Optimizing Deep Learning Inference for Resource-Limited Edge devices.

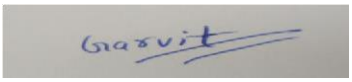
PHN-319 - Technical Communication - Report

Garvit (22123015)
B.Tech Engineering Physics (2022-26)

Supervisor: Prof. Anirban Mitra

Dept. of Physics, IIT Roorkee

November 12, 2024



Garvit

Prof. Anirban Mitra

Abstract

This paper focuses on optimization techniques for deep learning inference on resource-constrained edge devices. Applications of DL models on edge platforms, including smartphones and IoT devices, prove challenging in scenarios where memory, processing power and energy resources are limited. Unlike cloud-based environments, edge devices require efficient real-time processing with as little energy consumption as possible and maintaining data privacy. This report reviews the state-of-the-art in DL inference optimization, comprising model architecture design lightweight models, techniques for pruning and quantization, and integration of algorithm-hardware co-design for feasible, high performance inference on edge devices.

KEYWORDS | Edge AI; Model Compression; Neural Architecture Search (NAS) ; Algorithm-Hardware Codesign ; Low-Latency Processing ; Energy Efficiency.

Introduction

Machine learning (ML) algorithms are also a powerful tool that offers automatic decision making abilities by extracting and learning features from data. Deep learning (DL) is a term coined for ML because it has so far revolutionized several technological domains due to its ability of automatically learning complex features by multiple layers of neural networks. NNs, often called multilayer perceptrons (MLPs), have three layers, namely, input, hidden, and output layers. Then, forward and backward propagation take place for adapting the weights and bias of the model. All such processes are executed to get the optimal prediction by the model.

DL models are predominantly executed for inferencing in two environments: cloud processing and edge processing. The cloud processing offers a large scale computation resource which has its advantages but at the same time drawbacks owing to latency and security issues external data comes into play and out of play when data is being processed. Edge-based inference - where the data is processed as close to its source as possible - minimizes transmission time as well as encourages confidentiality. Nevertheless, there are challenges in carrying these DL models on the edge devices whose resources have limitations such as less memory and less processing power with energy constraints.

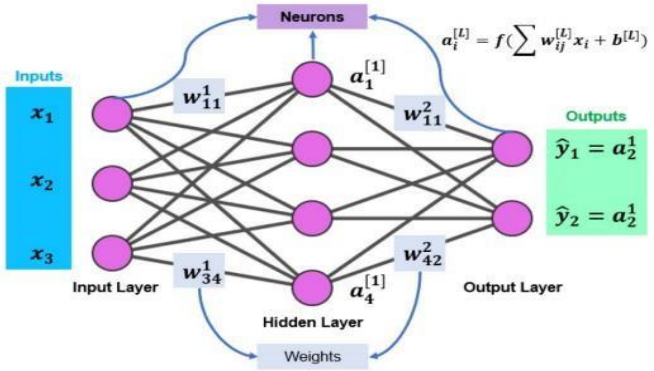


Fig. 1. *Overview of the terminology, operations, and processing in a basic NN model. Notations: x = example of training data, w_{ij}^L = weights connecting neuron i to j in layer L, $a_i^{[L]}$ = activation of neuron i in layer L, f = activation function, $b^{[L]}$ = bias in layer L, and $\hat{y}$ = predicted output.*

Optimization Techniques for Edge Devices

Several techniques are developed to make DL models deployable on resource-limited edge devices. Some of the main methods include:

1. **Model Pruning**: Pruning removes less important weights or neurons, thereby reducing the size of the model and the amount of computation needed. Pruning could be either structured (complete removal of filters) or unstructured (reduction of individual weights). Although structured pruning is usually better in terms of regularity in computation, unstructured pruning may lead to sparse matrices, which are quite hard to implement efficiently on hardware.

(a) $wij=0,\ if\ |wij|<\epsilon$ (b)$Fi=0,\ if\ contribution(Fi)<\delta$

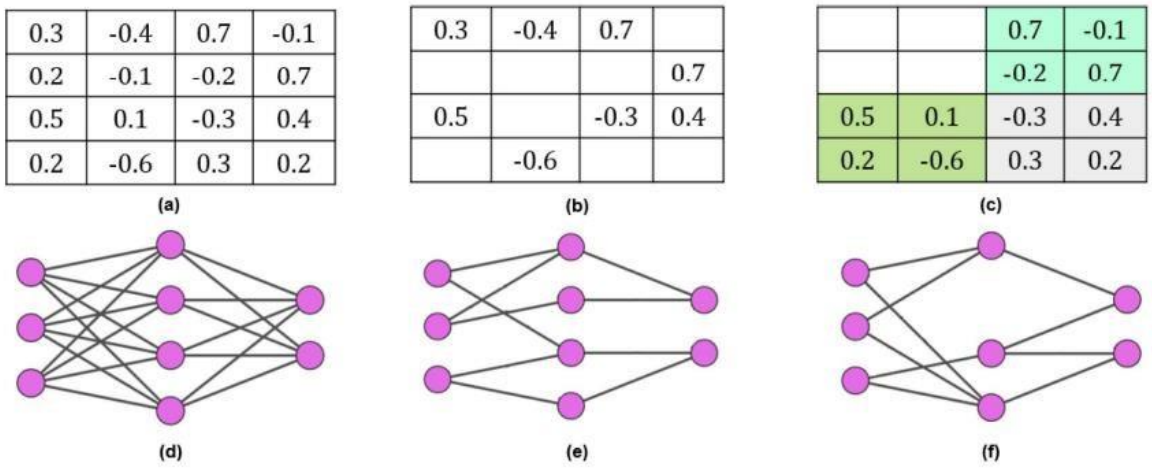
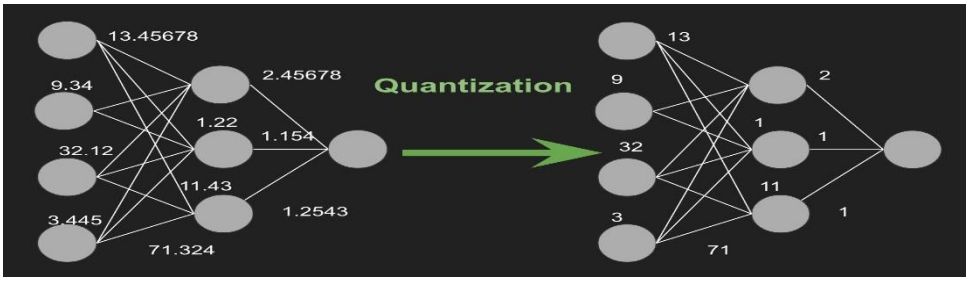


Fig. 10. *Illustration of pruning techniques. (a) Original weight matrix. (b) Magnitude-based unstructured pruning of 50% smaller magnitude weights. (c) Structured block pruning (pruning based on the lowest average of 2 × 2 weight groups). (d) Basic FNN architecture without pruning. (e) Result of unstructured 50% weight pruning on (d). Example of neuron pruning on (d).*

2. **Quantization:** Quantization reduces the weight and activation precision from 32-bit floating point down to a lower bit representation say, 8-bit integer. In this case, it reduces the memory and computational requirements while maintaining reasonable accuracy. Quantization is very useful for hardware accelerators because it allows more parallel processing with a lower energy consumption.

$$q=round(w/\alpha)$$



3. **Knowledge Distillation**: In this, a large "teacher" model trains the smaller "student" model with an understanding that moves the knowledge from the teacher, bringing down complexity without much loss in performance.

$$L=(1-\lambda)\times L_{true}+\lambda\times L_{teacher}$$

where L_{true} is the cross-entropy loss with ground truth labels, $L_{teacher}$ is the distillation loss from the teacher's outputs, and λ is a balancing hyperparameter.

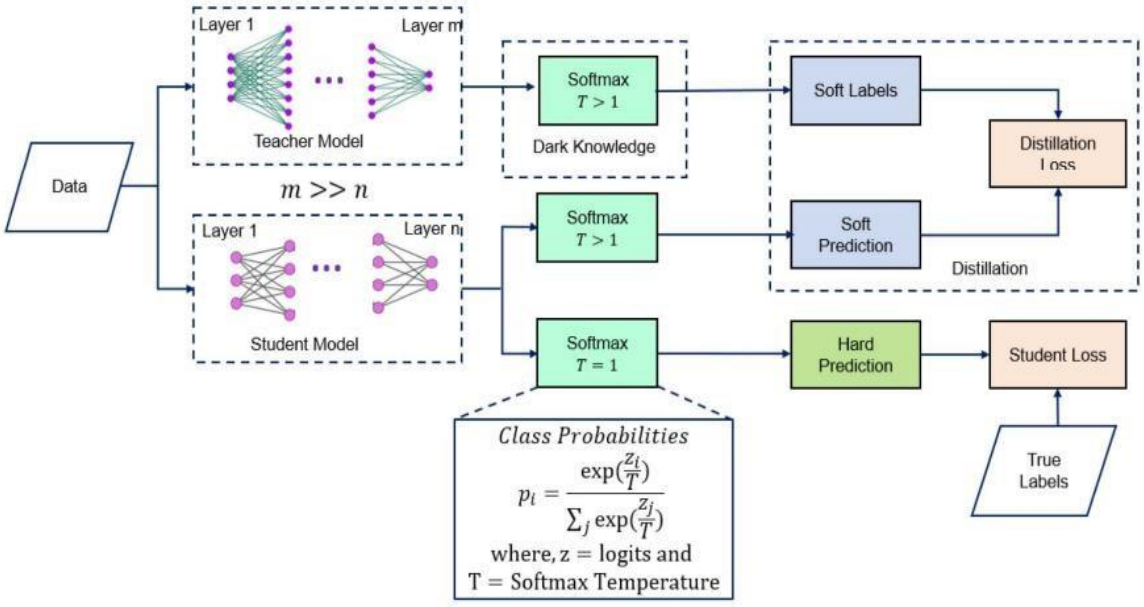


Fig. 11. *Illustration of KD [178]. The knowledge from the pretrained large teacher model is transferred to the shallow student model through the minimization of distillation loss.*

4. **Neural Architecture Search (NAS)**: NAS algorithms iteratively search through the possible solution space to find architectures that are optimal based on specific constraints, such as memory usage or processing time. Techniques like MobileNet have applied NAS to create well performing models suitable for the constraints on mobile devices. NAS finds architectures by evaluating possible structures within a defined search space:

$$\min_{\alpha} \quad \mathcal{L}(f(x;\alpha), y) + \Omega(\alpha)$$

where $\mathcal{L}$ represents the loss function, $f(x;\alpha)$ is the model prediction parameterized by architecture α, and $\Omega(\alpha)$ is a regularization term for complexity.

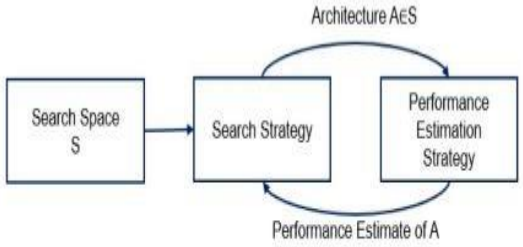


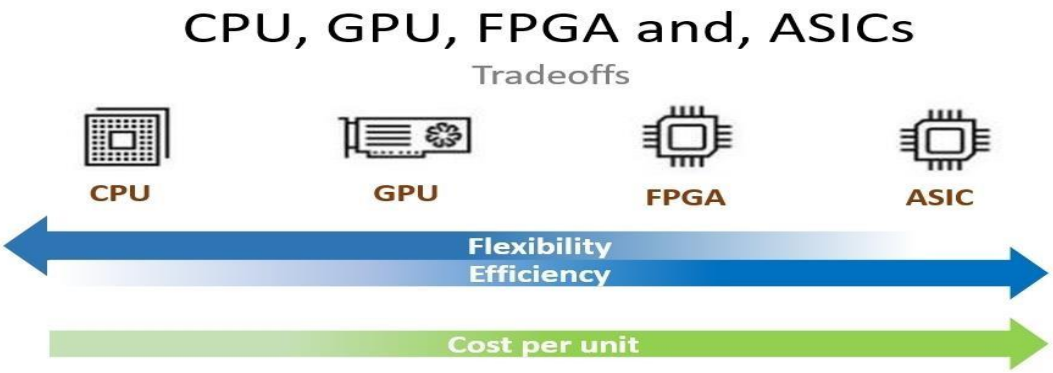
Fig. 7. *Components of NAS. NAS techniques consist of a search space S. Using the search strategy an architecture A is obtained, which is evaluated using the performance estimation strategy.*

Hardware Acceleration and Codesign

Hardware Acceleration and Codesign Algorithm-hardware codesign for matching DL models with hardware accelerators: Improving efficiency on edge devices. Common hardware platforms:

- Application-Specific Integrated Circuits (ASICs): These specialized chips, similar to the Google TPUs, are designed for DL operations and conserve energy only for certain models.
- Field-Programmable Gate Arrays (FPGAs): FPGAs are field-programmable and configurable settings within the scope of parallel processing, making them ideal for real-time applications.
- Advanced graphic processing units (GPUs) are quite power demanding of which in most cases enhances the work of deep learning as compared to any other tasks that require high degree of parallelism.

The aim is to increase throughput while minimizing Latency and power consumption factors.



Applications and Future Trends

A few overbearing inference can be evaluated on an edge device that is more cost effective in many aspects of its application.

- Healthcare: Monitoring and diagnostics in real time – most especially in ICU wards, extremely low latency will be needed to carry out DL inference to detect critical conditions.

- Autonomous Vehicles: Vehicles rely on fast processing to detect obstacles, traffic signals, and pedestrians, ensuring safety through rapid decision-making.

- Smart Wearables: Wearable devices can detect anomalies, track fitness data, and even predict health issues, all requiring efficient, real-time DL inference.

- Security and Surveillance: Edge inference enables real-time analysis in video surveillance, allowing systems to detect and respond to threats instantly.

As edge AI continues to grow, emerging trends focus on achieving greater energy efficiency, minimizing latency, and improving the robustness of DL models in variable real-world conditions. Moving forward, researchers are exploring methods for distributed processing and real-time DL model adaptation, which will further optimize DL performance on the edge.

References

[1] *Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review. December 2022 Proceedings of the IEEE·111(1):1-50*

[2] *Efficient Deep Learning Inference on Edge Devices Ziheng Jiang . Tianqi Chen . Mu Li*

[3] *AI Computing Chip Analysis for Software-Defined Vehicles [BLOG]*

[4] *Article: Understanding Model Quantization in Large Language Models*