2

From Complexity to Efficiency: Pruning Vision Transformers in Practice

Martyna Poreba, Ahmed Soulmani, Victor Bercy, and Michal Szczepanski

Université Paris-Saclay, CEA, List, France

Abstract

Vision Transformers (ViTs) achieve state-of-the-art performance in semantic segmentation. However, their high computational cost, especially when processing high-resolution images, remains a major obstacle for real-time and embedded applications. In this work, we provide a taxonomy of existing pruning approaches and observe that many efficient methods overlook fine-grained hardware and latency characteristics. We further investigate selected pruning strategies that merge redundant information at the patch or token level to effectively reduce computational complexity. To assess their practical relevance, we evaluate deployment performance on an NVIDIA Jetson Orin AGX platform, analysing trade-offs between latency, computational workload, and accuracy. While token pruning is commonly motivated by the assumption that fewer tokens lead to lower latency, our results reveal that this relationship is often non-linear due to GPU workload scheduling, framework overhead, and kernel design constraints.

Keywords: Vision Transformer, Semantic Segmentation, Token Pruning, Model Compression, Embedded AI.

2.1 Introduction and Background

Recent advances in Vision Transformers (ViTs) have reshaped the landscape of computer vision by introducing attention-based architectures capable of

capturing global context. In semantic segmentation, this capacity to model long-range dependencies has translated into substantial accuracy gains on complex datasets. However, this efficiency comes at a high computational and memory costs, which grows dramatically for high-resolution inputs. Such constraints hinder the deployment of ViTs in real-time or embedded settings, where latency, memory footprint, and energy efficiency are critical. To mitigate these limitations, several approaches have been explored, including model quantization, distillation, or architectural redesign. In parallel, a family of techniques referred to as pruning has been gaining popularity. These methods primarily target Transformer blocks, attention heads, linear layers, or tokens. While pruning blocks or heads modifies the model's architecture, token pruning distinguishes itself by shortening the sequence length during forward pass. The underlying premise is intuitive: by identifying and removing redundant or non-informative tokens at various stages of the network, the computational workload can be substantially reduced, thereby decreasing inference latency. A growing body of literature has proposed various pruning criteria applied either before the encoder or within its shallow layers. Their effectiveness is typically reported through reductions in Floating Point Operations (FLOPs), along with the corresponding impact on accuracy, for example mean Intersection over Union (mIoU) in segmentation. While many pruning approaches succeed in reducing theoretical FLOPs, they often overlook key aspects of practical deployment.

This paper presents a practical study of token pruning for semantic segmentation, emphasizing the gap between theoretical efficiency and real-world deployable performance. We first categorize existing methods and then empirically evaluate selected approaches that prune tokens at the patch level and within shallow encoder layers. Our central contribution is a comprehensive assessment of their end-to-end deployability, which encompasses not only accuracy but also TensorRT exportability and inference latency on an NVIDIA Jetson Orin platform. Our results demonstrate that the theoretical reductions in FLOPs from token pruning do not always translate linearly into latency gains on embedded GPUs, as actual runtime performance is influenced by factors such as memory access patterns, parallelization efficiency, and framework or kernel-level overheads. This highlights the practical trade-offs between pruning ratio, accuracy, and real-world performance that must be considered for efficient model design at the edge. The real technical bottleneck is exportability. We observe, that pruned models are often not compatible with standardized deployment formats such as ONNX, preventing their efficient execution on hardware-optimized runtimes like TensorRT.

2.2 Related Work

The success of Vision Transformers (ViTs) in image classification [1, 2] has quickly extended to dense prediction tasks such as semantic segmentation. However, adapting the Transformer paradigm to dense prediction required the design of novel architectures capable of mapping sequences to pixel-level representations. Pioneering works like SETR [3, 4] demonstrated this feasibility by employing a pure Transformer encoder followed by a CNN-style decoder. The focus then shifted towards improving efficiency and accuracy. SegFormer [5] introduced a hierarchical Transformer encoder that eliminates the need for positional encoding and was combined with a simple MLP decoder. The Segmenter [6] model adopted a more symmetrical encoder-decoder Transformer architecture, using mask tokens to directly predict semantic classes. More recently, SegViT [7] and SegViT2 [8] argued for a powerful decoder, introducing a lightweight ViT decoder with an attention-to-mask mechanism to enhance fine-grained detail.

Despite their high accuracy, the computational cost of these models, driven by the self-attention mechanism on a large number of patches, remains a significant bottleneck for real-world deployment. To mitigate this cost, token pruning reduces computational load by shortening the input sequence through the removal or merging of redundant or less informative tokens. The design of these approaches involves critical choices regarding both the scoring metrics (e.g., based on attention, similarity, or confidence) used to rank tokens, and the stage of application. Some techniques apply early pruning at the patch embedding level to maximize workload reduction, while others employ progressive pruning within the encoder layers to base decisions on more refined features (Figure 2.1). Pruning methods diverge also significantly in their adaptability to input content. While static pruning applies a uniform compression rate across all images, dynamic pruning tailors the number of preserved tokens to the specific complexity of each input.

Early approaches focused on attention-based scoring, exploiting the observation that tokens receiving little attention from the *[CLS]* token contribute marginally to the final representation and can therefore be safely discarded. EViT [9] and Evo-ViT [10] use the attentiveness value as a criterion to identify the top-k relevant tokens for classification and fuse the inattentive ones. EViT aggregates these scores across heads within each layer, while Evo-ViT accumulates class attention across layers to capture global token contributions. This concept was later refined by methods introducing more sophisticated attention mechanisms. For instance, SPViT [11] incorporates a saliency prediction module to guide token pruning, aiming to

preserve semantically important regions by leveraging attention distributions. ATS [12], in contrast, introduces a parameter-free sampling module that adaptively retains tokens based on *[CLS]*-attention scores through soft sampling, so less informative tokens are down-weighted. AS-ViT [13] employs head-weighted attention scores with learnable thresholds for instance-wise pruning to permanently discard non-informative tokens.

A recent breakthrough in ViT optimization lies in token merging techniques, which depart from traditional pruning by offering a softer alternative. Instead of permanently discarding tokens deemed redundant or non-informative, these methods progressively merge them, preserving information. The foundational work on ToMe [14] demonstrated that applying similarity-based fusion between consecutive Transformer layers provides an effective way to reduce token redundancy while maintaining model performance. Specifically, ToMe leverages the attention keys to compute token similarity and employs a fast bipartite matching algorithm to identify pairs of redundant tokens. These tokens are merged through weighted averaging, while proportional attention further ensures that the merging process reflects each token's relative contribution. This idea was subsequently adopted in other approaches [15–20], or [31].

Another line of work exploits confidence-based criteria to guide token pruning. The underlying intuition is that tokens with high predictive certainty can be safely removed from further computation. DToP [21] implements this idea through a multi-stage early-exit mechanism, where the maximum softmax probability from intermediate segmentation heads is used as a confidence score to halt tokens deemed sufficiently processed. STEP [22] extends this idea by systematically analysing the placement of intermediate pruning heads and demonstrates that their positioning plays a crucial role in overall effectiveness. DoViT [23] adopts a similar principle but adds a reconstruction module to preserve spatial consistency, which requires dedicated training and makes it less plug-and-play. PAUMER [24] further refines this direction by leveraging entropy over class probabilities as a confidence score, halting tokens with the highest certainty and thereby reducing redundant computation.

Although token pruning has demonstrated strong results in image classification, it does not always generalize to dense prediction tasks. In classification, preserving the *[CLS]* token is often sufficient for accurate global predictions, whereas segmentation requires maintaining fine-grained spatial information across all tokens, making token removal inherently problematic. Consequently, merging tokens too aggressively or permanently discarding

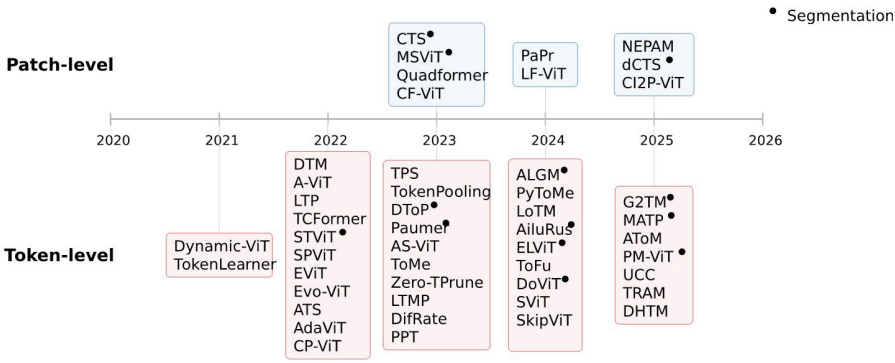

Figure 2.1 Overview of pruning techniques in ViT. The diagram distinguishes between patch-level and token-level approaches.

them leads to the loss of critical details required for accurate segmentation. To date, only a limited number of works have directly addressed token pruning in ViT-based segmentation models [16, 19, 21–29, 31].

Some of these methods employ patch-level pruning that adapts the tokenization process to the image content before the Transformer encoder. This results in a non-uniform token grids from the very beginning. Two complementary directions can be distinguished. First, top-down adaptive tokenization, where a mixed-scale patch grid allocates fine resolution to salient regions while coarsening homogeneous areas [25]. Second, bottom-up merging, where adjacent patches are grouped into larger semantic units after the initial tokenization like in CTS [26] or dCTS [22]. For instance, CTS merges tokens within fixed 2×2 patch groups, guided by a lightweight CNN-based policy network that decides whether the patches should share a token. dCTS extends this approach by allowing multi-scale merging across variable window sizes.

STViT [27] introduces a super-token mechanism to reduce redundancy in early layers of ViTs. Local tokens are first aggregated into super-tokens through sparse attention within limited neighbourhoods, then a compact self-attention is applied among super-tokens, before redistributing the information back to the original tokens. In contrast, AiluRus [28] applies an adaptive resolution strategy. Instead of processing all tokens at a uniform resolution, the method employs a spatial-aware density-based clustering algorithm at intermediate layers to identify representative tokens. Tokens in less informative regions (e.g., object interiors or homogeneous backgrounds) are merged into lower-resolution clusters, while tokens near boundaries or salient regions

are preserved at high resolution. ELViT [29] inserts a token clustering layer that aggregates locally redundant tokens into compact super-tokens, followed by a token reconstruction layer that restores high-resolution representations before the prediction head. More recently, building on ToMe's global bipartite matching, ALGM [16], introduces a hierarchical strategy. Instead of relying solely on progressive global merging, it first performs local merging within small windows to eliminate early redundancy, and then applies global merging once token representations become more discriminative. G2TM [31] proposes a more aggressive one-shot merging strategy that can be applied as early as the first encoder layer. In this approach, tokens are grouped into large graph-based super-tokens using a breadth-first search over the similarity graph.

2.3 Benchmark Setup

We benchmark two baseline architectures, SegViT and Segmenter, together with several state-of-the-art token merging mechanisms that reduce the token count at complementary levels. Specifically, we evaluate two patch-level merging methods: CTS, which performs a fixed 30% token merging, and dCTS, a dynamic variant that applies size-dependent thresholds (0.6, 0.8, 0.9, and 0.9) for 2×2, 4×4, 8×8, and 16×16 patch groups, respectively. We further consider DToP, an early-stopping module inserted after the 8th layer with a confidence threshold of 0.9, as well as STEP, a hybrid method that combines patch merging via dCTS with DToP, using the same configurations as their respective base methods. We also evaluate two recent token-level methods: ALGM, which uses a two-stage dynamic merging strategy (local at the first layer, global at the fifth) with a 0.95 threshold, and G2TM, which applies graph-based token merging at the second layer with the same threshold. For all these methods, we adopt the parameter configurations proposed in the original papers to ensure a fair and reproducible comparison.

Backbone. All methods are evaluated with a ViT-Base backbone. This model has 12 transformer layers, 12 attention heads, and a hidden embedding dimension of 768, resulting in about 86M parameters. This choice enables fair comparison across methods while keeping a suitable scale for embedded deployment and a balanced trade-off between accuracy and efficiency.

Dataset. We use the Cityscapes dataset [30], a standard benchmark for evaluating urban scene segmentation methods. To analyse how the models scale with input resolution, we report results at both 768×768 and 1024×1024.

Metrics. We assess the models in terms of both segmentation quality and computational efficiency. For accuracy, we report the mIoU, which is the standard metric in semantic segmentation. To capture runtime performance under real-time constraints, we measure the throughput in frames per second (FPS) with a batch size of one. Finally, we estimate the intrinsic computational complexity of each model in GFLOPs, computed using the fvcore library.

Implementation. All models are implemented in PyTorch. As none of the methods could be exported to ONNX, inference was conducted directly in PyTorch. Section 1.5 discusses the likely causes of these export failures.

Hardware. Experiments are conducted on the NVIDIA Jetson AGX Orin with 64 GB LPDDR5 and an Ampere GPU (2,048 CUDA cores, 64 Tensor Cores), operated in its maximum power mode (MaxN, 60 W).

2.4 Results

To provide an initial qualitative assessment, we illustrate the token partitioning obtained with different merging-based pruning strategies (Figure 2.2). At the patch level, we compare CTS and its dynamic variant dCTS, while at the token level, we show ALGM and G2TM. These visualizations highlight how merging operates at different granularities and serve as a reference point for contrasting patch-based and token-based pruning approaches. In general, all methods reduce the number of active tokens by approximately 40% on average, except for CTS, which removes a fixed 30% of tokens.

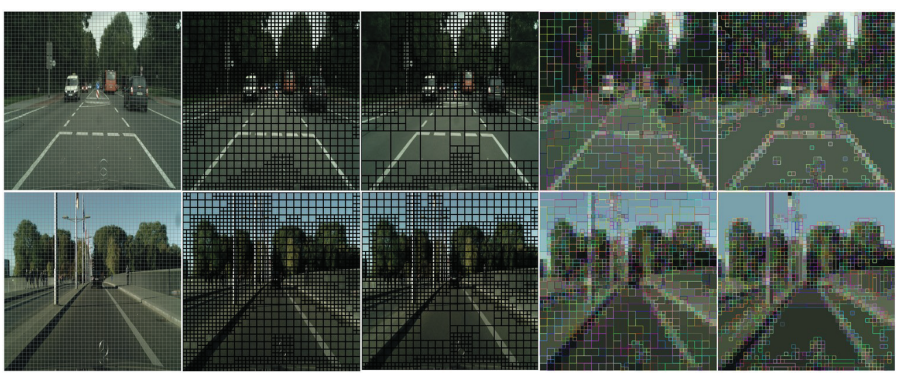

Figure 2.2 Visual examples of pruning with selected SOTA methods (from left to right): regular grid partitioning, CTS, dCTS, ALGM, and G2TM.

In terms of segmentation accuracy, Segmenter consistently outperforms SegViT, achieving higher mIoU across both resolutions. However, as shown in Figure 2.3, SegViT tends to better capture small or thin structures, whereas Segmenter provides more accurate and spatially consistent segmentation overall. SegViT is also notably more efficient, requiring fewer FLOPs and achieving higher FPS. While all evaluated pruning approaches demonstrate clear theoretical efficiency gains, their practical acceleration is mainly determined by the pruning method's design and its implementation efficiency (Table 2.1 and Table 2.2). For SegViT, when assessed at a resolution of 768×768, pruning achieves up to a 50% reduction in computational cost while preserving segmentation accuracy with only about a one-point

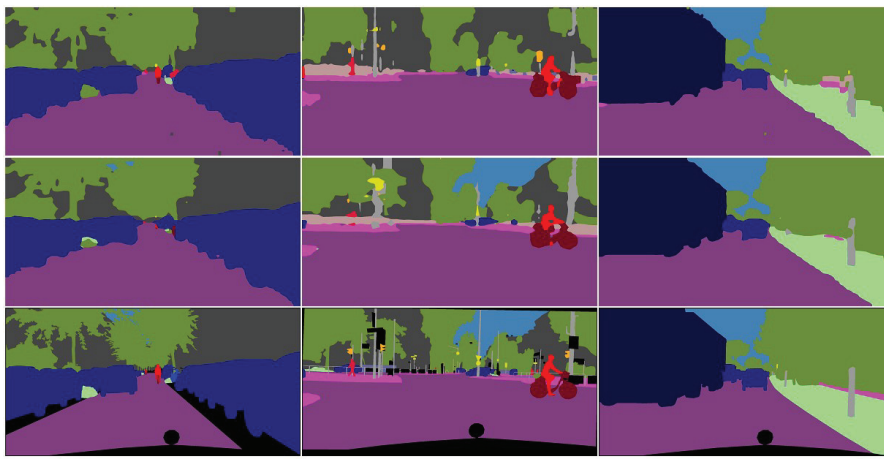

Figure 2.3 Segmentation results (top to bottom): SegViT with STEP, Segmenter with G2TM, and ground truth.

Table 2.1 Performance evaluation on Cityscapes (768×768) of state-of-the-art token reduction methods integrated into a ViT-Base backbone with different decoders

Method	**mIoU [%]**	**GFLOPs**	**FPS**
SegViT	73.7	301	8.0
+CTS	72.9	190	15.2
+DToP	73.5	233	6.6
+dCTS	72.8	182	15.0
+STEP	72.7	149	7.3
Segmenter	77.6	348	2.9
+CTS	77.6	172	5.3
+ALGM	76.4	199	4.7
+G2TM	76.0	230	4.0

decrease. At 1024×1024, the reduction in GFLOPs is even more pronounced, reaching about 70%, though this comes with a larger accuracy drop of up to three mIoU points. CTS and dCTS provide the best trade-off. In contrast, STEP achieves the largest reduction in computational cost across resolutions but shows limited runtime gains. Figure 2.4 illustrates the per-layer behavior of the model, revealing that throughput does not scale linearly with FLOP savings. Although fewer tokens are processed in deeper layers, each layer takes more time to execute, revealing the overhead introduced by the dynamic masking and control operations of the early-exit mechanism (as in DToP), which create bottlenecks due to irregular control flow and memory access patterns.

Table 2.2 Performance evaluation on Cityscapes (1024×1024) of state-of-the-art token reduction methods integrated into a ViT-Base backbone with different decoders

Method	mIoU [%]	GFLOPs	FPS
SegViT	75.2	670	3.1
+CTS	74.9	403	5.5
+DToP	74.6	500	3.4
+dCTS	72.7	247	9.8
+STEP	72.0	200	6.8
Segmenter	77.1	776	1.2
+CTS	72.2	448	1.9
+ALGM	76.6	381	2.2
+G2TM	75.1	424	2.0

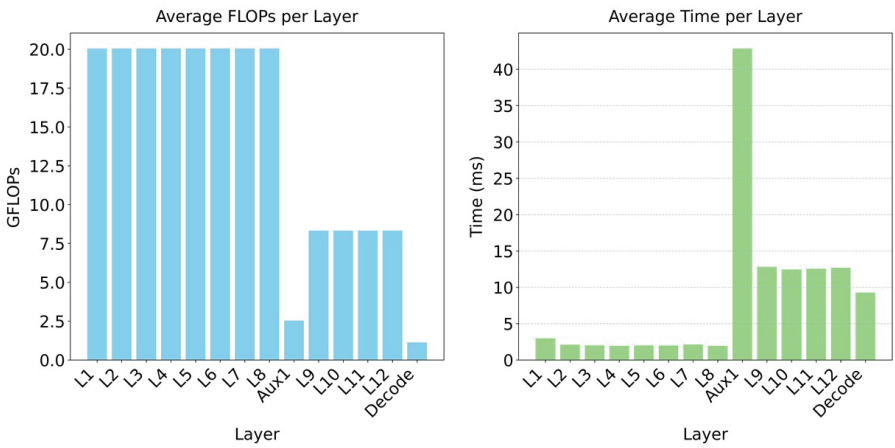

Figure 2.4 Per-layer complexity (GFLOPs) and throughput (FPS) at 1024×1024 for SegViT+STEP, comparing encoder and auxiliary pruning heads.

With the Segmenter, similar trends are observed: although pruning reduces the theoretical cost by nearly half, the real-time performance remains low, particularly at higher resolutions. At 768×768, CTS achieves the best trade-off, delivering both the largest FLOP reduction and the highest throughput. At 1024×1024, it remains efficient but achieves smaller FLOP reductions than G2TM and ALGM, yielding similar FPS.

2.5 Deployment Challenges in Practice

Deploying optimized TensorRT-based ViT models on the NVIDIA Jetson Orin platform presents several practical challenges, primarily related to model conversion into the ONNX format. While ONNX provides a standardized representation of computational graphs across different frameworks and hardware platforms, TensorRT uses ONNX as an intermediate format to enable GPU-accelerated inference through NVIDIA's optimized libraries. In practice, the export process often exposes inconsistencies when applied to complex model architectures. Empirical validation indicates that exporting all SegViT-based models (the baseline and pruned variants, including CTS, DToP, dCTS, and STEP) is technically feasible. However, the ONNX runtime outputs fail to numerically match those of the original PyTorch model, indicating a fundamental corruption of the computation graph. This issue already originates from the SegViT architecture itself, even before applying any pruning. In fact, exporting SegViT to ONNX is challenging due to its additional dependencies on MMSegmentation [32] and the MMCV operator set. The main issues come from operator incompatibilities and dynamic tensor shapes coded in the library itself. Some PyTorch and MMCV layers in the attention and decoding stages use reshaping operations that are hard to represent in the static ONNX graph. Consequently, successful export requires manual graph modifications.

In contrast to SegViT, Segmenter is implemented in pure PyTorch, which simplifies ONNX export process. Its clean and static design defines a single deterministic computation graph without conditional branching or external metadata dependencies. As a result, the exported ONNX model remains stable and reproduces the PyTorch outputs. However, when extending Segmenter with pruning blocks, none of the selected methods allow a successful ONNX export. Although CTS and G2TM can be exported, the generated ONNX graphs are found to be incorrect or incomplete. In contrast, ALGM systematically fails during export. This issue is probably due to non-static control flow, unsupported custom operations, variable tensor shapes, and

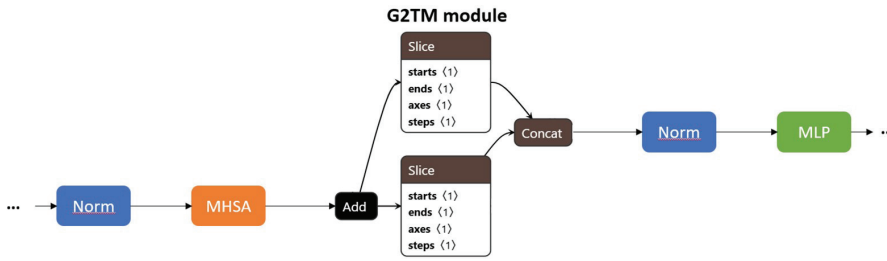

Figure 2.5 Wrong ONNX export of the G2TM module. G2TM is reduced to two brown Slice blocks, which separate the *[CLS]* token from the rest of the tokens, and a Concat operation. The internal computations responsible for the token merging behavior of G2TM are not translated at all and therefore do not appear in the ONNX graph.

in-place operations (e.g., *scatter_reduce*) used to save memory, which disrupted ONNX graph generation. For instance, the G2TM module is exported using only three ONNX blocks (Figure 2.5), corresponding to a simple slicing operation on the token sequence to separate the *[CLS]* token from the remaining tokens. It represents the first step of G2TM, which prevents this special token from being merged with others. The two groups are then directly concatenated with no token reduction applied. In practice, the G2TM module becomes transparent in the ONNX computation graph. This behavior indeed arises because G2TM relies on control-flow constructs such as *if–else* statements conditioned on runtime values or tensor shapes, as well as *for* loops whose iteration ranges are determined by computed values. Additionally, it also relies on third-party libraries (e.g., NetworkX) and performs operations that convert tensors into Python objects such as lists. These conversions cause data to leave the tensor computation graph, thereby preventing ONNX from tracing and representing the corresponding operations.

Overall, the ONNX exporter attempts to freeze the model into a single tensor-based computation graph. However, dynamic control flow, data structures such as lists or dictionaries, and unsupported operators prevent a full conversion, often leading to incomplete graphs or numerical discrepancies compared to PyTorch outputs.

2.6 Conclusions

Numerous pruning strategies based on token merging or discarding have been explored in the literature, but only a handful have been adapted or validated for semantic segmentation. While all selected pruning methods

achieve substantial theoretical gains with up to a twofold reduction in FLOPs, they do not always exhibit corresponding improvements in real throughput. Real-time performance often shows limited improvement because compute units are underutilized and batching is inefficient. Dynamic operations such as masking, token selection, and re-indexing are rarely optimized for specific hardware, adding extra runtime overhead. This issue is further amplified in dynamic pruning methods, where the number of active tokens varies per sample. Since modern accelerators are optimized for fixed-size and dense tensor operations, such variability leads to irregular memory access patterns and thread divergence. More fundamentally, these inefficiencies stem from a lack of algorithm–hardware co-design, as most token pruning strategies are developed under idealized computational assumptions, such as FLOPs reduction, without considering the behavior and limitations of actual deployment hardware.

A major challenge lies in deploying pruned models for optimized inference. In particular, exporting ViT architectures to TensorRT via ONNX often proves difficult due to compatibility and graph conversion issues. For many existing pruning methods, dynamic control flow remains one of the most significant obstacles to seamless ONNX export. Operations involving conditional branching (e.g., pruning rules that apply different thresholds depending on activation magnitude or token importance) and iterative loops (e.g., progressive token merging or pruning functions that stop once a confidence target is reached) introduce runtime variability that conflicts with ONNX's static computational graph representation. In fact, ONNX needs all execution paths and tensor shapes to be fixed when the graph is created. As a result, any operation that depends on dynamic values like *if–else* conditions based on tensor data or loops whose range is decided at runtime cannot be properly exported. Consequently, achieving a successful conversion requires either removing these dynamic behaviours altogether or introducing workarounds such as conditional masking, tensor-based indexing, or custom operators that preserve the model's functionality while adhering to ONNX's static graph constraints.

Acknowledgements

This work was conducted within the NEUROKIT2E project, funded by the EU Horizon Europe programme (Grant Agreement No. 101112268).

References

[1] A. Vaswani, *et al.,* "Attention is All you Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, vol. 30, pp. 6000–6010, 2017.

[2] A. Dosovitskiy, *et al.,* "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[3] S. Zheng, *et al.,* "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] L. Zhang, *et al.,* "Vision transformers: From semantic segmentation to dense prediction," in *Proceedings of the International Journal of Computer Vision*, pp. 1–21, 2024.

[5] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, no. 924, pp. 12077–1209, 2021.

[6] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for Semantic Segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7242–7252, 2021.

[7] B. Zhang, *et al.,* "SegViT: Semantic Segmentation with Plain Vision Transformers," in *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[8] B. Zhang, L. Liu, M. H. Phan, et al., "SegViT v2: Exploring Efficient and Continual Semantic Segmentation with Plain Vision Transformers," *International Journal of Computer Vision*, vol. 132, pp. 1126–1147, 2024.

[9] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, "Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations," in *International Conference on Learning Representations (ICLR)*, 2022.

[10] Y. Xu, *et al.*, "Evo-ViT: Slow-fast token evolution for dynamic vision transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 2964–2972, 2022.

[11] Z. Kong, *et al.*, "SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning," in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, Proceedings, Part XI*, pp. 620–640, 2022.

[12] M. Fayyaz, *et al.,* "Adaptive Token Sampling for Efficient Vision Transformers," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, pp. 396–414, 2022.

[13] X. Liu, T. Wu, and G. Guo, "Adaptive Sparse ViT: Towards Learnable Adaptive Token Pruning by Fully Exploiting Self-Attention," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)*, pp. 1222–1230, 2023.

[14] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token Merging: Your ViT but Faster," in *International Conference on Learning Representations (ICLR)*, 2023.

[15] M. Bonnaerens and J. Dambre, "Learned Thresholds Token Merging and Pruning for Vision Transformers," *Transactions on Machine Learning Research*, 2023.

[16] N. Norouzi, S. Orlova, D. De Geus, and G. Dubbelman, "ALGM: Adaptive Local-then-Global Token Merging for Efficient Semantic Segmentation with Plain Vision Transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15773–15782, 2024.

[17] M. Kim, S. Gao, Y.-C. Hsu, Y. Shen, and H. Jin, "Token fusion: Bridging the gap between token pruning and token merging," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1383–1392, 2024.

[18] D. H. Lee and S. Hong, "Learning to Merge Tokens via Decoupled Embedding for Efficient Vision Transformers," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[19] J. Mao, Y. Shen, J. Guo, Y. Yao, X. Hua, and H. Shen, "Prune and Merge: Efficient Token Compression for Vision Transformer with Spatial Information Preserved," *IEEE Transactions on Multimedia*, pp. 1–14, 2025.

[20] Y. Yang, Y. Zhou, X. Hu, and S. Duan, "KFF: K-feature fusion token merging for vision transformer," *Expert Systems with Applications*, vol. 288, Art. no. 128206, 2025.

[21] Q. Tang, B. Zhang, J. Liu, F. Liu, and Y. Liu, "Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation,"

in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 777–786, 2023.

[22] M. Proust, M. Poreba, M. Szczepanski, and K. Haroun, "STEP: SuperToken and Early-Pruning for efficient semantic segmentation," in *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, vol. 3, Porto, Portugal, pp. 56–61, 2025.

[23] Y. Liu, *et al.,* "Dynamic Token-Pass Transformers for Semantic Segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1816–1825, 2024.

[24] E. Courdier, P. T. Sivaprasad, and F. Fleuret, "PAUMER: Patch Pausing Transformer for Semantic Segmentation," *arXiv preprint arXiv:2311.00586*, 2023.

[25] J. D. Havtorn, A. Royer, T. Blankevoort, and B. E. Bejnordi, "MSViT: Dynamic Mixed-Scale Tokenization for Vision Transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 838–848, 2023.

[26] C. Lu, D. de Geus, and G. Dubbelman, "Content-aware Token Sharing for Efficient Semantic Segmentation with Vision Transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[27] H. Huang, X. Zhou, J. Cao, R. He, and T. Tan, "Vision Transformer with Super Token Sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22690–22699, 2023.

[28] J. Li, *et al.,* "AiluRus: A Scalable ViT Framework for Dense Prediction," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 30979–30996, 2023.

[29] Y. Yuan, W. Liang, H. Ding, Z. Liang, C. Zhang, and H. Hu, "Expediting Large-Scale Vision Transformer for Dense Prediction Without Fine-Tuning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 250–266, 2024.

[30] M. Cordts, *et al.,* "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] V. Bercy, M. Poreba, M. Szczepanski, and S. Bouchafa, "G2TM: Single-Module Graph-Guided Token Merging for Efficient Semantic Segmentation," in *Proceedings of the 21st International Conference on*

Computer Vision Theory and Applications (VISAPP 2026), Marbella, Spain, pp. 43–54, 2026.

[32] MMSegmentation Contributors, "MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark," *GitHub repository*, 2020.