university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Exploring Model Tensor Decomposition Techniques on Model Performance after TensorRT Conversion

Graduation Project Proposal

Kyriakos Antoniou (S5715881)

2025-08-29

Internal Supervisor: Dr. M.A. (Matias) Valdenegro Toro (Machine Learning, University of Groningen)
External Supervisors: Dr. rer. nat. Gunnar Schönhoff (Natural Sciences, DFKI) and Dr. Elie Mounzer (DFKI)

Artificial Intelligence /
University of Groningen, The Netherlands

1 Introduction

Deep learning models have achieved outstanding performance across a wide range of tasks, including image classification, natural language processing, and speech recognition, e.g., ResNet on ImageNet, BERT on NLP benchmarks, and DeepSpeech in ASR; (He, Zhang, Ren, & Sun, 2016; Devlin, Chang, Lee, & Toutanova, 2019; Hannun et al., 2014). However, their substantial computational and memory requirements often limit their deployment on resource-constrained devices such as embedded systems, mobile platforms, robotic systems, and IoT devices (Sze, Chen, Yang, & Emer, 2017). Addressing the challenge of reducing model complexity while maintaining accuracy is therefore essential for practical applications, particularly in real-time or energy-constrained environments. This limitation is increasingly pressing as modern hardware no longer benefits from the exponential improvements predicted by Moore's Law, which has plateaued due to physical limits in chip manufacturing.

TensorRT, developed by NVIDIA, is a widely adopted inference optimization framework that converts and accelerates deep learning models for GPU deployment. For instance, TensorRT has been used to significantly speed up inference in autonomous driving pipelines and real-time video analytics (NVIDIA, 2023). While TensorRT improves inference efficiency by performing layer fusion, kernel auto-tuning, and precision calibration, it does not inherently reduce the architectural redundancy or parameter count of the input model. As such, large and overparameterized networks may still be inefficient on resource-constrained embedded platforms such as the NVIDIA Jetson, even after TensorRT optimization.

This project investigates whether *tensor decomposition techniques* CP, Tucker, Tensor Train can serve as effective *preprocessing methods* to compress convolutional neural networks before TensorRT optimization. Some of these methods, particularly Tensor Train, were originally developed within quantum physics for efficiently representing quantum many-body states and are thus considered quantum-inspired. Others, like CP and Tucker, stem from classical multilinear algebra and data analysis. All three approaches have been adapted to machine learning, demonstrating significant potential in compressing and simplifying neural network architectures. These methods have been shown to reduce the number of parameters and computational operations in convolutional and fully connected layers, as demonstrated in previous work such as tensorized versions of VGG and ResNet (Lebedev & Lempitsky, 2016; Novikov, Podoprikhin, Osokin, & Vetrov, 2015). The objective is to determine whether these decompositions can reduce model size and increase inference speed without a significant drop in accuracy, specifically targeting robotic systems where computational resources are limited.

This project evaluates how tensor decomposition affects the performance and efficiency of deep neural networks before and after TensorRT optimization. The study focuses on how the performance gap between non-decomposed and decomposed networks changes once both have been optimized using TensorRT. TensorRT already exploits the highly parallel nature of GPUs to accelerate inference through techniques like layer fusion, kernel tuning, and tensor core utilization. This work investigates whether tensor decomposition can further enhance these benefits by restructuring large tensors into smaller components that better align with GPU execution patterns. Prior research has shown that decomposed models, when fine-tuned, can recover much of their original accuracy (Lebedev & Lempitsky, 2016). The overarching goal is to determine whether combining decomposition with TensorRT yields additional efficiency gains, enabling the deployment of larger models on resource-constrained platforms such as robotic systems.

2 Theoretical Framework

The theoretical foundation of this project lies in the observation that modern deep learning models, particularly convolutional and fully connected neural networks, can be naturally expressed using high-dimensional tensors. Weight parameters in deep networks often exist as multi-way arrays, such as 4D tensors in convolutional layers and 2D matrices in dense layers. As neural networks grow in depth and width to improve expressiveness, these weight tensors grow correspondingly large, leading to redundancy and overparameterization (Sze et al., 2017; Novikov et al., 2015).

Tensor networks were originally developed in quantum physics as computational frameworks for efficiently representing high-dimensional quantum states with limited computational resources. These methods, known as quantum-inspired tensor decomposition techniques, efficiently represent complex tensors through interconnected low-rank tensors. Early work, such as (Novikov et al., 2015), introduced the idea of *tensorizing* fully connected layers using Tensor Train (TT) decomposition, resulting in dramatic reductions in the number of parameters while maintaining predictive performance (Novikov et al., 2015). This line of research demonstrates that weight tensors often lie in low-rank subspaces, making them amenable to approximation via tensor decomposition methods.

The Canonical Polyadic (CP) decomposition approximates a tensor as a sum of rank-one tensors, allowing convolutional layers to replace full-rank filters with sequences of separable filters, significantly lowering computational costs (Lebedev & Lempitsky, 2016). Tucker decomposition, a higher-order generalization of PCA, decomposes a tensor into a core tensor and factor matrices, offering flexibility in rank selection and showing promising results in compressing VGG and ResNet models while preserving accuracy (Kim et al., 2015; Zhang, Zou, He, & Sun, 2015). Tensor Train (TT) decomposition reshapes tensors into a sequence of 3D cores linked in a chain structure, highly effective for compressing fully connected layers with high-dimensional spaces (Novikov et al., 2015). Tensor Ring (TR) decomposition differs from TT by introducing periodic boundary conditions, forming a closed loop of tensor cores. This structure increases expressive power and may offer additional compression benefits, particularly in fully connected layers (Zhao, Zhou, Xie, Zhang, & Cichocki, 2016).. These methods constitute *low-rank tensor approximations*, underpinning neural network compression strategies.

3 Research Question

The main research question of this project is whether applying tensor decomposition techniques, namely Canonical Polyadic (CP), Tucker, Tensor Train (TT), and Tensor Ring (TR) to pre-trained neural networks, followed by TensorRT optimization, leads to greater efficiency improvements compared to using either techniques or TensorRT alone. In particular, the project evaluates how the performance difference between decomposed and non-decomposed models changes after both are converted using TensorRT, to assess whether decomposition enhances the speedup TensorRT achieves on its own. A secondary question is whether fine-tuning the resulting models can restore or improve performance, both before (as shown in previous papers) and after conversion. This research is conducted in collaboration with an external institute that has previously worked extensively with tensor methods.

To investigate this question, the project will begin by selecting a set of pre-trained convolutional neural networks and applying each of the four tensor decomposition techniques CP, Tucker, TT, and TR to appropriate layers, such as convolutional or fully connected layers. The resulting compressed models will then be converted using NVIDIA's TensorRT framework. The performance metrics of model size, inference time, and memory usage will be recorded to evaluate the impact of decomposition prior to optimization.

In the second phase, the decomposed models will undergo fine-tuning using supervised learning on the original training dataset restoring or improving their accuracy. These fine-tuned models will then be converted with TensorRT, and their performance will be compared to that of their non-fine-tuned counterparts. A comparative analysis will be conducted across three groups: the original models, the decomposed-only models, and the decomposed-plus-fine-tuned models, in order to draw conclusions about the effectiveness of the full pipeline.

4 Methods

The primary metrics that will be used to evaluate the effects of decomposition and optimization are: *model size* (in megabytes),*classification accuracy* (the specific accuracy metric will depend on the dataset used and the scale of the model), and *inference latency* (in milliseconds) as a possible additional metric on robotic systems. These metrics are chosen because they directly align with the goals of model compression and efficient deployment. Model size reflects memory efficiency, latency indicates real-time applicability, and accuracy determines whether the compression and optimization pipeline maintains the predictive capability of the original model. These metrics also reflect the constraints of real-world systems, where hardware resources may be limited and deploying large models remains a significant challenge.

Each model will be evaluated in three states: the original pre-trained models, the decomposed models converted with TensorRT, and the decomposed models that have been fine-tuned post-decomposition and then converted. Comparing these configurations will allow us to assess the relative impact of tensor decomposition both before and after TensorRT conversion, as well as the additional performance benefits introduced by fine-tuning.

The experimental setup will cover three convolutional network architectures ResNet, DenseNet, and VGG each explored at three levels of depth: small, medium, and large. Specifically, ResNet18, ResNet50, and ResNet152 will represent the ResNet family; DenseNet121, DenseNet169, and DenseNet201 will be used for DenseNet; and VGG11_bn, VGG16_bn, and VGG19_bn for the VGG family. Each of the nine selected CNN models will be compressed independently using all three decomposition techniques CP, Tucker, and Tensor Train to enable a controlled comparison of their performance across architectures and depths. This uniform application ensures that the evaluation captures both the scalability of each decomposition method and how it interacts with different model structures. In addition to the primary experiments, Tensor Ring decomposition will not be included in the experimental evaluation, as the selected CNN architectures lack the translation-invariant structure that TR is designed to exploit, and preliminary analysis showed no clear advantage over Tensor Train in this context. This method is not part of the core experimental design but may be included if time and resources allow, to assess its potential advantages in compressing symmetric tensor structures.

5 Scientific Relevance for Artificial Intelligence

This graduation project explores the interaction between tensor decomposition techniques and TensorRT optimization to assess whether their combination enables more efficient deployment of deep neural networks. TensorRT accelerates inference by leveraging GPU-specific optimizations such as layer fusion, kernel tuning, and mixed-precision computation. Separately, tensor decomposition methods like CP, Tucker, Tensor Train (TT), and Tensor Ring (TR) reduce model size and computational complexity by restructuring large tensors into smaller, low-rank components. The central research goal is to determine whether decomposition can further enhance the inference speedups achieved by TensorRT, beyond what either approach provides in isolation.

As modern hardware no longer benefits from exponential growth in compute power, optimizing inference pipelines has become a key research priority. While both decomposition and TensorRT have been explored independently, their combined effect particularly in terms of how decomposition reshapes tensor structures to better align with GPU execution patterns remains underexplored.

By benchmarking performance across decomposed and non-decomposed models, both before and after TensorRT conversion, this project contributes a novel analysis of how tensor structure influences optimization outcomes. More broadly, this research contributes to making state-of-the-art deep learning models accessible for real-time and embedded applications by optimizing them not just for accuracy, but also for latency, memory usage, and energy efficiency. It supports the development of scalable, sustainable AI systems that can be deployed in constrained environments without sacrificing performance. The project is conducted in collaboration with the German Research Center for Artificial Intelligence (DFKI), extending their expertise in tensor methods into the domain of real-time model optimization and deployment.

In combining tensor decomposition and TensorRT optimization, this project contributes a deployment-focused perspective to the model compression literature. Rather than evaluating compression in isolation, it examines whether decomposed models maintain or amplify their efficiency gains after being converted for GPU execution using TensorRT. This directly addresses the challenges faced in real-time inference on embedded systems and contributes actionable insights into how structural compression affects inference frameworks. By assessing not just compression rates and accuracy, but also post-optimization behavior, the research informs practical decisions for deploying deep learning models in environments with limited computational resources.

6 Planning

The planning of this graduation project is divided into eight work phases (WPs) and one proposal phase (PP), each representing a major component of the research. The project begins with a literature study and concludes with thesis writing and the final presentation. The work packages and their estimated durations are outlined below.

Additionally, the Summer Holiday Period (HP1) and Winter Holiday Period (HP2) are included for transparency, along with potential delays associated with these periods, indicated as Holiday Delays (HD).

- **WP1: Literature study and theoretical groundwork**: Reviewing existing work on tensor decomposition techniques and TensorRT, and identifying relevant model architectures and tools for the experimental setup.

- **PP (Proposal Phase)**: Period allocated for preparing, writing, and submitting the graduation project proposal.

- **WP2: Experimental pipeline setup**: Setting up the core implementation pipeline using PyTorch, Tensorly, and Torch2TRT. Preparing baseline models and data loaders.

- **WP3: Model decomposition**: Applying CP, Tucker and TT decompositions to selected layers of candidate models

- **WP4: Fine-tuning experiments**: Fine-tuning the decomposed models and evaluating recovery or improvement of accuracy.

- **WP5: TensorRT conversion and benchmarking**: Converting the non-fine-tuned and fine-tuned models with TensorRT, benchmarking them on NVIDIA Jetson Nano hardware to measure inference latency, memory usage, model size, and accuracy in realistic embedded conditions.

- **WP6: Analysis and documentation**: Analyzing the results, performing comparisons, and documenting outcomes.
- **WP7: Thesis writing**: Writing the thesis document, integrating literature, methods, results, and discussion.
- **WP8: Final revision and presentation**: Final proofreading, formatting, and preparation for defense and presentation.
- **D (Delays)**: Delays that are expected to have occurred due to reduced activity during institutional holiday periods (e.g., communication gaps or rescheduled milestones).
- **HP1 (Holiday Period 1)**: Potential summer vacation days that may be taken during July and August.
- **HP2 (Holiday Period 2)**: Potential winter vacation days that may be taken during the December holiday season.

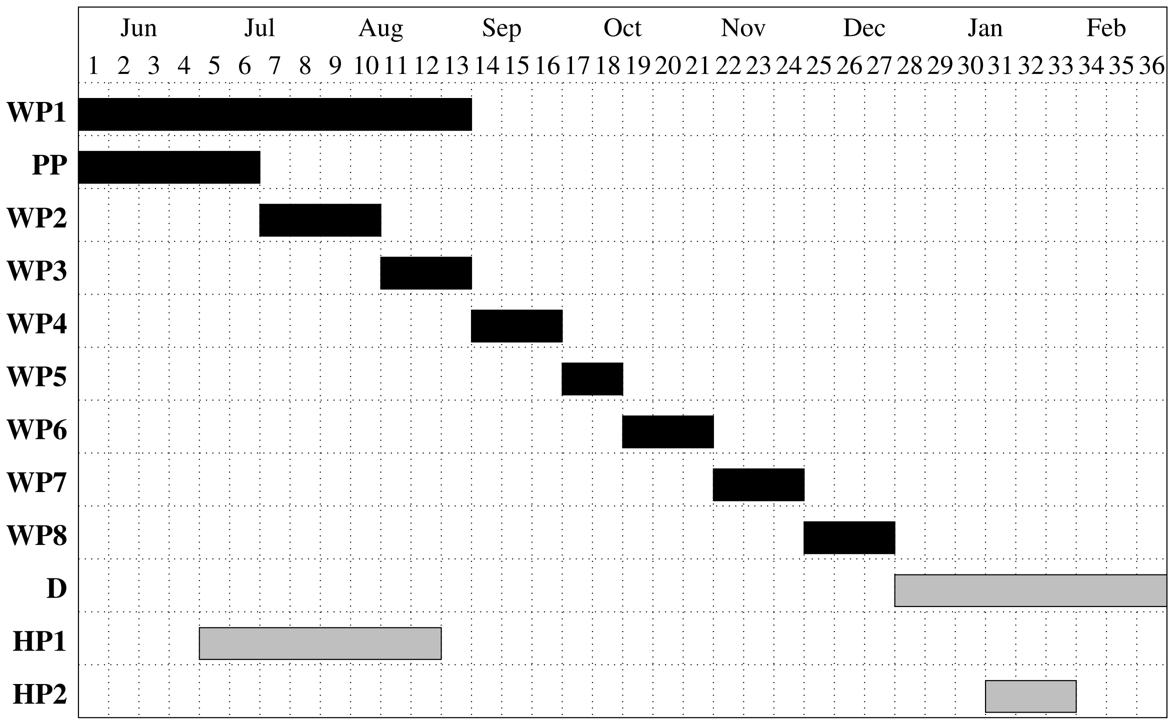

Figure 1: Expected Timeline of Work Packages (Simple Weekly Format)

7 Resources and Support

The project will be conducted using GPU-enabled environments via Google Colab, which provides access to modern NVIDIA GPUs and supports the required machine learning frameworks, including PyTorch and Tensorly. This platform is sufficient for the experiments involving model decomposition, fine-tuning, and performance benchmarking. Where necessary, additional setup will be performed to ensure compatibility with TensorRT workflows. While PyTorch and Tensorly will serve as the primary frameworks, other libraries or toolkits may be explored if they offer improved integration, decomposition capabilities, or conversion support for TensorRT and embedded deployment.

Inference benchmarking, however, will specifically be performed on NVIDIA Jetson hardware available at the Robotics Lab of the University of Groningen to provide realistic performance insights for embedded deployment.

Supervision will include periodic check-ins with the internal university supervisor to track academic progress and provide support. The external supervisor, affiliated with the host institute, will remain in regular contact to offer technical guidance and ensure alignment with the institute's ongoing research on tensor methods.

References

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171–4186).

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., … others (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR*.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., & Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

Lebedev, V., & Lempitsky, V. (2016). Fast convnets using group-wise brain damage. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2554–2564).

Novikov, A., Podoprikhin, D., Osokin, A., & Vetrov, D. P. (2015). Tensorizing neural networks. *Advances in neural information processing systems*, *28*.

NVIDIA. (2023). *TensorRT Developer Guide.* (https://docs.nvidia.com/deeplearning/tensorrt/)

Sze, V., Chen, Y.-H., Yang, T.-J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295–2329.

Zhang, X., Zou, J., He, K., & Sun, J. (2015). Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, *38*(10), 1943–1955.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., & Cichocki, A. (2016). Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*.