MLP}(\text{LN}(\boldsymbol{y})) + \boldsymbol{y}, \quad (2)$$

where "MSA" denotes multi-headed self-attention. "MLP" is a token-wise multilayer perceptron with two layers and one GELU nonlinearity. "LN" denotes layer normalization.

Multi-headed self-attention (MSA). The self-attention operator first applies three linear transforms $W_q, W_k, W_v \in \mathbb{R}^{D \times D}$ to its input $\boldsymbol{x}' = \text{LN}(\boldsymbol{x})$.

$$\boldsymbol{q} = \boldsymbol{x}' W_q \qquad \boldsymbol{k} = \boldsymbol{x}' W_k \qquad \boldsymbol{v} = \boldsymbol{x}' W_v. \quad (3)$$

$\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$ are the "query," "key," and "value" tensors, respectively. In practice, W_q, W_k, W_v are often fused into a single transform $W_{qkv} = [W_q, W_k, W_v]$. These transforms may include a bias; we omit the bias here for brevity.

The self-attention operator then computes a normalized similarity matrix (attention matrix) $A \in \mathbb{R}^{N \times N}$ between the tokens of $\boldsymbol{q}$ and $\boldsymbol{k}$.

$$A = \text{Softmax}\left(\boldsymbol{q}\boldsymbol{k}^T / \sqrt{D}\right). \quad (4)$$

Softmax normalization is applied along rows of the matrix.

The MSA output $\boldsymbol{y}'$ is an attention-weighted sum of the value tokens $\boldsymbol{v}$, followed by a linear projection W_p.

$$\boldsymbol{y}' = (A\boldsymbol{v})\, W_p. \quad (5)$$

Multi-headed self-attention (as opposed to single-headed self-attention) splits $\boldsymbol{q}$, $\boldsymbol{k}$, and $\boldsymbol{v}$ into H tensors of shape $\mathbb{R}^{N \times (D/H)}$ and applies self-attention in parallel across these H heads. Before applying W_p, the results of all heads are concatenated into a tensor with shape $\mathbb{R}^{N \times D}$.

Windowed attention. Standard MSA has a complexity of $\mathcal{O}(N^2)$ (quadratic in the number of tokens). To reduce this cost, many vision Transformers adopt *windowed* attention. Windowed attention constrains the attention computation to local windows. Information can be exchanged between windows by shifting the windows between blocks [42] or by interleaving global attention [38].

4. Eventful Transformers

Our goal is to accelerate vision Transformers for video recognition, in the situation where a Transformer is applied repetitively across frames or chunks of frames (*e.g.*, for video object detection or video action recognition, respectively). Our key idea is to exploit temporal redundancy by re-using computation from previous time steps. In this section, we describe how to modify Transformer blocks to add temporal redundancy awareness.

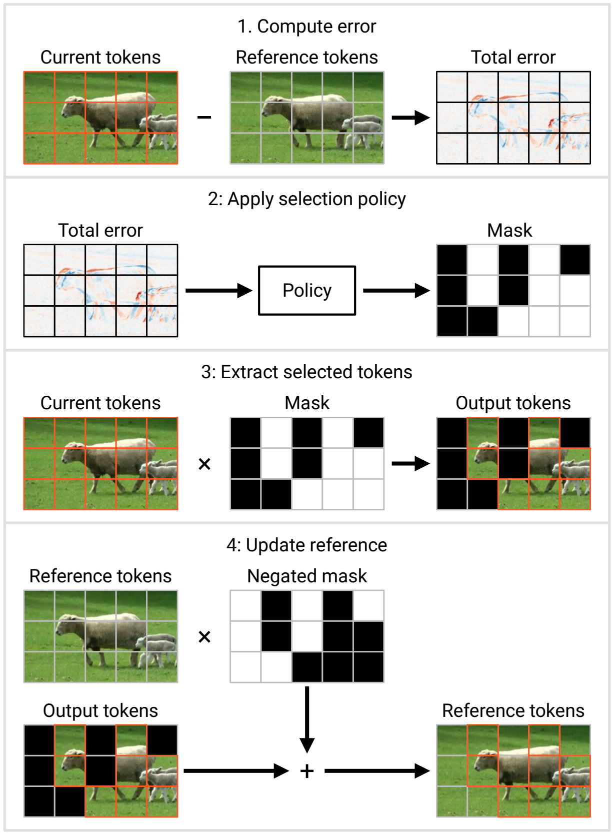

Figure 2. **Token Gating.** A gating module compares incoming tokens against a stored reference. If the difference between a token and its reference is large, then the token is selected to be updated. See Section 4.3 for details on selection policies. Images are from the VID [57] dataset.

In Section 4.1, we present a token-gating module that monitors temporal changes and determines which tokens to update. In Section 4.2, we integrate our token gating logic into a Transformer block, creating a redundancy-aware Eventful Transformer block. In Section 4.3, we explore policies for selecting which tokens to update.

4.1. Token Gating: Detecting Redundancy

In this subsection, we propose two modules: token gates and token buffers. These modules allow us to identify and update only those tokens that have changed significantly since their last update.

Gate module. A gate selects $M \leq N$ of its input tokens to send to downstream layers for re-computation. The gate maintains a set of *reference tokens* in memory, which we denote as $\boldsymbol{u} \in \mathbb{R}^{N \times D}$. The reference tensor contains the value of each token on the time step it was most recently updated. On each time step, tokens are compared against their references; those that deviate significantly from their reference are selected for an update.

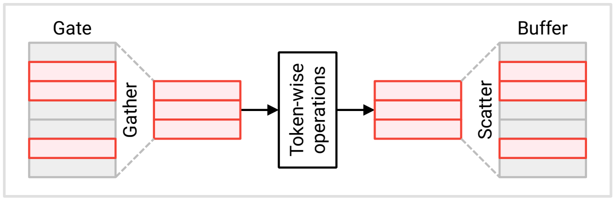

Figure 3. **Accelerating token-wise operations.** The gate reduces the number of active tokens from N to M. Subsequent token-wise operations operate on a smaller tensor and therefore have a lower computational cost (proportional to M).

Let $\boldsymbol{c} \in \mathbb{R}^{N \times D}$ denote the current input to the gate. On each time step, we update the gate's state and determine its output according to the following procedure (see Figure 2):

1. Compute the total error $\boldsymbol{e} = \boldsymbol{u} - \boldsymbol{c}$.
2. Apply a *selection policy* to the error $\boldsymbol{e}$. A selection policy returns a binary mask $\boldsymbol{m}$ (equivalently, a list of token indices) indicating which M tokens should be updated.
3. Extract the tokens selected by the policy. In Figure 2, we depict this as the product $\boldsymbol{c} \times \boldsymbol{m}$; in practice, we achieve this with a "gather" operation along the first axis of $\boldsymbol{c}$. We denote the gathered tokens as $\tilde{\boldsymbol{c}} \in \mathbb{R}^{M \times D}$. The gate returns $\tilde{\boldsymbol{c}}$ as its output.
4. Update the references for selected tokens. In Figure 2, we depict this as $\boldsymbol{u} \leftarrow \boldsymbol{e} \times (\sim \boldsymbol{m}) + \boldsymbol{c} \times \boldsymbol{m}$; in practice, we apply a "scatter" operation from $\tilde{\boldsymbol{c}}$ into $\boldsymbol{u}$.

On the first time step, the gate updates all tokens (initializing $\boldsymbol{u} \leftarrow \boldsymbol{c}$ and returning $\tilde{\boldsymbol{c}} = \boldsymbol{c}$).

Buffer module. A buffer module maintains a state tensor $\boldsymbol{b} \in \mathbb{R}^{N \times D}$ that tracks the most recent known value for each of its input tokens. When receiving a new input $f(\tilde{\boldsymbol{c}}) \in \mathbb{R}^{M \times D}$, the buffer scatters the tokens from $f(\tilde{\boldsymbol{c}})$ into their corresponding locations in $\boldsymbol{b}$. It then returns the updated $\boldsymbol{b}$ as its output. See Figure 3.

We pair each gate with a subsequent buffer. One simple usage pattern is as follows. The gate output $\tilde{\boldsymbol{c}} \in \mathbb{R}^{M \times D}$ is passed to a series of token-wise operations $f(\tilde{\boldsymbol{c}})$. The resulting tensor $f(\tilde{\boldsymbol{c}}) \in \mathbb{R}^{M \times D}$ is then passed to a buffer, which restores the full shape $\mathbb{R}^{N \times D}$.

4.2. Building Redundancy-Aware Transformers

In this subsection, we propose a modified Transformer block that exploits temporal redundancy. Figure 4 shows our design for an Eventful Transformer block. Our method accelerates token-wise operations (*e.g.*, the MLP), as well as the query-key and attention-value multiplications (Equations 4 and 5, respectively).

Token-wise operations. Many of the operations in a Transformer block are token-wise, meaning they do not involve information exchange between tokens. These include the

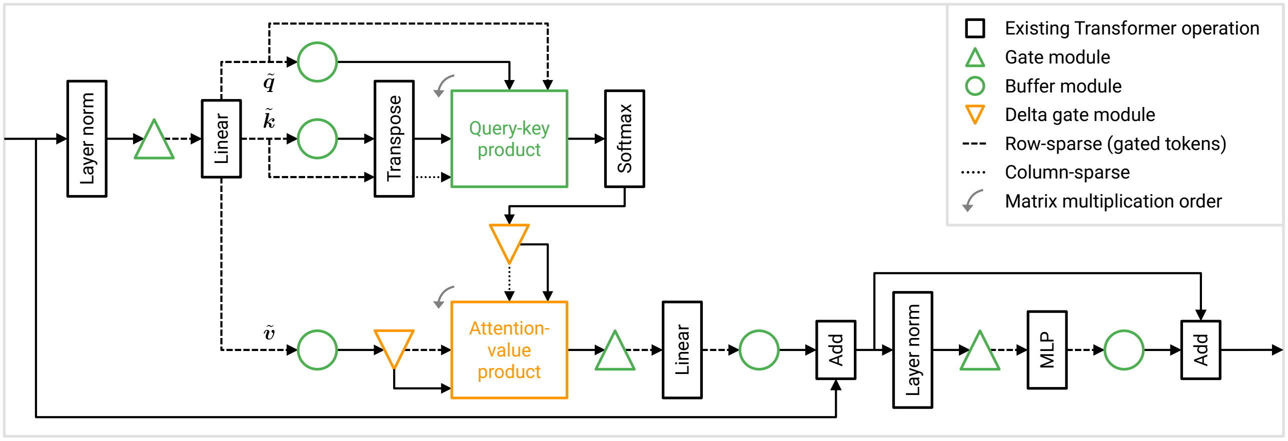

Figure 4. **An Eventful Transformer block.** To exploit temporal redundancy, we strategically apply token gating throughout the block and compute a modified, sparse self-attention update. Rectangles are standard Transformer components (see Section 3). For clarity, we have omitted some minor operations (*e.g.*, scaling after the first matrix multiplication) from this figure.

MLP and the linear transforms in the MSA. We can save computation in token-wise operations by skipping those tokens not selected by the gate. Due to token-wise independence, this does not change the result of the operation for the selected tokens. See Figure 3.

Specifically, we place a gate-buffer pair around each contiguous sequence of token-wise operations, including the W_{qkv} transform (Equation 3), the W_p transform (Equation 5), and the MLP. Note that we add buffers before the skip connections (Equations 1 and 2) to ensure that the tokens of the two addition operands are correctly aligned.

The cost of a token-wise operation is proportional to the number of tokens. A gate reduces the number of tokens from N to M. This, in turn, reduces the computational cost of downstream token-wise operations by a factor of N/M.

The query-key product. We now consider the query-key product $B = \boldsymbol{q}\boldsymbol{k}^T$ (part of Equation 4). Writing this matrix multiplication explicitly, we have

$$B_{ij} = \sum_p \boldsymbol{q}_{ip} \left(\boldsymbol{k}^T\right)_{pj}. \tag{6}$$

Element B_{ij} needs to be updated if either (a) there is a change in the i^{th} row of $\boldsymbol{q}$, or (b) there is a change in the j^{th} column of $\boldsymbol{k}^T$. Due to the gate that we inserted before the W_{qkv} transform (shown in Figure 4), only some rows of $\boldsymbol{q}$ and some columns of $\boldsymbol{k}^T$ have changed. Therefore, we only need to recompute a subset of the elements of B.

Let $\tilde{\boldsymbol{x}}' \in \mathbb{R}^{M \times D}$ denote the output of the gate before the W_{qkv} transform. We define $\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{x}}' W_q$ and $\tilde{\boldsymbol{k}} = \tilde{\boldsymbol{x}}' W_k$ (following Equation 3). Let $\boldsymbol{q}$ and $\boldsymbol{k}$ denote the outputs of the $\tilde{\boldsymbol{q}}$ and $\tilde{\boldsymbol{k}}$ buffers (shown in Figure 4). $\tilde{\boldsymbol{q}}$ contain $\tilde{\boldsymbol{k}}$ the subset of tokens from $\boldsymbol{q}$ and $\boldsymbol{k}$ that are being updated.

Figure 5 depicts our method for sparsely updating B. The product $\tilde{\boldsymbol{q}}\boldsymbol{k}^T$ contains the elements of B that need to be updated due to a change in $\tilde{\boldsymbol{q}}$. We compute $\tilde{\boldsymbol{q}}\boldsymbol{k}^T$, then scatter the result row-wise into the old B (the value of B from the last time step). We use an analogous approach for the $\tilde{\boldsymbol{k}}$-induced updates; we compute $\boldsymbol{q}\tilde{\boldsymbol{k}}^T$ and scatter the result column-wise into B.

The overall cost of these updates is $2NMD$, compared to a cost of N^2D to compute B from scratch. Note that the cost of our method is proportional to M, the number of tokens selected by the gate. We save computation when $M < N/2$ (when we update fewer than half of the tokens).

The above method for updating B involves some redundant computation. Some elements of the first scattered matrix $\tilde{\boldsymbol{q}}\boldsymbol{k}^T$ are also present in the second matrix $\boldsymbol{q}\tilde{\boldsymbol{k}}^T$. These overlapping elements are computed twice. Eliminating this redundancy would reduce the cost of our method to $NMD \leq N^2D$. This could be achieved by removing the tokens in $\tilde{\boldsymbol{q}}$ from $\boldsymbol{q}$ before computing $\boldsymbol{q}\tilde{\boldsymbol{k}}^T$. We would then scatter the result by indexing along both axes of B. We leave this as an optimization for future implementations.

The attention-value product. We now describe a method for updating the attention-value product $A\boldsymbol{v}$ (part of Equation 5). Writing this multiplication explicitly, we have

$$(A\boldsymbol{v})_{ij} = \sum_p A_{ip}\boldsymbol{v}_{pj}. \tag{7}$$

Because of the gate before the W_{qkv} transform, only some rows (tokens) of $\boldsymbol{v}$ change on each time step. However, there are some updated values in every *column* of $\boldsymbol{v}$. Therefore, every element of $A\boldsymbol{v}$ will change on each time step. This means we cannot use the same strategy that we used for B, where we only updated some of the output elements.

Instead, we propose a delta-based update strategy. Let A_o and $\boldsymbol{v}_o$ denote the last known values for A and $\boldsymbol{v}$. Let $A_\triangle$ and $\boldsymbol{v}_\triangle$ denote changes in A and $\boldsymbol{v}$. Define $A_n =$

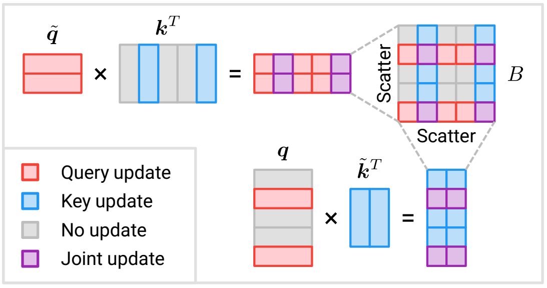

Figure 5. **The query-key product.** We reduce the cost of computing $B = qk^T$ by only updating a subset of its elements. We first compute changes induced by updated rows in q (top-left), then compute changes induced by updated columns in k^T (bottom).

$A_o + A_\Delta$ and $v_n = v_o + v_\Delta$. We can compute the updated attention-value product $A_n v_n$ as

$$\begin{aligned} A_n v_n &= (A_o + A_\Delta)(v_o + v_\Delta) \\ &= A_o v_o + A_o v_\Delta + A_\Delta v_o + A_\Delta v_\Delta \\ &= A_o v_o + (A_o + A_\Delta) v_\Delta + A_\Delta (v_o + v_\Delta) - A_\Delta v_\Delta \\ &= A_o v_o + A_n v_\Delta + A_\Delta v_n - A_\Delta v_\Delta. \end{aligned} \tag{8}$$

Therefore, on each time step, we can update Av by adding $A_n v_\Delta + A_\Delta v_n - A_\Delta v_\Delta$ to the previous result $A_o v_o$.

We obtain A_Δ, v_Δ, A_o, and v_o using *delta gate modules*. Delta gates are similar to the gates defined in Section 4.1, with one difference: instead of returning $\tilde{c}$, a delta gate returns u and $\tilde{e}$ (where $\tilde{e}$ is the result of gathering the selected indices from e). u represents the effective current value of the gate's output, corresponding to A_n or v_n in Equation 8. $\tilde{e}$ represents the amount of change on the current time step, corresponding to A_Δ or v_Δ in Equation 8.

Figure 6 illustrates our approach for efficiently computing the three delta terms in Equation 8. We remove the columns of A_n that correspond to zero rows in v_Δ (these columns will always be multiplied by zero). Let $\tilde{A}_n$ denote A_n with these columns removed. We remove rows of v_n analogously to produce $\tilde{v}_n$. We then compute

$$\tilde{A}_n \tilde{v}_\Delta + \tilde{A}_\Delta \tilde{v}_n - \tilde{A}_\Delta \tilde{v}_\Delta, \tag{9}$$

adding the result to the previous value of Av.

The product $\tilde{A}_\Delta \tilde{v}_\Delta$ assumes the columns of $\tilde{A}_\Delta$ are correctly aligned with the rows of $\tilde{v}_\Delta$. We achieve this alignment by forcing the A gate to select the same indices as the v gate. Using a separate policy in the A gate would be possible, but would require a re-alignment operation before computing $\tilde{A}_\Delta \tilde{v}_\Delta$. Further, forcing alignment allows us to eliminate a multiplication by rearranging Equation 9 as

$$\tilde{A}_n \tilde{v}_\Delta + \tilde{A}_\Delta (\tilde{v}_n - \tilde{v}_\Delta). \tag{10}$$

Equation 10 has a cost of $2MND$ (assuming the addition has a negligible cost), compared to $N^2 D$ for a standard multiplication. We see savings when $M < N/2$.

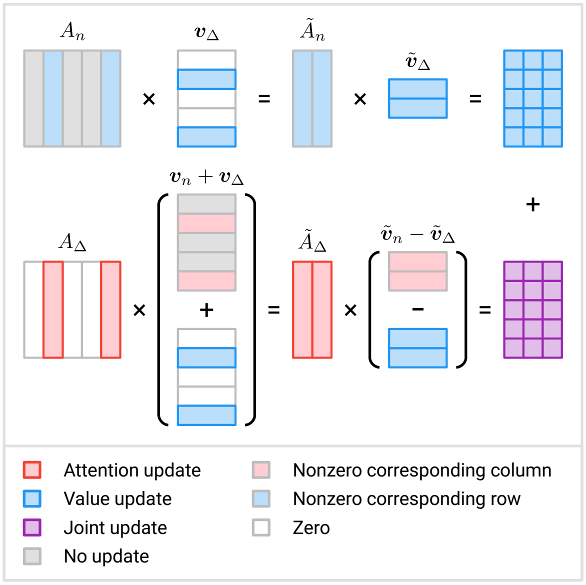

Figure 6. **The attention-value product.** We propose a delta-based strategy for sparsely updating the product Av. We reduce the cost of each sub-product by cutting rows and columns that do not contribute to the result (due to a zero multiplication).

4.3. Token Selection Policies

An important design choice for an Eventful Transformer is the token selection policy. Given a gate error tensor e, a policy generates a mask m indicating which tokens should be updated. We now discuss the design of selection policies.

Top-r policy. This policy selects the r tokens whose error e has the largest norm (we use the L2 norm). The top-r policy is lightweight and has a single parameter that can be easily tuned by hand. Varying r gives direct control over the model's computation cost. These properties make the top-r policy a good fit for applications with tight (potentially time-varying) computational constraints. We use a top-r policy in our main experiments.

Threshold policy. This policy selects all tokens where the norm of the error e exceeds a threshold h. A threshold policy is input-adaptive; the number of tokens selected depends on the amount of change in the scene. This input adaptivity can potentially lead to a better accuracy-cost tradeoff. However, the best value for the threshold h depends on the distribution of token vectors (which varies across layers) and is difficult to decide. In addition, a threshold policy does not give a fixed compute cost. This policy is likely better suited to applications with more flexible resources, where achieving the best possible accuracy-cost tradeoff is critical.

Other policies. More sophisticated token selection policies could lead to an improved accuracy-cost tradeoff. For example, we could use a learned policy (*e.g.*, a lightweight

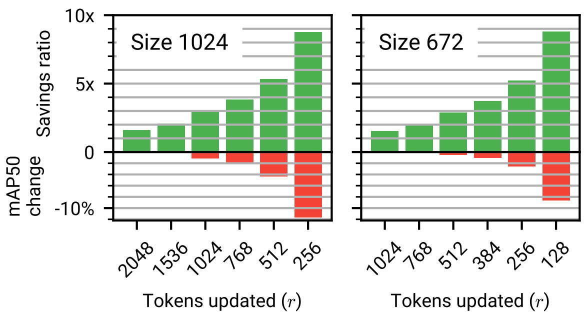

Figure 7. **Video object detection results.** Computation savings ratio (positive axis) and relative reductions in mAP50 score (negative axis) for our method. Results are for the ViTDet model [38] on the VID [57] dataset. See the supplement for tables.

policy network). However, training the policy's decision mechanism might be challenging, due to the general non-differentiability of the binary mask m. Another idea is to use an importance score (*e.g.*, as proposed in [19]) to inform the selection. We leave these ideas as potential topics for future work.

5. Experiments

In this section, we present our experiments and results. We evaluate our method for video object detection (Section 5.1) and video action recognition (Section 5.2). We show additional analysis in Section 5.3.

5.1. Video Object Detection

Task and dataset. We test our method on video object detection using the ILSVRC 2015 ImageNet VID dataset [57]. We report results on the validation set, which contains 555 videos with lengths of up to 2895 frames. Following prior works [10, 54], we evaluate the mean average precision (mAP) metric with an IoU threshold of 0.5.

Implementation details. We consider the ViTDet model from [38], which we apply to individual frames of an input video. ViTDet combines a plain Transformer backbone (based on ViT-B [16]) with a standard detection head [6, 26]. The backbone consists of 12 blocks with interleaved global and windowed self-attention (blocks 3, 6, 9, and 12 use global attention). Windowed self-attention uses a window size of 14×14 tokens (224×224 pixels). Token vectors are 768-dimensional. Self-attention operators have 12 heads and employ learned relative position embeddings.

Before the backbone, the model maps each 16×16 image patch to a token vector using a linear transform. The model expects fixed-size inputs (due to resolution-specific position embeddings). Therefore, following from [38], we rescale and pad all video frames to a uniform size (*e.g.*, 1024×1024) before applying the model.

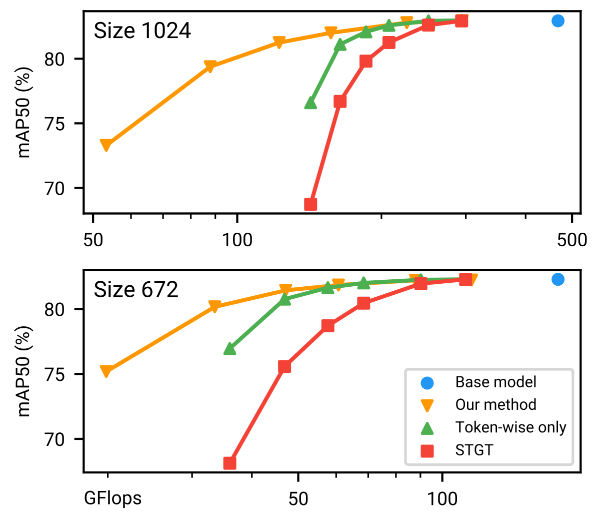

Figure 8. **Video object detection comparison and ablation.** The accuracy-cost tradeoff for our method, compared with STGT [37] and an ablation that only accelerates token-wise operations. See the supplement for tables.

We convert the model to an Eventful Transformer following the method in Section 4. In blocks that use windowed attention, we exploit temporal redundancy only within token-wise operations (not within the query-key or attention-value products). Our complete approach is compatible with windowed attention; however, windowing leads to some implementation challenges (ragged tensor shapes across windows, making batched computation more difficult). Note that for ViTDet, global self-attention represents the bulk of the self-attention compute cost.

Experiment protocol and baselines. We fine-tune the original ViTDet weights (trained on COCO) for VID object detection. See the supplement for training parameters. Note that we fine-tune *before* we add temporal redundancy awareness to the model. We train and evaluate at resolution 1024×1024. To understand the effect of token count (which strongly influences compute cost), we also evaluate at resolution 672×672. Rather than training a separate lower-resolution model, we adapt the 1024×1024 model by interpolating the learned position embeddings. The resulting adapted model retains most of its accuracy.

We compare against a version of the STGT method [37]. Due to unavailable source code, we were unable to evaluate all components of this method (notably, the use of a learned policy network). Instead, we consider a simplified version that uses the same top-r policy as our method. This setup enables a direct comparison of the core gating and update mechanisms. In addition, we evaluate an ablated version of our approach that only accelerates token-wise operations. We vary the policy r to explore the accuracy-compute trade-

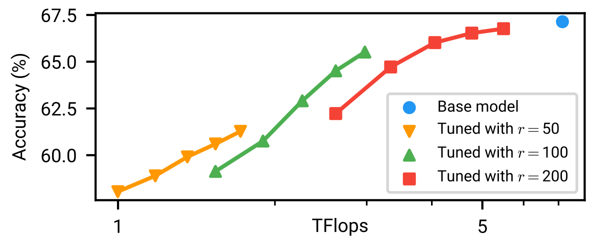

Figure 9. **Video action recognition results.** Our results for action recognition on EPIC-Kitchens 100 using the ViViT model [1]. We report the total TFlops per video (spatial + temporal sub-models). See the supplement for a table containing this data.

off. At resolution 1024, we test $r = 256, 512, 768, 1024,$ 1536, and 2048 (from a maximum of 4096 tokens). At resolution 672, we test $r = 128, 256, 384, 512, 768,$ and 1024 (from a maximum of 1764).

Results. Figure 7 shows our results. Our method gives significant savings with only minor reductions in accuracy. For example, at size 1024 with $r = 768$, our approach reduces the cost from 467.4 GFlops to 122.3 GFlops (3.8x lower) while reducing the mAP50 score from 82.93 to 81.25 (-1.68% in absolute mAP50). At size 672 with $r = 384$, we reduce the cost by 3.7x with a -0.85% change in mAP50.

In these experiments, some tokens correspond to padded space and are therefore "easy" from a temporal redundancy standpoint. However, even in a padding-free deployment (*e.g.*, with a single, known training and inference resolution) our method would still give strong computation savings. For example, consider resolution 1024 with $r = 768$. We are skipping 66% of all non-padded tokens here (based on a measured mean padding ratio of 44.6% on VID – corresponding to a $\sim$16:9 aspect ratio). This corresponds to a savings of >2x, with an accuracy drop of only 1.68%. Note that our ViViT experiments (Section 5.2 and supplementary) do not involve padding.

Figure 8 shows the accuracy-compute tradeoff for our method, along with baselines. Our approach gives a considerable improvement in the accuracy-compute tradeoff compared to STGT [37]. Further, adding redundancy awareness to the query-key and attention-value products reduces the cost significantly, especially at low r values.

5.2. Video Action Recognition

Task and dataset. We evaluate our method on action recognition using the EPIC-Kitchens 100 dataset [15]. EPIC-Kitchens 100 contains highly dynamic egocentric videos annotated with 97 verb and 300 noun classes. We consider the verb classification task. The training and validation set contains 67217 and 9668 action instances, respectively.

Implementation details. We use the ViViT model [1] with factorized spatial and temporal sub-models based on ViT-B.

Table 1. **Adding spatial redundancy to ViTDet.** "Spatial" is a model with pooling in $\boldsymbol{k}$ and $\boldsymbol{v}$. "Spatiotemporal" is a model with both pooling and temporal redundancy awareness.

Variant	r	mAP50 (%)	GFlops
Base model	–	82.93	467.4
Spatial	–	80.15	388.1
Spatiotemporal	2048	80.14	217.0
Spatiotemporal	1536	80.07	169.3
Spatiotemporal	1024	79.50	121.0
Spatiotemporal	768	78.69	96.3
Spatiotemporal	512	76.96	70.9
Spatiotemporal	256	71.35	44.5

The spatial sub-model (the bulk of the compute cost) is applied sequentially to 16 2-frame input clips. The outputs of the spatial model are concatenated and passed to the temporal model, which returns a class prediction. The prediction is the average over 12 video views (4 temporal views, each divided into 3 spatial crops). Each view has a shape of 320×320×32. Unlike ViTDet, ViViT adds a class embedding token (see [16]), does not use windowed self-attention, and does not use relative position embeddings.

We convert the spatial model to an Eventful Transformer. Naively replacing the spatial model with an Eventful version leads to a considerable drop in accuracy (about -10% with $r = 100$). We conjecture that the cause is a distribution shift in the inputs to the temporal model (see the supplement for further discussion). We recover most of the lost accuracy by fine-tuning the *non-Eventful* temporal model on the outputs of a frozen Eventful spatial model.

Experiment protocol. We start with ViViT pre-trained on EPIC-Kitchens 100 and fine-tune the temporal model as described above (on the EPIC-Kitchens training set). We fine-tune different model variants with policy r values of 50, 100, and 200 (out of a maximum of 401 tokens). See the supplement for training parameters. We report results using the top-1 accuracy metric, following standard protocol [15].

Results. Figure 9 shows our results for the Eventful ViViT model. We evaluate a range of r values for each of the fine-tuned variants. We test the original fine-tuned r-value, along with ±20% and ±40% of this value. We observe considerable computation savings with only moderate reductions in accuracy. For example, with $r = 140$, we reduce the cost by 2.4x while reducing the accuracy by only 1.62%. In addition, the model retains adaptivity despite being fine-tuned with a single-r value, exhibiting a favorable accuracy-compute tradeoff over a range of r-values.

5.3. Spatial Redundancy and Runtime

Considering spatial redundancy. Eventful Transformers exploit *temporal* redundancy and thus complement prior works that leverage *spatial* redundancy. Here we present

Table 2. **Runtimes (ms).** ViTDet runtimes are for the Transformer backbone only. ViViT runtimes include the temporal sub-model.

Model	Size	Variant	r	GPU	CPU
ViTDet	1024	Base model	–	86.6	5150
ViTDet	1024	Spatial	–	58.9	3116
ViTDet	1024	Temporal	512	69.9	3570
ViTDet	1024	Spatiotemporal	512	38.1	1682
ViTDet	672	Base model	–	28.3	1492
ViTDet	672	Spatial	–	23.3	1055
ViTDet	672	Temporal	256	21.6	838
ViTDet	672	Spatiotemporal	256	20.8	478
ViViT	320	Base model	–	950	5.45e4
ViViT	320	Temporal	50	545	2.15e4

a simple proof-of-concept experiment that considers spatial redundancy in Eventful Transformers.

Specifically, we adopt a variant of [68], which applies spatial pooling to the self-attention key and value tokens. We apply this method with 2×2 pooling to the global self-attention operators in the ViTDet model. We evaluate this method both with and without temporal redundancy awareness. In the temporal-redundancy model, we pool $\boldsymbol{k}$ and $\boldsymbol{v}$ after their respective buffers. We pool $\tilde{\boldsymbol{k}}$ by first pooling the active indices (equivalent to max-pooling the mask $\boldsymbol{m}$), then gathering the buffered $\boldsymbol{k}$ using the pooled indices.

Table 1 shows our results for resolution 1024 (see the supplement for resolution 672). We see that the spatial and temporal methods are complementary; both meaningfully contribute to reducing the computational cost. See Section 6 for further discussion on spatial redundancy methods.

Runtime. We show preliminary runtime results on a CPU (Xeon Silver 4214, 2.2 GHz) and a GPU (NVIDIA RTX 3090). See the supplementary material for experiment details. Table 2 shows our results. Adding temporal redundancy awareness leads to speedups of up to 1.74x on the GPU and 2.48x on the CPU. These results should be seen just as a proof of concept – we are confident that these speedups could be improved with further engineering effort (*e.g.*, by replacing vanilla PyTorch operators with custom kernels or using a high-performance inference framework).

Visualization of updates. Figure 10 shows an example video sequence. We visualize the model predictions (top), the token-wise L2 norm of the error $\boldsymbol{e}$ (middle), and the update mask $\boldsymbol{m}$ (bottom). We see that larger error values correspond to dynamic regions in the image.

6. Discussion

Memory overhead. Our method reduces floating point operations at the cost of higher memory usage. Each gate or buffer maintains a reference tensor ($\boldsymbol{u}$ or $\boldsymbol{b}$, respectively).

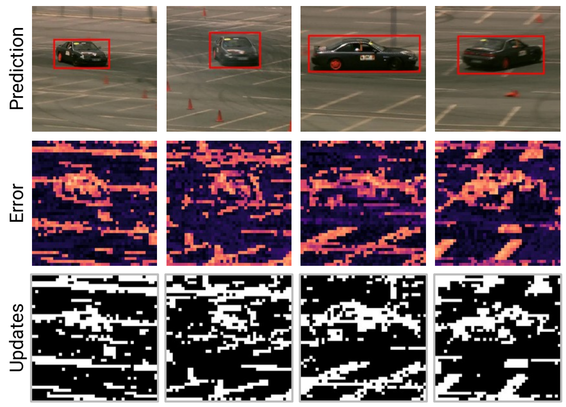

Figure 10. **Visualization of updates.** The error $\boldsymbol{e}$ and update mask $\boldsymbol{m}$, for the pre-QKV gate in block 3 of ViTDet. Video source: [57]

The memory overhead for token gates and buffers is generally modest. For, consider size-1024 ViTDet. The model has 4096 768-dimensional tokens, meaning a token gate or buffer takes 12.6/6.3 MB of memory at full/half precision.

However, gating or buffering the attention matrix A can require a larger amount of memory. For example, in the global attention layers of the size-1024 ViTDet model, the A matrix has shape 4096×4096×12. Buffering this A requires 805/403 MB at full/half precision. Fortunately, the situation dramatically improves if we reduce the number of tokens or use windowed attention (due to the quadratic size of A). For example, the A matrices for the size-672 ViTDet model (1764 tokens) and the ViViT model (301 tokens) occupy 149/75 MB and 4.3/2.2 MB, respectively. In applications where memory is tight and the A matrix is large, it is possible to save memory by removing temporal redundancy awareness in the query-key and/or attention-value products (each eliminates one A-shaped state tensor).

Integration with spatial redundancy methods. A promising avenue for future work the is further integration of our approach with spatial redundancy methods. Conceptually, these methods summarize a large, redundant set of tokens using a more compact set of tokens. The gating module in an Eventful Transformer assumes that most of its inputs vary smoothly over time. When combining our approach with spatial redundancy methods, we need to ensure that the compact spatial summary is relatively stable. For example, with adaptive token clustering [3, 45], we would need to sort the clusters in a mostly-consistent order.

There are many potentially interesting questions regarding joint modeling of spatial and temporal redundancy. For example, how is the temporal axis different from the spatial one? Should the temporal axis be modeled separately? We leave such questions for future work.

Acknowledgments. This research was supported in part by NSF CAREER award #1943149.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. ViViT: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, October 2021. 1, 2, 8

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? arXiv, 2021. 1

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. arXiv, 2022. 2, 9

[4] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240x180 130 db 3 µs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014. 2

[5] BuzzFarmers. flickr.com/photos/buzzfarmers/7318008726, 2011. Online, accessed March 2023, CC BY 2.0 license. 1

[6] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1483–1498, 2021. 7

[7] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. arXiv, 2016. 1

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, Cham, 2020. Springer International Publishing. 1

[9] Lukas Cavigelli, Philippe 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*. Association for Computing Machinery, September 2017. 3

[10] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. Memory enhanced global-local aggregation for video object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7, 15

[11] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. arXiv, 2019. 2

[12] Ting-Wu Chin, Ruizhou Ding, and Diana Marculescu. AdaScale: Towards real-time video object detection using adaptive scaling. In A. Talwalkar, V. Smith, and M. Zaharia, editors, *Proceedings of Machine Learning and Systems*, volume 1, pages 431–441, 2019. 3

[13] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with Performers. In *International Conference on Learning Representations*, 2021. 2

[14] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 3

[15] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The EPIC-KITCHENS dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 8

[16] 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. In *International Conference on Learning Representations*, 2021. 1, 3, 7, 8

[17] Matthew Dutson, Yin Li, and Mohit Gupta. Event neural networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 276–293, Cham, 2022. Springer Nature Switzerland. 2, 3

[18] 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, October 2021. 1

[19] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 396–414, Cham, 2022. Springer Nature Switzerland. 2, 7

[20] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3

[21] Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3690–3699. PMLR, July 2020. 2

[22] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. Star-transformer. arXiv, 2019. 2

[23] Amirhossein Habibian, Davide Abati, Taco S. Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2695–2704, June 2021. 2, 3

[24] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 3

[25] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. 3

[26] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, October 2017. 7

[27] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv, 2017. 3

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

[29] Kyuyeon Hwang and Wonyong Sung. Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 1–6, 2014. 3

[30] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5MB model size. arXiv, 2016. 3

[31] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, LUKASZ KAISER, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9895–9907. Curran Associates, Inc., 2021. 2

[32] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, July 2020. 2

[33] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics human action video dataset. arXiv, 2017. 14

[34] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. 2

[35] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. 3

[36] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, 2017. 3

[37] Yawei Li, Babak Ehteshami Bejnordi, Bert Moons, Tijmen Blankevoort, Amirhossein Habibian, Radu Timofte, and Luc Van Gool. Spatio-temporal gated transformers for efficient video processing. In *Advances in Neural Information Processing Systems Workshops*, 2021. 3, 7, 8

[38] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 280–296, Cham, 2022. Springer Nature Switzerland. 1, 3, 7

[39] Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EViT: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. 2

[40] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 2

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, Cham, 2014. Springer International Publishing. 15

[42] 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, October 2021. 1, 3

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 14, 15

[44] 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 M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21297–21309. Curran Associates, Inc., 2021. 2

[45] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. arXiv, 2021. 2, 9

[46] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. AdaViT: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12309–12318, June 2022. 3

[47] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. AR-Net: Adaptive frame resolution for efficient action recognition. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–104, Cham, 2020. Springer International Publishing. 2, 3

[48] Sparsh Mittal. A survey of techniques for approximate computing. *ACM Computing Surveys*, 48(4), March 2016. 2

[49] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. IA-RED^2: Interpretability-aware redundancy reduction for vision transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24898–24911. Curran Associates, Inc., 2021. 2

[50] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 377–386, October 2021. 2

[51] Mathias Parger, Chengcheng Tang, Christopher D. Twigg, Cem Keskin, Robert 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, June 2022. 2, 3

[52] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13937–13949. Curran Associates, Inc., 2021. 2

[53] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 525–542, Cham, 2016. Springer International Publishing. 3

[54] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7

[55] Hongyu Ren, Hanjun Dai, Zihang Dai, Mengjiao Yang, Jure Leskovec, Dale Schuurmans, and Bo Dai. Combiner: Full attention transformer with sparse computation cost. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22470–22482. Curran Associates, Inc., 2021. 2

[56] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. arXiv, 2020. 2

[57] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, April 2015. 1, 4, 7, 9

[58] Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: What can 8 learned tokens do for images and videos? arXiv, 2021. 2

[59] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016. 3

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 1, 3

[61] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2, 3

[62] Junke Wang, Xitong Yang, Hengduo Li, Li Liu, Zuxuan Wu, and Yu-Gang Jiang. Efficient video transformers with spatial-temporal token selection. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 69–86, Cham, 2022. Springer Nature Switzerland. 2

[63] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. arXiv, 2020. 2

[64] 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, October 2021. 1

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

[66] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 11960–11973. Curran Associates, Inc., 2021. 3

[67] Zuxuan Wu, Caiming Xiong, Yu-Gang Jiang, and Larry S Davis. LiteEval: A coarse-to-fine framework for resource

efficient video recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[68] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, May 2021. 9

[69] Ran Xu, Fangzhou Mu, Jayoung Lee, Preeti Mukherjee, Somali Chaterji, Saurabh Bagchi, and Yin Li. SmartAdapt: Multi-branch object detection framework for videos on mobiles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2538, June 2022. 2, 3

[70] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 3

[71] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 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, June 2022. 3

[72] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip H.S. Torr, Wayne Zhang, and Dahua Lin. Vision transformer with progressive sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 387–396, October 2021. 2

[73] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[74] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 408–417, 2017. 15

Supplementary Material

We use capital letters (*e.g.*, Figure A) to refer to the supplementary material and numbers (*e.g.*, Figure 1) to refer to the main paper.

In Section A, we provide further discussion on token selection policies, optimizations to the query-key product, and the ViViT temporal model. In Section B, we present additional experiments: action recognition on Kinetics-400, an evaluation of a threshold policy, and an ablation of the gate position. In Section C, we provide low-level details for the experiments in the main paper. In Section D, we include tables of results for the experiments in the main paper.

A. Further Discussion

The ViViT temporal sub-model. Recall that, for ViViT action recognition, we fine-tune the non-Eventful temporal model on the outputs of the Eventful spatial model. We now provide some intuition as to why this is necessary to preserve the prediction accuracy.

The outputs of an Eventful Transformer are approximations of the "correct" outputs (those of the original, non-Eventful Transformer). In the case of the ViViT spatial model, individual outputs are fairly close to the correct values. However, the *pattern of temporal changes* between outputs may be quite different from the original model. Token gates reduce the number of updated tokens on each frame, but each update tends to be larger (a single update may contain accumulated changes from several time steps). Given the nature of the prediction task – action recognition on highly dynamic videos – the temporal sub-model is sensitive to the pattern of temporal changes. Fine-tuning allows us to correct for the shifts in these temporal changes that result from using an Eventful spatial model.

Compatibility with spatial redundancy methods. We now provide further discussion regarding the compatibility of our method with spatial redundancy approaches. Abstractly, we can think of spatial redundancy methods as summarizing a set of tokens $\boldsymbol{x} \in \mathbb{R}^{N \times D}$ using a reduced set of tokens $\hat{\boldsymbol{x}} \in \mathbb{R}^{M \times D}$. The simple method in our experiments summarizes tokens using uniform pooling; however, we could also use adaptive pruning or merging.

Assume we apply a gate to the reduced tokens $\hat{\boldsymbol{x}}$. The gate assumes that the definitions of its input tokens are relatively stable. This assumption clearly holds for non-reduced or uniformly pooled tokens. However, we need to be careful when applying arbitrary reductions to $\boldsymbol{x}$.

For example, say we have an image containing a region of blue sky. An adaptive token merging method might combine all sky-colored tokens from $\boldsymbol{x}$ into a single token in $\hat{\boldsymbol{x}}$. Assume that on frame $t = 1$, the first token in $\hat{\boldsymbol{x}}$ represents the sky. Ideally, on frame $t = 2$, the first token in $\hat{\boldsymbol{x}}$ should again represent the sky. Note that this is not a strict constraint – our gating logic can deal with non-consistent definitions for a few tokens. However, if the definitions for all tokens in $\hat{\boldsymbol{x}}$ completely change between frames, then the gate will not be able to keep up (*i.e.*, the number of tokens with significant changes will exceed the policy r-value).

B. Additional Experiments

Video action recognition on Kinetics-400. We evaluate our method on the Kinetics-400 action recognition dataset [33]. Kinetics-400 contains over 300k video clips, each annotated with one of 400 action categories. We evaluate top-1 accuracy. We use the same ViViT model architecture as in our EPIC-Kitchens experiments; the only difference is the input size (224×224 rather than 320×320).

As in our EPIC-Kitchens experiments, we fine-tune the non-Eventful temporal model on the outputs of the Eventful spatial model. We fine-tune three variants of the model with r =24, 48, and 96 (out of a maximum of 197 tokens). We train for 10 epochs on a subset of the training set containing 39729 videos. We use the AdamW optimizer [43] with a learning rate of 2×10^{-6}, weight decay of 0.05, and a batch size of 16 videos. We add 50% dropout before the final classification layer.

Table A shows our results. The accuracy-compute tradeoff is generally consistent with our results on EPIC-Kitchens. For example, with r = 96, we sacrifice 1.48% accuracy for a speedup of approximately 2x.

A threshold policy. We evaluate the ViTDet object detection model with a threshold policy. The threshold policy selects all tokens where the L2 norm of $\boldsymbol{e}$ exceeds a threshold h. We test h = 0.2, 1.0, and 5.0. See Table B for results. The accuracy-compute tradeoff for the threshold policy is generally worse than for the top-r policy. For example, compare threshold h = 5.0 with r = 512 in Table C. This is likely due to the use of a constant threshold for all gates (we would ideally use a unique threshold for each gate).

Table A. **Kinetics-400 video action recognition.** Results for Kinetics-400 action recognition using the ViViT model. We report the total TFlops per video (spatial + temporal sub-models).

Variant	r	Accuracy (%)	TFlops
Base model	–	79.06	3.360
Temporal	96	77.62	1.814
Temporal	48	75.88	1.016
Temporal	24	75.16	0.618

C. Experiment Details

Fine-tuning ViTDet for VID. We initialize our model using COCO [41] pre-trained weights, and then trained on a combination of the ImageNet VID and ImageNet DET datasets, following common protocols in [10, 74]. We select images from the DET dataset that are of of the same 30 classes as in the VID dataset. The training uses a batch size of 8, a maximum input resolution of 1024×1024, an initial learning rate of 10^{-4}, and a weight decay of 0.1. We use the AdamW optimizer [43] with linear warmup for a total of 5 epochs, with 10x learning rate decay from the 3rd epoch.

Fine-tuning the ViViT temporal model. We fine-tune the temporal sub-model for 5 epochs. We use the AdamW optimizer [43] with a learning rate of 10^{-5}, weight decay of 0.05, and a batch size of 8 videos. We add 50% dropout before the final classification layer.

Arithmetic precision. We compute the product $A\boldsymbol{v}$ at half precision in the global self-attention operators of the Eventful model. Using half precision reduces the model's computational cost and memory footprint and has a negligible effect on accuracy. When evaluating runtimes, we also compute $A\boldsymbol{v}$ at half precision in the base model (this ensures a fair comparison).

Runtime experiments. For ViTDet, we evaluate CPU runtimes using one random video from VID (ID 00023010, containing 242 frames). On the GPU, we use 5 random videos. For ViViT, we evaluate CPU runtimes using 5 random videos from EPIC-Kitchens. On the GPU, we use 100 random videos. We use a consistent random seed across all experiment runs.

Operation counting. Our GFlop counts include the following types of operations: linear transforms, matrix multiplications, einsum operations (used in relative position embeddings), and additions. We count a multiply-accumulate as a single operation. In Eventful Transformers, we additionally count operations required for updating the gate (additions and subtractions) and the extra additions in the sparse attention-value update. We only report operations in the Transformer backbones (*e.g.*, we do not count anything in the object detection head).

Table B. **A threshold policy.** Results for a threshold policy with the 1024-resolution ViTDet model. The policy selects tokens where the error $\boldsymbol{e}$ exceeds a threshold h.

Variant	h	mAP50 (%)	GFlops
Base model	–	82.93	467.4
Temporal	0.2	83.00	431.8
Temporal	1.0	82.75	294.1
Temporal	5.0	78.11	133.5

D. Result Tables

In this section, we provide tables of results for experiments in the main paper. Table C corresponds to Figures 7 and 8, and Table D corresponds to Figure 9. Table E shows spatial redundancy results for the 672-resolution ViTDet model (the 1024-resolution results are in Table 1).

Table C. **Video object detection results.** Results for video object detection on VID using the ViTDet model. This table corresponds to Figures 7 and 8 in the main paper.

Size	Variant	r	mAP50 (%)	GFlops
1024	Base model	–	82.93	467.4
1024	Our method	2048	82.94	294.9
1024	Our method	1536	82.79	225.9
1024	Our method	1024	82.00	156.8
1024	Our method	768	81.25	122.3
1024	Our method	512	79.38	87.8
1024	Our method	256	73.29	53.3
1024	Token-wise only	2048	82.97	294.1
1024	Token-wise only	1536	82.93	250.7
1024	Token-wise only	1024	82.58	207.3
1024	Token-wise only	768	82.08	185.7
1024	Token-wise only	512	81.11	164.0
1024	Token-wise only	256	76.60	142.3
1024	STGT	2048	82.92	294.1
1024	STGT	1536	82.60	250.7
1024	STGT	1024	81.25	207.3
1024	STGT	768	79.81	185.7
1024	STGT	512	76.70	164.0
1024	STGT	256	68.73	142.3
672	Base model	–	82.28	174.5
672	Our method	1024	82.23	115.1
672	Our method	768	82.21	87.9
672	Our method	512	81.84	60.7
672	Our method	384	81.43	47.1
672	Our method	256	80.16	33.5
672	Our method	128	75.19	19.9
672	Token-wise only	1024	82.28	111.9
672	Token-wise only	768	82.25	90.2
672	Token-wise only	512	82.01	68.5
672	Token-wise only	384	81.64	57.7
672	Token-wise only	256	80.76	46.8
672	Token-wise only	128	76.96	36.0
672	STGT	1024	82.28	111.9
672	STGT	768	81.95	90.2
672	STGT	512	80.45	68.5
672	STGT	384	78.71	57.7
672	STGT	256	75.57	46.8
672	STGT	128	68.13	36.0

Table D. **Video action recognition results.** Results for video action recognition on EPIC-Kitchens using the ViViT model. This table corresponds to Figure 9 in the main paper.

Variant	Tuned r	Tested r	Accuracy (%)	TFlops
Base model	–	–	67.14	7.12
Temporal	200	280	66.77	5.49
Temporal	200	240	66.53	4.77
Temporal	200	200	66.02	4.05
Temporal	200	160	64.72	3.33
Temporal	200	120	62.23	2.62
Temporal	100	140	65.52	2.98
Temporal	100	120	64.51	2.62
Temporal	100	100	62.91	2.26
Temporal	100	80	60.76	1.90
Temporal	100	60	59.13	1.54
Temporal	50	70	61.27	1.72
Temporal	50	60	60.60	1.54
Temporal	50	50	59.91	1.36
Temporal	50	40	58.90	1.18
Temporal	50	30	58.05	1.00

Table E. **Adding spatial redundancy to 672-resolution ViTDet.** Results for adding spatial redundancy to the 672-resolution ViTDet model. 1024-resolution results are in the main paper.

Variant	r	mAP50 (%)	GFlops
Base model	–	82.28	174.5
Spatial	–	79.86	159.7
Spatiotemporal	1024	79.85	98.2
Spatiotemporal	768	79.81	75.5
Spatiotemporal	512	79.47	52.8
Spatiotemporal	384	79.02	41.4
Spatiotemporal	256	77.90	29.8
Spatiotemporal	128	73.40	18.0