

TiMM-ViT: Token-Importance-Aware Token Merging with Multi-Exit Training and an FPGA-based Vision Transformer Accelerator for Image Classification

Soomin Rho
smrho@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Sang-Ki Park
skpark1101@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Subin Park
subinpark@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Kwangrae Kim
kksilver91@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Min-Gwon Song
mgsong1005@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Ki-Seok Chung*
kchung@hanyang.ac.kr
Department of Electronic
Engineering, Hanyang University
Seoul, Korea

Abstract

Token merging (ToMe) reduces the computational cost of Vision Transformers (ViTs) by merging similar tokens. However, ToMe may inadvertently merge critical tokens, leading to accuracy degradation. Furthermore, the sequential execution of ViT and ToMe operations incurs increased latency on edge GPUs. To address these challenges, we propose TiMM, a token-importance-aware merging method that preserves essential tokens by considering both token importance and similarity. We also present a dedicated FPGA-based accelerator designed to exploit the concurrent execution between ViT and TiMM. Experimental results across DeiT-Tiny, Small, and Base demonstrate that TiMM achieves accuracy improvements of 0.23%–0.73% over ToMe, respectively, within a 1% accuracy drop margin relative to the baseline. When implemented on the Xilinx ZCU102, it demonstrates speedups of $1.45\times$ – $10\times$, along with energy efficiency gains of $1.03\times$ – $3.1\times$ compared to the NVIDIA Jetson TX2.

CCS Concepts

• Hardware → Hardware accelerators.

Keywords

Vision Transformers, Token Merging, Hardware Accelerator, FPGA

ACM Reference Format:

Soomin Rho, Sang-Ki Park, Subin Park, Kwangrae Kim, Min-Gwon Song, and Ki-Seok Chung. 2026. TiMM-ViT: Token-Importance-Aware Token Merging with Multi-Exit Training and an FPGA-based Vision Transformer Accelerator for Image Classification. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '26)*, August 05–07, 2026, Evanston, IL, USA. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3816440.3818568>

*Corresponding author.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ISLPED '26, Evanston, IL, USA

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2748-1/2026/08
<https://doi.org/10.1145/3816440.3818568>

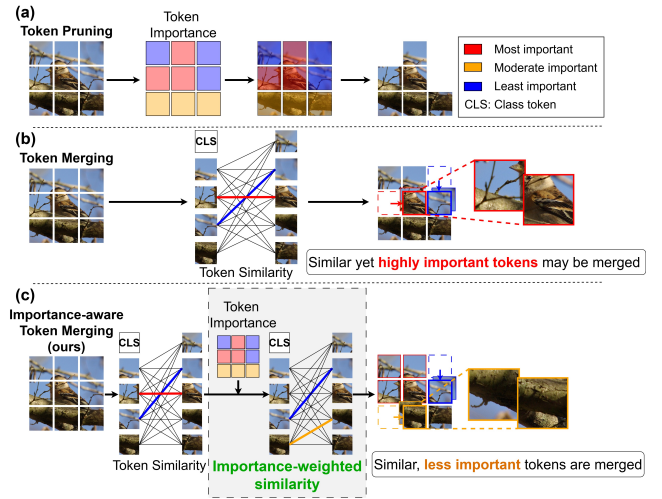


Figure 1: Comparison of existing token reduction methods: (a) token pruning, (b) token merging, and (c) importance-aware token merging.

1 Introduction

The Vision Transformer (ViT) is a DNN model that adapts the Transformer architecture for computer vision (CV) [4]. By leveraging the attention mechanism to capture global context, it has demonstrated superior performance across various CV tasks, including image classification, object detection, and serving as the vision encoder for vision-language models [4, 8, 12]. Because the computational cost and memory requirements of a ViT block, which consists of a multi-head attention (MHA) block and an MLP block, increase rapidly with the number of input tokens. This issue limits the widespread deployment of ViT on resource-constrained devices. To mitigate this problem, methods to reduce the number of tokens, such as token pruning and token merging, have been actively proposed in various CV tasks [1, 11, 13].

Token pruning, as shown in Fig. 1(a), is an importance-based token reduction method by removing tokens deemed less important. It can be broadly categorized into two types. One type utilizes

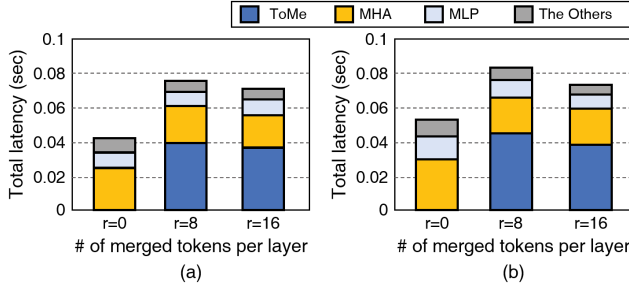


Figure 2: Latency breakdown of (a) ViT-Small and (b) ViT-Base on the Jetson TX2.

the attention of the class ([CLS]) token as a measure of token importance [9, 14]. The other type introduces an additional trainable network to predict token importance [3, 7]. Many of the existing token pruning studies require either finetuning or training from scratch to mitigate the uncertainty of [CLS] attention and construct prediction modules, which makes them difficult to deploy for practical applications.

On the other hand, token merging, as shown in Fig. 1(b), is a method to reduce the number of tokens by merging them based on similarities among tokens [1, 5]. Of these, ToMe [1] is a well-known method that reduces tokens by merging similar ones while minimizing information loss. Its ability to preserve information allows for off-the-shelf use without finetuning, making it highly practical for deployment.

However, there are two major limitations when applying ToMe to real-time image classification tasks on resource-constrained edge devices. First, since ToMe merges tokens based on their mutual similarity, it may inadvertently merge tokens critical for class prediction. This leads to the loss of essential discriminative information, consequently resulting in accuracy degradation. Second, as illustrated in Fig. 2, applying ToMe to edge GPUs actually increases end-to-end latency. This is primarily because: 1) the core operations of ToMe, such as sorting and matrix reordering, are computationally inefficient for GPU acceleration; and 2) ToMe does not exploit parallelism to hide latency overhead. To overcome these limitations, we propose TiMM, a token-importance-aware token merging method, along with its dedicated FPGA-based hardware accelerator. The contributions of this study are as follows:

- We propose **TiMM**, a token merging method that integrates both token pair importance and similarity into a unified criterion. TiMM extracts the importance of individual tokens from [CLS] attention and computes the token pair importance between two token sets. And we introduce a new scoring method, importance-weighted similarity (IWS), to consider both similarity and importance of each token pair, thereby preventing important tokens from being merged, as shown in Fig. 1(c).
- We introduce the **TiMM framework**, designed to refine token importance estimation by leveraging multi-exit training [6]. This approach trains intermediate class tokens to align more closely with the final class token, thereby enhancing the reliability of the importance used for merging decisions.
- We propose an **FPGA-based TiMM accelerator** that is capable of processing both ViT blocks and TiMM in parallel during inference. The proposed design consists of three primary components:

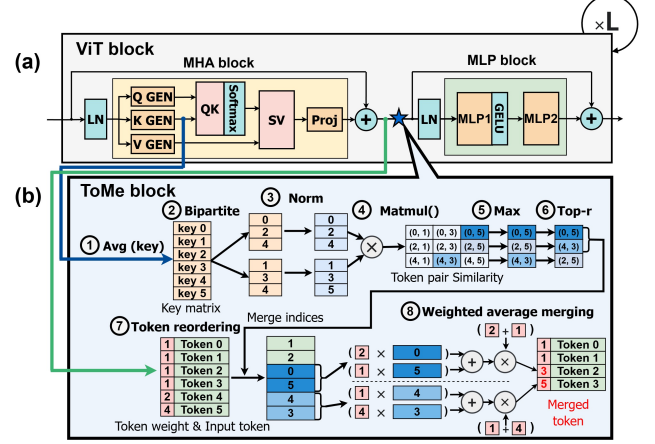


Figure 3: Illustration of (a) the ViT block and (b) the ToMe block.

a ViT accelerator, a multi-channel DMA, and a TiMM module. These components operate in a tightly-coupled and concurrent manner, enabling effective latency hiding and consequently minimizing the overall end-to-end latency.

Experimental results across DeiT-Tiny, Small, and Base demonstrate that TiMM yields accuracy improvements of 0.23%, 0.49%, and 0.73%, respectively, compared to ToMe, while maintaining the accuracy drop within a 1% margin relative to the baseline. Furthermore, when evaluated on the Xilinx ZCU102, the TiMM accelerator achieves speedups of 10×, 4×, and 1.45× and energy efficiency (FPS/W) gains of 3.1×, 2.4×, and 1.03× over the NVIDIA Jetson TX2, respectively.

2 Background

2.1 Vision Transformer

A ViT model consists of an embedding layer that partitions the image into patches to generate a token sequence, L ViT blocks that iteratively process these tokens to produce the [CLS] token as a global representation, and a head block that facilitates final class prediction based on the resulting [CLS] token. The internal structure of the ViT block is illustrated in Fig. 3(a), and the core components are Layernorm (LN), Add, MHA block, and MLP block. The MHA block generates query (Q), key (K), and value (V) via three linear projection layers, which are subsequently partitioned into multiple heads. Within each head, the scaled dot-product attention is computed through QK^T , $Softmax$, and SV operations. The resulting $Softmax$ output (S), termed the attention matrix, characterizes the contextual relationships among tokens.

2.2 ToMe

ToMe [1] is a similarity-based token reduction method that calculates similarities between tokens and merges the r most similar token pairs based on them. ToMe consists mainly of 1) similarity estimation, 2) bipartite matching, and 3) weighted merging processes, and Fig. 3(b) shows all steps (①~⑧). First, for similarity estimation, ToMe averages all heads of K in the MHA block, alternately partitions them into two sets, A and B (①, ②), and computes cosine similarities between them, resulting in a token-pair similarity matrix sim (③, ④). In bipartite matching, ToMe extracts the largest

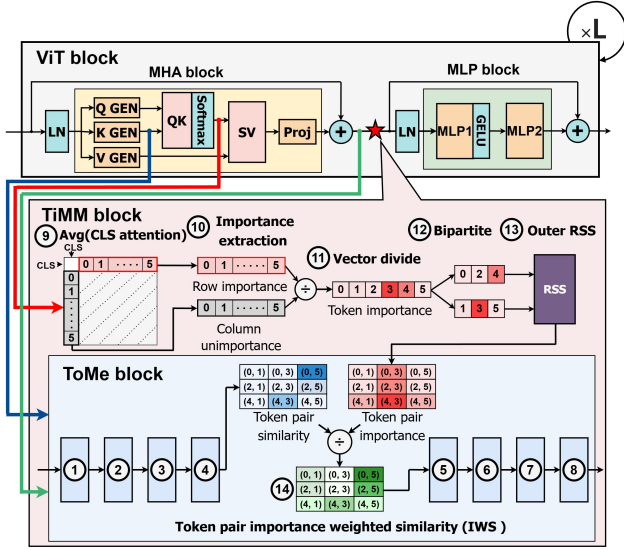


Figure 4: Architecture overview of TiMM.

value of each row of the sim (5). Based on the extracted similarity values, tokens corresponding to the top r pair indices are designated as merging candidates (6). In the weighted merging process, ToMe receives both the MHA block's output tokens and the token weights—which represent the number of tokens merged into each current token—as inputs (green arrow in Fig. 3(b)). Next, ToMe extracts the r pairs of merging candidates from the token set and realigns the remaining tokens (7). The selected r token pairs are merged pairwise into single tokens via weighted averaging based on their current weights; concurrently, the weights themselves are updated and tracked for subsequent layers (8).

3 TiMM & TiMM Framework

3.1 TiMM

TiMM is a method that preserves crucial tokens during the merging process by leveraging [CLS] attention as a measure of token importance. To identify merging candidates, we propose Importance-Weighted Similarity (IWS), which integrates token-pair importance and similarity into a unified criterion.

Fig. 4 shows the overall architecture of TiMM, which incorporates the ToMe process (1)–(8). TiMM specifically extracts the row and column vectors corresponding to the [CLS] token from the attention matrix to capture the relationships between the [CLS] token and all other tokens (9, 10). We define the row [CLS] attention as row importance (Imp_{row}), as it signifies each token's contribution to the generation of the [CLS] token during the SV operation. Consequently, a higher Imp_{row} indicates that the corresponding token contributes more significantly to the class representation. Conversely, the column [CLS] attention is regarded as column unimportance ($UnImp_{col}$), representing the degree to which each token is influenced by the [CLS] token. A larger $UnImp_{col}$ indicates that the token loses more of its unique information as it is updated by incorporating a high proportion of information from the [CLS] token during the MHA process. To integrate these two factors, we define the final token importance as $Imp = Imp_{row}/UnImp_{col}$ (11).

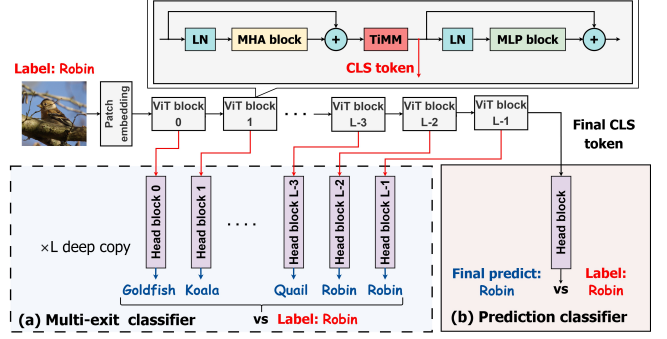


Figure 5: Overview of the TiMM framework: (a) multi-exit classifier and (b) prediction classifier.

TiMM constructs a token-pair importance matrix, Imp_{pair} , which quantifies the combined importance of token pairs from sets A and B . To compute Imp_{pair} , we introduce the Outer Root-Sum-of-Squares (Outer RSS) as a method for aggregating the importance scores of two individual tokens. First, the token importance vector Imp is partitioned into $Imp_A \in \mathbb{R}^m$ and $Imp_B \in \mathbb{R}^n$, consistent with the partitioning for bipartite matching (12). Then, the token-pair importance matrix Imp_{pair} is derived using the Outer RSS operation (13). The Outer RSS and the resulting Imp_{pair} are defined as follows:

$$Imp_{pair} = \text{OuterRSS}(Imp_A, Imp_B) \in \mathbb{R}^{m \times n} \quad (1)$$

$$Imp_{pair_{i,j}} = \sqrt{Imp_{A_i}^2 + Imp_{B_j}^2} \quad (i=1, \dots, m, j=1, \dots, n).$$

Outer RSS treats the two token importances as orthogonal dimensions that remain independent of one another, calculating their total magnitude within a joint importance space. We adopt RSS as it achieved superior Top-1 accuracy compared to alternative aggregation methods, such as element-wise summation and multiplication.

Finally, TiMM adopts IWS as the merging score (14), which is calculated by dividing the token-pair similarity by the token-pair importance, as defined in (2).

$$IWS_{i,j} = \frac{sim_{pair_{i,j}}}{Imp_{pair_{i,j}}} \quad (i=1, \dots, m, j=1, \dots, n) \quad (2)$$

Since IWS accounts for both token-pair similarity and importance, it assigns lower merging scores to important pairs, ensuring their preservation during the merging process.

3.2 TiMM Framework

TiMM can be applied in a plug-and-play manner without finetuning, yet its performance can be further enhanced through an additional finetuning process. Given that TiMM leverages token importance, we employed multi-exit training—a method to enhance the reliability of importance scores while simultaneously boosting classification performance. Multi-exit training is a strategy that supervises class predictions using intermediate [CLS] tokens from each ViT block. Consequently, the cosine similarity between intermediate and final [CLS] tokens is enhanced, aligning the representations across layers. This alignment improves the reliability of token importance metrics used for token reduction [6, 10].

Fig. 5 illustrates the overall architecture of the TiMM framework. Specifically, Fig. 5(b) shows the prediction classifier, which generates class predictions for an input image using the final [CLS] token and is optimized using the cross-entropy (CE) loss ($\mathcal{L}_{pred}$ in Eq. (3)).

In contrast, Fig. 5(a) presents the multi-exit classifier designed for multi-exit training. As proposed in [6], we instantiate L copies of the prediction classifier and attach each to the corresponding intermediate [CLS] token of the respective ViT blocks. During the training, each intermediate [CLS] token is processed through its head to generate a label prediction. The objective function, $\mathcal{L}_{\text{multi}}$ in Eq. (3), is then defined as the average cross-entropy (CE) loss across all heads.

Total loss for the TiMM framework, $\mathcal{L}_{\text{total}}$, is defined as Eq. (4) where the hyperparameter α adjusts the loss ratio between the prediction classifier and the multi-exit classifier, and λ controls the training intensity of the multi-exit classifier. The multi-exit classifier is not utilized during inference.

$$\mathcal{L}_{\text{pred}} = \text{CE}(p_{\text{final}}, y), \quad \mathcal{L}_{\text{multi}} = \frac{1}{L} \sum_{i=1}^L (\text{CE}(p_i, y)) \quad (3)$$

$$\mathcal{L}_{\text{total}} = (1 - \alpha) \mathcal{L}_{\text{pred}} + \alpha \lambda \mathcal{L}_{\text{multi}} \quad (4)$$

4 TiMM Accelerator

Despite the higher computational complexity of TiMM compared to ToMe, the proposed FPGA-based TiMM accelerator effectively hides its execution latency by overlapping it with the ViT block's processing. As shown in Fig. 6, the TiMM accelerator consists of 1) a ViT accelerator for processing ViT blocks, 2) a multi-channel DMA for off-chip data transfer, and 3) a TiMM module designed to support both LN and TiMM. These three components are designed to track each other's operational states, facilitating synchronization without any conflicts among their dependencies. All datapaths in Fig. 6 employ data streaming, where 32-dimensional vectors or scalars serve as words.

4.1 ViT Accelerator & Multi-channel DMA

The main components of the ViT accelerator in Fig. 6(a) are: 1) on-chip memories of token and weight & bias, 2) 64×32 MAC array unit for matrix multiplication, 3) vector add unit for bias addition in linear projection, and 4) a SIMD unit dedicated to nonlinear functions, including weight and bias operations of LN and the weighted-average merging of tokens.

The ViT accelerator keeps all tokens in on-chip memory at all times, eliminating off-chip memory transfers of intermediate tokens during inference. To achieve this, the token memory is organized into 6 token banks, each of which buffers a token matrix (e.g., up to a 197×768 matrix for ViT-Base). The stream pattern generator within the token memory supports diverse streaming patterns for token reordering and weighted-average merging. To facilitate these operations, it receives skip masks, candidate indices, and token weights from the TiMM module. The weight & bias memory stores parameters for linear projection and LN, employing a double-buffering mechanism to effectively overlap memory access latency during computation. Furthermore, by retaining all tokens in on-chip memory, each fetched weight and bias tile is fully reused across the entire token set. This design ensures that each parameter tile is transferred from off-chip memory only once, thereby minimizing off-chip memory bandwidth requirements.

A multi-channel DMA is employed for data transfer between the Processing System (PS) and Programmable Logic (PL) of the

ZCU102. The PS of the ZCU102 provides only up to a 128-bit AXI-HP port per channel, which lacks sufficient memory bandwidth. Furthermore, as the number of tokens decreases due to the TiMM, the computational load of the ViT reduces, causing memory access to become the major bottleneck. To mitigate this problem, the TiMM accelerator employs a 4-channel DMA with four AXI ports, which operates independently of the ViT block to hide memory latency.

4.2 TiMM Module

The TiMM module effectively hides its operational latency by running concurrently with the ViT accelerator. The TiMM module consists of 1) a TiMM memory, 2) a token merge manager, and 3) a TiMM engine. The TiMM memory is composed of memory blocks for *sim*, *Imp_{row}*, *UnImp_{col}*, and *index-value pairs*. Among these, the *UnImp_{col}* memory is connected to the Scalar Unit of the TiMM engine via the scalar stream, while the *index-value pairs* memory stores the outputs of the max operation (⑤) in Fig. 3, serving candidates for the *Top-r* selection (⑥).

The Token merge manager comprises a skip mask generator for token reordering, a candidate index FIFO, and a token weight table. It stores the *index-value pairs* resulting from the *Top-r* operation (⑥) into the candidate index FIFO while concurrently generating a skip mask. Guided by this mask, the token weight table sequentially reorders and updates the stored token weights according to the candidate indices. The skip masks, candidate indices, and token weights for token merging are delivered to the ViT accelerator to merge tokens (⑦, ⑧ in Fig. 3).

Fig. 6(b) illustrates the architecture of the TiMM engine, which executes LN and TiMM's diverse computations via configuration signals. Its internal structure includes: 1) a VPU featuring a vector adder cascaded with a vector multiplier to support various vector operations, including squaring, 2) a reduction tree that reduces vectors to scalars through add or compare operations to support *sum*, *max*, and *Top-r*, 3) a scalar unit dedicated to executing essential TiMM scalar operations, including the calculation of normalization factors (μ , σ , and ℓ_2), the squaring and inversion of *UnImp_{col}*, and the generation of averaged weights, 4) a time division multiplexing (TDM) buffer and a broadcast buffer that facilitate data exchange between vector and scalar streams, which can be bypassed to the input of the VPU to be used as operands and 5) a shuffle buffer dedicated to partitioning *Imp_{row}*.

For resource efficiency, the TiMM engine implements only a single LUT-based square-root and inverse unit within its scalar unit. Consequently, such vector operations must be decomposed into scalars and processed via the TDM buffer, which incurs a significant cycle overhead. To mitigate this latency, we propose an inverse-square IWS, denoted as $\widehat{\text{IWS}}$:

$$\widehat{\text{IWS}}_{i,j} = \frac{\text{imp}_i^2 + \text{imp}_j^2}{\text{sim}_{ij}|\text{sim}_{ij}| + \epsilon}, \quad (i=1, \dots, m, j=1, \dots, n). \quad (5)$$

$\widehat{\text{IWS}}$ effectively performs the same ranking function as the original IWS, since the primary objective of IWS is to compare magnitudes for token selection rather than computing absolute values. However, substituting IWS with $\widehat{\text{IWS}}$ inverts the magnitude relationships and introduces potential discontinuities near zero, which can affect monotonic comparisons. To resolve these issues, as illustrated in

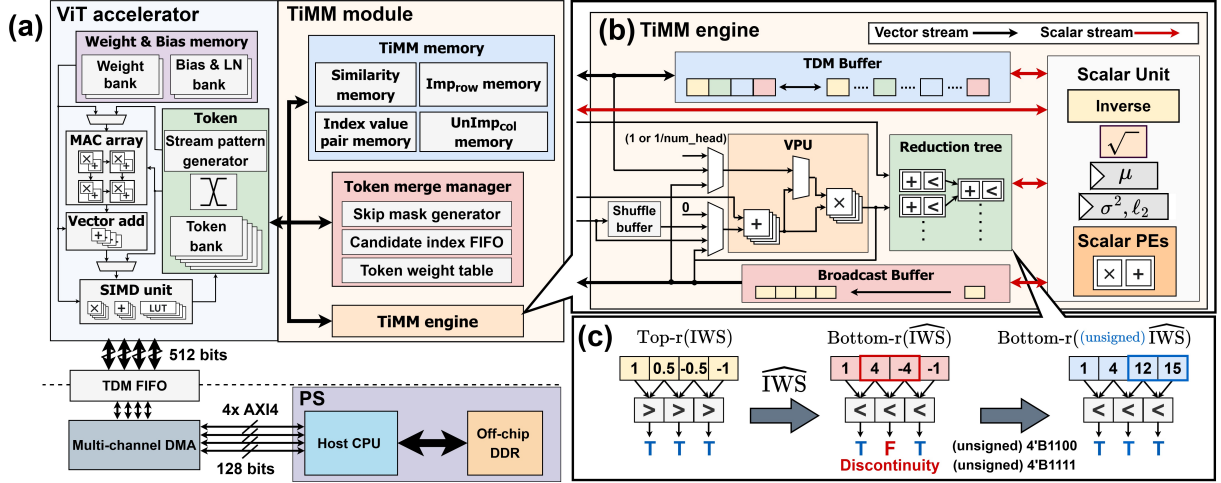


Figure 6: Overall architecture of the TiMM accelerator: (a) ViT accelerator, multi-channel DMA and TiMM module, (b) architecture of the TiMM engine, and (c) magnitude reversal and discontinuities problem and the unsigned-int-based comparator.

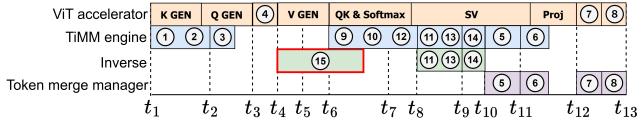


Figure 7: Timing diagram of the TiMM accelerator with $\widehat{\text{IWS}}$.

Fig. 6(c), we replace $\max(\text{IWS})$ and $\text{Top-r}(\text{IWS})$ to $\min(\widehat{\text{IWS}})$ and $\text{Bottom-r}(\widehat{\text{IWS}})$, and utilize an unsigned integer comparator.

As illustrated in Fig. 7, $\widehat{\text{IWS}}$ scheme applies the inverse operation to sim at t_4 (15). This contrasts with the IWS method, which must wait until imp is processed at t_9 before initiating the operation. This scheduling allows the engine to execute the inverse operation at earlier cycles. Consequently, the execution of TiMM is tightly coupled and synchronized with the MHA block, ensuring that the token merging overhead is completely hidden.

5 Evaluation

5.1 Experimental Setup

To evaluate the TiMM, we employed three ViT variants—DeiT-Tiny, Small, and Base [15]—integrated with TiMM on the ImageNet-1K dataset [2], and used pretrained models provided by [16]. To compare with ToMe, we evaluated both finetuning-free and finetuned accuracies using four RTX 3090 GPUs. We additionally compared the finetuned accuracy with prior token reduction methods [1, 3, 5, 7, 9, 14]. Finetuning was conducted over 30 epochs, with a grid search performed for α (ranging from 0.1 to 0.2) and λ (from 1.5 to 3.5). The TiMM accelerator was developed in Verilog HDL utilizing 16-bit fixed-point arithmetic. Our design was implemented on the Xilinx ZCU102 platform operating at 200 MHz via Vivado 2021.2 with power optimization strategies, and performance was evaluated as a standalone application within the Vitis 2021.2. We compare our design against HeatViT [3] and its baseline, the Jetson TX2 edge GPU (TX2), for L blocks of ViTs. To ensure a fair evaluation, HeatViT was implemented on the same FPGA using identical data precision, while the TX2 was selected as it leverages

Table 1: Comparison of accuracy (%) and GMACs for different numbers of token merges (r) on finetuning-free DeiT models.

ViT model	# of merge	$r=0$	$r=4$	$r=8$	$r=12$	$r=16$
DeiT-Small	GMACs	4.60	4.01	3.43	2.85	2.30
	ToMe (Acc %)	79.80	79.74	79.42	79.08	78.14
	TiMM (Acc %)	79.80	79.77	79.46	79.19	78.29
DeiT-Base	GMACs	17.60	15.34	13.12	10.92	8.78
	ToMe (Acc %)	81.98	81.66	81.32	80.28	77.79
	TiMM (Acc %)	81.98	81.75	81.32	80.54	78.14

the same TSMC 16-nm FinFET process technology as the ZCU102. Power consumption was estimated using the Vivado Power Report. To ensure a fair comparison, only the on-chip power of the TX2 was considered, as reported through the Tegrastats utility.

5.2 Token Reduction Evaluation

Table 1 presents the accuracy and GMACs for various r values across finetuning-free DeiT models. TiMM outperforms ToMe in accuracy under equivalent computational budgets (GMACs), achieving a peak improvement of 0.35% on DeiT-Base at $r = 16$. These results indicate that TiMM offers superior plug-and-play performance compared to ToMe. This is because TiMM’s token-importance-aware similarity prevents important tokens from being merged.

Table 2 compares the accuracy and GMACs of our proposed method against prior token reduction studies using finetuned ViT models. With the pretrained DeiT models serving as our baselines, we configured r for TiMM with multi-exit training to ensure that any accuracy degradation was strictly limited to a 1% margin relative to the baseline. Within this margin, TiMM consistently outperforms ToMe, achieving accuracy improvements of 0.23%, 0.49%, and 0.73% for DeiT-Tiny, Small, and Base, respectively, at equivalent GMAC levels. Additionally, at similar GMAC reduction ratios, TiMM achieves a superior accuracy trade-off compared to previous token-reduction methods. Table 3 presents an ablation study investigating the impact of multi-exit training on ToMe and TiMM. While both TiMM (w/o multi-exit training) and ToMe (w/ multi-exit

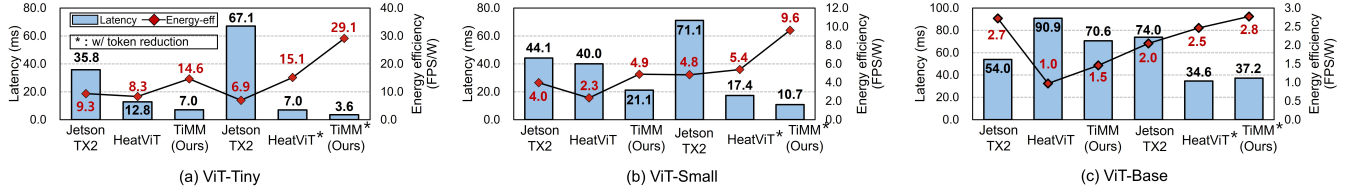


Figure 8: Latency and energy efficiency comparison across diverse hardware using token reduction methods.

Table 2: Comparison with various token reduction methods with finetuning.

ViT model	Method	GMACs (Ratio)	Accuracy (Diff)
DeiT-Tiny	Baseline	1.26 (x1.00)	72.17 (0.00)
	ToMe	0.63 (x0.50)	71.35 (-0.82)
	CP-ViT	0.71 (x0.57)	71.24 (-0.93)
	Heat-ViT	0.90 (x0.72)	72.15 (-0.02)
	TiMM+ME* (Ours)	0.63 (x0.50)	71.58 (-0.59)
DeiT-Small	Baseline	4.60 (x1.00)	79.80 (0.00)
	ToMe	2.30 (x0.50)	78.83 (-0.97)
	EViT	2.30 (x0.50)	78.30 (-1.50)
	GTP-ViT	2.30 (x0.50)	78.60 (-1.20)
	CP-ViT	2.64 (x0.57)	79.08 (-0.72)
	Heat-ViT	2.64 (x0.57)	79.38 (-0.42)
	TiMM+ME* (Ours)	2.30 (x0.50)	79.32 (-0.48)
DeiT-Base	Baseline	17.60 (x1.00)	81.98 (0.00)
	ToMe	9.83 (x0.56)	80.37 (-1.61)
	CP-ViT	10.27 (x0.58)	81.10 (-0.88)
	TMViT	11.06 (x0.63)	81.50 (-0.48)
	Heat-ViT	10.49 (x0.60)	81.08 (-0.90)
	TiMM+ME* (Ours)	9.83 (x0.56)	81.10 (-0.88)

*ME: Multi-Exit.

Table 3: Ablation studies of multi-exit training.

ViT model	Method	GMACs	Accuracy (Diff)
DeiT-Small	ToMe	2.30	78.83 (+0.0)
	ToMe+ME*	2.30	79.09 (+0.26)
	TiMM	2.30	78.94 (+0.11)
	TiMM+ME* (Ours)	2.30	79.32 (+0.49)
DeiT-Base	ToMe	9.83	80.37 (+0.0)
	ToMe+ME*	9.83	80.78 (+0.41)
	TiMM	9.83	80.62 (+0.25)
	TiMM+ME* (Ours)	9.83	81.10 (+0.73)

*ME: Multi-Exit.

training) slightly outperformed the ToMe, TiMM combined with multi-exit training, achieved the highest overall accuracy. This is because the TiMM framework improves the reliability of token importance, which enables it to better select important token pairs.

5.3 TiMM Accelerator Evaluation

Fig. 8 shows the performance comparison of the TiMM accelerator with the TX2 and HeatViT. The TiMM accelerator utilizes our proposed merging scheme, while the TX2 adopts ToMe (with an identical r) and HeatViT employs its native pruning strategy. Even in the absence of token reduction, the TiMM accelerator delivers speedups of 5 $\times$ and 2 $\times$ for ViT-Tiny and ViT-Small, respectively, over the TX2. With token reduction, TiMM achieves speedups of 10 $\times$, 4 $\times$, and 1.45 $\times$, along with energy efficiency (FPS/W) gains of

Table 4: Resource utilization and power consumption of the TiMM accelerator on a ZCU102 FPGA board.

Resource usage (utilization)				Power
CLB LUTs	CLB registers	BRAM tile	DSPs	(W)
193.3K (70.53%)	233.2K (42.6%)	699 (76.64%)	2154 (85.5%)	9.69

3.1 $\times$, 2.4 $\times$, and 1.03 $\times$ for ViT-Tiny, ViT-Small, and ViT-Base, respectively. These significant improvements are attributed to the TiMM accelerator’s architecture, which effectively overlaps the execution of its three internal components, and the resource-efficient design of the TiMM module itself.

TiMM achieves approximately 2 $\times$ and 1.6 $\times$ improvements in both speedup and energy efficiency on ViT-Tiny and ViT-Small, respectively, compared to HeatViT. For ViT-Base, HeatViT marginally outperforms TiMM. However, HeatViT employs a more aggressive token reduction strategy, operating at 7.75 GMACs (as reported in the hardware results of [3]) compared to 9.83 GMACs for TiMM. Notably, TiMM maintains the accuracy drop within a 1% margin relative to the baseline. Furthermore, while HeatViT requires over 270 training epochs, our method achieves convergence in only 30 epochs. Finally, TiMM even supports plug-and-play deployment without the need for additional finetuning, offering superior practicality for real-world applications.

Table 4 shows the FPGA resource utilization of TiMM. The ViT accelerator used 2048 DSPs for the 64x32 MAC array and 516 and 114 of BRAM tiles for token memory and weight & bias memory, respectively. A TiMM module is implemented using approximately 9.5K CLB LUTs, 61 BRAM tiles, and 36 DSPs. The power consumption was estimated to be 9.69 W.

6 Conclusion

This study proposed 1) TiMM, a token-importance-aware token merging method, 2) the TiMM framework, which enhances the reliability of token importance, and 3) the TiMM accelerator, which hides the latency of TiMM by overlapping it with the latency of the ViT blocks. Compared to ToMe, TiMM improves accuracy by 0.23%–0.73% while strictly limiting accuracy loss to within 1%. Implementation on the Xilinx ZCU102 reveals that TiMM provides up to 10 $\times$ speedup and 3.1 $\times$ energy efficiency relative to the NVIDIA Jetson TX2, with consistent gains across all model scales.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (RS-2024-00409492).

The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

References

- [1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token Merging: Your ViT But Faster. In *ICLR*.
- [2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. doi:10.1109/CVPR.2009.5206848
- [3] Peiyan Dong, Mengshu Sun, Alec Lu, Yanyue Xie, Kenneth Liu, Zhenglun Kong, Xin Meng, Zhengang Li, Xue Lin, Zhenman Fang, et al. 2023. Heatvit: Hardware-efficient adaptive token pruning for vision transformers. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 442–455.
- [4] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
- [5] Zhanzhou Feng and Shiliang Zhang. 2023. Efficient vision transformer via token merger. *IEEE Transactions on Image Processing* 32 (2023), 4156–4169.
- [6] Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. 2024. On layer-wise representation similarity: Application for multi-exit models with a single classifier. In *NeurIPS 2024 Workshop on Symmetry and Geometry in Neural Representations*.
- [7] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. 2022. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European conference on computer vision*. Springer, 620–640.
- [8] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. 2022. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*. Springer, 280–296.
- [9] Y Liang, C Ge, Z Tong, Y Song, P Xie, et al. 2022. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*.
- [10] Dongyang Liu, Meina Kan, Shiguang Shan, and Xilin Chen. 2024. A Simple Romance Between Multi-Exit Vision Transformer and Token Reduction. In *ICLR*.
- [11] Yifei Liu, Mathias Gehrig, Nico Messikommer, Marco Cannici, and Davide Scaramuzza. 2024. Revisiting token pruning for object detection and instance segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2658–2668.
- [12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmlR, 8748–8763.
- [13] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. 2025. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 22857–22867.
- [14] Zhuoran Song, Yihong Xu, Zhezhi He, Li Jiang, Naifeng Jing, and Xiaoyao Liang. 2022. Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction. *arXiv preprint arXiv:2203.04570* (2022).
- [15] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.
- [16] Ross Wightman et al. 2019–. PyTorch Image Models (timm). <https://github.com/rwightman/pytorch-image-models>.