Demo: An Experimental Platform for AI Model Partitioning on Resource-constrained Devices

Dimitrios Kafetzis
Athens University of Economics and Business,
Department of Informatics
Athens, Greece

Iordanis Koutsopoulos
Athens University of Economics and Business,
Department of Informatics
Athens, Greece

Abstract

Partitioning Artificial Intelligence (AI) models such as Deep Neural Networks (DNNs) or Transformer-based Architectures is essential for minimizing latency in resource-constrained edge computing environments, which is critical for applications such as real-time video analytics, autonomous vehicles, smart IoT systems, and most importantly, on-device Large Language Models (LLMs) [1]. This paper presents a demonstration of an experimental platform showcased for DNN partitioning for the inference stage, which comprises a WiFi network of Raspberry Pi 4 devices. The platform supports research on optimizing inference in distributed setups, focusing on performance and resource usage and it can support several architectures, ranging from simple and deep NNs, to more advanced transformer-based architectures.

CCS Concepts

• **Computing methodologies** → **Cooperation and coordination**.

Keywords

NN Architecture Partitioning, Inference, Edge Computing, Resource-constrained Devices, Experimental Platform.

ACM Reference Format:
Dimitrios Kafetzis and Iordanis Koutsopoulos. 2024. Demo: An Experimental Platform for AI Model Partitioning on Resource-constrained Devices. In *The Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MOBIHOC '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3641512.3690629

1 Introduction

AI models are integral to many applications like image recognition and natural language processing, but often exceed the computational limits of edge devices like IoT devices and mobile phones. This challenge is more pronounced with transformer-based Generative AI (GenAI) architectures, which are too large for resource-constrained devices. As we anticipate a surge in inference loads from LLMs on edge devices, efficient scheduling and partitioning of these workloads become crucial challenges. This demo offers a proof of concept for partitioning DNNs—linear architectures whose simpler partitioning strategies pave the way for more complex policies in transformer-based architectures.

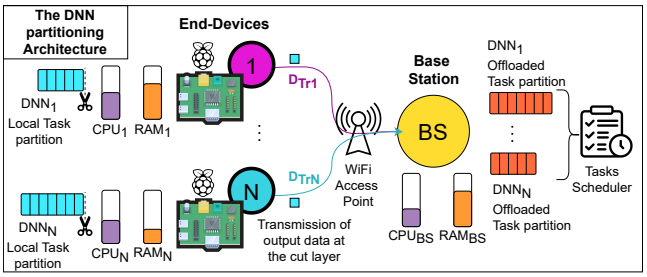

Figure 1: The architecture of the DNN partitioning procedure on the demonstrated platform.

In our previous work [2], we build a custom simulator and we addressed the problem of DNN partitioning at the inference stage and offloading inference tasks to a Base Station (BS) node. We demonstrated significant reductions in total execution delay and balanced task execution delays even as a result of heuristic greedy algorithms. As a second step, we now design and implement an experimental platform based on a WiFi network of Raspberry Pi 4 devices. While several works, such as Feltin et al. [1], which focuses on distributing inference workloads over multiple edge devices in a simulated environment with real implementation of the DNN models, and Wang et al. [3] which presented a testbed for Split Learning training in IoT, our work targets the inference stage evaluating the partitioning in a real experimentation environment, providing a tool-set for real-time DNN inference scenarios, going up to transformer-based architectures.

In this demo, we make an effort to develop a customizable platform for experimentation with partitioning schemes of different NN architectures. The platform is deployed in a wireless network of resource-constrained devices, including a PC acting as the BS. The end-devices handle part of the DNN computation load locally according to the computed partitioning scheme. This setup allows the investigation of factors affecting DNN partitioning performance, including network latency, computational load, and task scheduling at the BS. Its modular design separates key components like DNN partitioning and scheduling, allowing for customization and extension. The platform supports future research and advanced scenarios like GenAI and LLMs inference at the edge.

[1] Gemma Team and Google Deepmind. 2024. Gemma 2: Improving Open Language Models at a Practical Size.

This work is licensed under a Creative Commons Attribution International 4.0 License.
MOBIHOC '24, October 14–17, 2024, Athens, Greece
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0521-2/24/10
https://doi.org/10.1145/3641512.3690629

2 Platform Architecture

The platform architecture (Fig.1) for NN architectures partitioning involves the following hardware and software infrastructure.

2.1 Hardware Infrastructure

i) **End-Device:** Raspberry Pi 4 units, each equipped with a Cortex-A72 CPU, 4GB of RAM, and WiFi connectivity. The platform includes custom power monitors for each Raspberry Pi, enabling future research on NN architectures partitioning and power consumption. *ii)* **Base Station:** a powerful PC with an Intel i7-5820K CPU and 32GB of RAM and WiFi connectivity.

2.2 Software Infrastructure

The software infrastructure includes necessary components and libraries (e.g. TensorFlow, Pytorch) for developing and executing NN inference tasks under various network scenarios. Each scenario varies in topology, such as the distance of end-devices from the BS, and node configurations regarding compute capacity and memory availability. We implemented the AlexNet DNN model using TensorFlow, trained with the CIFAR10 dataset. All Raspberry Pi devices run on Raspberry Pi 4 Bullseye 64-OS, customized for AI development, while the BS operates on Ubuntu 22.04.

2.2.1 Automation Scripts: Bash and Python scripts manage deployment, DNN model partitioning, and execution flow between Raspberry Pi devices. Monitoring scripts track CPU, memory usage, and network latency. DNN task execution is benchmarked per layer, with metrics collected by the BS, which sets up separate Secure Shell (SSH) channels with each end-device.

2.2.2 Key Procedures: DNN inference is executed based on partitioning schemes calculated by the BS, which uses task schedulers (e.g., Shortest Job First (SJF) or Round Robin (RR)) to schedule DNN tasks from devices. The BS centrally orchestrates the remote execution performance monitoring of end-devices, which execute up to the partition point and send output data at the cut layer to the BS for task completion.

2.2.3 NN Architectures Partitioning Manager (NAPM):. This tool (Fig. 2) configures, and orchestrates the DNN partitioning experimentation scenarios on the platform. It offers a user-friendly interface for defining network nodes, their placement and networking parameters configuration. It supports the creation of experimental scenarios and their execution, allowing users to set network conditions and apply latency and load parameters. The configuration of various scenarios in terms of latency between end-devices and the BS is performed by autogenerated Bash scripts using the *tc* Linux tool. For computational and memory load application on the end-devices, custom Bash scripts using the *stress* and *cpulimit* Linux tools, create artificial computation and memory load based on user-defined input parameters through the NAPM tool during scenario creation and configuration.

3 Experimentation

In a typical experimentation scenario, researchers can investigate DNN partitioning performance under varying network conditions

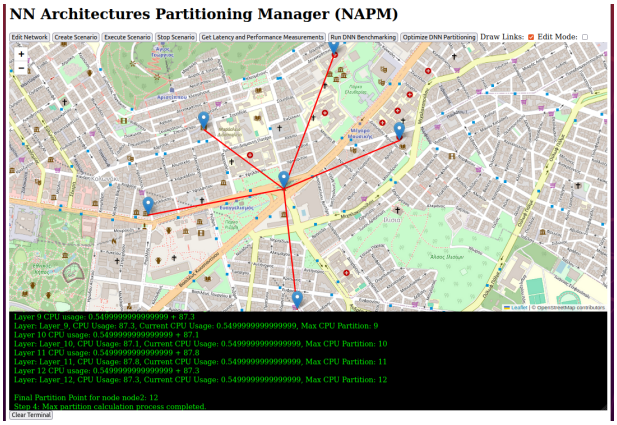

Figure 2: The main screen of the NAPM tool's GUI.

and computational loads. First, the user configures the computational load of the end-devices and the network topology through the NAPM tool. The BS may request the end-devices' computational capacities and DNN layer benchmarking results after initiating the scenario. Using this data, the BS calculates a partitioning scheme, either using our heuristic algorithm from [2] for minimizing total delay or delay variance across tasks, or a custom algorithm uploaded by the user, and transmits it to the end-devices. During execution, each device processes its DNN tasks up to the partition point and sends the output data to the BS, where the task scheduler completes the offloaded tasks and computes the final execution delays.

The presented platform allows users to implement their own partitioning algorithms, load different AI models, and measure performance metrics like latency. Future plans include adding an energy consumption measurement module, and associated algorithms. Additionally, we plan to work on accommodating and testing partitioning schemes for transformer-based architectures.

Acknowledgments

This work was conducted in the context of the Horizon Europe project PRE-ACT (Prediction of Radiotherapy side effects using explainable AI for patient communication and treatment modification). It was supported by the European Commission through the Horizon Europe Program (Grant Agreement number 101057746), by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 22 00058, and by the UK government (Innovate UK application number 10061955).

References

[1] T. Feltin, L. Marchó, J. Cordero-Fuertes, F. Brockners, and T. Clausen. 2023. DNN Partitioning for Inference Throughput Acceleration at the Edge. *IEEE Access* 11 (2023), 52236–52249.

[2] D. Kafetzis and I. Koutsopoulos. 2024. DNN Partitioning and Inference Task Offloading in 6G Resource-Constrained Networks. In *2024 EuCNC/6G Summit.*

[3] Z. Wang, L. Boccardo, and Y. Deng. 2024. An Edge-Enabled Wireless Split Learning Testbed: Design and Implementation. *IEEE Communications Letters* 28, 6 (2024), 1337–1341.