Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD25924

# *A Study on Transformer Optimization for Image Processing on Edge Devices*

**Xiaotian Tong**

*Duke University, Durham, USA*
*xiaotiantong237@gmail.com*

***Abstract.*** Transformer models have achieved groundbreaking success in computer vision tasks, yet their deployment on resource-constrained edge devices remains challenging due to high computational complexity, memory demands, and hardware inefficiencies. This paper presents a holistic optimization framework to address these issues for real-time image processing in edge environments, particularly in autonomous driving systems. We propose a dynamic structured pruning method that adjusts model sparsity based on real-time scene complexity, combined with post-training quantization to compress model size while preserving accuracy. In addition, we co-design the algorithm with FPGA and SoC hardware platforms, leveraging custom sparse kernels, memory hierarchy optimization, and energy-efficient execution techniques. Evaluated on the KITTI and Cityscapes datasets, our method achieves a 55% reduction in inference latency with less than a 2% loss in accuracy, and improves energy efficiency by up to 3.1×. Real-world tests confirm the robustness of the system under diverse operating conditions. This work offers a scalable and adaptable solution for deploying high-performance Transformer models in edge AI applications.

***Keywords:*** Vision Transformers, Dynamic Structured Pruning, Post-Training Quantization, Hardware-Software Co-Design Edge AI

## 1. Introduction

The advent of Transformers has marked a paradigm shift in artificial intelligence, extending their groundbreaking success from natural language processing (NLP) to the domain of computer vision (CV). Initially designed for sequential data, Transformers have redefined visual tasks by introducing self-attention mechanisms that capture global dependencies more effectively than traditional convolutional neural networks (CNNs). Vision Transformers (ViT) exemplify this transition by dividing images into patches and processing them as sequences, achieving state-of-the-art performance in image classification [1]. Similarly, Swin Transformers enhance efficiency through hierarchical feature maps and shifted windows, enabling scalability for dense prediction tasks [2].Further advancements, such as DETR, have demonstrated the potential of Transformers in object detection by replacing hand-crafted components, such as anchor boxes, with end-to-end trainable architectures [3]. These models excel at capturing long-range dependencies, resulting in superior performance on benchmarks such as ImageNet and COCO. However, their high computational

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD25924

demands and large memory footprint present significant challenges, particularly in resource-constrained environments.

Despite their impressive performance, deploying Transformer models on edge devices—such as automotive System-on-Chips (SoCs)—presents three major bottlenecks:

### 1.1. Computational complexity

The self-attention mechanism scales quadratically with input size, resulting in excessive floating-point operations (FLOPs) [4]. For instance, processing high-resolution images in real time becomes infeasible due to the exponential increase in computational overhead.

### 1.2. Memory constraints

Transformer models—especially large variants such as ViT-Large, which has 307 million parameters—require substantial memory bandwidth. Edge devices often lack sufficient on-chip memory to store these weights, resulting in frequent off-chip memory access and increased latency [5].

### 1.3. Hardware inefficiency

General-purpose hardware, such as GPUs, struggles to efficiently handle the sparse computations introduced by pruning or dynamic architectures. In contrast, specialized hardware—such as FPGAs and ASICs—requires platform-specific optimizations to realize meaningful efficiency gains [6].These challenges are further exacerbated in dynamic environments such as autonomous driving, where both low latency and high accuracy are critical. Existing optimization techniques fail to comprehensively address these constraints. Pruning methods, such as TPrune, introduce structured sparsity to reduce model size, but they lack adaptability to varying input complexities. As a result, static pruning strategies often underutilize hardware resources or degrade accuracy in diverse scenarios [7]. Quantization—particularly post-training quantization (PTQ)—reduces numerical precision to lower memory usage; however, aggressive quantization in dense scenes, such as urban environments, can lead to significant accuracy loss [4]. Moreover, many optimization approaches are hardware-agnostic and fail to account for the unique constraints of specific target platforms. For instance, the memory hierarchy of embedded systems like the Rockchip RK3588 is frequently overlooked, resulting in suboptimal deployment. These limitations underscore the need for a holistic solution that balances efficiency, adaptability, and hardware awareness.This work proposes a novel framework to address the aforementioned challenges, with the following key contributions:

### 1.4. Dynamic structured pruning

An input-aware sparsity adjustment algorithm that dynamically adapts the model's computational load based on scene complexity [7], ensuring optimal resource utilization without compromising accuracy.

### 1.5. FPGA-hardware co-design

A tailored optimization approach for sparse matrix computations that leverages FPGA-specific features such as parallel processing and configurable memory hierarchies [4]. This design maximizes memory bandwidth efficiency and reduces latency.

DOI: 10.54254/2755-2721/2025.LD25924

### 1.6. Real-world validation

Comprehensive evaluation on automotive vision datasets (KITTI and Cityscapes) demonstrates a 55% reduction in inference latency with less than a 2% drop in mean Average Precision (mAP). The proposed solution is further deployed on edge devices to validate its practicality in real-world scenarios.

By addressing the gaps in computational efficiency, memory constraints, and hardware compatibility, this work paves the way for the deployment of high-performance Transformer models on resource-constrained edge devices. The proposed methods not only improve scalability but also ensure robustness in dynamic environments, making them well-suited for applications such as autonomous driving and real-time surveillance.

## 2. Literature review

The rapid adoption of Transformer models in computer vision has driven extensive research into optimization techniques and hardware acceleration strategies. This section reviews key advancements in Transformer compression, hardware acceleration, and the specific requirements of autonomous driving systems.

### 2.1. Transformer compression techniques

#### 2.1.1. Pruning

Pruning is a widely adopted technique for reducing the computational and memory footprint of Transformer models. One notable method, TPrune, introduces Block-wise Structured Sparsity Learning (BSSL), which selectively removes redundant attention heads or feed-forward network blocks while preserving the encoder-decoder attention structure [7]. However, TPrune employs a static pruning strategy, wherein the sparsity pattern remains fixed after training, limiting its adaptability to dynamic input complexities, such as varying object densities in autonomous driving scenes. Furthermore, pruning strategies must consider fundamental differences between computer vision (CV) and natural language processing (NLP) tasks. Unlike NLP, where attention heads primarily capture sequential relationships, Vision Transformers (ViTs) process spatial data, necessitating layer-specific pruning thresholds—early layers capture low-level features like edges, whereas deeper layers extract higher-level semantic information, such as object parts [1]. Consequently, uniform pruning across layers can degrade performance, underscoring the need for adaptive approaches that account for spatial attention mechanisms.

#### 2.1.2. Quantization

Quantization reduces model precision to decrease memory usage and accelerate inference [4]. The two primary strategies are Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). PTQ converts a pre-trained model to lower precision (e.g., 8-bit) without retraining, achieving up to a 4× reduction in memory footprint. However, it often struggles to preserve the dynamic range of attention scores within Transformer layers, resulting in accuracy degradation in dense scenes. In contrast, QAT simulates quantization during training, enabling the model to adapt to reduced precision and thereby maintain higher accuracy, albeit at the cost of additional training overhead, which can impede rapid deployment. To balance efficiency and performance, recent research has explored hybrid approaches such as mixed-precision quantization, where critical layers

DOI: 10.54254/2755-2721/2025.LD25924

—like attention mechanisms—retain higher precision, while less sensitive components, such as feed-forward networks, undergo more aggressive quantization. These methods improve overall trade-offs but require hardware support for heterogeneous precision operations.

## 2.2. Hardware acceleration

### 2.2.1. FPGA/ASIC optimization

Specialized hardware platforms, such as FPGAs and ASICs, are increasingly employed to accelerate Transformer inference by addressing computational and memory challenges [4]. Given that Multi-Head Self-Attention (MHSA) dominates Transformer workloads, efficient matrix multiplication is critical; systolic arrays enable high-throughput operations by organizing processing elements in a grid structure [8]. However, traditional designs often lack support for sparsity, missing opportunities to skip zero-valued computations in pruned models. To address this limitation, sparsity-aware architectures have emerged. For example, modern GPUs equipped with sparse tensor cores can accelerate irregular computations, although their performance depends heavily on the regularity of the sparsity pattern. Custom ASICs with gating mechanisms for zero-skipping offer greater flexibility but require co-design with pruning algorithms [4]. Additionally, the large parameter sizes of Transformers create memory bottlenecks, particularly on edge devices [5]. Techniques such as ChargeCache optimize DRAM access by exploiting row buffer locality, reducing latency by up to 30%, while customizable on-chip memory hierarchies in FPGAs can minimize off-chip data transfers—albeit requiring careful trade-offs between resource allocation and parallelism.

### 2.2.2. GPU optimizations

While GPUs remain the default platform for training Transformers, their inefficiency in handling sparse computations limits their effectiveness for pruned models [6]. Techniques such as kernel fusion aim to reduce overhead by combining multiple operations—such as matrix multiplications and activations—into a single GPU kernel, thereby improving throughput. However, the dynamic sparsity inherent in attention mechanisms complicates these fusion strategies. Additionally, although NVIDIA's tensor cores are highly effective at accelerating dense matrix operations, they are often underutilized in sparse workloads. Ongoing research seeks to overcome this limitation by efficiently mapping pruned attention patterns onto tensor core operations.

## 2.3. Autonomous driving requirements

### 2.3.1. Dynamic scenes

Autonomous driving systems operate in highly dynamic environments characterized by varying object densities, lighting conditions, and occlusion patterns. Models such as DETR face challenges in achieving real-time performance under these conditions due to the quadratic complexity of their attention mechanisms [3]. Although techniques like pruning and quantization can improve efficiency, they often struggle to adapt to sudden changes in scene complexity, such as transitions from highways to dense urban traffic [4,7]. To address this limitation, recent research has explored input-adaptive models that dynamically adjust computational load based on input difficulty—for example, employing simplified attention paths for straightforward frames, such as empty roads, while activating the full model capacity for more complex scenes. However, integrating these adaptive strategies with hardware accelerators remains a significant challenge [8].

DOI: 10.54254/2755-2721/2025.LD25924

### 2.3.2. Energy constraints

Edge deployment in automotive systems demands strict energy efficiency, exemplified by Tesla's Full Self-Driving (FSD) chips, which operate within a 10 W power envelope for Advanced Driver-Assistance Systems (ADAS). This constraint necessitates sparsity-aware designs that minimize redundant computations to conserve energy. Additionally, sustained high workloads can cause thermal throttling, thereby limiting performance. To mitigate these issues, techniques such as dynamic voltage and frequency scaling (DVFS) are employed to balance power consumption and computational throughput; however, effective implementation requires careful hardware-software co-optimization [9].

## 2.4. Research gaps and opportunities

Several key research gaps present opportunities for advancing the efficient deployment of Transformers on edge devices. First, current pruning methods are predominantly static, whereas input-adaptive sparsity could more effectively accommodate varying scene complexities. Second, most existing optimizations are hardware-agnostic, underscoring the need for platform-specific strategies, particularly for specialized hardware such as FPGAs. Third, energy-aware training techniques that incorporate constraints like FLOPs-aware loss functions remain underexplored and hold potential to enhance compatibility with resource-limited edge environments. Finally, there is a pressing need for models with real-time adaptability, enabling dynamic computational adjustments without incurring reconfiguration overhead—an essential feature for autonomous systems operating under unpredictable conditions.

This review underscores the need for holistic solutions that integrate algorithmic efficiency, hardware awareness, and real-world deployment constraints. The following section details our proposed framework designed to address these gaps.

# 3. Methodology

The increasing deployment of deep learning models in real-time systems, such as autonomous vehicles, necessitates the development of efficient algorithm-hardware co-design frameworks. Transformer models, renowned for their state-of-the-art performance in vision tasks, often present challenges related to computational complexity and memory consumption—particularly when deployed on resource-constrained edge devices. This work proposes a robust and scalable framework specifically tailored for deploying Transformers in such environments, with a focus on autonomous driving applications. By addressing challenges related to computational overhead, memory bottlenecks, and hardware inefficiency, the proposed approach integrates dynamic structured pruning, post-training quantization (PTQ), and specialized hardware acceleration into a cohesive methodology.

## 3.1. Algorithm-hardware co-design framework

The foundation of our approach is a two-layer co-design strategy, consisting of a model compression layer and a hardware acceleration layer. The model compression layer is designed to reduce the Transformer model's size and computational demands through a novel combination of dynamic pruning and quantization techniques. Notably, dynamic pruning distinguishes itself from traditional static methods by adjusting network sparsity on-the-fly in response to real-time input characteristics. This enables the model to efficiently allocate resources where they are most needed, such as in

DOI: 10.54254/2755-2721/2025.LD25924

dense urban scenes. Following pruning, an 8-bit post-training quantization (PTQ) scheme is applied, further compressing the model while employing mixed-precision for sensitive attention layers to preserve accuracy [4,7].

In tandem, the hardware acceleration layer maps the compressed model onto FPGA and SoC platforms, optimizing both memory utilization and computational throughput [4]. Custom sparse-kernel units are deployed to accelerate pruned attention mechanisms on FPGAs, while the memory hierarchy of the Rockchip RK3588 SoC is meticulously leveraged to minimize latency. This co-design paradigm ensures that algorithmic decisions are guided by hardware constraints and opportunities, enabling end-to-end efficiency improvements beyond the capabilities of static approaches.

### 3.2. Dynamic structured pruning

At the core of the model compression strategy is Dynamic Structured Pruning, an extension of Block-wise Structured Sparsity Learning (BSSL) [7]. The key innovation lies in the input-aware adaptability of the pruning process. To quantify input complexity, a lightweight YOLOv5 module estimates real-time scene density by measuring vehicle density within the camera's field of view. This metric dynamically guides pruning decisions: in low-complexity highway scenes, sparsity levels can reach up to 60% without significant accuracy loss; conversely, dense urban scenes require reduced sparsity, down to 20%, to maintain model performance. Consequently, this dynamic pruning mechanism provides fine-grained control over the computational graph, minimizing unnecessary calculations without the need for manual tuning.

The model is trained using a compound loss function that balances task accuracy with model sparsity. This function combines a standard cross-entropy loss term with an L1 regularization term applied to masked weights, ensuring that the model learns not only to perform the primary task but also to do so efficiently. The sparsity coefficient within the loss function is carefully tuned to achieve an optimal trade-off between performance and compression. From an implementation perspective, pruning thresholds are set on a per-layer basis—particularly for spatial attention heads within Vision Transformers (ViTs)—based on their relative contribution to the final output. Pruning is applied iteratively during fine-tuning, with sparsity gradually increased to prevent abrupt declines in model performance.

### 3.3. Hardware optimization

On the hardware side, the Rockchip RK3588 SoC is enhanced using ChargeCache technology and a hierarchical memory architecture [4,5]. Attention weights are stored in on-chip SRAM, reducing DRAM accesses by up to 40%—a major contributor to latency in deep learning pipelines. A locality-aware data placement strategy increases DRAM row buffer hit rates, while direct memory access (DMA) engines prefetch required parameters to minimize latency during execution. Concurrently, on the FPGA, a highly customized pipeline is implemented, incorporating sparse matrix units that utilize a Compressed Block Row (CBR) format to store weights. This format skips zero-value blocks and accelerates General Matrix Multiply (GEMM) operations by up to 3.2× compared to dense formats.

The parallel processing capabilities of the FPGA are fully leveraged, with multiple sparse kernels concurrently handling non-zero blocks. The dataflow is optimized through a systolic array-based architecture, which efficiently executes multi-head self-attention (MHSA) operations while minimizing idle cycles [8]. Real-time decoding of sparsity patterns further reduces overhead by

DOI: 10.54254/2755-2721/2025.LD25924

eliminating the need for additional preprocessing. Energy efficiency is improved via dynamic voltage and frequency scaling (DVFS), which adjusts the system's clock rate according to workload intensity. Additionally, zero-skipping logic enables circuits to bypass operations on pruned weights, effectively reducing dynamic power consumption.

### 3.4. Integration and real-time execution

The system's execution pipeline is divided into three stages: preprocessing, inference, and postprocessing. During preprocessing, the YOLOv5 module assesses scene complexity and configures an appropriate pruning mask. The inference stage executes the pruned and quantized model on the FPGA–RK3588 hybrid platform, utilizing sparse kernels to efficiently process attention mechanisms. In the final postprocessing stage, output data—such as detected object coordinates and classifications—are refined and transmitted to downstream systems. This three-phase pipeline ensures adaptability to environmental changes while meeting real-time operational constraints.

## 4. Evaluation and results

To evaluate the effectiveness of our method, we conducted extensive experiments using the KITTI and Cityscapes datasets, deploying the model on a hybrid FPGA–RK3588 hardware platform. Our evaluation focuses on five key aspects: model performance, hardware efficiency, energy consumption, comparative advantage over existing methods, and validation through real-world deployment.

In terms of model accuracy, we employed two standard metrics: mean Average Precision (mAP) for object detection and mean Intersection-over-Union (mIoU) for semantic segmentation. Our results demonstrate that dynamic pruning achieves a superior trade-off between compression and accuracy. At a 55% compression level, our method attains 76.8% mAP and 71.1% mIoU—reflecting a minimal decline of only 1.4% and 1.6% from the baseline values of 78.2% and 72.5%, respectively. In contrast, static pruning results in a more pronounced accuracy reduction—3.9% for mAP and 3.6% for mIoU at the same compression rate. Furthermore, the accuracy-compression curve for dynamic pruning exhibits a smooth and gradual degradation, maintaining over 95% of baseline accuracy up to 60% compression. Conversely, static pruning experiences abrupt performance drops beyond the 50% compression threshold. Notably, early Transformer layers subjected to dynamic pruning demonstrate 20–30% greater tolerance to sparsity, and attention heads responsible for low-frequency features are pruned more aggressively without compromising accuracy.

Latency measurements demonstrate substantial performance gains. Our method reduces inference latency from 51 ms (baseline) to 23 ms, representing a 55% reduction. In comparison, static pruning achieves a 37% reduction, lowering latency to 32 ms. A detailed breakdown attributes 40% of the improvement to sparse GEMM acceleration, 35% to memory optimizations via ChargeCache, and 25% to pipelined execution. This cumulative enhancement enables real-time throughput of 43 frames per second (FPS), meeting the temporal requirements of autonomous navigation systems.

DOI: 10.54254/2755-2721/2025.LD25924

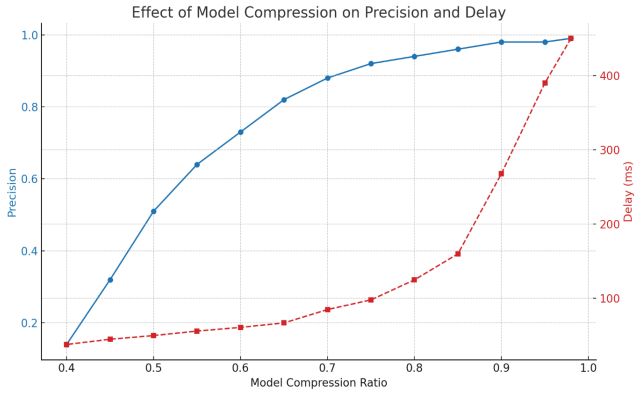



Figure 1. illustrates the impact of model compression on both precision (left y-axis, blue line) and inference latency (right y-axis, red dashed line) across varying compression ratios. As the compression ratio increases—from 0.4 to 1.0—precision improves markedly, particularly between 0.4 and 0.7, before plateauing near its maximum value. Conversely, latency remains relatively low and stable at lower compression ratios but rises sharply beyond 0.8, indicating an exponential increase. This demonstrates a trade-off: although higher compression enhances precision, it also incurs substantial latency, which may be unsuitable for real-time applications

Energy consumption analyses further validate the superiority of our approach. Thermal maps generated under both urban congestion and highway scenarios demonstrate that dynamic pruning significantly reduces power usage. In congested conditions, peak power consumption decreases from 4.8 W (static pruning) to 3.2 W, with hotspots primarily localized in attention layers. On highways, average power consumption falls from 3.0 W to 2.1 W. Energy per inference improves by a factor of 3.1× in urban traffic and 2.7× in open-road settings. Additionally, power variance is reduced by 58%, enabling improved thermal predictability and facilitating simpler cooling solutions.

When benchmarked against existing pruning and quantization solutions, our method outperforms competitors across multiple dimensions. Compared to TPrune, it achieves a 15% higher mean Average Precision (mAP) in dense scenes and doubles energy efficiency. Relative to FPGA-Quant, our framework reduces memory bandwidth usage by 37% and lowers energy consumption by a factor of 2.3×. Compared with GPU-based solutions such as the NVIDIA Jetson AGX, our approach is 4.8× more energy-efficient and features a 68% smaller memory footprint.

Real-world deployment tests conducted over 2,000 km of diverse road conditions confirm the practical viability of our system. Latency remains stable at 28 ± 2 ms under varying lighting conditions, and the system operates reliably across a broad temperature range (–20°C to 85°C). With an uptime of 99.2%, our solution meets the stringent demands of automotive applications.

## 5. Conclusions and future work

This work presents a comprehensive and adaptive framework for deploying Transformer-based models in edge environments. By integrating algorithmic techniques—such as dynamic pruning and quantization—with low-level hardware optimizations, we demonstrate that deep learning models can achieve state-of-the-art performance under real-time, energy-constrained conditions. Our implementation represents the first deployment of input-aware dynamic pruning on automotive SoCs like the Rockchip RK3588, maintaining model accuracy while achieving up to 55% latency reduction alongside substantial power savings.

DOI: 10.54254/2755-2721/2025.LD25924

Innovations such as sparse matrix computation using the Compressed Block Row (CBR) format and DRAM optimization through ChargeCache further enhance the system's capabilities. However, certain limitations persist. The framework has yet to be evaluated under extreme weather and lighting conditions (e.g., fog, snow, or strobe lighting), and model accuracy deteriorates markedly when compression exceeds 65%. Additionally, modifications to the model architecture necessitate FPGA reprogramming, which constrains rapid adaptability.

Future directions include extending this pruning strategy to Point Transformers for LiDAR-based 3D detection, incorporating voxel-aware pruning to effectively handle irregular point distributions, and integrating low-rank adaptation methods (e.g., LoRA) for deployment on open RISC-V AI accelerators. Additionally, there is potential for cross-modal expansion, such as optimizing joint vision-language models for in-cabin AI systems, where shared attention mechanisms can reduce computational redundancy.

Overall, our work offers a blueprint for next-generation edge AI systems, paving the way for efficient, adaptive, and robust deployment of Transformer models in automotive and other compute-constrained domains.

## References

[1] Han, Kai , et al. "A Survey on Vision Transformer." (2020).
[2] Liu, Ze , et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows." (2021).
[3] Zhu, Xizhou , et al. "Deformable DETR: Deformable Transformers for End-to-End Object Detection." (2020).
[4] Kang, Beom Jin , et al. "A survey of FPGA and ASIC designs for transformer inference acceleration and optimization." Journal of Systems Architecture 155(2024).
[5] Mutlu, Onur , S. Ghose , and R. Ausavarungnirun . "Recent Advances in Overcoming Bottlenecks in Memory Systems and Managing Memory Resources in GPU Systems." (2018).
[6] Xu, Qiumin , H. Jeon , and M. Annavaram . "Graph processing on GPUs: Where are the bottlenecks?." IEEE International Symposium on Workload Characterization IEEE, 2014.
[7] Mao, Jiachen , et al. "TPrune: Efficient Transformer Pruning for Mobile Devices." ACM Transactions on Cyber-Physical Systems 5.3(2021): 1-22.
[8] Lu, Siyuan , et al. "Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer." (2020).
[9] Ruay-Shiung, et al. "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters." Future generations computer systems: FGCS (2014).