MATHEMATICAL MODELING AND COMPUTING, Vol. 12, No. 4, pp. 1211–1220 (2025)

Inference-Time Optimization for Fast and Accurate Visual Object Tracking

Borsuk V., Yakovyna V.

Lviv Polytechnic National University,
12 S. Bandera str., 79013, Lviv, Ukraine

(Received 28 August 2025; Revised 28 October 2025; Accepted 20 November 2025)

Visual object tracking has recently benefited from the adoption of transformer architectures, which provide strong modeling capacity but incur high computational and memory costs, limiting real-time deployment. Existing efficiency-focused trackers primarily address this challenge through architectural redesign, often trading accuracy for speed. In this work, we explore an alternative and complementary direction: inference-time optimization. Using HiT as our baseline, we integrate memory-efficient attention into its hierarchical transformer blocks, reducing high-bandwidth memory accesses during self-attention without altering the model's representational capacity. Our experiments show that the proposed optimization reduces average latency from 7.82 ms to 6.45 ms, increasing throughput from 127 FPS to 155 FPS, while preserving tracking accuracy. These results demonstrate that inference-time optimizations can significantly improve the practicality of transformer-based trackers for real-time applications, opening a new path toward efficient, high-performance tracking beyond architectural modifications.

Keywords: *visual object tracking; transformers; inference-time optimization; real-time tracking; deep learning; efficient computer vision.*

2010 MSC: 68T45, 68U10 **DOI:** 10.23939/mmc2025.04.1211

1. Introduction

Visual object tracking (VOT) is a fundamental problem in computer vision with wide-ranging applications in areas such as autonomous driving, augmented reality, human–computer interaction, and video surveillance. In its standard formulation, a tracker is provided with a target exemplar (typically a bounding box in the first frame) and tasked with localizing that target in subsequent frames despite challenges such as occlusion, background clutter, and appearance variation. Over the past decade, deep learning-based methods have significantly advanced the field, achieving remarkable improvements in both robustness and accuracy across diverse benchmarks.

A major driver of these advances has been the adoption of transformer architectures, which leverage global self-attention to effectively capture long-range dependencies between the template and search regions. Models such as TransT [1], STARK [2], and MixFormer [3] have set new state-of-the-art results, particularly on large-scale tracking datasets like LaSOT [4] and GOT-10k [5]. However, the powerful modeling capacity of transformers comes at a high computational and memory cost, making these trackers difficult to deploy in real-time scenarios or on resource-constrained platforms.

To mitigate this, much of the existing literature has focused on architectural innovations aimed at designing more efficient models. For instance, lightweight trackers such as FEAR [6], HiT [7], and LightTrack [8] introduce compact attention mechanisms or hybrid convolutional-transformer designs that balance speed and accuracy. While these approaches are effective, they often require nontrivial redesign of the model architecture and may trade off performance for efficiency.

In contrast, relatively little attention has been given to inference-time optimization techniques that can accelerate existing transformer-based trackers without altering their high-level design. Recent advances in memory-efficient attention and related inference optimizations have shown great promise in reducing both latency and GPU memory usage in large-scale vision and language models, yet their

© 2025 Lviv Polytechnic National University

potential remains largely unexplored in the context of visual tracking. Such methods can exploit the inherent redundancy in attention computation — e.g., by reordering operations, reducing intermediate memory usage, or using kernel-based approximations — while preserving the representational power of the model.

In this work, we explore the application of inference-time optimization techniques, particularly memory-efficient attention, to transformer-based visual object tracking. Our goal is to demonstrate that significant improvements in runtime efficiency can be achieved without compromising tracking accuracy or requiring architectural redesign. This line of research opens up a complementary perspective to existing efficiency-focused works: instead of making models smaller or simpler, we optimize the way they are executed. We evaluate our approach on standard tracking benchmarks, showing that inference-level optimizations can close much of the efficiency gap between high-performance transformer trackers and lightweight designs, making state-of-the-art models more practical for real-world deployment.

2. Related works

The rapid evolution of visual object tracking has been closely intertwined with advances in deep learning, efficient model design, and transformer-based architectures. This section reviews the most relevant developments that form the foundation of our work. We first outline the progression of visual object tracking from classical correlation-filter methods to deep learning-based Siamese trackers. We then discuss the emergence of efficient tracking architectures optimized for real-time performance, followed by an analysis of transformer-based attention mechanisms and their computational challenges. Finally, we highlight recent efforts in inference-time optimization, which motivate our exploration of efficiency improvements without structural model redesign.

2.1. Visual object tracking

Visual object tracking (VOT) has long been a central task in computer vision, with applications ranging from video surveillance and autonomous navigation to augmented reality and human-computer interaction. Early benchmarks such as the Visual Object Tracking (VOT) challenges [9] and the Online Tracking Benchmark (OTB) [10] were dominated by methods based on hand-crafted features and discriminative correlation filters [11–13]. While these classical approaches were computationally lightweight and relatively robust to short-term appearance changes, their limited representational power restricted their ability to handle complex variations such as heavy occlusions, large scale changes, or background clutter.

With the rise of deep learning, convolutional neural networks quickly displaced hand-crafted features, leading to more robust and generalizable trackers. Among them, Siamese-based trackers emerged as a particularly influential family of methods.

2.2. Siamese trackers

Siamese trackers formulate tracking as a similarity learning problem, where a matching function is learned offline between a target template and candidate regions in the search frame. This approach provided a favorable balance between accuracy and speed, making it suitable for real-time applications. Early works such as SiamFC [14] demonstrated that correlation-based Siamese networks could outperform traditional methods while running at real-time speeds.

Subsequent advances extended Siamese designs with anchor-free detection [15], multi-level feature fusion, and transformer-based modules. In particular, STARK [2] introduced a transformer encoder-decoder to capture global relationships between the template and search features, achieving state-of-the-art performance on long-term tracking benchmarks.

With the rapid development of transformer neural network architectures, large transformer-based trackers such as TransT [1], MixFormer [3], and OSTrack [16] have achieved state-of-the-art results on LaSOT [4] and GOT-10k [5], albeit at significant computational cost. Despite their success, most large Siamese trackers emphasize accuracy over efficiency, with less focus on inference-level optimization.

2.3. Efficient visual object trackers

As tracking applications often run on resource-constrained hardware, efficiency has become a central research focus. To this end, most existing works primarily target architectural design improvements. Lightweight trackers such as LightTrack [8] leverage neural architecture search to optimize FLOPs and parameter count for mobile deployment. FEAR [6] combines efficient hybrid attention mechanisms with convolutional backbones to achieve real-time tracking without significant loss of accuracy. Similarly, HiT [7] and HCAT [17] integrate hierarchical attention and feature fusion strategies to deliver strong performance under strict latency constraints.

While these methods demonstrate impressive results, they highlight a common trend: efficiency is typically sought through architectural modifications, often requiring careful model redesign or lightweight feature engineering. Complementary lines of work — such as pruning, quantization, distillation, and NAS — likewise operate at the model-design level and may trade accuracy for speed or demand nontrivial re-training. In contrast, relatively little work has explored inference-time optimizations that accelerate existing trackers without changing their high-level structure. This leaves a gap between lightweight architectures designed from scratch and the potential to optimize the execution of state-of-the-art transformer-based trackers.

Our work targets precisely this gap. Instead of altering the network topology, we optimize how the most expensive operator — self-attention — is executed at run time. Concretely, we integrate memory-efficient attention [18] into a strong hierarchical transformer baseline (HiT), reducing peak memory traffic and latency by better exploiting on-chip SRAM and minimizing redundant HBM accesses. Because the operator is drop-in and mathematically exact, the resulting tracker preserves accuracy while delivering tangible speedups and memory savings. This execution-level approach is orthogonal to prior architectural efforts and can, in principle, be applied to a broad family of transformer-based trackers (e.g., TransT, MixFormer, OSTrack) with minimal code changes. In doing so, it turns efficiency into a deployment property — achieved via kernel scheduling and fusion — rather than a model property that must be baked into the design.

2.4. Attention mechanism in transformers

Transformers have become central to modern visual trackers due to their ability to model long-range dependencies between the template and search regions. At the core of this capability is the self-attention mechanism. Given input queries Q, keys K, and values V, attention computes a similarity matrix $A = \mathrm{softmax}(QK^\top/\sqrt{d})$ that captures pairwise relationships between tokens. The attended features are then obtained as AV. While highly expressive, this operation has a quadratic complexity in both time and memory with respect to the sequence length, which quickly becomes a bottleneck when applied to high-resolution visual features.

In practice, the main computational and memory overhead arises from explicitly storing the attention matrix A, which requires $O(n^2)$ memory for n tokens. For visual tracking, where large search regions are processed at every frame, this cost severely limits real-time deployment of transformer-based trackers on resource-constrained devices.

2.5. Inference-time optimization for transformers

To address the quadratic overhead of self-attention, the broader deep learning community has proposed inference-time optimization techniques such as memory-efficient attention [18] and kernelized approximations. Unlike architectural redesign, these methods retain the same model structure but optimize the execution of attention operations. Memory-efficient attention, for instance, avoids explicitly materializing the attention matrix by computing the softmax normalization in a streaming fashion, thereby reducing peak memory usage and improving runtime efficiency.

These techniques have already shown substantial speedups and memory savings in large-scale language and vision models, yet their adoption in visual object tracking remains limited. Importantly, inference-time optimizations are complementary to architectural approaches: they can be integrated into existing transformer-based trackers with minimal changes, narrowing the efficiency gap between

high-accuracy models and lightweight designs. Exploring this under-investigated direction is the central motivation of our work.

2.6. Quantization and pruning for efficient inference

Beyond kernel-level optimizations, other inference acceleration techniques such as quantization and pruning have proven effective in reducing computational and memory demands of deep networks. Quantization compresses model parameters and activations to lower-precision formats (e.g., FP16, INT8, or mixed precision), substantially reducing memory footprint and improving throughput with minimal accuracy loss [19]. Modern GPUs and edge accelerators natively support quantized inference, making it a practical approach for real-time tracking on embedded devices.

Pruning, in contrast, removes redundant parameters or entire channels from the model to reduce FLOPs and latency. Structured pruning methods [20] eliminate filters or attention heads based on their importance scores, while unstructured pruning produces highly sparse networks that benefit from hardware-level sparse computation. Both approaches have been successfully applied to CNN-based trackers, but applying them to transformers remains an active research challenge due to their dense attention patterns.

While quantization and pruning modify the representation of weights and activations, memory-efficient attention operates at the execution level, optimizing how computations are performed rather than what is computed. These strategies are therefore complementary: quantization and pruning reduce the model's arithmetic cost, while memory-efficient attention minimizes runtime memory overhead. Combining such orthogonal optimizations represents a promising avenue toward real-time, high-accuracy transformer-based trackers deployable on low-power hardware.

2.7. Hardware performance considerations

The efficiency of modern visual object trackers depends not only on algorithmic design but also on hardware characteristics, particularly the GPU memory hierarchy. Contemporary GPUs, such as NVIDIA's A100, feature 40–80 GB of high-bandwidth memory (HBM) with up to 2 TB/s bandwidth, alongside limited on-chip SRAM (around 192 KB per streaming multiprocessor) offering an order of magnitude higher bandwidth [18, 21]. While HBM provides capacity, SRAM delivers speed — making efficient SRAM utilization crucial for real-time performance.

GPU kernels operate by loading data from HBM to on-chip memory, performing computations, and writing results back. Operations are often either compute-bound (e.g., matrix multiplications) or memory-bound (e.g., activations, normalizations), depending on their arithmetic intensity. As computation speed has outpaced memory bandwidth, many transformer-based tracking operations have become memory-bound [18].

Kernel fusion is a common strategy to reduce redundant memory transfers by combining multiple operations into a single kernel. However, during training, intermediate results must still be stored in HBM for backpropagation, limiting the benefits of naive fusion. Efficiently leveraging SRAM — such as through memory-efficient attention techniques like FlashAttention — can therefore substantially reduce latency and improve throughput, enabling real-time deployment of transformer-based trackers on modern GPUs.

3. Methods

To study the effect of inference-time optimizations in visual object tracking, we adopt HiT [7] as our baseline model. HiT is a hierarchical vision transformer that achieves a strong trade-off between accuracy and efficiency by progressively reducing spatial resolution while enriching feature representations. While the proposed memory-efficient attention can be integrated into any transformer-based tracker, we focus our analysis on HiT, as it is among the most efficient and well-balanced tracking architectures available. In our approach, we replace the standard self-attention layers within HiT's hierarchical transformer blocks with a memory-efficient attention implementation and evaluate its impact on both runtime efficiency and tracking accuracy.

3.1. Memory-efficient attention

Standard self-attention is defined as:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N\times N}, \quad \mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N}, \quad \mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N\times d}, \tag{1}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V} \in \mathbb{R}^{N\times d}$ are the query, key, and value matrices, N is the sequence length, and d is the feature dimension. This formulation requires explicitly constructing and storing the $N \times N$ attention matrix. The quadratic memory cost $O(N^2)$ arises not only from storing $\mathbf{S}$ and $\mathbf{P}$ but also from repeated high-bandwidth memory (HBM) accesses when multiplying by $\mathbf{V}$. For visual object tracking, where N scales with the resolution of the search region, this memory pressure becomes a critical bottleneck on GPU hardware.

Blockwise decomposition. Memory-efficient attention [18] reorders the computation so that intermediate $N \times N$ matrices are never fully materialized. Instead, the input matrices are partitioned into tiles that fit into fast on-chip SRAM. For a block of queries $\mathbf{Q}_i \in \mathbb{R}^{B\times d}$, we stream in tiles of keys and values $(\mathbf{K}_j, \mathbf{V}_j)$ and compute their contribution to the partial attention output $\mathbf{O}_i$. The output is accumulated incrementally across blocks, avoiding global memory writes for the entire attention matrix.

Softmax across blocks. A central difficulty in blockwise attention is the softmax normalization, which inherently couples all columns of $\mathbf{K}$ and thus cannot be computed independently per block. To enable tile-based computation, we rely on a numerically stable, decomposed formulation of softmax. For a vector $x \in \mathbb{R}^B$, the softmax can be expressed as

$$\mathrm{softmax}(x) = \frac{f(x)}{\ell(x)}, \quad f(x)_i = \exp(x_i - m(x)), \quad \ell(x) = \sum_i f(x)_i, \tag{2}$$

where $m(x) = \max_i x_i$ ensures numerical stability.

When processing multiple tiles $x^{(1)}, \ldots, x^{(K)}$, corresponding to consecutive blocks of attention scores, the global normalization must be reconstructed from local statistics. Suppose each block $x^{(k)}$ is normalized using its own local maximum $m(x^{(k)})$ and sum $\ell(x^{(k)})$. The correct normalization over the concatenated vector $x = [x^{(1)}, \ldots, x^{(K)}]$ can be obtained by maintaining running global statistics:

$$m^* = \max_k m(x^{(k)}), \quad \ell^* = \sum_{k=1}^{K} \exp\big(m(x^{(k)}) - m^*\big)\ell(x^{(k)}). \tag{3}$$

Thus, to merge partial results, we rescale each local sum $\ell(x^{(k)})$ by a factor depending on the difference between its local maximum $m(x^{(k)})$ and the global maximum m^*. Similarly, the locally normalized exponentials $f(x^{(k)})$ can be rescaled to be consistent with the global denominator ℓ^*.

This procedure ensures that softmax over the concatenated vector can be computed exactly from blockwise results while never constructing the full attention matrix. In practice, (m^*, ℓ^*) are maintained incrementally across tiles, so that each new block updates the running normalization statistics. This decomposition is the core enabler of memory-efficient attention: it permits computing attention in blocks (by splitting Q, K, and V into tiles), while guaranteeing numerical stability and correctness of the global softmax.

The final softmax value for an entry $x_i^{(k)}$ is therefore obtained by rescaling its local contribution to the global normalization:

$$\mathrm{softmax}\big(x_i^{(k)}\big) = \frac{\exp\big(x_i^{(k)} - m(x^{(k)})\big)}{\ell^*} \cdot \exp\big(m(x^{(k)}) - m^*\big). \tag{4}$$

This expression makes explicit how local computations (using $m(x^{(k)}), \ell(x^{(k)})$) align with global statistics (m^*, ℓ^*), ensuring that the result is mathematically identical to computing softmax over the full concatenated vector.

Complexity and efficiency. In standard self-attention, both computation and memory scale quadratically with the sequence length N. Forming the similarity matrix $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$ and computing $\mathbf{P}\mathbf{V}$ require $O(N^2 d)$ operations and $O(N^2)$ memory. This quadratic scaling becomes a bottleneck as N grows with the spatial resolution of the search region.

Memory-efficient attention avoids constructing $\mathbf{S}$ and $\mathbf{P}$ explicitly by computing attention in tiles that fit into fast on-chip SRAM. Each tile of $(\mathbf{K}, \mathbf{V})$ is streamed once from HBM, processed locally with $\mathbf{Q}$, and immediately accumulated into the output. This reduces high-bandwidth memory accesses from $O(N^2)$ to $O(Nd)$ and the memory footprint to $O(N)$.

Although the total FLOPs remain $O(N^2 d)$, practical runtime improves due to reduced data movement and better SRAM utilization. Empirically, this yields up to $2-3\times$ lower memory usage and $1.5-2\times$ faster inference without altering the exact attention outputs.

3.2. Integration into HiT

Memory-efficient attention is a general optimization strategy applicable to a wide range of transformer-based visual trackers. It can be incorporated into any architecture that employs self-attention without modifying the model's overall structure or training objectives. In this study, however, we focus on HiT [7] as a representative case due to its strong balance between accuracy and efficiency, making it an ideal testbed for evaluating inference-time optimizations.

To enhance the inference efficiency of HiT, we integrate memory-efficient attention into its hierarchical transformer blocks. Specifically, all self-attention layers in the lightweight hierarchical vision transformer (including both the Multi-Head Attention and Shrink Attention modules) are replaced with their memory-optimized counterparts. The convolutional patch embedding layers, dual-image position encoding, and the subsequent Bridge Module and head network remain unchanged, ensuring architectural consistency with the original HiT design.

The key idea is to reduce high-bandwidth memory usage during the attention computation without altering the representational capacity of the model. In practice, this is achieved by recomputing intermediate activations during the backward pass and adopting kernel-level optimizations for the attention mechanism. Such modifications preserve the attention outputs while lowering the memory footprint and improving runtime efficiency.

By applying the optimized attention across all hierarchical stages, we maintain the original multi-resolution feature extraction process and the joint encoding of template and search region information, but with significantly reduced computational overhead. Importantly, since the feature aggregation in the Bridge Module and the global-context reweighting in the prediction head remain intact, the tracking accuracy is unaffected.

This integration demonstrates a complementary direction to architectural redesign: instead of trading representational power for efficiency, we accelerate inference by optimizing the execution of existing components. In Section 4, we empirically evaluate the resulting tracker on standard benchmarks, reporting improvements in latency, throughput, and memory usage while maintaining the same accuracy.

3.3. Implementation details

We build upon the official HiT [7] codebase and retain its original training and inference configurations to ensure a fair comparison. The integration of memory-efficient attention is implemented as a direct substitution for the standard attention modules within the hierarchical transformer blocks. Specifically, all instances of Multi-Head Attention (MHA) and Shrink Attention (SA) are replaced with memory-efficient attention layers.

During this replacement, we explicitly extract the query, key, and value projections ($\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$) from the original MHA implementation and feed them into the optimized attention operator. This ensures that the parameterization, scaling, and normalization behavior remain identical to the baseline. The modification is therefore confined to the attention computation kernel, leaving the linear projections, feed-forward layers, and normalization modules unchanged.

All other architectural components of HiT, including the convolutional patch embedding, dual-image positional encoding, Bridge Module, and prediction head, are preserved as in the original design. Inference and evaluation follow the same experimental protocols and hyperparameter settings as the HiT-Base model. Our implementation leverages PyTorch's fused CUDA kernel support for memory-efficient attention, enabling kernel-level streaming of query–key–value tiles and minimizing high-bandwidth memory traffic.

4. Results

We evaluate the impact of our proposed inference-time optimizations on the HiT baseline model. Our primary metrics are overall inference latency and frames per second (FPS), measured under identical hardware and implementation settings. Tracking accuracy is also reported to confirm that performance remains unchanged. All experiments are conducted on an NVIDIA L40S GPU with batch size set to 1, representing the online inference regime commonly used in tracking.

We focus our analysis on GPU hardware, as it is the most common platform for training and evaluating deep visual trackers. Performance characteristics on other accelerators, such as TPUs, follow similar trends [22]. Modern GPUs employ a hierarchical memory system comprising multiple levels of memory with differing capacities and bandwidths. The on-chip SRAM (shared memory) provides extremely high bandwidth but is limited in size, while high-bandwidth memory (HBM) offers large capacity at the cost of higher latency and lower throughput.

4.1. Inference efficiency

Table 1 summarizes the efficiency results. The baseline HiT model achieves an average latency of 7.82 ms per frame, corresponding to 127 FPS. After integrating memory-efficient attention into its hierarchical transformer blocks, the average latency is reduced to 6.45 ms, yielding 155 FPS. This corresponds to a relative speedup of approximately 17.5% in latency and 22.0% in FPS, achieved without any architectural modifications.

Table 1. Comparison of inference efficiency between the baseline HiT model and our optimized version.

Model	Latency (ms)	FPS
HiT (baseline)	7.82	127
HiT + Memory-Efficient Attention	6.45	155

4.2. Memory footprint

In addition to runtime, we measure GPU memory consumption during inference. The optimized HiT reduces peak memory usage by 28.4%, primarily due to recomputation-based attention kernels that avoid storing large intermediate tensors. This improvement directly benefits deployment in memory-constrained environments such as UAV platforms, embedded devices, and mobile GPUs.

4.3. Tracking accuracy

Table 2. Performance comparison across multiple tracking benchmarks. The use of the proposed optimization with memory-efficient attention maintains accuracy on all datasets.

Method	LaSOT			TrackingNet			GOT-10k		
	AUC	P_{Norm}	P	AUC	P_{Norm}	P	AO	$SR_{0.5}$	$SR_{0.75}$
HiT-Base (baseline)	64.6	73.3	68.1	80.0	84.4	77.3	64.0	72.1	58.1
HiT-Base (optimized)	64.6	73.3	68.1	80.0	84.4	77.3	64.0	72.1	58.1

Since our modifications only change how attention computations are scheduled and executed, they do not affect the precision of the operations. Consequently, the optimized model achieves the same accuracy as the baseline across all evaluated benchmarks. Table 2 shows that the accuracy remains the same across several visual object tracking benchmarks. This confirms that inference-time optimizations can yield substantial efficiency gains while fully preserving the representational power of the original architecture.

Because the memory-efficient attention implementation is mathematically equivalent to standard self-attention — differing only in computation order and memory access patterns — the results are deterministic and bitwise identical. Therefore, the corresponding confidence intervals have zero width, as there is no stochastic variation between runs. This further supports that the optimization alters only execution efficiency, not the representational behavior or prediction consistency of the model.

4.4. Discussion

These results highlight the potential of inference-time optimizations as a complementary approach to architectural redesign. Unlike lightweight models that often sacrifice accuracy for speed, our method improves runtime efficiency while maintaining state-of-the-art tracking performance. The reduced latency and memory footprint demonstrate that transformer-based trackers can be made more practical for real-time and resource-constrained deployment.

A notable strength of our approach is its generality: since memory-efficient attention preserves the exact outputs of standard self-attention, it can be integrated into existing transformer-based trackers (e.g., TransT, MixFormer, OSTrack) without retraining or architectural changes. This makes inference-time optimization particularly suitable for efficient model deployment.

Limitations. The main limitation of our method is that it does not reduce the asymptotic computational complexity of attention, which remains $O(N^2d)$. The observed speedups arise from improved memory locality rather than fewer operations, and thus depend on hardware architecture and memory hierarchy. Moreover, the current implementation targets GPU hardware; adapting it to other accelerators (e.g., TPUs or edge NPUs) may require platform-specific kernel redesign.

Future Work. Future work could combine inference-time optimization with model-level techniques such as quantization, pruning, or distillation to achieve further gains. Extending memory-efficient computation to other components (e.g., cross-attention or feed-forward layers) and developing adaptive scheduling strategies for edge devices are also promising directions.

In summary, inference-time optimization provides an architecture-agnostic pathway toward efficient visual tracking, complementing existing model compression efforts by improving how attention is executed rather than how it is designed.

5. Conclusions

In this work, we investigated inference-time optimization as a complementary direction for improving the efficiency of transformer-based visual object trackers. Using HiT as our baseline, we integrated memory-efficient attention into its hierarchical transformer blocks. Our experiments demonstrated that this modification reduces inference latency from 7.82 ms to 6.45 ms, increasing throughput from 127 FPS to 155 FPS, while preserving tracking accuracy across benchmarks.

Unlike most existing efficient trackers that focus on architectural redesign, our approach shows that substantial efficiency gains can be achieved by optimizing the execution of existing models. Memory-efficient attention restructures the computation of self-attention without lowering precision or changing the model's representational capacity, making it highly practical for real-world deployment.

Looking ahead, inference-time optimizations could be extended to other components of visual trackers, such as cross-attention modules, feature fusion layers, or diffusion-based models. We believe that combining architectural innovations with execution-level optimizations will be key to enabling high-performance, real-time tracking on a wide range of platforms, from edge devices to large-scale systems.

[1] Chen X., Yan B., Zhu J., Wang D., Yang X., Lu H. Transformer Tracking. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 8122–8131 (2021).

[2] Yan B., Peng H., Fu J., Wang D., Lu H. Learning Spatio-Temporal Transformer for Visual Tracking. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). 10428–10437 (2021).

[3] Cui Y., Cheng J., Wang L., Wu G. MixFormer: End-to-End Tracking with Iterative Mixed Attention. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 13598–13608 (2022).

[4] Fan H., Bai H., Lin L., Yang F., Chu P., Deng G., Yu S., Harshit, Huang M., Liu J., Xu Y., Liao C., Yuan L., Ling H. LaSOT: A High-quality Large-scale Single Object Tracking Benchmark. **129**, 439–461 (2021).

[5] Huang L., Zhao X., Huang K. GOT-10k: A Large High-Diversity Benchmark for Generic Object Tracking in the Wild. IEEE Transactions on Pattern Analysis and Machine Intelligence. **43** (5), 1562–1577 (2021).

[6] Borsuk V., Vei R., Kupyn O., Martyniuk T., Krashenyi I., Matas J. FEAR: Fast, Efficient, Accurate and Robust Visual Tracker. Computer Vision – ECCV 2022. 644-663 (2022).

[7] Kang B., Chen X., Wang D., Peng H., Lu H. Exploring Lightweight Hierarchical Vision Transformers for Efficient Visual Tracking. 2023 IEEE/CVF International Conference on Computer Vision (ICCV). 9578–9587 (2023).

[8] Yan B., Peng H., Wu K., Wang D., Fu J., Lu H. LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 15175–15184 (2021).

[9] Kristan M., Matas J., Leonardis A., Vojíř T., Pflugfelder R., Fernăndez G., Nebehay G., Porikli F., Čehovin L. A Novel Performance Evaluation Methodology for Single-Target Trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence. **38** (11), 2137–2155 (2016).

[10] Wu Y., Lim J., Yang M.-H. Online Object Tracking: A Benchmark. 2013 IEEE Conference on Computer Vision and Pattern Recognition. 2411–2418 (2013).

[11] Bertinetto L., Valmadre J., Golodetz S., Miksik O., Torr P. H. S. Staple: Complementary Learners for Real-Time Tracking. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1401–1409 (2016).

[12] Henriques J. F., Caseiro R., Martins P., Batista J. High-Speed Tracking with Kernelized Correlation Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence. **37** (3), 583–596 (2015).

[13] Vojir T., Noskova J., Matas J. Robust Scale-Adaptive Mean-Shift for Tracking. Image Analysis. 652–663 (2013).

[14] Bertinetto L., Valmadre J., Henriques J. F., Vedaldi A., Torr P. H. S. Fully-Convolutional Siamese Networks for Object Tracking. Computer Vision – ECCV 2016 Workshops. 850–865 (2016).

[15] Tian Z., Shen C., Chen H., He T. FCOS: Fully Convolutional One-Stage Object Detection. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). 9626–9635 (2019).

[16] Ye B., Chang H., Ma B., Shan S., Chen X. Joint Feature Learning and Relation Modeling for Tracking: A One-Stream Framework. Computer Vision – ECCV 2022. 341–357 (2022).

[17] Chen X., Wang D., Li D., Lu H. Efficient Visual Tracking via Hierarchical Cross-Attention Transformer. Computer Vision – ECCV 2022 Workshops. 461–477 (2022).

[18] Dao T., Fu D. Y., Ermon S., Rudra A., Ré C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. Preprint arXiv:2205.14135 (2022).

[19] Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2704–2713 (2018).

[20] Molchanov P., Tyree S., Karras T., Aila T., Kautz J. Pruning Convolutional Neural Networks for Resource Efficient Inference. Preprint arXiv:1611.06440 (2016).

[21] Huerta R., Shoushtary M. A., Cruz J. L., Gonzalez A. Dissecting and Modeling the Architecture of Modern GPU Cores. MICRO '25: Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture®. 369–384 (2025).

[22] Jouppi N. P., Young C., Patil N., Patterson D., Agrawal G., Bajwa R., Bates S., Bhatia S., Boden N., Borchers A., et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. ISCA '17: Proceedings of the 44th Annual International Symposium on Computer Architectu. 1–12 (2017).

Оптимізація етапу застосування моделі для швидкого й точного візуального відстеження об'єктів

Борсук В. Ю., Яковина В. С.

Національний університет "Львівська політехніка", вул. С. Бандери, 12, 79013, Львів, Україна

Візуальне відстеження об'єктів останнім часом отримало значний поштовх завдяки використанню трансформерних архітектур, які забезпечують високу точність, але водночас потребують великих обчислювальних і пам'ятних ресурсів, що обмежує їх застосування в режимі реального часу. Більшість існуючих трекерів, орієнтованих на ефективність, вирішують цю проблему шляхом архітектурних змін, часто жертвуючи точністю заради швидкості. У цій роботі досліджуємо альтернативний і водночас комплементарний напрям – оптимізацію обчислень. Використовуючи HiT як базову модель, інтегруємо оптимізований механізм самоуваги у її ієрархічні трансформерні блоки, зменшуючи кількість звернень до високошвидкісної пам'яті (HBM) під час обчислення уваги без зміни репрезентативної здатності моделі. Наші експерименти показують, що запропонована оптимізація зменшує середній час обчислень з 7.82 мс до 6.45 мс та збільшує пропускну здатність із 127 FPS до 155 FPS, при цьому зберігаючи точність відстеження. Ці результати демонструють, що оптимізація часу інференсу може суттєво підвищити практичність використання трансформерних трекерів у реальному часі, відкриваючи новий шлях до ефективного високопродуктивного відстеження без потреби у змінах архітектури.

Ключові слова: *візуальне відстеження об'єктів; трансформери; оптимізація часу інференсу; відстеження в реальному часі; глибинне навчання; ефективне комп'ютерне бачення.*