Lightning Talk: Efficient Embedded Machine Learning Deployment on Edge and IoT Devices

Sudeep Pasricha
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, CO, United States
sudeep@colostate.edu

2023 60th ACM/IEEE Design Automation Conference (DAC) | 979-8-3503-2348-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/DAC56929.2023.10247845

***Abstract* –There has been rapid growth in the use of machine learning (ML) software in emerging edge and IoT systems. ML software deployments enable analytics and pattern recognition for multi-modal data (e.g., audio, images/video, wireless signals, air quality) obtained from embedded sensors and transceivers. However, resource constraints in edge and IoT platforms make it challenging to meet quality-of-service and real-time goals. The growing complexity of ML also exacerbates these issues. We discuss the challenges of ML software deployment in edge and IoT platforms, present strategies to ease deployment, and discuss case studies from the automotive, indoor navigation, and hardware/software co-design domains.**

Keywords—embedded software, machine learning, edge computing, IoT computing, model optimizations

I. Embedded ML Software in Edge and IoT Platforms

Machine learning (ML) software is being actively deployed in edge and IoT platforms in our everyday lives. Our homes have smart thermostats (e.g., Nest), smart speakers with voice assistants (e.g., Amazon's Alexa), surveillance IP cameras (e.g., Wyze), and virtual reality gaming headsets (e.g., Meta Quest) that rely on ML software. Our cars are using ML for vehicle localization; detection of pedestrians, traffic/road signs, and lanes; and advanced driver assistance systems [1]. Even our smartphones today use ML in almost every aspect of how we interact with these devices [2].

While in many of these cases, ML software is deployed in the cloud and accessed via API calls from devices connected to the Internet, there is a growing push to implement ML models on the resource-constrained devices. The primary motivation here is to avoid the uncertainty and overheads associated with wireless communication. Not only does wireless communication lead to high power consumption in devices (e.g., up to 70% [3]) and reliability issues due to wireless signal interference or lack of available wireless network coverage at locations, but the latency associated with communication can exceed real-time requirements in many applications (e.g., in real-time automotive perception systems).

For these reasons, there has been growing interest in supporting on-device ML software execution on edge and IoT platforms, ranging from base stations and mobile devices with GPUs and hardware accelerators to simpler CPU-based systems. At the extreme end of this device spectrum are tiny microcontrollers which are being considered as platforms for ML deployment to achieve low-power audio keyword spotting (i.e., recognizing a word or phrase), anomaly detection and forecasting from sensor data, and pattern recognition (e.g., detecting faces, images, objects, gesture, activities, text) [4]. However, there remain many open challenges with deploying ML on resource-limited edge and IoT platforms, as discussed next.

II. ML Deployment Challenges

ML deployments on edge and IoT devices face many important challenges: *1) Power dissipation:* the low cost and compact nature of many edge/IoT platforms limits the use of fans or liquid cooling. As a result, power dissipation has to be carefully controlled to meet stringent TDP (thermal design power) goals that can be as small as a few milliWatts. Meeting reasonable performance goals (e.g., sub-second runtimes) with a large class of ML software algorithms, such as deep learning techniques, can easily exceed these requirements; *2) Energy consumption:* many edge/IoT platforms are battery-driven, and as such have a limited energy store at their disposal. Compute intensive ML models, such as the neural network family of models that require large numbers of matrix multiplication operations, can drain the battery quickly, and reduce uptime in devices; *3) Memory footprint:* edge and IoT platforms are typically memory limited, to reduce device area, power, and cost. Even relatively simple deep learning ML models have peak memory requirements of several GBs, which cannot be supported in these devices; *4) Computation complexity:* To support high accuracy, executing large and complex ML models are crucial. These models can have requirements that exceed hundreds of GFLOPs, whereas edge and IoT platforms may only be capable of delivering a few GFLOPs, and up to tens of GFLOPs in the best case; *5) Real-time constraints*: many applications have real-time constraints, requiring ML models to not only generate the correct results, but also to do so within a time constraint. For instance, indoor navigation with smartphones, and automotive perception-to-actuation have strict timing constraints that require ML predictions within 50-200 milliseconds [5], [6]. Meeting such goals is difficult with the limited resources in edge and IoT devices.

III. Strategies for Efficient ML Deployment

Due to the resource limitations of edge and IoT platforms, training ML models on them is usually impractical. But even deploying and running pre-trained models for inference runs into the issues discussed in the previous section. To unlock the full potential of ML on edge and IoT devices, there is a need for strategies to optimize ML runtime behavior. ML models need to fit in limited memory and utilize limited processing capabilities, which requires creative approaches that can limit the size of the input and the number of layers in the ML model, optimize the parameters and computations within ML models, or make use of lightweight non-neural network-based ML algorithms to accomplish application goals.

The approaches that have shown the most promise to reduce ML overheads across edge and IoT platforms include: *1) Model selection:* sometimes the extra few % of accuracy with predictions that comes with more complex ML models can be sacrificed in favor of simpler ML models with fewer parameters that provide "good enough" accuracy. For example, the family of single-stage object detectors, including SSD, YOLOx, and RetinaNet have much fewer parameters and faster inference speeds than the more accurate two stage object detectors, such as Faster R-CNN [7], *2) Quantization:* the 32-bit floating point values of weights in parameterized ML models (such as neural networks) can lead to significant memory footprint and computation overheads. Therefore, converting these weight parameters (and also activations) to fixed point integer values with lower bitwidths (e.g., 8 bits) is desirable. In many cases quantization with retraining can lead to minimal reduction in accuracy while leading to orders of magnitude reduction in inference time and memory footprint [8], *3) Pruning:* removing redundant, non-informative weights in a pre-trained ML model (e.g., DNN or CNN), or training such models with sparsity constraints not only allows compressed storage of such models in memory, but can also speed up their inference [9]; *4) Knowledge distillation:* by shifting knowledge from a large teacher model into a smaller one, and by learning the class distributions output via softmax, it is possible to compress deep

This research is supported by grants from NSF (CNS-2132385, CCF-1813370)

and wide networks into shallower ones, where the shallower model mimics the functions learned by the complex model [10]; *5) Loop optimizations:* as ML models are often custom coded (e.g., using C or C++) for resource-constrained platforms, and involve time-consuming operations on tensors in loops, optimizations such as loop tiling, loop unrolling, and loop reordering can change the data access patterns in a manner that improves spatial and temporal locality that can be better exploited by caches, to improve performance [11].

IV. Case Study: Automotive Platforms

Efficient deployment of ML in automotive platforms is a topic of much research, given the rise of autonomous vehicles. Here we present recent examples of ML software optimization approaches for the automotive domain. In [12], a framework for optimizing ML-based perception architectures for autonomous vehicles was proposed. The main contribution of the framework was an iterative, semi-structured pruning approach that was able to reduce inference time, energy use, and memory footprint for single stage object detectors such as YOLOv5s and RetinaNet. In [13], a framework for co-optimizing the selection of locations and orientations for sensors, object detectors, and sensor fusion algorithms was proposed for semi-autonomous vehicles and demonstrated for the Audi-TT and BMW-Minicooper vehicles. The framework involved selecting appropriate object detectors for a given vehicle and sensing goal using multi-modal sensors, while also fine tuning the object detectors using neural architecture search (NAS) techniques to improve inference accuracy and performance. In [14], an efficient ML-based framework for anomaly detection in automotive networks was proposed. The framework made use of a simplified ML model with a powerful temporal convolutional neural attention mechanism to learn to detect patterns in sequence data. The model reduced memory footprint, inference time, and model parameters by several orders of magnitude compared to the state-of-the-art [15].

V. Case Study: Indoor Navigation

Efficient deployment of ML for indoor localization with mobile devices is an emerging domain that is poised to usher in highly precise emergency response; seamless robot, human, and UAV navigation; and location-based services within indoor, dense urban, and subterranean environments, where GPS signals cannot penetrate. Here we present recent examples of ML software optimization approaches for the indoor navigation domain. In [16], a framework for simultaneous quantization and pruning of ML models used in indoor localization was proposed. Using these two ML model compression techniques, the framework was able to deploy a convolutional autoencoder and a CNN classifier on resource-scarce devices with an inference latency of just a few milliseconds to meet the needs of real-time navigation. Moreover, the bitwidth reduction in model parameters and model pruning allowed the ML model to fit in less than 100KB of memory, while maintaining acceptable indoor localization accuracy. In [17], an early exit strategy was leveraged to speed up ML inference on mobile devices. The strategy involved training an ML model in a manner that allowed making predictions with high confidence in a majority of scenarios after executing just a few layers of the ML model. Such an early exit during model inference was able to achieve up to 42% reduction in inference latency and 45% reduction in inference energy. In [18], a large vision transformer ML model architecture was simplified and adapted for deployment on mobile devices, with a small enough footprint (less than 250K parameters) to accomplish a localization prediction in 50 milliseconds or less, while outperforming ML models (in terms of accuracy) that were several orders of magnitude larger.

VI. Case Study: HW/SW Co-Design

If co-designing the ML software and hardware platform is possible, it opens up new avenues for optimizing performance and energy-efficiency for ML software on edge and IoT platforms. As an example, [19] proposed a co-design approach that trained hyper-quantized binary neural network ML models (with 1 bit weights and 4 bit activations) and simultaneously customized an optical hardware accelerator platform to execute these models efficiently. Similarly, [20] proposed a co-design approach that trained sparse neural network ML models and simultaneously customized the optical hardware accelerator platform to execute these models efficiently. These approaches were able to improve ML energy-efficiency and power consumption by over 10× compared to an optical hardware accelerator [21] that was designed for high performance server platforms. In [22], inference with large transformer models such as BERT [23] was optimized, enabling it to be efficiently executed by co-designing the device, circuit, and architectures layers in the hardware together with the ML software model. The deployed ML models showed orders of magnitude reduction in energy-per-bit compared to CPU, GPU, FPGA, and other ML accelerator platforms.

VII. Conclusions

In this paper, we presented a brief overview of some of the key challenges with deploying ML software on edge and IoT platforms. Given the many resource limitations of edge and IoT platforms, novel approaches are needed to deploy powerful ML models on these platforms. We discussed some effective strategies to improve ML performance, energy-efficiency, and reduce its memory footprint on edge and IoT platforms. We presented case studies of efficient ML deployment from the automotive and indoor navigation application domains. Lastly, we motivated the use of hardware/software co-design to further optimize efficiency metrics.

References

[1] V. Kukkala, et al., "Advanced driver assistance systems: a path toward autonomous vehicles", IEEE Consumer Electronics, 2018.

[2] "Apple using machine learning for almost everything, and privacy-first approach actually better", 2020 [Online]: https://9to5mac.com/2020/08/06/apple-using-machine-learning/

[3] S. Branco, et al., "Machine learning in resource-scarce embedded systems, FPGAs, and end-devices: A survey." Electronics 8.11, 2019.

[4] H. Han, et al. "TinyML: A systematic review and synthesis of existing research." In IEEE ICAIIC, pp. 269-274, 2022.

[5] C. Langlois, et al., "Indoor localization with smartphones", IEEE Consumer Electronics, 6(4), 2017.

[6] K. Strobel et al. "Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions." IEEE/RSJ IROS, 2020.

[7] A Balasubramaniam et al., "Object detection in autonomous vehicles: Status and open challenges", arXiv, 2022.

[8] A. Gholami, et al. "A survey of quantization methods for efficient neural network inference." arXiv, 2021.

[9] J. Liu, et al. "Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey." arXiv 2020.

[10] J. Gou, et al. "Knowledge distillation: A survey." IJCV, 2021.

[11] J. Gao, et al. "A systematic survey of general sparse matrix-matrix multiplication." ACM Computing Surveys 55.12: 1-36, 2023.

[12] A. Balasubramaniam, et al., "R-TOSS: A framework for real-time object detection using semi-structured pruning", IEEE/ACM DAC, 2023.

[13] J. Dey, et al., "Robust Perception Architecture Design for Automotive Cyber-Physical Systems", IEEE ISVLSI, 2022

[14] S. V. Thiruloga, et al., "TENET: Temporal CNN with attention for anomaly detection in automotive cyber-physical systems", IEEE/ACM ASPDAC, Jan 2022

[15] V. K. Kukkala, et al., "INDRA: Intrusion Detection using Recurrent Autoencoders in Automotive Embedded Systems", IEEE TCAD, 2020.

[16] L. Wang, et al., "CHISEL: Compression-aware high-accuracy embedded indoor localization with deep learning", IEEE ESL, 2022.

[17] S. Tiku, et al., "QuickLoc: adaptive deep-learning for fast indoor localization with mobile devices", ACM TCPS, 17(4), Oct 2021.

[18] D. Gufran, et al., "VITAL: vision transformer neural networks for smartphone heterogeneity resilient and accurate indoor localization", IEEE/ACM DAC, 2023.

[19] F. Sunny, et al., "ROBIN: A robust optical binary neural network accelerator", ACM TECS, Volume 20, Issue 5s, Oct 2021.

[20] F. Sunny, et al., "SONIC: A sparse neural network inference accelerator with silicon photonics for energy-efficient deep learning", IEEE/ACM ASPDAC, Jan 2022.

[21] F. Sunny, et al., "CrossLight: A cross-layer optimized silicon photonic neural network accelerator", IEEE/ACM DAC, 2021

[22] S. Afifi, et al., "TRON: transformer neural network acceleration with non-coherent silicon photonics", ACM GLSVLSI, 2023.

[23] J. Devlin, et al., "BERT: pre-training of deep bidirectional transformers for language understanding," in CoRR, 2018.