Advances in Artificial Intelligence and Machine Learning; Review 6 (4) 5995-6013 Received 26-04-2026; Accepted 08-08-2026; Published online 15-08-2026

Model Compression Techniques for Scalable Deployment of Generative AI in Automotive Systems

Arunachalam Thirunavukkarasu arunachalam.thirunavukkarasu@dlr.de
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Germany

Domenik Helms domenik.helms@dlr.de
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)
Germany

Christian Ojeda Bernal christian.ojeda-bernal@valeo.com
Valeo Schalter und Sensoren GmbH
Germany

Sven Mantowsky sven.mantowsky@zf.com
ZF Friedrichshafen AG
Germany

Syed Saqib Bukhari saqib.bukhari@zf.com
ZF Friedrichshafen AG
Germany

Róbert Lajos Bücs robert.buecs@aptiv.com
Aptiv Services Deutschland GmbH
Germany

Corresponding Author: Arunachalam Thirunavukkarasu

Abstract

The deployment of generative AI models within automotive systems is fundamentally constrained by the limited computational resources, memory capacity, and energy budgets of in-vehicle hardware. A companion article examined the scalability challenges, current applications, and evolving E/E architectures that frame this problem. The present paper builds on that foundation by providing an in-depth survey of model compression and optimization techniques that enable the deployment of large neural networks on resource-constrained automotive platforms. We first present an overview of deployment strategies including edge computing, hardware selection, optimized inference frameworks, AI compilers, dedicated accelerators, and knowledge distillation. We then conduct a detailed review of three core compression methodologies: pruning (both with and without retraining), quantization (post-training and quantization-aware approaches for both vision and language models), and low-rank tensor decomposition (including Canonical Polyadic, Tucker, Tensor Train, and Tensor Ring methods, as well as Neural Architecture Search-based compression). The techniques

Citation: Arunachalam Thirunavukkarasu, et al. Model Compression Techniques for Scalable Deployment of Generative AI in Automotive Systems. **Advances in Artificial Intelligence and Machine Learning**. 2026;6(4):332. https://dx.doi.org/10.54364/AAIML.2026.64332

were evaluated by analyzing the underlying principles, reviewing the current available methods, and discussing how the techniques can be used for particular situations in an automotive environment. By using a comparative analysis, trade-offs between compression ratio, speedup for inference, accuracy maintained, and hardware compatibility were all evaluated together so that some conclusions could be drawn. The results of this evaluation provide evidence that multiple compression techniques should be used in combination to achieve the best opportunity for deploying generative AI in real time within an automotive environment. We identify this as an open research direction for the automotive AI community.

Keywords: Automotive AI, Generative AI, Knowledge distillation, Low-Rank approximation, Model compression, Neural network pruning, Quantization

1. INTRODUCTION

The automotive industry is experiencing a major transformation as a result of integrating AI into vehicles. Generative AI like LLMs, Diffusion Models for Image and Video Synthesis, and Vision-Language Models have the potential to revolutionize autonomous driving, cabin interactions, predictive maintenance, and synthetic data generation. Nonetheless, deploying these models on vehicles faces major challenges regarding computational power impact, memory footprint, energy consumption, and real-time latency requirements. Addressing these challenges is especially timely: the automotive sector is moving rapidly toward centralized, high-performance compute platforms and software-defined vehicles, yet the on-board hardware available for AI inference remains orders of magnitude more constrained than the data-center hardware on which generative models are typically trained. Bridging this gap is therefore a prerequisite—rather than an optional enhancement—for bringing generative AI capabilities into production vehicles, which motivates the focused study of model compression presented in this paper.

The previous article [1], addressed how generative AI can be scaled up for automotive applications and what it currently looks like in terms of application use (current generation systems). It also looked at the state of electrical/electronic architectures as they relate to generative AI integration. As a result of our research, we determined that there is a major structural road block to creating AI enabled vehicles; The structural view of new vehicles is being created by means of a combination of centralized computing paradigms and software defined vehicle (SDV) architectures however the models used to create generative AI models will still require significant compression and optimization prior to being able to operate with automotive resources.

This paper attempts to fill the gap between today's best generative models (specifically those deployed to the automotive edge) and the additional constraints involved in those deployments by identifying and discussing model compression and optimization strategies. We will review and describe three main classes of techniques that reduce both the size and inference cost of these models while maintaining acceptable performance on the original tasks – pruning, quantization, and low-rank tensor decomposition. Each of these techniques addresses a different aspect of model complexity; when combined together, they represent a very powerful set of tools/techniques available to automotive AI engineers. We also review complementary deployment strategies—edge computing, hardware selection, optimized inference frameworks, AI compilers, dedicated accelerators, and knowledge distillation—which form the operational context for compression techniques.

The main contributions of this paper are as follows. First, we provide a structured overview of system-level deployment strategies—edge computing, hardware selection, optimized inference frameworks, AI compilers, dedicated accelerators, and knowledge distillation—that establish the operational context for automotive AI. Second, we present a detailed, automotive-oriented survey of three core model compression families: pruning, quantization, and low-rank tensor decomposition, explicitly linking individual methods to in-vehicle use cases. Third, we provide a comparative analysis that distils the trade-offs among compression ratio, inference speedup, accuracy retention, and hardware compatibility into practical guidance for deployment. Finally, we identify open research challenges specific to the automotive domain—including safety validation of compressed models and the absence of a unified benchmarking framework—that we hope will guide future work.

The remainder of this paper is organized as follows. Section 2 reviews deployment strategies for automotive AI. Sections 3, 4, and 5 survey pruning, quantization, and low-rank decomposition respectively. Section 6 provides a comparative analysis, Section 7 discusses future directions, and Section 8 concludes.

2. DEPLOYMENT STRATEGIES FOR AUTOMOTIVE AI

Before applying model-level compression, several system-level strategies improve deployment efficiency. This section provides a brief overview of six complementary strategies that form the operational foundation for automotive AI deployment.

2.1 Edge Deployment

Deploying AI models at the edge—directly within vehicle hardware rather than on cloud infrastructure—minimizes communication latency and enhances real-time performance for safety-relevant applications. Edge deployment also improves data privacy by keeping sensitive sensor data within the vehicle. The trade-off is that edge hardware imposes strict constraints on model size and throughput, making the compression techniques discussed in subsequent sections essential.

2.2 Hardware Selection and Adaptation

Selecting an appropriate hardware platform (e.g., CPU, GPU, NPU, or specialized accelerator) significantly affects inference performance. It is important to perform systematic benchmarking across proposed platforms in order to identify optimal deployment configurations. In addition, adapting model architectures to take advantage of hardware-specific parallelism could result in significant improvements to performance beyond those provided by software-level optimization alone.

2.3 Optimized Inference Frameworks

Deployment efficiency is not optimized in training frameworks like TensorFlow or PyTorch. For example, running a model that has already been trained through a purpose-built inference framework like NVIDIA TensorRT or OpenVINO, TFLite will provide an increase in throughput while reducing latency due to various hardware-specific operations fused together, optimized for different memory layouts, and/or precision calibrated.

2.4 AI Compilers

AI compilers automatically transform trained models into highly-optimized machine code for specific hardware. Compilers create intermediate representations (for example, ONNX) containing optimized versions of the model, including fused-operations, constant-folding, and scheduling of memory accesses. To get the best results from a compiler, one needs to understand two things: 1 - The architecture of the AI model; and 2 - The capabilities of the AI compiler, which may require modifying parts of the model (for example, replacing unsupported activation functions) in order to generate the most efficient results possible.

2.5 Dedicated Accelerators

General-purpose processors cannot compete with a dedicated AI accelerator, like NVIDIA's Jetson, that includes an integrated image processing unit (IPU) or dedicated neural network (NN) inference processor. An AI accelerator can support NN inference rates much higher than general-purpose processors while consuming a fraction of their power.

2.6 Knowledge Distillation

Knowledge distillation [2], trains smaller student models to imitate larger teacher models by utilizing soft probability distributions instead of using hard labels to convey richer amounts of information related to how they relate to one another outside of their class. In addition, the student architecture may be designed to reflect the computational profile of the hardware being used in the vehicle and, therefore, can be very beneficial for automotive deployment. However, careful consideration must be given when choosing an appropriate student architecture, training process, and knowledge transfer method, to achieve a successful outcome.

3. PRUNING TECHNIQUES

Pruning techniques decrease how many parameters exist in a neural network through the elimination of parameters from a model's weight, neuron, or structural elements. The idea is that the vast majority of the parameters of modern models are over-parameterized, and therefore, can be safely removed at all levels without adversely affecting model performance. Pruning methods are

different based on two key characteristics; granularity (structural versus unstructured) and retraining requirements. In an automotive context, pruning is particularly attractive for in-vehicle language assistants and perception models, where structured pruning yields compact networks that run on the general-purpose ECU and domain-controller hardware already present in vehicles, without the specialized sparse-compute units that unstructured pruning would otherwise require.

Structured pruning removes groups of parameters (e.g., filters, channels, attention heads, or transformer blocks) and consequently produces a smaller model that may be executed on a general-purpose machine. With unstructured pruning, individual weights (e.g., each weight is one parameter) are removed, thereby resulting in sparse matrices that require specialized hardware or software libraries to realize practical speedup. Conventional methods are necessary for retraining in order to regain lost accuracy and, thus, necessitate a significant computational expense in the case of large language models (LLMs) or diffusion models. Additionally, some newer methods have shown effective pruning with no required retraining, thereby achieving significantly lower costs.

3.1 Pruning Without Retraining

Retraining-free pruning methods offer cost-efficient alternatives for practitioners with access to consumer or data-center GPUs such as the NVIDIA V100 [3].

3.1.1 WANDA

The authors of [4], use input activation norms and weight magnitudes to calculate the ”importance” of the weights, examining each output for the magnitude per weight of all inputs’ activations. WANDA creates a sparse network without requiring retraining, while providing much better performance than conventional pruning based on weight magnitude. WANDA has better zero-shot performance than other pruning techniques. Finally, the authors showed that larger sparsely connected networks will generally outperform smaller, densely connected networks; therefore, this is a significant finding for deployment in the automotive industry, where the ability to ”scale” across broad capabilities is important.

3.1.2 BONSAI

BONSAI [5], is a structured pruning method that does not use gradients to evaluate the importance of each sub-module, instead using only forward pass to derive the importance with almost no additional memory overhead compared to baseline inference. The authors recommend iteratively pruning models, and fine-tuning the model afterwards using LoRA-based [6], methods to recover any performance lost due to pruning. Similar to WANDA, BONSAI-pruned models also have demonstrated robust zero-shot capabilities.

3.1.3 FLAP

FLAP [7], is a structured pruning method that does not require retraining. The baseline bias compensation mechanism is the key innovation that prevents output feature map damage due to pruning in FLAP. Pruning is possible on multiple layers of the model due to the standardised importance score comparison method used to determine pruning criteria. This allows FLAP to achieve up to a 66% inference speedup at a 20% pruning ratio on the LLaMA-7B model, and a 31% speedup at a 50% pruning ratio. For automotive deployment, the combination of structured output and a retraining-free workflow makes FLAP well suited to latency-sensitive in-vehicle functions, such as conversational assistants, where the resulting dense sub-network maps directly onto existing automotive accelerators.

3.1.4 F3Pruning

F3Pruning [8], targets Text-to-Video (T2V) models. Observational evidence of redundant temporal attention (TA) having TA values equal to or near zero suggests they can be safely removed from TA processing and provide reasonable reallocation of TA. As a result, this method prunes redundant TA weights. The removal of the redundant TA weights achieved a 1.35x speed up and 22% improvement in FVD quality metric at UCF-101 using CogVideo comparing to not pruning.

3.1.5 ZeroTPrune

ZeroTPrune [9], works in two stages. I-Stage prunes irrelevant tokens (e.g., background regions) using a Weight Page Ranking algorithm analogous to Google PageRank applied to the transformer's attention graph. The S-Stage prunes tokens similar to each other (e.g., repetitive structures). Applied to DeiT-S [10], ZeroTPrune increases throughput by 45.3% with only 0.4% accuracy loss. Because automotive driving scenes are typically dominated by large redundant regions such as road surface and sky, token-level pruning of this kind is especially relevant for transformer-based in-vehicle perception, where it can raise throughput with negligible impact on detection accuracy.

3.2 Pruning With Retraining

Retraining-based methods typically achieve higher compression ratios and better accuracy retention at additional computational cost.

3.2.1 Shortened LLaMA

Shortened LLaMA [11], performs depth pruning by removing entire transformer blocks. Block selection is based on importance scores (Taylor expansion, perplexity). A key finding is that depth pruning achieves better speedups than width pruning at small batch sizes typical of real-time applications. For LLaMA-7B, 1.5x speedup is achieved at a 35% pruning ratio. With continued pretraining

(CPT), the pruned Vicuna-7B [12], achieves 1.5x higher average accuracy than WANDA and FLAP at pruning ratios over 50%.

3.2.2 LAPTOP-Diff

LAPTOP-Diff [13], compresses diffusion models via layer pruning with a one-shot criterion based on *additivity*—the property that a network's output distortion from multiple perturbations approximately equals the sum of individual perturbations. Compressing U-Nets of SDXL [14], and SDM-v1.5 [15], yields only a 4.0% PickScore decline at 50% pruning ratio, compared to 8.2% for competing methods.

3.2.3 LoRAPrune

LoRAPrune [16], addresses compatibility issues between LoRA and existing pruning methods. It combines LoRA weights with Taylor expansion to estimate weight importance and quantizes frozen weights to 8-bit for further memory savings. Speedup is proportional to pruning rate (50% faster at 50% pruning), and the method nearly matches LLaMA-7B's full performance with only a 20% pruned model.

3.2.4 LLMPruner

LLMPruner [17], is among the earliest task-agnostic structural pruning methods for LLMs. Its two key elements are a dependency graph for analyzing model structure to identify coherent groups, and grouped importance estimation for those groups. LoRA is used to implement recovery after pruning post training. All experiments were performed using NVIDIA RTX 4090 (24 GB) and a single machine, showing units don't need special hardware to be used. Model performance is still around 95% at 20% Pruning.

3.3 Additional Pruning Approaches

The SIMPLE [18], algorithm implements learnable masks with sparsity-inducing penalties upon some components of a pretrained language model (e.g., attention heads, feedforward networks (FFNs), hidden dimensions). When compressing those components, SIMPLE employs a causal distillation objective to minimize fine-tuning errors resulting from compression. Because it has been demonstrated to be effective in several studies, including those conducted using GPT-2 and BART/mBART, SIMPLE is applicable in other domains such as language modelling, summarisation, and translation. The authors in [19], systematically assess unstructured versus structured approaches to weight pruning of Transformers and systematically explore hyperparameters involved in weight pruning, including pruning thresholds, pruning rates, and the number of epochs for retraining. PuMer [20], is designed specifically for vision-language models and relies on two techniques to achieve this goal: text-informed pruning, specifically retaining image tokens that are relevant to text based upon cross-attention, and Modality-aware merging, where semantically similar tokens

within each modality are combined. Across four vision-language (VL) tasks, PuMer achieves up to 2× speedup and over 50% memory reduction while maintaining less than 1% drop in accuracy. ECoFLaP [21], is a two-stage method that first determines the required sparsity ratio using global importance scores, and second employs local, layer-wise unstructured weight pruning in order to provide high-sparsity gains for large vision-language models. Finally, GOHSP [22], uses a graph-based attention head ranking algorithm to identify and optimize heterogeneous structured sparsity patterns across ViT modules.

4. Quantization Techniques

Quantization of Neural Networks describes processes that transform a model from its original floating-point (FP) model to a more compact fixed-point (FX) model which can potentially compress a model significantly, but at the cost of accuracy in some cases too. There are two main families of quantization: Quantization Aware Training (QAT) simulates the error caused by quantizing during the training phase; the second family, Post-Training Quantization (PTQ), makes the transition to an FX model from an already trained FP model. This paper will primarily focus on reviewing PTQ for practical reasons with respect to automotive deployments (there is no need to have training infrastructure or have the complete training dataset available). Quantization is arguably the most directly deployable compression family for the automotive domain, because the resulting integer (e.g., Int8) arithmetic is natively supported by the AI accelerators and embedded SoCs already used in vehicles, simultaneously reducing memory footprint, inference latency, and energy per inference—all first-order constraints for in-vehicle hardware.

4.1 General Quantization Frameworks

4.1.1 Comprehensive Quantization Overview

The work in [23], provide generic pipelines for both QAT and PTQ. They argue that PTQ suffices for 8-bit quantization with near floating-point accuracy and should be preferred when model performance is acceptable; QAT is required only for sub-8-bit quantization with reasonable accuracy. The authors recommend asymmetric quantization for activations and symmetric for weights, with per-tensor being most common except when weight distributions vary significantly across channels (common in depthwise separable layers), where per-channel is advisable. Models suffering large PTQ degradation may need advanced techniques like Cross-Layer Equalization (CLE) for QAT initialization.

4.1.2 BRECQ

BRECQ [24], quantizes pretrained weights below 8-bit without retraining. By analyzing second-order quantization error using Taylor expansion, the method shows this error can be transformed to layer outputs rather than parameters. Block-wise reconstruction of outputs during PTQ balances cross-layer dependency modeling and overfitting avoidance, outperforming layer-wise and network-wise alternatives. The method uses Fisher Information Matrix calculations and a genetic algorithm

for mixed-precision search. On ImageNet and COCO, BRECQ achieves state-of-the-art 4-bit and 2-bit PTQ, outperforming some QAT methods while being orders of magnitude faster.

4.1.3 LSQ+ for Low-Bit Quantization

Bhalgat et al. (2020) [25], address low-bit quantization (2–4 bits) for networks with negative-valued activations such as Swish. Existing methods like LSQ [26], assume unsigned activations, causing significant accuracy loss. LSQ+ proposes a learnable asymmetric scheme with trainable scale and offset parameters, plus MSE-based initialization for better stability. For EfficientNet-B0 with Swish, LSQ+ improves over LSQ by 1.6–1.8% at 4-bit and up to 5.6% at 2-bit quantization.

4.2 Quantization Methods for Large Language Models

4.2.1 LLM.int8()

LLM.int8() [27], quantizes LLMs from 16/32-bit to Int8 without performance loss via two complementary methods. Vector-wise quantization applies separate normalization constants per row/column, isolating outlier values. For models exceeding 6.7 billion parameters, mixed-precision decomposition additionally isolates outlier parameters in specific hidden dimensions using FP16, while the remaining 99.9% of dimensions use Int8.

4.2.2 SmoothQuant

SmoothQuant [28], is a training-free PTQ algorithm that quantizes both weights and activations of LLMs to Int8, halving memory versus FP16 and accelerating inference via widely supported integer kernels. Its core innovation is a mathematically equivalent per-channel scaling transformation that shifts quantization difficulty from activations to weights, making activation quantization tractable with only a slight increase in weight quantization difficulty.

4.2.3 AQLM

AQLM [29], achieves extreme quantization of LLM weights to 2–3 bits per parameter using multi-codebook quantization. Each vector in key weight matrices is divided into sub-vectors approximated via learned codebook vectors; the original weight is reconstructed by selecting and summing codewords from each codebook. AQLM is the first method that is Pareto-optimal in size-accuracy trade-off at sub-3-bit quantization, though it is computationally more demanding than other PTQ methods, which may present challenges for real-time automotive deployment.

4.2.4 HQQ

According to Half-Quadratic Quantization (HQQ) [30], data-free calibration can be used instead of relying on external calibration datasets which may contain biases. HQQ minimizes quantization errors in weight parameters by applying a sparsity-promotion loss function using Half-Quadratic splitting to optimize that loss function. On the ImageNet dataset, HQQ's 4-bit ViT-B-32 model achieved a +3.1% improvement in top-1 accuracy compared to the BitsAndBytes approach. HQQ's 3-bit ViT-H-14 model significantly outperformed the full-precision (FP32) ViT-L-14 (which has much higher precision) by an impressive +2.4% under extreme quantization conditions, while HQQ's 2-bit variant also outperformed the FP32 ViT-B-32 by an incredible +5.2%. Finally, in comparison with several other algorithms: GPTQ; AWQ; and BNB on WikiText; HQQ produces competitive perplexity scores as well.

4.2.5 AWQ

AWQ [31], is designed for deploying LLMs on edge devices. The fundamental insight is that a small fraction (the top 1%) of weight channels, identified by their activation magnitudes, are salient; protecting these salient channels through per-channel scaling preserves accuracy, while the remaining weights are quantized to a low bit-width, yielding a substantial reduction in model size with minimal accuracy loss. AWQ uses a data-driven, per-channel scaling approach based on an automated search for optimal scaling factors and does not require retraining.

4.3 Quantization for Diffusion Models

4.3.1 ViDiT-Q

ViDiT-Q [32], targets diffusion transformers (DiTs), where the standard U-Net quantization techniques cause significant quality degradation. Its three quantization schemes address four levels of data variance: (i) token-wise quantization handles variance in DiT linear layers that account for 77% of latency and nearly all memory; (ii) dynamic quantization computes parameters online, addressing classifier-free guidance variance and timestep-wise variance across diffusion stages; and (iii) timestep-aware channel balancing applies different masks per timestep to handle channel-wise variance. For sensitivity-prone layers, the mixed-precision variant ViDiT-Q-MP assigns higher bit-widths selectively. Diffusion-transformer quantization of this kind is directly relevant to automotive workflows that rely on generative video and image synthesis—for example, world-model simulation and synthetic scenario generation for training and validating autonomous driving systems—where it enables such models to run within tighter compute budgets.

5. LOW-RANK TENSOR DECOMPOSITION

Low-rank approximation approximates a matrix or tensor with one of lower rank, reducing computational complexity while retaining essential features. Since neural network layers fundamen-

tally operate through matrix and tensor multiplications, low-rank decomposition offers a principled compression approach. A tensor is a multi-dimensional array: vectors are 1st-order, matrices are 2nd-order, and higher-order tensors represent structures such as 4D CNN weight kernels.

The most common tensor decomposition techniques include Canonical Polyadic Decomposition (CPD), Tucker Decomposition, Tensor Train (TT), Tensor Ring (TR), and Block Term Decomposition (BTD). This section surveys state-of-the-art applications of the tensor decomposition techniques used in neural network compressions. Low-rank decomposition is most naturally suited to the convolutional perception networks at the core of automotive sensing pipelines—such as object detection, lane segmentation, and pedestrian recognition—where decomposing the 4D convolutional kernels reduces both parameter count and multiply-accumulate operations, lowering the latency and energy of safety-relevant perception on embedded hardware.

5.1 Matrix and Kernel Decomposition Methods

The authors in [33], express the kernels of a Convolutional Neural Network (CNN) using low rank kernels and linear combinations in two ways: 1) by taking advantage of the spatial dimension using rank-1 filters 2) by taking advantage of the redundancy of input and output channels. Their two methods are able to obtain a speed-up of 2.5 while maintaining accuracy and 4.5 with a loss of only 1% accuracy. In [34], the authors developed an SVD-based decomposition for convolutional kernels, which provides a closed form solution as opposed to iterative methods e.g., CANDECOMP/PARAFAC (CP). The original 4D kernel is reduced to a 2D kernel for use by SVD, after which the two 4D kernels are restored from the SVD, thereby splitting one convolution into two layers that run consecutively. Batch Normalization can solve some of the problems caused by the increased number of layers, such as the exploding and vanishing gradient problem. This allows low-rank NIN models to achieve an accuracy score of 91.31% on the CIFAR-10 dataset as well as provide significant speed gains on AlexNet, VGG, and GoogleNet networks.

The work in [35], improve CNN evaluation by exploiting linear structure: one technique targets the initial color-channel layer, another handles higher-layer redundancy via filter clustering. Using SVD and clustering with reduced Mahalanobis and covariance metrics, the method achieves CPU/GPU speedups with under 1% accuracy loss and 2–3x parameter reduction in convolutional layers (5–10x in fully connected layers). Zhang et al. (2016) [36], extend these ideas to include non-linear units, using the low-rank space of the output response and applying GSVD for asymmetric reconstruction that accounts for errors from previously approximated layers. Rank selection uses the empirical relationship between PCA energy and classification accuracy. On ImageNet, VGG-16 achieves 4x speedup with only a 0.3% increase in top-5 error.

5.2 CP Decomposition-Based Methods

Lebedev et al. (2014) [37], decompose 4D weight kernels into four rank-1 tensors using CP decomposition, with ranks determined via non-linear least squares optimization and subsequent full fine-tuning. The approach achieves 8.5x CPU speedup with only 1% accuracy drop on a 36-class dataset. Minster et al. (2023) [38], enhance CP decomposition stability by incorporating QR decomposition within ALS to compute minimum-norm solutions, particularly helpful when

standard CP-ALS encounters ill-conditioning. Yang and Liu (2024) [39], address rank selection and optimization stability by determining layer sensitivity as both a rank-selection heuristic and ordering criterion for layer-by-layer approximation with iterative fine-tuning. On ImageNet with VGG and ResNet, the method achieves competitive compression and accuracy, though the heuristic sensitivity metric and iterative fine-tuning are computationally expensive.

5.3 Tucker and Higher-Order Decomposition

According to Kim et al. (2016) [40], they compressed both VGGNet and GoogleNet by applying Tucker decomposition to them through the use of Bayesian matrix factoring in order to determine the ranks of these models. Mode-3 and mode-4 matricizations were used in order to form Tucker-2 decompositions of the higher convolution levels, while also performing an one-shot fine-tuning on the models afterwards. Oseledets (2011) [41], first introduced the concept of Tensor Train (TT) decomposition to alleviate the NP-hard canonical rank problem of CP decomposition. TT approximates a tensor through a series of three-dimensional tensors, with the ranks of these tensors being based on the auxiliary matrices from QR and SVD sequences (without recursion), thus providing stable numerical results while greatly reducing the number of parameters used. Zhao et al. (2016) [42], provides an enhanced method over TT decomposition by developing Tensor Ring (TR) decomposition; all cores in TR are treated equally according to their circular multilinear product, unlike the sequential multilinear product used in TT decomposition. The 4D kernel tensor can be modeled as a stack of three 2D layers with regards to SVD and ALS for core optimization, and as the order of the tensor increases, the parameters used in TR scale linearly. TR is therefore also effective for unsupervised feature representation due to its flexibility through its rank structure.

5.4 Specialized Compression Methods

According to Kossaifi et al. (2017) [43], they replaced the flattened output of CNNs and the Fully Connected (FC) layers with a tensor regression layer that maintains all of the various modalities through Tucker decomposition for tensor contraction which enables using low rank weight tensors for tensor regression mapping; this method is especially efficient for multi-dimensional datasets (such as those created by MRI scans). The authors in [44] developed two types of matrices (Compressed Entropy Row (CER), Compressed Shared Elements Row (CSER)) which utilize the properties of low-entropy weight distributions to achieve 42 times size reduction, 5 times speed increase, and 90 times energy savings. Further, the work in [45] extends this line with Huffman coding-based methods: Huffman Address Map (HAM) and sparse HAM (sHAM) replace weight addresses with Huffman codes for compressed representation.

Kim et al. (2019) [46], optimize rank configuration across the entire network rather than layer-by-layer using non-iterative accuracy metrics. For VGG-16, the approach achieves 25% FLOP reduction with 0.7% accuracy improvement in just 3 minutes on a CPU, versus 4 hours on 8 GPUs for prior methods. Idelbayev and Carreira-Perpiñán (2020) [47], propose a mixed discrete-continuous framework simultaneously optimizing ranks and matrix elements; a compressed VGG achieves faster performance than ResNet with nearly identical classification error. Kamalakara et al. (2022) [48], study training low-rank tensors from scratch versus decomposition after training, showing that spectral initialization and L2 regularization with differentiated learning rates significantly improve

from-scratch low-rank training. Cao et al. (2017) [49], compare Tucker, CP, and TT decompositions applied to Tensor Regression Layers (TRL) against Global Average Pooling (GAP): on CIFAR-10/100 with ResNet, GAP shows better compression and accuracy than CP-TRL; overall, TRL performs better in shallow networks while GAP maintains better balance in deep networks.

5.5 Neural Architecture Search for Tensor Compression

In [50], NAS is integrated with Tucker decomposition-based tensor compression to optimize CNNs for embedded systems. The method controls per-layer compression parameters through NAS, achieving up to 85% parameter reduction with only 0.1–1% accuracy loss. Compressed models occupy just 20% of original memory with up to 2.5x inference speedup. There is still a high computation cost for current methodsusing NAS; however, researchers should look to further develop search efficiency using new methods such as Reinforcement Learning and Evolutionary Algorithms.

6. COMPARATIVE ANALYSIS

TABLE 1, provides a comparative overview of representative methods from each compression family, summarizing their characteristics, compression metrics, and automotive applicability.

Several observations emerge. Retraining-free pruning methods like WANDA and FLAP offer the most accessible path for automotive teams with limited compute budgets, being rapidly applicable to pretrained models. Quantization methods targeting Int8 (e.g., SmoothQuant, AWQ) offer perhaps the most direct deployment path by reducing both computation and memory while being supported by widely available hardware. Low-rank decomposition provides the strongest theoretical foundations for CNN-based perception models. Combining techniques—pruning followed by quantization, or low-rank decomposition with fine-tuning—typically yields the best overall results. TABLE 2, summarizes the trade-offs across compression families.

7. FUTURE RESEARCH DIRECTIONS

Several open challenges and research opportunities remain for the automotive AI community. **Combined compression pipelines**: the optimal ordering and configuration of pruning, quantization, and low-rank decomposition for specific architectures and hardware targets remains open, requiring systematic studies of technique interactions and automated pipeline search. **Multimodal generative model compression**: automotive applications increasingly rely on vision-language-sensor models; early work like PuMer [20] and ECoFLaP [21] must be extended across the full spectrum of automotive multimodal models. **Hardware-aware compression**: compression techniques should be co-designed with specialized accelerators to exploit features such as structured sparsity support and low-bit arithmetic units. **Safety-critical validation**: compressed models must be validated not only for average-case but also for worst-case behavior in safety-critical scenarios, developing efficient validation methodologies for compressed models remains an essential but largely unaddressed challenge. This concern is particularly acute in the automotive domain, where deployed models fall under functional safety standards such as ISO 26262 and the safety-of-the-intended-functionality

Table 1: Comparative Overview of Model Compression Techniques

Method	Category	Model Target	Key Metric	Retrain	HW Req.	Auto. Relevance
WANDA [4]	Pruning	LLMs	Competitive perplexity	No	Consumer GPU	Fast, universal applicability
FLAP [7]	Pruning	LLMs	66% speedup @20%	No	Consumer GPU	Real-time edge inference
Shortened LLaMA [11]	Pruning	LLMs	1.5x speedup @35%	Yes	Single GPU	Small-batch real-time use
LLMPruner [17]	Pruning	LLMs	95% perf. @20% prune	Yes	RTX 4090	Accessible, task-agnostic
SmoothQuant [28]	Quant.	LLMs	Int8 W+A, 50% mem.	No	General HW	Broad HW compatibility
AWQ [31]	Quant.	LLMs	Edge-optimized	No	Edge devices	Direct edge deployment
BRECQ [24]	Quant.	CNNs	SOTA 4-bit/2-bit PTQ	No	General HW	Perception model compression
HQQ [30]	Quant.	LLMs/Vision	Data-free, +3.1% acc	No	General HW	No calibration data needed
ViDiT-Q [32]	Quant.	Diff. Trans.	DiT-specific schemes	No	Edge devices	Video/image generation
CP Decomp. [37]	Low-rank	CNNs	8.5x speedup	Yes	CPU/GPU	Sensor processing models
Tucker [40]	Low-rank	CNNs	High compression	Yes	CPU/GPU	Established method
TT Decomp. [41]	Low-rank	DNNs	Stable, linear params	No	CPU/GPU	Numerically stable
NAS+TC [50]	Low-rank +NAS	CNNs	85% param. reduction	Yes	GPU cluster	Embedded vision systems

Table 2: Trade-off Summary Across Compression Technique Families

Criterion	Pruning	Quantization	Low-Rank Decomposition
Compression Ratio	Moderate to High	High (4–16x typical)	Moderate to High
Inference Speedup	Moderate (structured) to Low (unstructured)	High (hardware-supported)	Moderate to High
Accuracy Retention	Good with fine-tuning	Good at 8-bit; degrades at extreme low-bit	Good with fine-tuning
Compute Cost to Apply	Low (no retrain) to High	Low (PTQ) to Moderate (QAT)	Moderate to High
Hardware Requirements	General (structured) or Specialized (unstructured)	Broad support for Int8	General purpose
Combinability	Combines well with quantization	Combines well with pruning	Combines with distillation
Maturity for LLMs	Rapidly advancing	Most mature	Early stage
Maturity for CNNs	Well established	Well established	Most established

standard ISO 21448 (SOTIF). Compression can alter a model's tail behavior in ways that average-case accuracy metrics do not reveal: pruning or aggressive low-bit quantization may disproportionately affect rare but safety-critical inputs, such as vulnerable road users in unusual poses or adverse weather. Consequently, compressed automotive models require validation that explicitly targets worst-case and out-of-distribution behavior, alongside redundancy and fallback strategies for fail-operational designs. The regulatory landscape adds further requirements, as frameworks such as the EU AI Act classify many automotive AI functions as high-risk, implying obligations on traceability, robustness, and documentation that compressed-model pipelines must be able to satisfy.

Toward a unified benchmarking framework: a persistent limitation of the current literature, including this survey, is the absence of a unified benchmarking framework that evaluates compression techniques along automotive-relevant dimensions—latency on representative ECU and SoC hardware, energy per inference, accuracy retention under safety-critical input distributions, and robustness to out-of-distribution inputs. Most reported results rely on general-purpose benchmarks (e.g., ImageNet, COCO, WikiText) that do not reflect automotive deployment realities; developing such a framework, ideally with public reference implementations on representative automotive hardware, would substantially accelerate progress in the field. **Adaptive inference**: dynamic models adjusting complexity based on input difficulty could enable more efficient resource utilization while maintaining safety guarantees. **Sustainable and energy-aware compression**: with the transition to electric vehicles, inference energy directly impacts vehicle range; future compression research should explicitly optimize for energy efficiency alongside accuracy and speed.

8. Conclusion

This paper has presented a comprehensive survey of model compression and optimization methods for utilizing generative AI in resource-limited automotive environments. Following on from our previous work [1], we have identified several technical solutions that help bridge the gap between model capabilities and hardware limitations across the three primary classes of methods discussed in this paper: pruning (e.g., retraining-free methods such as WANDA, BONSAI, and FLAP; retraining-based methods such as Shortened LLaMA, LAPTOP-Diff, and LLMPruner), quantization (e.g., general frameworks such as BRECQ; LLM-centric methods such as SmoothQuant, AWQ, and HQQ; diffusion-oriented approaches such as ViDiT-Q), and low-rank tensor decomposition (e.g., CP, Tucker, Tensor Train, and Tensor Ring-based algorithms; NAS-based methods).

Our comparative analysis reveals that no single technique dominates across all criteria. The best deployment method will incorporate numerous methods depending upon what model architecture an organization uses and the hardware target being deployed on, especially in the automotive industry, which will be able to apply these methods to scale up the usage of generative AI inside of vehicles to provide better performance, safety, and user experience in a resource limited environment. As such, future studies should focus on having integrated compression pipelines, hardware-aware optimization methods, methodologies for verifying that compressed models are safe, and energy-efficient compression to support the transitions to electric vehicles. If the AI research community continues to work collaboratively with automotive engineers, the next generation of vehicles will begin to reach the full potential of generative AI.

9. ACKNOWLEDGEMENT

The research leading to these results is funded by the German Federal Ministry for Economic Affairs and Energy within the project “NXT GEN AI METHODS – Generative Methoden für Perzeption, Prädiktion und Planung”. The authors would like to thank the consortium for the successful cooperation.

References

[1] Thirunavukkarasu A, Helms D, Ojeda Bernal C, Mantowsky S, Bukhari S, et al. Generative AI in Automotive Industry. Adv Artif Intell Mach Learn. 2025;5:260.

[2] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network. 2015. Arxiv preprint: https://arxiv.org/pdf/1503.02531.

[3] https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf.

[4] Sun M, Liu Z, Bair A, Kolter JZ. A Simple and Effective Pruning Approach for Large Language Models. 2024. Arxiv preprint: https://arxiv.org/pdf/2306.11695v3.

[5] Kolawole S, Dery L, Kagy JF, Smith V, Neubig G, et al. Everybody Prune Now: Structured Pruning of Llms With Only Forward Passes. 2024. Arxiv preprint: https://arxiv.org/pdf/2402.05406v1.

[6] Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, et al. Lora: Low-Rank Adaptation of Large Language Models. 2021. Arxiv preprint: https://arxiv.org/pdf/2106.09685.

[7] An Y, Zhao K, Yu T, Tang M, Wang J. Fluctuation-Based Adaptive Structured Pruning for Large Language Models. 2023. Arxiv preprint: https://arxiv.org/pdf/2312.11983.

[8] Su S, Liu J, Gao L, Song J. F3-Pruning: A Training-Free and Generalized Pruning Strategy Towards Faster and Finer Text-To-Video Synthesis. In Proceedings of the AAAI Conference on Artificial Intelligence. 2024;38:4961-4969.

[9] Wang H, Dedhia B, Jha NK. Zero-Tprune: Zero-Shot Token Pruning Through Leveraging of the Attention Graph in Pre-Trained Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024:16070-16079.

[10] Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, et al. Training Dataefficient Image Transformers and Distillation Through Attention. 2020. Arxiv preprint:: https://arxiv.org/pdf/2012.12877.

[11] Kim BK, Kim G, Kim TH, Castells T, Jeong S, et al. Shortened Llama: A Simple Depth Pruning for Large Language Models. 2024. Arxiv preprint: https://arxiv.org/pdf/2402.02834.

[12] https://www.lmsys.org/blog/2023-03-30-vicuna/ .

[13] Zhang D, Li S, Chen C, Xie Q, Lu H. Laptop-Diff: Layer Pruning and Normalized Distillation for Compressing Diffusion Models. 2024. Arxiv preprint: https://arxiv.org/abs/2404.11098.

[14] Podell D, English Z, Lacey K, Blattmann A, Dockhorn T, et al. Sdxl: Improving Latent Diffusion Models for High-Resolution Image Synthesis. 2023. Arxiv preprint: https://arxiv.org/pdf/2307.01952.

[15] Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-Resolution Image Synthesis With Latent Diffusion Models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022:10674-10685.

[16] Zhang M, Chen H, Shen C, Yang Z, Ou L, et al. Loraprune: Pruning Meets Low-Rank Parameter-Efficient Fine-Tuning. 2024. Arxiv preprint: https://arxiv.org/pdf/2305.18403.

[17] Ma X, Fang G. Wang X. Llm-Pruner: On the Structural Pruning of Large Language Models. 2023. Arxiv preprint: https://arxiv.org/pdf/2305.11627.

[18] Tao C, Hou L, Bai H, Wei J, Jiang X, et al. Structured Pruning for Efficient Generative Pre-Trained Language Models. In Findings of the Association for Computational Linguistics: ACL. 2023:10880-10895.

[19] Gholami S. Can Pruning Make Large Language Models More Efficient? Inredefining Security With Cyber AI. IGI Global Scientific Publishing. 2024:1-14.

[20] Cao Q, Paranjape B, Hajishirzi H. Pumer: Pruning and Merging Tokens for Efficient Vision Language Models. 2023. Arxiv preprint: https://arxiv.org/pdf/2305.17530.

[21] Sung YL, Yoon J, Bansal M. Ecoflap: Efficient Coarse-To-Fine Layer-Wise Pruning for Vision-Language Models. 2023. Arxiv preprint: https://arxiv.org/pdf/2310.02998v1.

[22] Yin M, Uzkent B, Shen Y, Jin H, Yuan B. Gohsp: A Unified Framework of Graph and Optimizationbased Heterogeneous Structured Pruning for Vision Transformer. In Proceedings of the AAAI conference on artificial intelligence. 2023;37:10954-10962.

[23] Nagel M, Fournarakis M, Amjad RA, Bondarenko Y, Van Baalen M, et al. A White Paper on Neural Network Quantization. 2021. Arxiv preprint: https://arxiv.org/pdf/2106.08295.

[24] Li Y, Gong R, Tan X, Yang Y, Hu P, et al. Brecq: Pushing the Limit of Post-Training Quantization by Block Reconstruction. 2021. Arxiv preprint: https://arxiv.org/pdf/2102.05426.

[25] Bhalgat Y, Lee J, Nagel M, Blankevoort T, Kwak N. Lsq+: Improving Low-Bit Quantization Through Learnable Offsets and Better Initialization. 2020. Arxiv preprint: https://arxiv.org/pdf/2004.09576.

[26] Esser SK, McKinstry JL, Bablani D, Appuswamy R, Modha DS. Learned Step Size Quantization. 2019. Arxiv preprint: https://arxiv.org/pdf/1902.08153v1.

[27] Dettmers T, Lewis M, Belkada Y, Zettlemoyer L. llm.int8(): 8-Bit Matrix Multiplication for Transformers at Scale. 2022. ArXiv preprint: https://arxiv.org/pdf/2208.07339.

[28] Xiao G, Lin J, Seznec M, Wu H, Demouth J, et al. Smoothquant: Accurate and Efficient Post-Training Quantization for Large Language Models. In International conference on machine learning. PMLR. 2023:38087-38099.

[29] Egiazarian V, Panferov A, Kuznedelev D, Frantar E, Babbenko A, et al. Extreme Compression of Large Language Models via Additive Quantization. 2024. Arxiv preprint: https://arxiv.org/pdf/2401.06118.

[30] https://github.com/dropbox/hqq

[31] Lin J, Tang J, Tang H, Yang S, Chen W, et al. Awq: Activation-Aware Weight Quantization for Llm Compression and Acceleration. Proceedings of machine learning and systems. 2024;6:87-100.

[32] Zhao T, Fang T, Huang H, Liu E, Wan R, et al. Vidit-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation. 2024. Arxiv preprint: https://arxiv.org/pdf/2406.02540v1.

[33] Jaderberg M, Vedaldi A, Zisserman A. Speeding up Convolutional Neural Networks With Low Rank Expansions. 2014. Arxiv preprint: https://arxiv.org/pdf/1405.3866.

[34] Tai C, Xiao T, Zhang Y, Wang X, EW. Convolutional Neural Networks with Low-Rank Regularization. 2016. Arxiv preprint: https://arxiv.org/pdf/1511.06067 .

[35] Denton E, Zaremba W, Bruna J, LeCun Y, Fergus R. Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation. 2014. Arxiv preprint: https://arxiv.org/pdf/1404.0736.

[36] Zhang X, Zou J, He K, Sun J. Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Trans Pattern Anal Mach Intell. 2016;38:1943-1955.

[37] Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V. Speeding-up Convolutional Neural Networks Using Fine-Tuned Cp-Decomposition. 2014. Arxiv preprint: https://arxiv.org/pdf/1412.6553v1.

[38] Minster R, Viviano I, Liu X, Ballard G. Cp Decomposition for Tensors via Alternating Least Squares With Qr Decomposition. Numerical Linear Algebra with Applications. 2023;30:e2511.

[39] Yang C, Liu H. Stable Low-Rank Cp Decomposition for Compression of Convolutional Neural Networks Based on Sensitivity. Appl Sci. 2024;14:1491.

[40] Kim YD, Park E, Yoo S, Choi T, Yang L, et al. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications. 2016. Arxiv preprint: https://arxiv.org/pdf/1511.06530.

[41] Oseledets IV. Tensor-Train Decomposition. SIAM J Sci Comput. 2011;33:2295-2317.

[42] Zhao Q, Zhou G, Xie S, Zhang L, Cichocki A. Tensor Ring Decomposition. 2016. Arxiv preprint: https://arxiv.org/pdf/1606.05535.

[43] Kossaifi J, Lipton ZC, Kolbeinsson A, Khanna A, Furlanello T, et al. Tensor Regression Networks. 2017. Arxiv preprint: https://arxiv.org/pdf/1707.08308v1.

[44] Wiedemann S, Müller KR, Samek W. Compact and Computationally Efficient Representation of Deep Neural Networks. IEEE Trans Neural Netw Learn Syst. 2020;31:772-785.

[45] Mariño GC, Ghidoli G, Frasca M, Malchiodi D. Compression Strategies and Space-Conscious Representations for Deep Neural Networks. In2020 25th International Conference on Pattern Recognition (ICPR). IEEE. 2020:9835-9842.

[46] Kim H, Khan MU, Kyung CM. Efficient Neural Network Compression. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019:12561-12569.

[47] Idelbayev Y, Carreira-Perpiñán MA. Low-Rank Compression of Neural Nets: Learning the Rank of Each Layer. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020:8046-8056.

[48] Kamalakara SR, Locatelli A, Venkitesh B, Ba J, Gal Y, et al. Exploring Low Rank Training of Deep Neural Networks. 2022. Arxiv preprint: https://arxiv.org/pdf/2209.13569.

[49] Cao X, Rabusseau G. Tensor regression networks with various low-rank tensor approximations. 2017. Arxiv preprint: https://arxiv.org/pdf/1712.09520

[50] Thirunavukkarasu A, Helms D. Using Network Architecture Search for Optimizing Tensor Compression. InInternational Embedded Systems Symposium. Cham: Springer Nature Switzerland. 2023:139-150.