InterArch: Video Transformer Acceleration via Inter-Feature Deduplication with Cube-based Dataflow

Xuhang Wang[1=], Zhuoran Song[1=*], Xiaoyao Liang[1]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

ABSTRACT

In the realm of video-oriented tasks, Video Transformer models (VidT), an evolution from vision Transformers (ViT), have demonstrated considerable success. However, their widespread application is constrained by substantial computational demands and high energy consumption. Addressing these limitations and thus improving VidT efficiency has become a hot topic. Current methodologies solve this challenge by dividing a video into several features and applying intra-feature sparsity. However, they neglect the crucial point of inter-feature redundancy and often entail prolonged latency in fine-tuning phases. In response, this paper introduces InterArch, a tailored framework designed to significantly enhance VidT efficiency. We first design a novel inter-feature sparsity algorithm consisting of hierarchical deduplication and recovery. The deduplication phase capitalizes on temporal similarities at both block and element levels, enabling the elimination of redundant computations across features in both coarse-grained and fine-grained manners. To prevent long-latency fine-tuning, we employ a lightweight recovery mechanism that constructs approximate features for the sparsified data. Furthermore, InterArch incorporates a regular dataflow strategy, which consolidates sparse features and effectively translates sparse computations into dense ones. Complementing this, we develop a spatial array architecture equipped with augmented processing elements (PEs), specifically optimized for our proposed dataflow. Extensive experiment results demonstrate that InterArch can achieve satisfactory performance speedups and energy saving.

1 INTRODUCTION

In recent years, based on the attention mechanism, various models have gained satisfactory accuracy in the field of Computer Vision (CV). A representative model, Vision Transformer [7] (ViT), has surpassed traditional convolutional neural networks (CNNs) in accuracy. With the success of ViTs on image recognition tasks, researchers have created Video Transformers (VidTs), expanding the applications of ViT to video understanding tasks (e.g., video classification [12], video action recognition [2], and video target detection [16]). Unfortunately, the high accuracy is at the cost of inefficiencies [6], especially in video-oriented tasks with substantial

This work is partly supported by National Natural Science Foundation of China (Grant No. 62202288). = Equal Contribution. *Zhuoran Song is the corresponding author.

DAC '24, June 23–27, 2024, San Francisco, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3656237

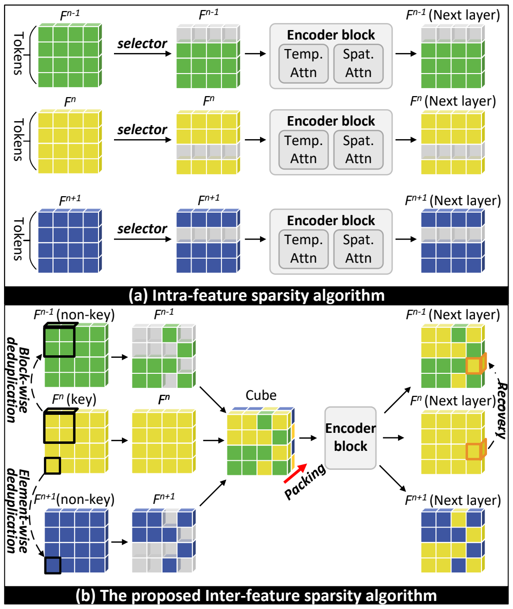


Figure 1: Comparison between the intra-frame sparsity algorithm and our proposal.

frames. Thus, there is an imperative need to alleviate the enormous computational and storage pressure of VidT.

Applying sparsity to individual feature frame (denoted as F) is one of the promising trends to accelerate ViT-related models [18]. A notable example, HeatViT [6], shown in Fig. 1 (a), implements a selector to dynamically assess and prune less significant tokens in each feature F. However, despite these advancements, there are several critical limitations in current approaches: *LIMITATION 1*, they primarily concentrate on redundancy within individual features, overlooking the potential of addressing inter-feature similarity. This oversight limits opportunities for further performance enhancement. *LIMITATION 2*, they typically necessitate extensive fine-tuning to compensate for accuracy losses. This is especially burdensome for video-oriented tasks involving numerous frames, where the retraining process becomes excessively time-consuming and resource-intensive. *LIMITATION 3*, as the number of tokens varies between features, the feature-by-feature dataflow struggles to efficiently utilize fixed-shape Processing Element (PE) arrays, leading to suboptimal resource utilization. *LIMITATION 4*, their specialized architectures developed lack scalability and flexibility, posing difficulties when there are algorithmic changes.

We categorize the aforementioned limitations into three distinct challenges. **Algorithmic Challenge**: The first challenge involves designing an algorithm capable of exploring inter-feature redundancy without extensive fine-tuning to overcome the *LIMITATION 1* and *LIMITATION 2*. **Dataflow Challenge**: Addressing *LIMITATION 3* requires the development of a dataflow that can efficiently handle features of varying shapes. **Architectural Challenge**: The final challenge revolves around creating an architecture that can seamlessly execute various sparsity algorithms without requiring significant hardware modifications to tackle *LIMITATION 4*.

In this paper, we propose a VidT acceleration framework called InterArch, which involves three innovations. From the **algorithm** aspect, we design an inter-feature sparsity algorithm to remove the similarity between features by exploiting the natural redundancy between them. The foundational concept involves a hierarchical deduplication process, consisting of both block-wise and element-wise stages. Considering that the background is largely static even with abrupt object movements, we employ block-wise deduplication to coarsely eliminate repetitive background information across features. Moreover, in scenes with slowly moving objects, our element-wise deduplication finely targets and removes redundancies caused by temporally similar objects. As illustrated in Fig 1 (b), the proposed algorithm initially divides multiple features into key and non-key features. Subsequent steps involve chunking non-key features into blocks to identify and eliminate redundancy during block- and element-wise deduplication. Unlike conventional methods that discard unimportant values outright, our framework includes a lightweight recovery mechanism. This process approximates non-key features by referencing key features and preserving crucial information, thereby mitigating significant accuracy losses and obviating the need for retraining. As for **dataflow**, in contrast to traditional methods that process features serially, we establish a creative cube-based dataflow that stacks the key and non-key features along the temporal dimension, compressing zeros within non-key features to create a dense and balanced cube. The resulting cube, with its reduced dimensions, facilitates a notable speedup by leveraging the complementary nature of the features. Based on this dataflow, we establish an **architecture** by inserting slight modifications to the PEs of the traditional spatial architecture [4]. Note that the proposed architecture is well-scalable. We can support any sparse algorithms by stacking the sparse features into dense cubes. The main contributions of this paper include:

- We propose a framework for VidT models, InterArch. We design a novel inter-feature sparsity algorithm to reduce the redundancy of features based on inter-feature similarity. More importantly, we create an extensible dataflow with characteristics that maintain the shape of features to improve the PE utilization. To the best of our knowledge, InterArch is the first framework dedicated to accelerating VidTs' inference, offering a new perspective on efficient VidT solutions.
- We design an InterArch architecture to support the proposed algorithm. Specifically, we slightly modify the PEs in a dense spatial array, making it perform proposed dataflow. Moreover, we design an inter-column balance scheme that allows free PEs to fetch subsequent data in advance based on an augmented MAC function with simple input matching and holding capabilities to perform conditional MAC operations.
- We validate the effectiveness of design on various VidT models for video-oriented tasks. Compared to CPU, GPU, as well as two state-of-the-art ViT accelerators ViTCoD and HeatViT, we gain 81.5×, 40.6×, 1.87×, and 1.59× speedup and 77.7×, 18.1×, 1.81×, and 1.74× energy efficiency while maintaining the model accuracy.

2 BACKGROUND AND MOTIVATION

2.1 VidT Models

We take one of the most representative VidT models TimeSformer [2] as an example to illustrate the basic working mechanism of VidT. As shown in Fig. 2, initially, TimeSformer receives multiple frames and slices each frame into multiple patches, which act as tokens in the sequence. After executing through an embedding layer, each frame is transformed into a feature F and sent into L encoding blocks for feature extraction. The core computations of each block are temporal attention and spatial attention. The inputs of the temporal attention are formed by tokens in the same spatial location across features. For example, the first tokens of each features in orange boxes are aggregated together to form a new feature F_t^{n-1}. Each newly generated feature is regarded as an input of the temporal attention. Within the temporal attention, each feature transforms into Q, K, and V by linear projections. For example, we obtain Q by matrix multiplication between F and a weight W_q during linear projections. Then, Q, K, and V conduct self-attention computation, which is $Softmax(\frac{QK^T}{\sqrt{d_k}})V$, to obtain output. Before feeding into the spatial attention, each feature is reshaped back to its original spatial distribution. After that, the spatial attention performs the same computations as the temporal attention. Finally, the spatial attention generates input features of the next layer.

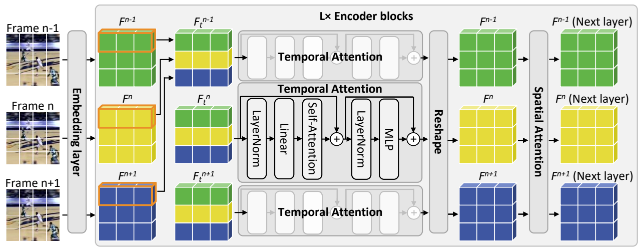


Figure 2: The structure of TimeSformer.

2.2 Related Work

To reduce the computation of the attention mechanism, researchers propose a series of sparse attention algorithms [15, 17] and architecture designs [5, 6, 18].

On the algorithm side, various researchers impose sparsity on a single frame. DynamicViT [15] inserts a prediction module before encoder blocks to obtain the significance of each token and discard the tokens with low significance. Moreover, Evo-vit [17] retains those tokens that are more similar to CLS tokens.

On the hardware side, researchers accelerate the attention mechanism by software-hardware co-design. For example, ViTCoD [18] provides a fixed sparsity pattern for the attention score based on the training dataset and reorders the pattern into denser and sparser parts. By efficiently supporting the two parts with dedicated hardware, ViTCoD can reduce the overhead of the attention-based model. Moreover, Vitality [5] reduces the computational complexity by converting the softmax computation into a linear operation through a first-order Taylor expansion and reversing the order of calculations

Table 1: Redundancy ratio of input features

Models	Datasets	Redundancy ratio	
		Temporal	Spatial
TimeSformer	K400	68.1%	55.2%
	K600	68.4%	50.5%
	SSv2	65.3%	58.4%
Motionformer	K400	68.9%	63.9%
	K600	68.3%	61.3%
	SSv2	54.6%	57.2%

in self-attention. Additionally, HeatViT [6] proposed an MLP-based token selector to cascade prune tokens.

In summary, although existing works try to dig out the redundant information within features, they still fail to apply the attention-based model to video-oriented tasks effectively as they overlook inter-feature similarity. Therefore, reducing the redundant information between features in the temporal dimension is crucial for accelerating attention-based models for video tasks. In this paper, we detect the similarity between features and remove redundant computations. One may argue that existing methods for skipping inter-frame redundancy in CNNs [8, 10] could be adapted for VidT. These approaches typically employ a learnable DNN model to select informative frames. However, such techniques are not well suited for optimizing VidT, whose structure and feature representation in VidTs significantly differ from those in CNNs.

2.3 Motivation

To verify whether there exist sufficient inter-feature redundancy, we conducted an analysis using prominent VidT models, Motionformer [14] and TimeSformer [2] on three datasets: Kinetics-400 [11], Kinetics-600 [11], and SSv2 [9]. We select the middle feature in a batch as key feature and the rest as non-key features. A value in the non-key feature is regarded as redundant if its difference with the corresponding position in the key feature is less than a pre-defined threshold 0.169. Fig. 1 shows the average ratio of redundant data is 65.6% in temporal attention and 57.8% in spatial attention, which gives us a large room for efficient VidT.

3 ALGORITHM AND DATAFLOW

3.1 Inter-feature Sparsity Algorithm

3.1.1 Deduplication. We develop an inter-feature sparsity algorithm that contains deduplication and recovery. The core idea of the algorithm is to locate and eliminate elements in non-key features that are similar to key features. To locate redundancy, we add a hierarchical duplication detection that generates a *F-ID* matrix for each feature to record the positions of the similarity. Initially, the *F-ID* matrix is filled with features' IDs. In hierarchical duplication detection, we start by grouping multiple features into a batch. Within this batch, we designate the *n*th feature F^n as the key feature, while the remaining features are categorized as non-key features. For each non-key feature, such as F^{n-1}, we divide it into blocks, covered by black boxes in Fig. 3. We then compute the L1 Norm between corresponding blocks in F^{n-1} and F^n. As shown in Eqn. 1, the L1 Norm is defined as the sum of the absolute values of the differences (denoted as ΔF_i^{n-1}) between corresponding elements in two blocks. If the L1 Norm falls below a predefined threshold D,

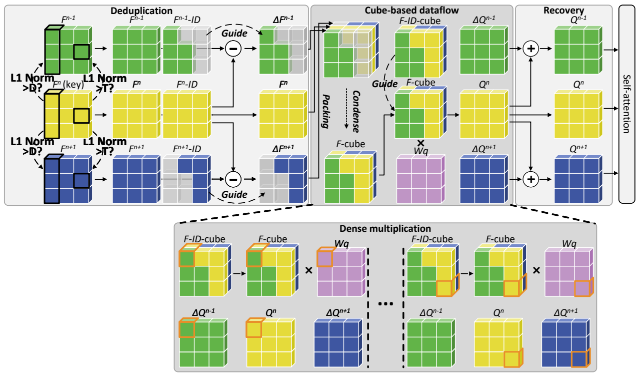


Figure 3: An overview of ours algorithm and dataflow.

the entire block in F^{n-1} is deemed redundant. This step primarily targets larger, more obvious redundancies. Following the block-wise duplication detection, we delve into a finer granularity by examining the reserved blocks. The L1 Norm is calculated between elements in F^{n-1} and F^n. Any value in F^{n-1} whose L1 Norm with the corresponding element in F^n is less than another predefined threshold T is considered redundant. For each redundancy in F^{n-1}, we set the corresponding position in F^{n-1}-*ID* as zero. Through this meticulous process of block-wise and element-wise duplication detection, our algorithm efficiently uncovers redundancy within the non-key features.

Subsequently, we perform element-wise subtraction between F^{n-1} and F^n to obtain their difference ΔF^{n-1}. The subtractions can help us build a lightweight recovery, which will be further discussed in Section 3.1.2. Next, we apply sparsity to ΔF^{n-1} guided by F^{n-1}-*ID*. Finally, the dense F^n and sparse ΔF^{n-1} can experience dense and sparse linear projections to gain performance improvement.

$$\text{L1 Norm} = \sum_i |F_i^{n-1} - F_i^n| = \sum_i |\Delta F_i^{n-1}| \tag{1}$$

$$F^{n-1} \times W_q \approx (\Delta F^{n-1} + F^n) \times W_q$$
$$\underbrace{F^{n-1} \times W_q}_{Q^{n-1}} \approx \underbrace{F^n \times W_q}_{Q^n} + \underbrace{\Delta F^{n-1} \times W_q}_{\Delta Q^{n-1}} \tag{2}$$

3.1.2 Recovery. Recovery is another important stage of our algorithm for avoiding severe accuracy loss. As shown in Fig 3, the recovery involves an element-wise addition between ΔQ^{n-1} and Q^n to reconstruct Q^{n-1}. This method is grounded in the distributive law of multiplication, as outlined in Eqn. 2. The sparsity induced in ΔF^{n-1} is activated only when an element is identified as similar to the corresponding element in the key feature F^n. As a result, the sum $\Delta F^{n-1} + F^n$ can be seen as approximately equal to F^{n-1}. In other words, ΔQ^{n-1} can be estimated by adding Q^n and ΔQ^{n-1}.

3.2 Cube-based Dataflow

In this section, we delve into cube-based dataflow, a novel approach devised to streamline the process of sparse matrix multiplication. The conventional method of multiplying a sparse matrix ΔF with a weight matrix W_q to generate ΔQ typically requires complex control overheads. To address the problem, as shown in Fig 3, we stack successive sparse matrices along a batch dimension and squeeze out their zero values to form a dense *F*-cube. Alongside the *F*-cube, we also generate a *F-ID*-cube by the same operation. The *F-ID*-cube

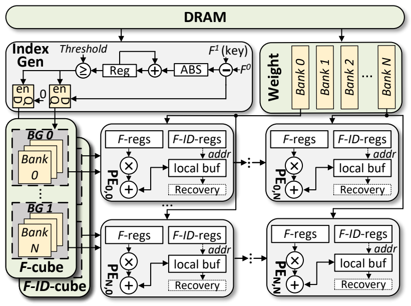


Figure 4: An overview of InterArch architecture.

serves as a reference structure, indicating the original locations of the non-zero elements. Finally, we perform dense matrix multiplication between the F-cube and the weight matrix. For illustrative purposes, we use the weight matrix W_q as an example in the figure.

The dense matrix multiplication primarily consists of numerous vector-scalar products. Each vector-scalar product entails multiplying a vector with a weight. Specifically, this vector, identified as F-Vec, is located in a position of F-cube that houses multiple non-zero elements. Following the multiplication, the remaining task is to accurately allocate the product results to their designated positions within the output matrices. This allocation process is meticulously controlled by a corresponding vector, labeled as F-ID-Vec, in the F-ID-cube. Subsequent to the dispatching, these results are accumulated with the partial sums.

The lower part of Fig. 3 shows how to generate ΔQ^{n-1}, Q^n, and ΔQ^{n+1} by a F-cube and a weight matrix W_q. The process involves 9 vector-scalar products; we take the first step as an example. The F-Vec in the top-left corner of the F-cube, boxed by an orange block, containing two elements belonging to $n-1$th and nth features, are multiplied with one weight. Correspondingly, F-ID-Vec in the top-left corner of the F-ID-cube records "n-1" and "n". This step requires two multiplications, whose results are dispatched into ΔF^{n-1} and F^n and added with the partial sums of ΔQ^{n-1} and Q^n.

Overall, the cube-based dataflow engaged by the rigid F-cube creatively performs dense multiplication between F-cube and weights, which is able to avoid inefficiency due to shape variation of features.

4 ARCHITECTURE

4.1 Architecture Overview

As shown in Fig. 4, we propose InterArch architecture, which contains buffers for F-cube, F-ID-cube, and weight, a spatial array, and index generators. The F-cube buffer is organized into bank groups, each comprising multiple banks, with each group accommodating a row of F-Vecs. In each bank group, elements of a F-Vec are distributed across different banks. Similarly, the F-ID-cube buffer accommodates the F-ID-cube following the same principle. The PEs of the spatial array are tasked with executing vector-scalar products between F-Vecs and weights, adhering to an output-stationary dataflow paradigm. To be specific, the leftmost PEs fetch the F-Vecs and corresponding F-ID-Vec from the F-cube and F-ID-cube buffers, while other PEs receive them from their left-side PEs. In the meanwhile, the weights are broadcasted to the PE columns. Once all necessary F-Vecs and weights are in place, the PEs begin to work and save the product result in their local buffers. Moreover, the InterArch architecture is equipped with the index generator for performing deduplication and generating F-cube and F-ID-cube.

4.2 PE Design

To achieve vector-scalar products, we modify PEs in the spatial architecture [4]. As shown in Fig. 4, the PE contains registers for F-Vecs (abbreviated as F-regs) and their F-ID-Vec (abbreviated as F-ID-regs), a multiplier, an adder, and a local buffer for retaining partial sums. For a N-length F-Vec, PE requires N cycles to accomplish the vector-scalar product. At each cycle, the multiplier reads an element of the F-Vec from a F-reg and multiplies it with the broadcasted weight. Then, based on its F-ID-Vec, PE accumulates the result with the corresponding partial sum in the local buffer by the adder. Moreover, the PE contains a recovery module to execute the recovery, which consists of multiple adders. Each adder reads a ΔQ and a Q from the local buffer and adds them together.

4.3 Inter-column Balance Scheme

Fig. 5 (a) shows a 3×3 array performing the dense multiplication between the F-cube and the weight matrix. As each PE column should be synchronized to ensure the correct outcome, the diverse lengths of F-Vecs in the F-cube cause pipeline stall. This issue is further exemplified in Figure 5 (b). For instance, $PE_{2,0}$ completes its vector-scalar product at cycle 1. However, this PE then enters a period of under-utilization, as it must wait for $PE_{0,0}$ and $PE_{1,0}$ to finish their computations.

To mitigate the pipeline stalls caused by uneven F-Vec lengths, we introduce an inter-column balance scheme. The essence of this scheme lies in keeping the PEs busy by preemptively fetching F-Vecs and temporally holding more weights. Specifically, during periods when a PE is idle, this scheme utilizes the free broadcast line to interpolate and store an additional weight into PEs. Simultaneously, the stalled PE can then read ahead the F-Vecs and commence calculations using these buffered weights, thereby minimizing idleness.

Fig. 5 (c) shows the execution timeline after employing the inter-column balance scheme. At cycle 0, $PE_{0,0}$, $PE_{1,0}$, and $PE_{2,0}$ read the first column of F-Vecs and multiply them with the weight $W_{0,0}$. At cycle 1, to avoid a stall, $PE_{2,0}$ fetches the F-Vec at the position [2,1]—the third row and second column (denoted as F-cube[2,1] in the figure) and acquires an additional $W_{1,0}$ by leveraging the broadcast line. As a result, $PE_{2,0}$ can be busy by performing F-cube[2,1]$\times W_{1,0}$.

4.4 Index Generator

Index generators have two functions: (1) perform the deduplication, and (2) generate F-cube and F-ID-cube. The index generator, shown in Fig. 4, equips a subtractor, an absolute value generator (ABS), an adder, a register, a comparator, and two latches. To perform the deduplication, we first compute the L1 Norm by following the Eqn. 2. Specifically, the subtractor calculates a difference ΔF_i^{n-1} at the ith cycle. Then, the ABS generates the absolute value of F_i^{n-1}. After that, the adder accumulates the absolute value and stores a partial sum of the L1 Norm in the register. After obtaining the final L1 Norm, the comparator compares the L1 Norm with a threshold to determine whether the non-key feature contains redundancy. If the L1 Norm exceeds the threshold, the corresponding values in ΔF^{n-1} are not redundant. Then, the comparator controls the two latch outputs these values in ΔF^{n-1} with corresponding F-IDs

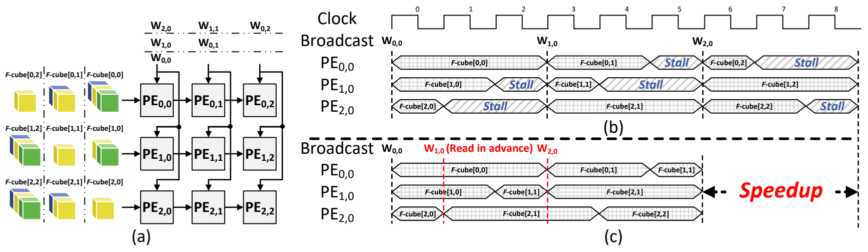


Figure 5: (a) Perform multiplication in a 3×3 array. (b) Unbalanced executing timing. (c) Balanced executing timing.

Table 2: Area Breakdown of InterArch.

Modules	Parameter	Area (mm^2)
Spatial array	16 × 16 PEs	2.086
Index generators	16×7 index generators	0.329
On-chip buffers	96KB *F*-cube-buffer 18KB *F-ID*-buffer 12KB Weight buffer	0.964
Total	UMC 28nm: Area = 3.379 mm^2	

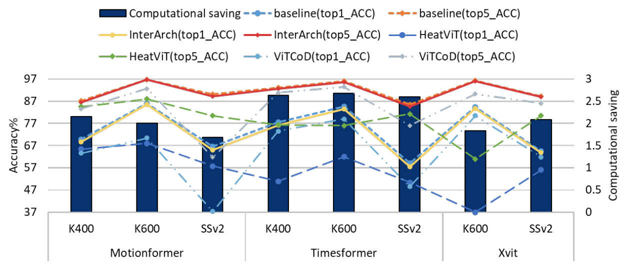
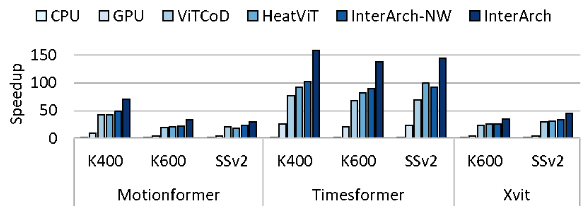
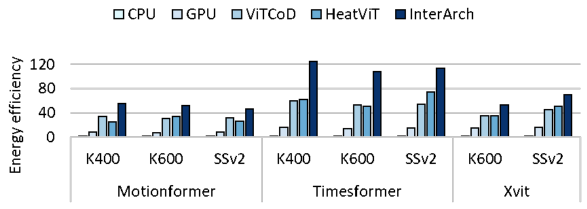


Figure 6: Comparisons of accuracy and computational saving.



(a) Speedup



(b) Energy efficiency

Figure 7: Speedup and Energy Efficiency of InterArch

to a bank of *F*-cube buffer according to the *F-ID*s. For example, non-zero elements in ΔF^0 and their *F-ID*s "0" will be sent into the first bank in the first bank group of *F*-cube and *F-ID*-cube buffers, respectively. As a result, the index generator picks up non-zero values in the ΔF^{n-1} and constructs *F*-cube and *F-ID*-cube during filling buffers. Moreover, to reduce extra latency caused by the index generator in the critical path, we employ multiple index generators.

5 EVALUATION

5.1 Evaluation Methodology

Software implementation. To verify the effectiveness of the InterArch algorithm, we select TimeSformer [2], Motionformer [14], and Xvit [3] with Kinetics-400 (K400), Kinetics-600 (K600), and SSv2 [9] datasets. These datasets are prevalent large-scale benchmarks for video-oriented tasks. All models are implemented based on the official source code and are executed using PyTorch v1.8.1. Since our method does not require fine-tuning, we directly utilize the official pre-trained models for validations. Moreover, the block size in the proposed hierarchical duplication detection is 4 and the batch size is set as 8 in the following experiments.

Hardware implementation. We implement the proposed InterArch architecture in Verilog and synthesize it by Synopsys Design Compiler to get the area and power consumption under 28nm technology, with a design frequency of 500MHz. For SRAM-based on-chip buffers, like the *F*-cube buffer, F-ID-buffer, and weight buffer, we use CACTI 7 [1] to model their area and power consumption. For off-chip DRAM, we follow the work [13], which models the high bandwidth memory based on HBM2 with 256 GB/s bandwidth and 1.2 pJ/bit energy for data access. Table 2 provides the detailed design parameter and area breakdown for the InterArch.

Platforms for comparison. We compare our architecture with widely used hardware platforms, including a server CPU (Intel Xeon Gold 6226R) and a GPU (Nvidia A100). For CPU and GPU, we respectively evaluate their executing latency via the Time library and CUDA Event integrated into PyTorch and real-time power by Nvidia-smi and power gadget. We also compare two state-of-the-art ViT accelerators including ViTCod [18] and HeatViT [6] in the same computation resources budget. For a fair comparison, we turn off their long-latency retraining processing.

5.2 Experiment Result

5.2.1 Accuracy, Performance, and Energy Efficiency. Fig 6 shows the accuracy and computational saving of baseline and the proposed InterArch on TimeSformer, Motionformer, and Xvit models with K400, K600, and SSv2 datasets. We can see that the InterArch drops 1% and 0.5% at the top 1 and top 5 accuracy on average. ViTCoD and HeatViT suffer sharp decreases of 9.7% and 18.1% at top 1 accuracy as well as 7.9% and 13.2% at top 1 and top 5 accuracy, respectively. Moreover, InterArch can gain 2.205× computational saving on average over baseline, which leaves the proposed InterArch architecture a large space to improve efficiency.

Fig. 7 (a) shows the speedup of the InterArch over CPU, GPU HeatViT, and InterArch without inter-column balance scheme (denoted as InterArch-NW). InterArch outperforms CPU, GPU, ViTCoD, and HeatViT by 81.5×, 40.6×, 1.87×, and 1.59×, respectively. Compared to CPU and GPU, the speedup comes from the proposed inter-feature sparsity algorithm discards redundancy in features. Compared to ViTCoD and HeatViT, the speedup originates from two aspects. First, the untrained feature sparsity modules are inefficient. For example, HeatViT can only prune 25% features on average, while InterArch explores 61% redundancy. Second, the sparsity modules introduce significant overhead. Moreover, compared to InterArch-NW, InterArch further gains 1.5× speedup as the inter-column balance scheme can fetch the follow-up data to free PEs to keep our computing resources busy.

As shown in Fig. 7 (b), the energy efficiency of InterArch is 77.7×, 18.1×, 1.81×, and 1.74× over CPU, GPU, ViTCoD, and HeatViT. Compared to CPU and GPU, InterArch only needs to compute and access the non-zero values, which reduces the energy consumed by the spatial array and off-chip memory. Compared to ViTCoD and HeatViT, energy savings can be attributed to the following factors: 1) the inter-feature sparsity algorithm explores more redundancy so that InterArch saves more computation and data movement energy; 2) As features and weights can be well reused in the spatial array, InterArch benefits from lower buffer access times.

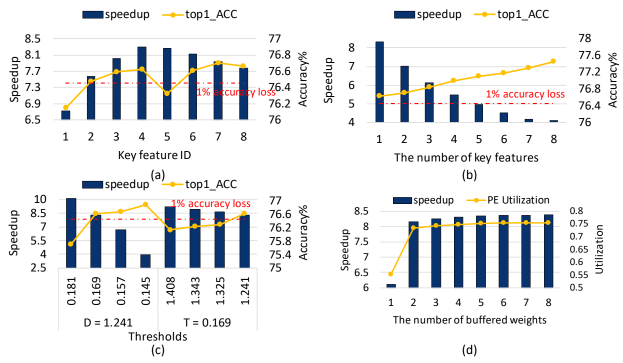


Figure 8: Design space exploration.

5.2.2 Design Space Exploration. In this section, we use TimeSformer on K400 to validate the speedup and accuracy when changing the location and the number of key features, given the batch size of eight. Fig. 8 (a) shows the impact of the key feature location. The speedup is normalized to GPU. We change the key feature from the first feature (numbered as 1) to the last feature (numbered as 8). From this plot, we find that the fourth feature gains maximal speedup while maintaining negligible accuracy loss (1%). As centralized features have smaller average time gap between others, setting these features as key features increases the opportunity for deduplication, leading to higher performance. Thus, we select the fourth feature as the key feature to achieve the best performance.

Fig. 8 (b) explores the impact of the number of key features. We diverse the number from one to eight. As the number rises, the plot shows that accuracy loss and speedup progressively decrease. As we only perform sparsity on non-key features, more key features mean we reserve more information about features, causing higher accuracy but lower performance. Overall, we can find that setting one feature within each batch is enough to gain satisfactory results.

Fig. 8 (c) plots the impact of thresholds D and T, mentioned in the proposed algorithm. We fix one threshold and vary the other. With the thresholds decreasing, we gain higher accuracy and lower speedup since a larger threshold misses more opportunities for deduplication. In summary, we set the D as 1.241 and T as 0.168 as they accomplish ≤1% accuracy loss and gain best performance.

Fig. 8 (d) depicts the impact of the number of buffered weights in each PE. We increased the number from one to eight. With the number increasing, the inter-column balance scheme can read more F-Vec in advance to perform a more balanced dataflow. Thus, the experiments show higher speedup and PE utilization. However, the higher performance is at the cost of more weight registers in PEs. In a word, we prefer the number of buffered weights as two due to the significant performance improvement.

6 CONCLUSION

This paper proposes InterArch, a framework for efficient video Transformer (VidT). On the algorithm side, InterArch introduces a post-training inter-feature sparsity algorithm to reduce the redundancy between features. On the dataflow side, InterArch stack features along temporal dimension into a dense cube. By performing dense computation between dense cube and weight matrix, InterArch avoids hardware control overhead for sparse calculations. On the architecture side, InterArch houses a spatial array with slightly modified PE from the traditional dense accelerator. Experiments show that InterArch can deliver satisfactory performance gain while maintaining accuracy.

REFERENCES

[1] Balasubramonian et al. 2017. CACTI 7: New tools for interconnect exploration in innovative off-chip memories. *TACO* 14, 2 (2017), 1–25.
[2] Bertasius et al. 2021. Is space-time attention all you need for video understanding?. In *ICML*, Vol. 2. 4.
[3] Bulat et al. 2021. Space-time Mixing Attention for Video Transformer. In *NeurIPS*.
[4] Yu-Hsin Chen et al. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE JETCAS* 9, 2 (2019), 292–308.
[5] Jyotikrishna Dass et al. 2023. Vitality: Unifying low-rank and sparse approximation for vision transformer acceleration with a linear taylor attention. In *HPCA*. IEEE, 415–428.
[6] Peiyan Dong et al. 2023. Heatvit: Hardware-efficient adaptive token pruning for vision transformers. In *HPCA*. IEEE, 442–455.
[7] Dosovitskiy et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[8] Hehe Fan et al. 2018. Watching a small portion could be as good as watching all: Towards efficient video classification. In *IJCAI*.
[9] Raghav Goyal et al. 2017. The" something something" video database for learning and evaluating visual common sense. In *ICCV*. 5842–5850.
[10] Kensho Hara et al. 2017. Learning spatio-temporal features with 3d residual networks for action recognition. In *ICCV*. 3154–3160.
[11] Will Kay et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).
[12] Tianyang Lin et al. 2022. A survey of transformers. *AI Open* (2022).
[13] O'Connor et al. 2017. Fine-grained DRAM: Energy-efficient DRAM for extreme bandwidth systems. In *MICRO*. IEEE, 41–54.
[14] Mandela Patrick et al. 2021. Keeping your eye on the ball: Trajectory attention in video transformers. *NeurIPS* 34 (2021), 12493–12506.
[15] Yongming Rao et al. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. 34 (2021), 13937–13949.
[16] Wodajo et al. 2021. Deepfake video detection using convolutional vision transformer. *arXiv preprint arXiv:2102.11126* (2021).
[17] Yifan Xu et al. 2022. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *AAAI*, Vol. 36. 2964–2972.
[18] Haoran You et al. 2023. Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design. In *HPCA*. IEEE, 273–286.