University of Cyprus

DEPARTMENT OF COMPUTER SCIENCE

DepGraphRL: A Deep Reinforcement Learning-Driven, Energy-Aware, Dependency Graph-Based Pruning Framework for Edge-IoT Devices

Asfa Jamil

A dissertation submitted to the University of Cyprus

in partial fulfillment of the requirements

for the degree of Masters of Science in Artificial Intelligence

01, 2024

VALIDATION PAGE

Masters of Science Candidate: Asfa Jamil

Masters of Science Dissertation Title: DepGraphRL: A Deep Reinforcement Learning-Driven, Energy-Aware, Dependency Graph-Based Pruning Framework for Edge-IoT Devices

The present dissertation was submitted in partial fulfillment of the requirements for the Degree of Masters of Science in Artificial Intelligence at the Department of Computer Science and was approved on ***01 08, 2024*** *by the members of the Examination Committee.*

Examination Committee:

Research Supervisor: Professor Elpida Keravnou-Papailiou

Committee Member: Associate Professor Vasos Vassiliou

Committee Member: Assistant Lecturer Nouman Ashraf

DECLARATION OF Masters of Science in Artificial Intelligence CANDIDATE

The present dissertation was submitted in partial fulfillment of the requirements for the degree of Masters of Science in Artificial Intelligence of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

Asfa Jamil

Abstract

Recent technological advancements have revolutionized the role of the Internet of Things (IoT) in various commercial and industrial sectors, including applications in smart city, agriculture, healthcare, e-commerce, and industrial automation. Further, the landscape of IoT innovation has steered toward automation and real-time adaptability with the fusion of Artificial Intelligence knowledge in the domain of Deep Learning, Machine Learning and Reinforcement Learning. However, due to the small size of IoT devices, they often face computational capability and energy efficiency limitations. To overcome the computational constraints of IoT devices, the integration of cloud computing has been rigorously explored in different IoT applications. Though cloud computing provides powerful and robust computation servers to the IoT devices and allows offloading of various application tasks, the issue of high network and transmission latencies still persists. This necessitates the shift of paradigm towards the edge computing, transitioning consumer IoTs towards Edge-IoT devices. Though the Edge-IoT devices ensure the Quality of Services by leveraging computation near to IoT devices, the computation of complex tasks on battery constraint Edge-IoT devices require efficient and intelligent energy management schemes, especially in the context of operating Deep Neural Networks, such as Convolutional Neural Network (CNN), for image classification.

A practical and renowned approach to reduce the energy consumption of Edge-IoT devices for operating CNN models is to deploy efficient energy-aware, less structurally complex neural network models on Edge-IoT devices using *network pruning*. However, this deployment comes with various constraints. The first constraint revolves around the accurate measurement of energy consumption in any baseline CNN model. The traditional methods

of estimating energy consumption by using theoretical formulas often fall short considering practical scenarios where different natural factors also contribute in evaluation methods. Thus, the evaluation of energy consumption requires empirical measurement methods alongside the theoretical approaches. The second constraint is the limitations of existing pruning methodologies, particularly the lack of generalizability across various neural network architectures and the complexities introduced by structural coupling in pruning processes. The third and final constraint delves into the optimization challenges in pruning CNN models for Edge-IoT devices, considering the multifaceted nature of variables within a dynamic environment, such as layer count, energy consumption, and accuracy requirements. These challenges necessitate a novel approach, combining empirical energy measurements with advanced pruning techniques and optimization strategies.

To resolve these constraints, this thesis proposes a novel DepGraphRL framework to optimise the CNN models on Edge-IoT devices. The proposed framework utilizes Deep Reinforcement Learning (DRL) to provide an energy-aware, fully automated, and generalizable layer-wise pruning technique for neural networks by employing DepGraph for structural pruning and empirical methods for energy consumption calculation of CNNs at edge devices. The proposed framework considers the pruning of neural networks as a Markov Decision Problem. It utilizes an Advantage Actor-Critic algorithm for optimal policy training to decide which layers to prune while balancing available energy and the required accuracy tradeoff. A unique feature of the framework is the utilization of a dependency graph-based structural pruning approach, i.e. DepGraph; this approach ensures generalizability of the same pruning technique to the different neural networks in an automated fashion and also handles issues of structural coupling while preserving the original model architecture and Performance. The empirical methods used in calculating the energy consumption of neural networks on Edge-IoT devices support the applicability of the proposed framework in

real-world applications compared to theoretical methods.

Using different pruning ratios and feature importance ranking techniques, the proposed framework has been extensively evaluated on different neural network architectures like ResNet-18, VGG-19, and MobileNet. Compared to the baseline Depgraph pruning method, the proposed DepgraphRL framework reduces overall energy consumption while obtaining accuracy comparable to the baseline neural network models. This shows the effectiveness of the proposed energy-aware framework in optimizing the performance of neural networks in energy-constraint environments by providing energy-efficient pruned architectures of baseline CNN models with comparable accuracy to the baseline.

Acknowledgments

In the name of Allah, the most Gracious, the most Merciful, I truly believe in his ultimate power and that he is my ultimate strength. I want to extend my sincerest gratitude to Dr. Eplida Keravnou for her constant support and guidance during my work on this thesis. I am deeply thankful to my thesis co-supervisor, Dr. Nouman Ashraf, who helped me shape and reform my thesis work. Words can't express my gratitude to my husband, Abdullah Bin Masood, for his mentorship and emotional support during the whole process. And my parents, Dr. Jamil Ahmad and Sajida Rehman, I owe everything to them.

Contents

1 Introduction **1**

1.1 Thesis Motivation 2

1.2 Thesis Contributions 4

1.3 Thesis Organization 6

2 Background Knowledge and Literature Review **7**

2.1 Energy Consumption in CNNs 7

2.2 Network Pruning Techniques 8

3 Proposed DepGraphRL Framework **10**

3.1 Problem Formulation 10

3.2 Energy Consumption Analysis: Theoretical vs Hardware Perspectives . . . 13

3.3 DepGraph: Group-Level Pruning in Neural Networks 17

3.3.1 Understanding Dependencies in Neural Networks 17

3.3.2 Dependency Graph 18

3.4 DepGraphRL Framework: Proposed Scheme 22

3.4.1 Definition of State, Action, and Reward 23

3.4.2 Advantage Actor-Critic (A2C) Algorithm 25

4 Experiments And Performance Evaluation **28**

4.1 Experimental Parameters And Setup 28

4.1.1 Deep Neural Network (DNN) Models 29

4.1.2 Datasets: 30

4.2 Performance Evaluation 31

4.2.1 Comparison of Theoretical Energy with Hardware-Based Energy . 33

4.2.2 Impact of Number of Filters on Pruning Decisions 35

4.2.3 Energy-Accuracy Comparison 36

5 Conclusion and Future Work **44**

Nomenclature

A2C	Advantage Actor-Critic
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
DepGraph	Dependency Graph-based pruning technique
DepGraph	Dependency Graph
DRL	Deep Reinforcement Learning
Edge-IoT	Edge computing in Internet of Things
FLOPs	Floating Point Operations
IoT	Internet of Things
MDP	Markov Decision Problem
NVML	NVIDIA Management Library
RA	Reference Architecture
RL	Reinforcement Learning

List of Algorithms

1 Empirical Energy Consumption Measurement for CNN Layers on GPUs . . 16
2 Dependency Graph [1] 22
3 Grouping [1] 22

List of Figures

3.1 Parameters grouped together with interdependencies, structured in various formats. It's essential to prune all the emphasized parameters at the same time [1]. 18

3.2 Grouping of layers is accomplished through recursive propagation on the Dependency Graph, beginning with the output of the fourth convolutional layer, f_4^+. In this scenario, the input f_4^- and output f_4^+ of the convolutional layer do not have an Intra-layer Dependency, owing to the differing pruning strategies demonstrated earlier [1]. 20

3.3 Proposed model for energy-aware pruning of neural networks using a dependency graph, enhanced by deep reinforcement learning within the DepGraph framework, specifically designed for Edge-IoT devices. 23

4.1 A comparative analysis between the evaluation of the theoretical energy ($E_{\text{Theoretical Energy}}$) and hardware energy ($E_{\text{Hardware Energy}}$) for the "ResNet-18" CNN model, employing an $l_1 - norm$ filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework. 34

4.2 A comparative analysis between the evaluation of the theoretical energy ($E_{\text{Theoretical Energy}}$) and hardware energy ($E_{\text{Hardware Energy}}$) for the "ResNet-18" CNN model, employing "Random Pruning" filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework. . . 35

4.3 A comparative analysis between the evaluation of the theoretical energy ($E_{\text{Theoretical Energy}}$) and hardware energy ($E_{\text{Hardware Energy}}$) for the "ResNet-18" CNN model, employing "Taylor Expansion" filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework. . . 36

4.4 A comparative analysis in the context of a number of filters between $l_1 - norm$, Random Pruning, and Taylor expansion for the pruning of "ResNet-18" CNN model considering both the DepGraph and the proposed DepGraphRL framework. 37

4.5 Tradeoff between energy (J) and accuracy (%) over ten pruning cycles at a pruning ratio of 0.1 for the ResNet-18 convolutional neural network model applied to the CIFAR-10 dataset. 38

4.6 Tradeoff between energy (J) and accuracy (%) across ten iterations of pruning at a 0.3 ratio for the ResNet-18 CNN model applied to the CIFAR-10 dataset. 39

4.7 Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning at a ratio of 0.5 in the ResNet-18 CNN model, applied to the CIFAR-10 dataset. 40

4.8 Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning for MobileNet applied to the CIFAR-10 dataset. 42

4.9 Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning for VGG applied to the CIFAR-10 dataset. 43

Chapter 1

Introduction

The rapid evolution of information and communication technologies has catalyzed the emergence of the Internet of Things (IoT) [2, 3]. This innovation is pivotal in various commercial [4] and industrial sectors [5, 6], offering significant benefits such as real-time process tracking [7] and advanced image processing [8]. Its applications span across diverse areas including smart homes [9], smart grids [10], agriculture [11], e-commerce [12], and smart cities [13], among others. The integration of Artificial Intelligence (AI) with IoT has further transformed this landscape, steering towards automation and real-time adaptability, leveraging Machine Learning, Deep Learning, and Reinforcement Learning techniques [14–17]. Nonetheless, IoT devices often face limitations in computational capacity and energy efficiency, largely due to their compact size [18, 19].

To mitigate these computational challenges, in literature, various state-of-the-art Reference Architectures (RA) have been proposed. Initially, these architectures depended heavily on cloud computing servers to enhance the computational capabilities of IoT devices and applications [20]. However, the significant transmission latencies associated with cloud computing [21] have prompted a shift towards mobile-edge computing. This transition to Edge-IoT architecture has brought computation closer to the IoT devices, significantly improving the quality of service in computational and processing contexts [22].

Despite the significant progress made in tackling computational constraints, the efficient management of energy in Edge-IoT devices continues to be a critical challenge. This thesis aims to confront the prevalent energy management issues in these devices. Specifically, the thesis research work focuses on the intelligent energy-aware fully automated and generalizable deployment of neural networks, including Convolutional Neural Networks (CNN), on Edge-IoT devices.

1.1 Thesis Motivation

In the domain of Deep Neural Networks, CNNs are regarded as an important class, predominantly used for processing data with a grid-like topology, such as images [23]. A CNN typically consists of a series of layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers, the core building blocks, are particularly effective in capturing spatial and temporal dependencies in an image through the application of relevant filters [24]. This ability makes CNNs ideally suited for tasks such as image and video recognition, image classification, and many other tasks that require processing visual inputs [23] [25].

The computational depth and complexity of CNNs lead to substantial energy consumption, a factor critically analyzed in various studies [26–31]. This high energy demand primarily arises from several key factors. Firstly, the intricate architecture of CNNs, which encompasses numerous layers and parameters, necessitates extensive computational operations that are inherently energy-intensive [26, 27, 32]. These operations require significant processing power, contributing to the overall energy expenditure of deploying CNNs, particularly in resource-constrained environments such as Edge-IoT devices [33, 34]. Secondly, CNNs require frequent memory accesses for retrieving and storing weights and intermediate data. This process of memory access and management significantly adds to the energy usage [35, 36]. Lastly, the energy consumption escalates further due to the substantial data movement between the processor and memory, a factor that becomes particularly prominent in distributed systems where data transfer can involve considerable energy consumption [37, 38].

Based on above presented arguments, it is vital to understand the energy dynamics of CNNs, especially when optimizing these networks for energy efficiency in scenarios such as Edge-IoT devices where power resources are limited. Prior research in this area aims to develop strategies and frameworks that can balance computational demands with energy constraints, ensuring the sustainable deployment of CNNs in various applications [28, 29, 39]. A prominent strategy to mitigate the high energy consumption of CNN models, as identified in the literature, is "*network pruning*", a technique that reduces computational load and energy requirements [40, 41]. Network pruning in CNNs involves reducing the network size by selectively eliminating neurons or connections. This reduction in complexity aims to maintain performance while enhancing energy efficiency [42, 43]. In the literature, multiple techniques of network pruning have been presented such as weight pruning, unstructured pruning, structured pruning, layer-wise pruning, magnitude-base pruning, iterative pruning, automated pruning, importance score, and lottery ticket hypothesis [29, 44–48]. However, in this thesis, two network pruning techniques are focused, i.e., structured and unstructured pruning, and an overview of these is presented before moving ahead.

Structured pruning involves removing entire units like neurons or layers, aligning well with hardware architectures but at the risk of losing key network components, potentially reducing accuracy [49,50]. Unstructured pruning, on the other hand, targets individual weights or connections. While it preserves a larger portion of network functionality, it leads to sparse matrice that present challenges to standard hardware which are generally optimized for dense matrix operations [29,51]. The challenges with structured pruning mainly revolve around the potential reduction in network performance due to the removal of significant network elements [1,52]. Unstructured pruning, while more flexible, results in sparse structures that can complicate hardware implementation, potentially negating the benefits of a reduced network size [53,54].

Both pruning methods offer unique advantages and face distinct challenges. The choice between them depends on the specific requirements of the application, including the balance between accuracy and computational efficiency, and the hardware characteristics for CNN deployment [55–57]. Current research is focused on optimizing these pruning techniques to achieve the best trade-off between maintaining network performance and enhancing energy efficiency in Edge-IoT devices.

In the ongoing evolution of pruning techniques for CNNs, the Dependency Graph (DepGraph) based pruning method, introduced by Fang *et al.* in [1], marks a significant advancement. DepGraph pruning stands out for its innovative approach to structural pruning, addressing the limitations inherent in traditional methods. It is specifically designed to efficiently reduce network complexity while maintaining performance, a critical aspect in the optimization of neural networks. The core strength of the DepGraph method lies in its ability to model the dependencies between layers in a neural network. This facilitates the selective removal of redundant parameters without significantly impacting the network's overall performance. By focusing on the structural integrity of the network during pruning, DepGraph ensures that the resultant network is not only simpler but also retains much of its original accuracy and functionality.

In essence, the DepGraph pruning method is a major step forward in the field of network pruning. It adeptly balances the need to reduce network complexity with the imperative to maintain high levels of performance, making it a particularly promising solution for deploying efficient CNNs in Edge-IoT devices [54, 57].

Another critical issue in ensuring an energy-aware scheme for Edge-IoT devices concerning CNN models is related to the calculation of energy consumption for each layer in the baseline CNN model [33, 34]. The prevalent method in existing research involves using a theoretical formula set to estimate the FLOPs and memory metrics for each layer [32]. These metrics help calculate the baseline model's total energy, which is then used for pruning. However, this method has a significant limitation when it comes to practical real Edge-IoT applications. With advancements in GPU metrics evaluation libraries, like NVML from NVIDIA, new methods have emerged for empirically measuring energy consumption at each layer. These empirical values often differ considerably from theoretical calculations, underscoring the importance of using empirical methods for energy measurement before pruning a baseline model.

To effectively solve the problem of energy-efficient network pruning of CNN models for Edge-IoT devices, various strategies and algorithms have been proposed in the literature. However, these strategies are often required to manage extensive information about the CNN model during the network pruning process, including the number of layers, energy consumption at each layer, target energy consumption, baseline energy consumption, pruned model's accuracy, and target accuracy. Further, with the increasing number of Edge-IoT devices, the problem complexity of energy-efficient network pruning of CNN models to solve for prior proposed optimization strategies rises exponentially. Moreover, the dynamic nature of the Edge-IoT devices' environment over time adds to this problem's complexity.

Given these aforementioned discussed challenges, it is crucial to develop a novel framework that utilizes empirical methods for measuring energy consumption in the baseline model and integrates the DepGraph approach for structural pruning. The proposed framework should also be capable of handling the complexity of the problem and equipping Edge-IoT devices with a dynamic, energy-aware, DepGraph-based group-level pruning scheme.

1.2 Thesis Contributions

In light of the above, Deep Reinforcement Learning (DRL) has emerged as a pivotal platform in optimizing neural networks, particularly in the dynamic context of Edge-IoT devices. DRL combines deep learning with reinforcement learning, enabling an agent to make decisions by interacting with its environment. This combination is ideal for handling the complex, high-dimensional state spaces typical in Edge-IoT applications [58, 59]. Techniques such as model-based reinforcement learning [60] and runtime neural pruning [61] utilize DRL to dynamically adjust the network structure during training or even in real-time operation. This dynamic adaptability is crucial for Edge-IoT devices, which operate under varying computational demands and power constraints. DRL's ability to make context-aware, real-time decisions enables the optimization of the balance between computational load and performance requirements [62, 63].

Therefore, in this thesis, a novel DepGraphRL framework is proposed that incorporates all these features. The proposed DepGraphRL framework represents a novel integration of DRL with the advanced DepGraph pruning method, specifically tailored for optimizing CNN models in Edge-IoT devices. This framework is designed to address the unique challenges presented by the dynamic and resource-constrained nature of these devices, offering a sophisticated solution that combines the strengths of DRL and DepGraph-based structural pruning. At its core, the DepGraphRL framework leverages the DepGraph method for structural pruning, which efficiently models dependencies between network layers and ensures the removal of redundant parameters with minimal impact on performance [1]. Integrated with this is a DRL-based optimization process that dynamically adapts the network structure in real time, making context-aware decisions to optimize computational load and performance requirements based on empirical energy evaluation methods. The contribution of the thesis are highlighted as follows:

- A novel energy-aware DepGraphRL framework, driven by DRL technology, is proposed in this thesis to address the dynamic nature of Edge-IoT devices and their energy-accuracy demands to operate CNN models for image classification.
- The framework also employs a DepGraph based structural pruning method that removes the obstacle of structural coupling by explicitly modelling the dependency between layers and comprehensively grouping coupled parameters for pruning.
- The framework provides an energy-aware fully automatic and general pruning approach to energy-constraint Edge-IoT devices for incorporating any CNN model using Depgraph in comparison to traditional architecture-specific pruners, which offers pruning schemes that need to be manually redesigned for different models making these non-generalizable to new architectures.
- It also leverages the empirical energy evaluation criteria as compared to theoretical evaluation methods to ensure the adaptability of the proposed framework in real-world

scenarios.

- The thesis mathematically formulated the energy-aware structural pruning-based optimization problem for Edge-IoT devices and systematically refined it into a Markov Decision Problem (MDP) by characterizing the state-space, action-space, and reward function.

- To address the formulated optimization problem, the thesis incorporated an Advantage Actor-Critic (A2C) based algorithm, a renowned DRL technique, which is used to train the DepGraphRL agent to obtain the sub-optimal policy.

- Extensive performance evaluations considering different scenarios are conducted to demonstrate the efficacy of the proposed DepGraphRL framework with other baseline pruning schemes.

In summary, the DepGraphRL framework represents a significant advancement in neural network optimization for Edge-IoT devices. Its innovative approach, combining DRL with structural pruning, paves the way for more efficient, adaptable, and powerful Edge-IoT solutions [54, 64].

1.3 Thesis Organization

The thesis is organized as follows: Chapter 2 presents an in-depth background and literature review of the state-of-the-art work. Chapter 3 presents a detailed overview of the DepGraphRL framework while Chapter 4 demonstrates the efficacy of the proposed framework through extensive simulations and experiments conducted under various scenarios. Finally, Chapter 5 concludes the thesis work with future directions.

Chapter 2

Background Knowledge and Literature Review

2.1 Energy Consumption in CNNs

The deployment of CNNs in different IoT applications leads to high energy consumption due to their complex and deep structures. Prior research has contributed significantly in the context of methods to evaluate the energy consumption of CNN models and offer various approaches and optimization method to reduce the energy consumption.

For instance, Fontaine *et al.* [65] discuss the potential of deploying energy efficient neural networks with comprising performance on embedded devices. This study highlights the benefits of processing data on the edge while utilizing energy efficient computationally less expensive models on Edge-IoT devices. Further, the authors also provides insights in their study to optimization techniques to facilitate the deployment of CNNs in energy-constrained Edge-IoT devices. Another notable work by Ye *et al.* in [66] explore the challenges and emerging technologies for low-power AI, focusing on IoT chips. They identify power efficiency, latency, and memory constraints as primary concerns and propose solutions like event-driven architectures and nonuniform sampling techniques. Their research also emphasizes the role of energy harvesting in enhancing the lifespan of IoT devices, pointing towards a synergy between hardware and software optimizations to achieve energy efficiency.

The work conducted by Alenazi *et al.* in [67] delves into neural network embedding in IoT networks. In their work, they demonstrate that optimized neural networks can save up to 86% of bandwidth. Further, their proposed framework, which utilizes Service Oriented Architecture (SOA), showcases the significant energy savings achievable through strategic network optimizations.

Similarly, Govindaraj and Nachimuthu [68] present a capsule neural network-based learning model for IoT networks. Their findings highlight effective network energy optimization, particularly in wireless personal communication systems, thereby contributing to the broader discussion on energy-efficient AI models in IoT.

Zhang [69] focuses on real-time detection of energy consumption in IoT networks. The paper presents a routing planning method that significantly reduces energy consumption, demonstrating the potential of software-level optimizations in managing the energy demands of IoT networks.

These studies collectively underscore the importance of both hardware and software optimizations in reducing the energy consumption of CNNs deployed in IoT and edge computing scenarios.

2.2 Network Pruning Techniques

To optimize neural networks for energy efficiency by reducing structural complexity, neural network technique is often explored. Multiple techniques have been explored in the literature. In this section, an overview of recent advances in this direction has been discussed particularly considering structual and nonstructural pruning.

Structured Pruning: In 2018, Crowley *et al.* [70] explored structured pruning for neural network compression, a pruning technique that focuses on eliminating whole neurons, channels or layers. Lin *et al.* [71] in 2018 introduced innovative methodologies for optimal structured CNN pruning via generative adversarial learning. Bragagnolo *et al.* In 2021, structural pruning was further explored by [72] and Wang *et al.* [73] to remove structural redundancy of neural networks for network compression. Recently, [1] proposed DepGraph, a dependency graph-based structural pruning technique to overcome obstacles of structural coupling and non-generalizability of pruning techniques. This pruning technique models inter and intra-layer dependencies to overcome these obstacles.

Unstructured Pruning: Unstructured pruning removes individual weights or connections that can lead to irregular network structures. Molchanov *et al.* [74] have explored the effectiveness of unstructured pruning in providing energy-efficient networks with better energy accuracy tradeoff in energy-constrained environments. Further, this technique has been explored by [58, 75–79]

Other methods include magnitude-based pruning [80], weight pruning [81] automated pruning [48].

Energy-Aware Pruning Algorithms Above mentioned pruning techniques, do not take in account actual energy consumption by the neural networks. This aspect has also been explored to optimize the pruning technique. In their research, He *et al.* [82] explored the combined impact of pruning redundant network components and also the approximation of computations that can lead to a reduction in energy consumption without significant impact on performance. Yang *et al.* [29] highlighted how the energy consumption of state-of-the-art neural networks like AlexNet and GoogLeNet consider energy limitations o f IoT devices. Montazeri *et al.* [83] and Ghazisaeedi *et al.* [84] have explored energy-aware strategies networking and computing, contributing to the optimization of energy management.

Reinforcement Learning-Based Pruning Techniques For automation of the dynamic requirement of pruning, reinforcement learning (RL) has been explored in literature to enhance computational and energy efficiency considering pruning as an optimization problem. this section explores, some of the recent works in this direction. Bencsik and Szemenyei [60] proposed an RL-based automated neural network pruning technique considering the complex trade-offs involved in pruning, achieving optimal network simplification while maintaining performance. Malik *et al.* [62] utilized a constrained RL to find a balance between the pruning process and learning objectives of the neural network.

Liu *et al.* [85] utilized meta-learning to automate the process of neural network pruning. Lin *et al.* [61] explored RL for pruning during runtime by dynamically pruning network structure based on runtime resources and constraints of energy. In recent development, Zawish *et al.* [86] focused on energy-aware DRL-driven model compression framework for energy harvesting devices utilizing energy management scheme and theoretical energy consumption.

Chapter 3

Proposed DepGraphRL Framework

This chapter presents a novel DepGraphRL framework which is designed with the aim to equip Edge-IoT devices with real-time, energy-efficient pruning of dependency graph-based neural networks. Initially, the chapter presents the challenge of energy-aware pruning and formulated optimization problem. It then delves into the comparison of theoretical vs hardware based CNN energy consumption analysis and specifics of dependency graph based pruning algorithm. The chapter subequently presents the system model, integrating the DepGraphRL framework with Edge-IoT devices. It also define state, action, and reward space that refine the optimization problem into a Markov Decision Process (MDP). To address the formulated optimization problem, the chapter details the A2C algorithm, a DRL approach, employed to train the agent within the DepGraphRL framework. This training aims to develop a sub-optimal policy.

3.1 Problem Formulation

In this section, a formulation of an energy-aware problem is presented. The formulated problem is related to Edge-IoT devices that require an energy-efficient strategy to perform image classification tasks by deploying CNN models. In the considered scenario, the Edge-IoT devices are constrained by their limited energy resources. Consider a baseline CNN model, denoted as M, which is composed of $K \in \mathbb{Z}^k$ layers. The energy consumption of this baseline model is expressed as $E_{\text{Baseline}}(t)$, while its baseline accuracy level is denoted by $\text{Accuracy}_{\text{Baseline}}(t)$. In this thesis, the 'Cross Entropy Loss' evaluation metric is considered to calculate the model's accuracy. To ensure the minimum energy consumption in Edge-IoT device while operating a CNN model and have a long battery life, the aim here is to prune the high energy-consuming layers in a CNN model at the expense of reduction in model accuracy.

Based on this, an EdgeIoT device can provide the agent (i.e., an optimization algorithm) with its set priorities in terms of target accuracy, denoted as $\text{Accuracy}_{\text{Target}}(t)$, of the pruned CNN model so that energy consumption can be reduced. Using this, the optimization problem can be formulated as:

$$\begin{aligned} \text{Minimize:} \quad & E_{\text{pruned}}(t) \\ \text{Subject to:} \quad & C1 : E_{\min}(t) \le E_{\text{Pruned}}(t) \le E_{\text{Target}}(t), \\ & C2 : E_{\text{Pruned}}(t) \le E_{\text{Baseline}}(t), \\ & C3 : \text{Accuracy}_{\text{pruned}}(t) \ge \text{Accuracy}_{\text{Target}}(t), \end{aligned} \tag{3.1}$$

The constraints in our optimization challenge are outlined as follows:

- **Constraint** $C1$: This ensures that the energy consumption of the modified, or 'pruned', CNN model ($E_{\text{Pruned}}(t)$) at any given time (t) does not exceed the target energy consumption specified by the Edge-IoT device. Also, it must be at least as much as the minimum energy consumption ($E_{\min}(t)$) of the original model. This minimum is defined by the energy used by the least energy-consuming layer (k) in the original model.

- **Constraint** $C2$: This guarantees that the energy consumption of the pruned model at any time is always less than or equal to the energy consumption of the original, or baseline, model. It's a way to ensure that the pruning process makes the model more energy-efficient.

- **Constraint** $C3$: This condition requires that the accuracy of the pruned CNN model ($\text{Accuracy}_{\text{Pruned}}(t)$) is always at least equal to the target accuracy level ($\text{Accuracy}_{\text{Target}}(t)$) set by the user.

The main goal of this optimization is to reduce the energy usage of the pruned model, within the limits set by $E_{\min}$ and $E_{\text{Baseline}}(t)$, while also making sure that the model's accuracy does not drop below the set target level. In the literature, different approaches and framework have been proposed that address this optimization problem. However, they fail to consider three significant challenges that are associated with this problem in real-world scenarios.

The first issue concerns the calculation of energy consumption for each layer in the baseline CNN model. The prevalent method in existing research involves using a theoretical formula set to estimate the FLOPs and memory metrics for each layer. These metrics help calculate the baseline model's total energy, which is then used for pruning. However, this method has a significant limitation. With advancements in GPU evaluation libraries, like NVML from NVIDIA, new methods have emerged for empirically measuring energy consumption at each layer. These empirical values often differ considerably from theoretical calculations, underscoring the importance of using empirical methods for energy measurement before pruning a baseline model.

The second issue pertains to the method of pruning a baseline model, particularly with respect to the structural pruning of neural networks. Structural pruning typically accelerates models by removing grouped parameters, but the grouping patterns vary significantly across different architectures like CNNs, Graph Neural Networks (GNNs), Recurrent Neural Networks (RNNs), and Transformers. This variation makes architecture-specific pruners, which depend on manually designed grouping schemes, non-generalizable to new architectures. A major hurdle in this context is structural coupling, which necessitates simultaneous pruning across different layers and assumes that all removed parameters are consistently non-essential, thereby preventing structural complications and significant performance loss post-pruning. To address this, Fang *et al.* [1] recently introduced a general and fully automated method, the Dependency Graph (DepGraph), for modelling layer dependencies and effectively grouping parameters for pruning. This innovation underscores the need to integrate the DepGraph method in Edge-IoT devices for a fully automated and generalizable structural neural network pruning method.

Lastly, various strategies and algorithms have been proposed to solve this optimization problem effectively, especially for pruning CNN models in Edge-IoT devices. These devices might need to prune different baseline models, each with its own number of layers (K) and varying energy consumption. Hence, any optimization method must manage extensive information about the model, including the number of layers, energy consumption at each layer, target energy consumption, baseline energy consumption, pruned model accuracy, and target accuracy. The increasing number of Edge-IoT devices amplifies the complexity of this problem exponentially. Moreover, the dynamic nature of the environment over time adds to this complexity, rendering traditional optimization methods less effective and categorizing this optimization problem as a Mixed Integer Non-Linear Programming (MINLP) problem, which is generally challenging to solve optimally (NP-hard).

Given these three challenges, it is crucial to develop a novel framework that utilizes empirical methods for measuring energy consumption in the baseline model, integrates the DepGraph approach for structural pruning, and is driven by DRL techniques. These techniques should be capable of handling the complexity of the problem and equipping Edge-IoT devices with a dynamic, energy-aware, DepGraph-based group-level pruning scheme. Therefore, in this thesis, a novel DepGraphRL framework is proposed that incorporates all these features. Before proceeding with an overview and discussion of DepGraphRL, the thesis will present a discussion on the empirical method for measuring energy consumption and the details of the DepGraph approach in the subsequent sections.

3.2 Energy Consumption Analysis: Theoretical vs Hardware Perspectives

A crucial aspect of the proposed DepGraphRL framework is the comprehensive analysis of energy consumption for individual layers of the CNN. This analysis is essential as it guides the DepGraphRL framework in understanding the energy impacts of various pruning decisions.

The methodology for analyzing energy consumption encompasses two distinct approaches: a theoretical model and an empirical measurement approach. The theoretical model, grounded in FLOPs and memory requirements, provides a foundational estimate of the energy demands of each CNN layer. This approach is vital for establishing a baseline understanding of energy consumption. On the other hand, the empirical measurement approach, based on real-time energy monitoring on edge devices, offers practical insights. This hardware-based perspective reveals the actual energy usage in real-world scenarios, complementing the theoretical model. Together, these dual perspectives enable a more nuanced and accurate assessment of energy consumption, crucial for optimizing the DepGraphRL model's performance and ensuring energy-efficient operations in Edge-IoT environments.

A standard CNN is composed of multiple convolutional and fully connected layers. Among all layers of a CNN, convolutional layers perform the bulk of the computation, thus requiring higher energy for execution on low-powered IoT devices. A convolution operation is a basic element in a standard CNN whose kernels are defined by a 4-D tensor given as:

$$W \in \mathbb{R}^{C_{\text{in}} \times X \times Y \times C_{\text{out}}} \tag{3.2}$$

where X and Y represent kernel's spatial dimensions, while C_{in} and C_{out} are the number of input and output channels, respectively.

If $I \in \mathbb{R}^{C \times U \times V}$ are the input feature maps with a spatial dimensions of U and V, then the output feature map f with the spatial dimensions (x, y) can be calculated using:

$$T(f,x,y) = \sum_{c=1}^{C}\sum_{x'=1}^{X}\sum_{y'=1}^{Y} I(c, x-x', y-y') \cdot W(c, x', y', f) \tag{3.3}$$

Theoretical-Based Energy Consumption

For a CNN model M with K convolutional layers, the FLOP in $k \in K$ layer can be calculated using:

$$F^k_{\text{FLOP}} = C^k_{\text{in}} \times (\Omega^k)^2 \times C^k_{\text{out}} \times S^k_{\text{out}} \tag{3.4}$$

and total FLOPs in the model M are calculated as:

$$\begin{aligned} F_{\text{FLOPs}} &= \sum_{k=1}^{K} \left[C^k_{\text{in}} \times (\Omega^k)^2 \times C^k_{\text{out}} \times S^k_{\text{out}} \right] \\ F_{\text{FLOPs}} &= \sum_{k=1}^{K} F^k_{\text{FLOP}} \end{aligned} \tag{3.5}$$

Similarly, the memory consumption of a layer $k \in K$ in CNN model M with K convolutional layers can be calculated using:

$$M^k_{\text{Memory}} = C^k_{\text{in}} \times (\Omega^k)^2 \times C^k_{\text{out}} \times 4 \tag{3.6}$$

while total memory consumption is given as:

$$\begin{aligned} M^k_{\text{Memory}} &= \sum_{k=1}^{K} \left[C^k_{\text{in}} \times (\Omega^k)^2 \times C^k_{\text{out}} \times 4 \right] \\ M_{\text{Memory}} &= \sum_{k=1}^{K} M^k_{\text{Memory}} \end{aligned} \tag{3.7}$$

However, the energy requirement of a CNN is not only reflected by memory and FLOPs. In fact, the total energy consumption of a CNN is a sum of energy consumption in executing arithmetic operations. The former is a product of energy required for DRAM access (referred as E_{access} in a 45-nm CMOS, i.e., 640 pJ [87] and model memory, while the latter is a product of energy required for a single FLOP (referred as E_{FLOP}), i.e., 2.3 pJ [87] and model's total FLOPS. Thus, the theoretical energy consumption for a CNN model M and K convolutional layers can calculated using:

$$E_{\text{Theoretical Energy}} = \sum_{k=1}^{K} \left[\left(M_{\text{Memory}} \times E_{\text{access}} \right) + \left(M_{\text{FLOPs}} \times E_{\text{FLOPs}} \right) \right] \tag{3.8}$$

$$E_{\text{Theoretical Energy}} = \sum_{k=1}^{K} E^k_{\text{Theoretical Energy}} \tag{3.9}$$

where C_{in} and C_{out} are the number of input and output channels, respectively, while Ω^2 and S_{out} represent the size of filters and size of feature maps in the output layer, respectively. Moreover, M_{FLOPs}, M_{Memory}, and $E_{\text{Theoretical Energy}}$ denote the model's total FLOPs, memory, and energy. The M^k_{FLOPs}, M^k_{Memory}, and $E^k_{\text{Theoretical Energy}}$ denote the k-th layer's FLOPs. memory, and energy, respectively.

Empirical/Hardware-Based Energy Consumption

In the "Empirical/Hardware-Based Energy Consumption" section, we move away from theoretical models and focus on measuring energy consumption in real-time on edge devices, such as GPUs. Unlike theoretical models, this method uses tools like NVML[1] to directly measure the energy used by the devices. An important part of this process is the warm-up phase. During this phase, the device's GPU runs multiple times to reach a stable, consistent operational state. This ensures that the GPU is performing at a regular power level before we start measuring. This step is crucial because it helps avoid variations in power readings that can happen when a GPU switches from being idle or in a low-power state to being actively used.

[1] `https://developer.nvidia.com/nvidia-management-library-nvml`

Algorithm 1: Empirical Energy Consumption Measurement for CNN Layers on GPUs

```
Input: $K$, number of repetitions $repeats$
Output: $E^k_{\text{Hardware Energy}}$ for each layer $k \in K$
Procedure MeasureEnergyConsumption($K$, $repeats$):
    Initialize NVML or a similar tool for GPU monitoring;
    // Warm-up Phase:
    foreach $k \in K$ do
        Run layer $k$ multiple times to stabilize GPU;
    end foreach
    // Measurement Phase:
    foreach $k \in K$ do
        $\text{Time}_{\text{elapsed}} \leftarrow 0$;
        $E^k_{\text{Hardware Energy}} \leftarrow 0$;
        for $i \leftarrow 1$ to $repeats$ do
            Start timing;
            Record $P_{\text{Start}_i}$;
            Execute layer $k$;
            Record $P_{\text{End}_i}$;
            Stop timing;
            $\text{Time}_{\text{elapsed}_i} \leftarrow$ time elapsed;
            $\text{Time}_{\text{elapsed}} \leftarrow \text{Time}_{\text{elapsed}} + \text{Time}_{\text{elapsed}_i}$;
            $E^k_{\text{Hardware Energy}_i} \leftarrow (\frac{P_{\text{Start}_i}+P_{\text{End}_i}}{2}) \times \text{Time}_{\text{elapsed}_i}$;
            $E^k_{\text{Hardware Energy}} \leftarrow E^k_{\text{Hardware Energy}} + E^k_{\text{Hardware Energy}_i}$;
        end for
        Output $E^k_{\text{Hardware Energy}}$ for layer $k$;
    end foreach
return $E^k_{Hardware\ Energy}$ for each $k \in K$
```

To get accurate and reliable energy consumption data, the GPU layer is executed several times. This repetition helps average out any irregularities or changes in power readings, giving a more stable and representative measure of energy consumption. We also calculate the time it takes for each layer to run using precise timing events. These events mark the beginning and end of the layer's operation and are synchronized with the GPU for accurate timing. The total time taken is then divided by the number of runs to get an average execution time for each layer.

Considering both the warm-up phase and the repeated measurements, we recalculate the energy consumption for each layer $E^k_{\text{Hardware Energy}}$ using the following formula:

$$E^k_{\text{Hardware Energy}} = \frac{1}{\text{repeats}} \sum_{i=1}^{\text{repeats}} \left(\frac{\text{Power}_{\text{start}_i} + \text{Power}_{\text{end}_i}}{2} \right) \times \text{Time}_{\text{elapsed}_i} \tag{3.10}$$

In this formula, $\text{Power}_{\text{start}_i}$ and $\text{Power}_{\text{end}_i}$ are the power readings at the beginning and end of each run, respectively. $\text{Time}_{\text{elapsed}_i}$ is the time it takes for each execution, and repeats is the total number of measurements for averaging. From (3.10), the total energy $E_{\text{Hardware Energy}}$ of a CNN model M with K layers can be calculated as:

$$E_{\text{Hardware Energy}} = \sum_{k=1}^{K} E^k_{\text{Hardware Energy}} \tag{3.11}$$

This hardware-based method provides a more empirical and accurate way to assess the energy needs of each CNN layer, taking into account the unique characteristics of edge devices. By including the warm-up phase, repeated measurements, and precise timing, this approach gives essential insights into the energy efficiency of CNNs in real-world edge computing scenarios. A detailed algorithm for measuring the hardware-based energy of each layer in a CNN model is presented in Algorithm 1.

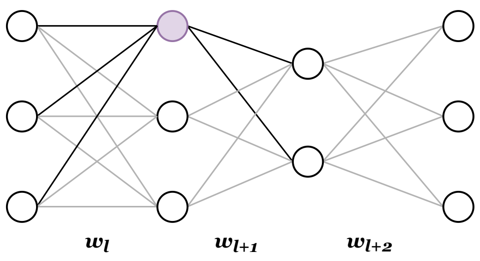
3.3 DepGraph: Group-Level Pruning in Neural Networks

This thesis introduces a method for structurally pruning neural networks, focusing on the constraints of parameter dependency. The considered approach for this purpose is based on DepGraph group-level pruning which is adapted from the work of Fang *et al.* [1].

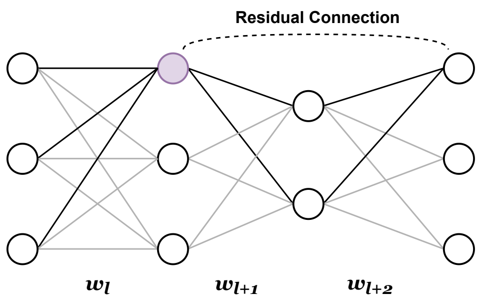
3.3.1 Understanding Dependencies in Neural Networks

The concept of pruning is explored using fully-connected (FC) layers as a foundation. Consider a basic linear neural network with three consecutive layers, as shown in Fig. 3.1a. These layers are defined by 2-D weight matrices w_l, w_{l+1}, and w_{l+2}. Structural pruning streamlines the network by removing neurons, leading to dependencies between parameters, such as $w_l \Leftrightarrow w_{l+1}$. This means that if w_l is pruned, w_{l+1} must also be pruned. Specifically, pruning the k-th neuron connected to w_l implies the removal of the connecting bridge.

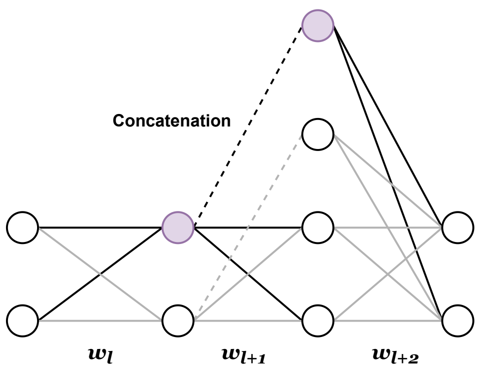
Previous studies have addressed layer dependencies in deep neural networks through customized and model-specific strategies [46, 88]. However, dependencies, as depicted in



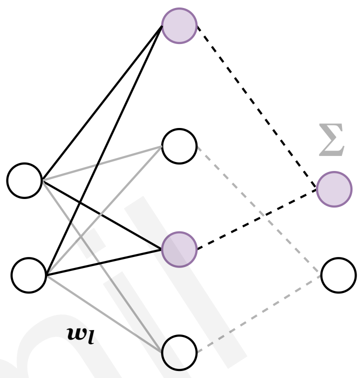
(a) Basic dependency



(b) Residual dependency



(c) Concatenation dependency



(d) Reduction dependency

Figure 3.1: Parameters grouped together with interdependencies, structured in various formats. It's essential to prune all the emphasized parameters at the same time [1].

Fig. 3.1b, 3.1c, and 3.1d, are varied and complex. Manually analyzing these dependencies on a case-by-case basis is impractical, especially when simple dependencies can intertwine to form more intricate patterns. The Dependency Graph offers a comprehensive and automated solution to this challenge, effectively managing the dependency aspect in structural pruning.

3.3.2 Dependency Graph

Grouping for Structural Pruning

To facilitate structural pruning in neural networks, it's crucial to group layers based on their inter-dependencies. The aim is to create a grouping matrix $G \in R^{L\times L}$, where L represents the number of layers in the network. Here, $G_{ij} = 1$ signifies a dependency between the i-th and j-th layers. Lets define $\text{Diag}(G) = \mathbf{1}^{1\times L}$ to include self-dependency for simplicity. Using G, it's easy to identify all layers interconnected with the i-th layer, denoted as $g(i)$:

$$g(i) = \{j|G_{ij} = 1\} \tag{3.12}$$

However, determining these grouping patterns in complex neural networks, which may have thousands of layers, is challenging. The dependencies indicated by G_{ij} depend not only on the i-th and j-th layers but also on intermediate layers. Since these relationships are often non-local and implicit, they cannot be easily defined by simple rules. To address this, DepGraph doesn't directly estimate G but instead employs an alternative method for modelling dependencies, from which G can be efficiently derived.

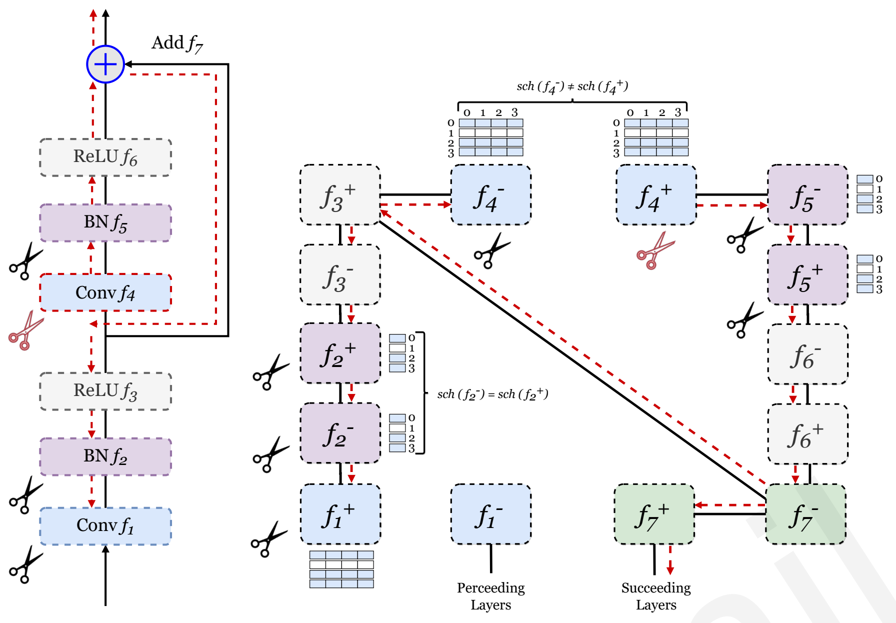
Constructing the Dependency Graph

Consider a group of weights $g = \{w_1, w_2, w_3\}$ with dependencies $w_1 \Leftrightarrow w_2$, $w_2 \Leftrightarrow w_3$, and $w_1 \Leftrightarrow w_3$. Notably, some dependencies, like $w_1 \Leftrightarrow w_3$, can be inferred from others. Starting with w_1 and exploring its connections, such as $w_1 \Leftrightarrow w_2$, leads us to $w_2 \Leftrightarrow w_3$ through a recursive process. This culminates in a transitive relation: $w_1 \Leftrightarrow w_2 \Leftrightarrow w_3$.

In this context, the group's relationships can be represented with just two dependencies. Similarly, the previously mentioned grouping matrix G contains redundancies for dependency modelling. Fang *et al.* [1] propose a Dependency Graph D, which efficiently represents local inter-dependencies between adjacent layers, as an alternative to G. D focuses only on direct connections between adjacent layers, acting as a transitive reduction of G. It contains the same vertices as G but minimizes the number of edges. For any $G_{ij} = 1$, there is a path in D between vertices i and j, allowing for the inference of G_{ij} by examining paths in D.

Decomposition of Network

Building a dependency graph at the layer level can be challenging in practical scenarios. This difficulty arises because some fundamental layers, like fully connected layers, employ different schemes such as $w[k, :]$ and $w[:, k]$. These schemes alter the dimensions of the inputs and outputs differently, as we have discussed in Section 3.3.1. Additionally, networks incorporate non-parameterized operations like skip connections, which influence the dependencies among layers [81]. To address these complexities, Fang *et al.* introduced a novel approach to break down a network, denoted as $\mathcal{F}(x; w)$, into more detailed and simpler elements, represented as $\mathcal{F} = \{f_1, f_2, \cdots, f_L\}$. Here, each component f is either a parameterized layer, such as a convolution layer, or a non-parameterized operation like adding residuals. In contrast to focusing on the relationships at the layer level, DepGraph emphasizes the dependencies between the inputs and outputs of the layers. More specifically, the input and output of a component f_i are defined as f_i^- and f_i^+, respectively. This designation simplifies the process of modelling dependencies and enables the application of varied pruning schemes to the same layer.



(a) CNN *(b) Propagation on Dependency Graph*

Figure 3.2: Grouping of layers is accomplished through recursive propagation on the Dependency Graph, beginning with the output of the fourth convolutional layer, f_4^+. In this scenario, the input f_4^- and output f_4^+ of the convolutional layer do not have an Intra-layer Dependency, owing to the differing pruning strategies demonstrated earlier [1].

Dependency Modeling

Using the notation above introduced, let's redraw the neural network as shown in (3.13). In this model, two main types of dependencies are presented: interlayer and intralayer dependencies. The equation below illustrates this:

$$\underbrace{(f_1^-, f_1^+) \leftrightarrow (f_2^-, f_2^+)}_{\text{Inter-layer Dep}} \cdots \leftrightarrow \underbrace{(f_L^-, f_L^+)}_{\text{Intra-layer Dep}} \tag{3.13}$$

In this representation, the symbol $\leftrightarrow$ shows the connection between two adjacent layers. By examining these dependencies, some basic yet broad rules for modelling dependencies can be established, which are as follows:

- **Inter-layer Dependency:** This type of dependency, $f_i^- \Leftrightarrow f_j^+$, occurs consistently in layers that are connected, specifically where $f_i^- \leftrightarrow f_j^+$.
- **Intra-layer Dependency:** This occurs when the input and output of a layer, f_i^- and f_i^+ respectively, share the same pruning approach, indicated by $sch(f_i^-) = sch(f_i^+)$.

Firstly, inter-layer dependencies can be estimated by looking at the network's structure. For layers that are connected, a dependency always exists as their inputs and outputs correspond to the same parts of the network. Next, consider intra-layer dependencies. This type of dependency requires that both the input and output of a layer are pruned at the same time. Many layers, such as batch normalization, meet this criterion because their inputs and outputs are pruned in the same way, as denoted by $sch(f_i^-) = sch(f_i^+)$.

However, layers like convolutions often have different pruning approaches for their inputs and outputs, leading to no dependency between the two. This distinction is important in our model. The formal approach to dependency modelling is as follows:

$$D(f_i^-, f_j^+) = \underbrace{\mathbb{I}[f_i^- \leftrightarrow f_j^+]}_{\text{Inter-layer Dep}} \vee \underbrace{\mathbb{I}[i = j \wedge sch(f_i^-) = sch(f_j^+)]}_{\text{Intra-layer Dep}} \tag{3.14}$$

In this equation, $\vee$ and $\wedge$ are the logical "OR" and "AND" operations, respectively, and $\mathbb{I}$ is an indicator function that returns "True" if the specified condition is met. The first part of the equation deals with inter-layer dependencies caused by connections in the network, while the second part deals with intra-layer dependencies that arise from layers having the same pruning scheme for their inputs and outputs.

It's important to note that the DepGraph is a symmetric matrix, meaning that $D(f_i^-, f_j^+) = D(f_j^+, f_i^-)$. Therefore, all pairs of inputs and outputs can be analyzed to estimate the network's dependency graph. The DepGraph of a CNN block with residual connections is shown in Fig. 3.2. Algorithms 2 and 3 summarize the procedures for dependency modelling and grouping.

Algorithm 2: Dependency Graph [1]

```
   Input: A neural network $\mathcal{F}(x; w)$
   Output: DepGraph D($\mathcal{F}, E$)
 1 $f^- = \{f_1^-, f_2^-, \ldots, f_L^-\}$ decomposed from the $\mathcal{F}$;
 2 $f^+ = \{f_1^+, f_2^+, \ldots, f_L^+\}$ decomposed from $\mathcal{F}$;
 3 Initialize DepGraph $D = \mathbf{0}_{2L\times 2L}$ ;
 4 for $i = \{0, 1, \ldots, L\}$ do
 5     for $j = \{0, 1, \ldots, L\}$ do
 6         $D(f_i^-, f_j^+) = \underbrace{\mathbb{I}[f_i^- \leftrightarrow f_j^+]}_{\text{Inter-layer Dep}} \vee \underbrace{\mathbb{I}[i = j \wedge sch(f_i^-) = sch(f_j^+)]}_{\text{Intra-layer Dep}}$
 7     end for
 8 end for
 9 return $D$
```

Algorithm 3: Grouping [1]

```
   Input: DepGraph D($\mathcal{F}, E$)
   Output: Groups G
 1 $G = \{\}$;
 2 for $i \in \{1, 2, \ldots, 2 * ||\mathcal{F}||\}$ do
 3     $g = \{i\}$;
 4     repeat
 5         $UNSEEN = \{1, 2, \ldots, 2 * ||\mathcal{F}||\} - g$;
 6         $g' = \{j \in UNSEEN \mid \exists k \in g, D_{kj} = 1\}$;
 7         $g = g \cup g'$ ;
 8     until $g' = \emptyset$;
 9     $G = G \cup \{g\}$;
10 end for
11 return $G$
```

3.4 DepGraphRL Framework: Proposed Scheme

The system model of the DepGraphRL framework, which integrates an A2C-agent, Edge-IoT devices, and a dependency graph-based neural network pruning, is depicted in Fig. 3.3. This model considers IoT devices equipped with edge computing capabilities. The

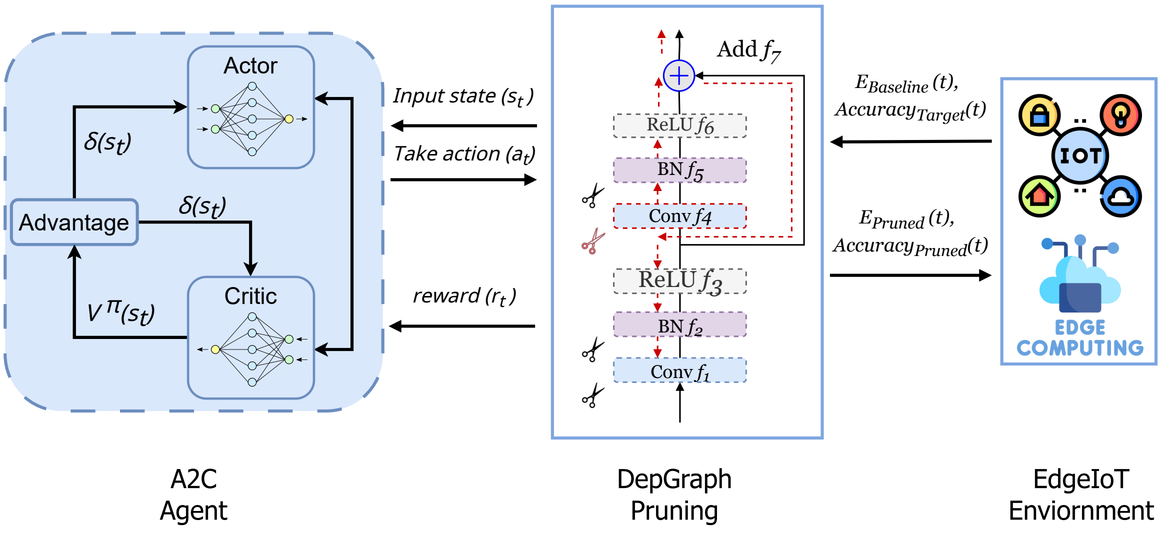


Figure 3.3: Proposed model for energy-aware pruning of neural networks using a dependency graph, enhanced by deep reinforcement learning within the DepGraph framework, specifically designed for Edge-IoT devices.

DepGraphRL framework features an energy-aware dependency graph-based neural network pruning scheme, driven by a DRL-based A2C agent. It assesses the status of Edge-IoT devices and subsequently provides a compressed CNN model for execution. Unlike traditional compression schemes, the DepGraphRL framework prunes a pre-trained CNN in an energy-aware fashion, pruning each CNN layer according to its contribution to the overall energy complexity of the CNN.

Prior to deployment on the edge device, the DepGraphRL model requires pre-training on a GPU or cloud server. Following this, any CNN model can be compressed using DepGraphRL on the device itself. This approach contrasts with cloud offloading, which, as reported in several studies, consumes significantly more energy than local inference on energy-harvesting edge devices [87, 89]. Hence, the proposed scheme avoids computational offloading to the cloud, reducing the associated energy expenditure. The subsequent section presents a comparison of CNN energy consumption using theoretical and hardware-based approaches, followed by a detailed discussion of the DepGraphRL-based scheme. The thesis will demonstrate the efficacy of the DepGraphRL scheme, defining its state, action, and reward functions tailored for this specific task.

3.4.1 Definition of State, Action, and Reward

This part formulates the problem in (3.1) as an MDP and uses the DepGraphRL as an agent while treating the Edge-IoT device as an environment. The related state space, action space and reward function are introduced below.

1. **State Space (S):** At each moment, denoted as time t, the proposed A2C-based DepGraphRL agent receives a set of information, known as the state space s_t, from its environment. This state space comprises detailed parameters about the environment at that specific time. The state space is represented as follows:

$$s_t = \Bigg\{ \Big[I_k, C_{\text{in}}^k, C_{\text{out}}^k, E_{\text{Hardware Energy}}^k, R_k \Big], \Big[I_{k+1}, C_{\text{in}}^{k+1}, C_{\text{out}}^{k+1}, E_{\text{Hardware Energy}}^{k+1}, R_{k+1} \Big], \\ \dots, \Big[I_K, C_{\text{in}}^K, C_{\text{out}}^K, E_{\text{Hardware Energy}}^K, R_K \Big] \Bigg\}^T \quad (3.15)$$

   In this formulation, I_k, C_{in}^k, and C_{out}^k indicate the index, input channels, and output channels of the k^{th} layer in a CNN model with K layers, respectively. $E_{\text{Hardware Energy}}^k$ represents the current energy status of the hardware for layer k, and R_k shows the proportion of energy used by this layer in the overall CNN structure. These details about individual layers help the agent distinguish one convolutional layer from another, enhancing its decision-making. Specifically, $E_{\text{Hardware Energy}}^k$ and R_k are crucial for the agent to identify layers based on their energy contribution to the overall complexity of the model.

2. **Action (A):** Based on the given state space, the agent takes an action according to the previously defined policy. In each iteration, the agent must decide whether to prune a specific layer k or not. This decision is represented as a_t^k, which can be either 0 (do not prune) or 1 (prune). Given that a network might have K layers, there are 2^K possible pruning strategies, meaning the A2C actor neural network could have as many as 2^K output nodes. This causes the complexity of the A2C actor neural network to increase exponentially with the number of layers in the CNN.

   To manage this complexity, the thesis proposes modeling the actions in a scalar continuous action space, denoted as $\tilde{a}_t^k$, which ranges between 0 and 1. Subsequently, the decision to prune a specific layer is quantized into a binary choice, either 0 or 1. This approach simplifies the decision-making process and makes it more manageable, even as the number of layers in the CNN increases.

3. **Reward (R):** The thesis introduces a reward function to assess the effectiveness of the A2C compression policy. This function is inspired by the work referenced in [48]. The reward function from [48] is designed to encourage not only the maintenance

of accuracy but also the reduction of $E_{\text{Hardware Energy}}$ or the model size. On the other hand, an alternative reward function, defined as “reward = -Error”, tends to compress the model aggressively without focusing on reducing complexity. However, empirical observations suggest that accuracy is directly proportional to the hardware energy consumption ($E_{\text{Hardware Energy}}$). Therefore, in the proposed model, the reward function is crafted to incentivize both accuracy and energy efficiency. It is given by the following equation:

$$r_t = -\text{Error} \cdot \log(E_{\text{Hardware Energy}}) \tag{3.16}$$

Here, the error is different between target accuracy and obtained accuracy after pruning. This reward function takes into account not just the negative error (i.e., $-$Error), but also the hardware energy consumption ($E_{\text{Hardware Energy}}$). As a result, the model compression is executed with careful consideration of both dynamic accuracy and the energy budget necessary for specific IoT applications. This approach ensures a balanced optimization between model performance and resource efficiency.

3.4.2 Advantage Actor-Critic (A2C) Algorithm

The A2C algorithm is a reinforcement learning technique that falls under the policy-based methods category within the domain of DRL. A2C combines the strengths of both policy-based and value-based methods to efficiently learn and optimize policies for decision-making tasks. In this subsection, we will delve into the working of the A2C algorithm, highlighting its key components and how it balances policy and value aspects for reinforcement learning.

Key Components of A2C

1. **Dual Neural Networks**:

    - **Actor (Policy Network)**: This neural network is responsible for selecting actions based on the current policy. The policy network approximates the policy $\pi_\theta(a|s)$, where θ represents the network parameters. It outputs a probability distribution over possible actions given the current state.
    - **Critic (Value Network)**: The value network evaluates the quality of the actions taken by the Actor within specific states. It approximates the state-value function $V^\pi(s)$, which estimates the expected return from a particular state. Essentially,

the Critic network approximates a function $Q^{\pi}(s, a)$, quantifying the value attributed to actions selected by the Actor within particular states.

2. **Gradient Computation**: The primary objective of A2C is to maximize the expected rewards by optimizing the policy. This is achieved by computing the gradient of the objective function $\mathcal{J}(\theta)$ with respect to the policy parameters θ. The gradient is computed as follows:
$$\nabla_{\theta}\mathcal{J}(\theta) = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) Q^{\pi}(s_t, a_t)] \tag{3.17}$$

3. **Advantage Estimation**: One crucial aspect of A2C is the introduction of the "advantage," denoted as $\mathcal{A}^{\pi}(s_t, a_t)$. The advantage quantifies the relative advantage of taking action a_t in state s_t compared to the average expected return. It can be estimated through the Temporal Difference (TD) error as follows:
$$\mathcal{A}^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t) \approx r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) = \delta(s_t) \tag{3.18}$$

4. **Policy Gradient Update**: The policy gradient is updated by incorporating the advantage (TD error) as follows:
$$\nabla_{\theta}\mathcal{J}(\theta) = \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \cdot \delta^{\pi}(s_t)] \tag{3.19}$$
This updated gradient is then used to adjust the parameters of the policy network to maximize expected rewards:
$$\theta_{a_{t+1}} \leftarrow \theta_{a_t} + \alpha_{A2C}\nabla_{\theta}\mathcal{J}(\theta) \tag{3.20}$$

5. **Value Network Update**: The value network is updated by minimizing the loss function, which is the square of the TD error:
$$\mathcal{L} = \delta^{\pi}(s_t)^2 = [r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)]^2 \tag{3.21}$$

Synchronous Operation

A unique feature of A2C is its synchronous operation, where multiple agents learn simultaneously. This synchronous training ensures uniformity among agents, making A2C suitable for scenarios that require decentralized operations.

Advantages of A2C

- **Balanced Approach**: A2C effectively balances both policy-based and value-based reinforcement learning aspects, leveraging the strengths of each approach.
- **Advantage Function**: The use of the advantage function helps in reducing the high variance problem often encountered in pure policy-based methods like REINFORCE.
- **Efficient Value Approximation**: A2C efficiently approximates the state-value function, resulting in a more streamlined learning process.
- **Synchronous Training**: Synchronous training ensures uniformity among agents, making it suitable for decentralized scenarios.

The A2C algorithm is particularly well-suited for complex environments with extensive state and action spaces, making it a valuable tool in the realm of Deep Reinforcement Learning.

Chapter 4

Experiments And Performance Evaluation

This chapter explores the experimental setup and examines the impact of the DepGraphRL framework for energy aware pruning of neural networks such as ResNet-18, VGG-19 and MobileNet. It first presents a theoretical and empirical energy comparison after pruning baseline CNN using the baseline DepGraph pruning scheme and the proposed DepGraphRL framework. Additionally, it delves into the influence of different filter ranking techniques for convolutional layers considering different pruning ratios. It also evaluates the impact of a number of filters of each layer of energy consumption in case of pruning actions being taken by DepGraph and DepGraphRL framework. Further, the chapter presents the performance of the proposed DepGraphRL model and DepGraph scheme, specifically looking at accuracy and energy trade-offs across various iterations and pruning ratios using three pruning mechanisms: l_1-norm, Taylor expansion, and random pruning. It concludes by comparing the proposed model with leading pruning techniques across diverse datasets and models.

4.1 Experimental Parameters And Setup

In this thesis, an A2C based agent is trained with a learning rate α_{A2C} of 2×10^{-4} over 5000 episodes with 20 steps in each episode. Both actor and critic neural networks in the A2C agent comprised of a single fully connected hidden layer using sigmoid activation. The input layer's dimension corresponds to the number of input state dimensions, while the output layer's dimension matches the number of layers in the observed CNN. All experiments and results are conducted using PyTorch version 1.7.0. with Python 3.8. The models were pretrained on a GPU-integrated machine having the following specifications: 3.7 GHz Intel Core i7-8700k CPUs, 32 GB of memory, and NVIDIA GeForce RTX 2080Ti, running on Windows 10 operating system.

Subsequent subsections provide details of benchmark CNN models and a dataset used to assess the DepGraphRL framework. It is important to highlight here that the proposed DepGraphRL approach can be extended to other CNN models and data sets.

4.1.1 Deep Neural Network (DNN) Models

The experimental evaluations include widely recognized CNNs like ResNet-18, MobileNet, and VGG-19, pivotal in benchmarking model compression methods. For the experimentation purposes in this thesis work, ResNet-18, MobileNet, and VGG-19 have been modified for CIFAR-10 dataset as these models are originally trained on ImageNet dataset. The details regarding architecture of the considered DNN models are briefly overviewed as follows:

- **ResNet-18: [90]** ResNet-18 is a streamlined variant of the Residual Network (ResNet) architecture, featuring 18 layers designed for deep learning in computer vision. Central to its design is 8 residual blocks, each comprising two layers with skip connections that enable inputs to bypass some layers, effectively mitigating the vanishing gradient problem in deep networks. The architecture begins with a 7x7 convolutional layer followed by max pooling, progresses through these residual blocks, and concludes with global average pooling and a fully connected layer for classification.

- **MobileNet: [91]** MobileNet, a deep learning architecture optimized for mobile devices, comprises a unique configuration of layers emphasizing efficiency. It begins with a single standard convolutional layer, followed by 13 depthwise separable convolutional blocks, each consisting of a depthwise convolution for spatial filtering and a pointwise convolution for feature combination, totalling 26 layers in these blocks. Each convolution is accompanied by batch normalization and ReLU activation, with some blocks incorporating a stride of 2 for downsampling. The architecture culminates with global average pooling and a final fully connected layer for classification. This setup results in a total of 28 significant layers, including the initial and final layers.

- **VGG-19: [92]** VGG-19, a prominent model in the VGG series, is characterized by its depth and simplicity, comprising 19 learnable layers that include 16 convolutional layers and 3 fully connected layers. The architecture predominantly uses small 3x3 convolutional filters across all layers, a design choice enabling the capture of fine details within images. Each convolutional layer is accompanied by a ReLU (Rectified Linear Unit) activation function, introducing non-linearity. The model also incorporates 5 max-pooling layers, strategically placed to perform spatial downsampling, thus reducing the dimensionality of the feature maps. The convolutional and pooling layers are followed by three dense layers, with the first two having 4096 units each, and the third designed for classification with 10 units, aligning with the number of classes in datasets like Cifar-10. The network culminates with a softmax layer for output classification. VGG-19's architecture is known for its uniformity, using only 3x3 filters, which simplifies scaling and adaptation for various vision tasks, though it is computationally intensive. This configuration makes VGG-19 highly effective for image recognition and as a feature extractor in diverse applications.

4.1.2 Datasets:

CIFAR-10 is a widely used dataset in the field of machine learning and computer vision. It's an acronym for the Canadian Institute For Advanced Research, and the "10" in CIFAR-10 refers to the fact that the dataset contains 10 classes of images. Key characteristics of CIFAR-10 include:

- **Dataset Size and Composition**: CIFAR-10 consists of 60,000 32x32 colour images, divided into 10 classes, with each class containing 6,000 images. The dataset is typically split into 50,000 training images and 10,000 test images.
- **Classes**: The 10 different classes represent everyday objects. These classes are aeroplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The variety in classes helps in training models to recognize a wide range of objects.
- **Image Resolution**: The images in CIFAR-10 are 32x32 pixels, which is relatively small. This low resolution makes the dataset manageable for training neural networks, especially when computational resources are limited.
- **Usage**: CIFAR-10 is often used in academic settings and for benchmarking the performance of new machine learning and computer vision algorithms, especially for tasks like image classification, object recognition, and similar applications.
- **Challenge**: Despite its small image size, CIFAR-10 is considered challenging due to the low resolution of the images and the high level of similarity between some classes (like automobiles and trucks, or cats and dogs).

CIFAR-10's popularity stems from its suitability as a benchmark dataset. It provides a balance between being manageable for training (due to the small image size and the moderate size of the dataset) and challenging enough to test the effectiveness of various image recognition algorithms.

4.2 Performance Evaluation

The purpose of the proposed DepGraphRL framework is to provide an energy-aware pruned model of the baseline CNN model based on the dynamic energy requirements of Edge-IoT devices. To validate the effectiveness and efficacy of the proposed framework, evaluation is performed on three different CNN models that are ResNet-18, MobileNet, and VGG-19 on the CIFAR-10 dataset. The baseline CNN models are pruned iteratively for 10 iterations via two experimental setups including a baseline DepGraph pruning framework for random selection of group pruning decisions and the proposed DepGraphRL pruning framework driven with an A2C-based agent for sub-optimal selection of group pruning decisions based on theoretical and hardware layer-wise energy consumption.

During the experimentation evaluations, three distinct pruning ratios are considered i.e., $Pr = \{0.1, 0.3, 0.5\}$. In contrast to traditional pruning techniques where layers are pruned based on the proportion of filters, the proposed DepGraphRL framework provides a generalizable pruning technique for CNN models based on the energy consumption of each layer. Furthermore, the proposed framework removes the obstacle of structural coupling by explicitly modelling the dependency between layers and comprehensively grouping coupled parameters for pruning. Thus, DepGraphRL provides an energy-aware fully automatic and a general pruning approach to energy-constraint Edge-IoT devices for incorporating any CNN model. Further, to show the efficacy of the proposed framework in a real-world scenario, the evaluation of DepGraphRL demonstrates hardware-based energy consumption of a CNN model alongside theoretical energy consumption. This insight will support the applicability of the proposed DepGraphRL framework in practical Edge-IoT devices based on their empirical energy consumption evaluation as compared to theoretical energy consumption evaluation methods.

The evaluation of the DepGraphRL framework also includes different filter importance ranking techniques which demonstrate that the proposed framework can work with a wide variety of filter ranking approaches including l_1-norm, Random, and Taylor Expansion. The details of these filter ranking approaches are briefly explained below:

- l_1**-norm** [46]: In this technique, the significance of a specific filter F within each layer is determined by calculating the sum of the absolute values of its weights, $\sum |F|$, which is the l_1-norm ($||F||_1$). Subsequently, the filters are arranged based on their l_1-norm values. Then, a certain proportion of filters that have the lowest l_1-norm values are eliminated.

- **Taylor Expansion [45]:** The Taylor expansion method for pruning in CNNs frames pruning as an optimization problem. The objective is to find a weight set W_0 with a bounded number of non-zero elements that minimizes the change in the cost function

$$|\Delta C(h_i)| = |C(D \mid W_0) - C(D \mid W)| \tag{4.1}$$

This change approximates the impact on the loss function from removing a specific parameter. Given a parameter i and its output h_i, the cost function is assumed to depend equally on the parameters and the outputs computed from these parameters:

$$C(D \mid h_i) = C(D \mid (w, b)_i) \tag{4.2}$$

Assuming independence of parameters, the change in cost is given as:

$$|\Delta C(h_i)| = C(D, h_i = 0) - C(D, h_i) \tag{4.3}$$

where $C(D, h_i = 0)$ is the cost if output h_i is pruned, while $C(D, h_i)$ is the cost if it is not pruned.

To approximate $\Delta C(h_i)$, the first-degree Taylor polynomial is employed. For a function $f(x)$, the Taylor expansion at point $x = a$ is

$$f(x) = \sum_{p=0}^{P} \frac{f^{(p)}(a)}{p!}(x-a)^p + R_p(x) \tag{4.4}$$

where $f^{(p)}(a)$ is the p-th derivative of f evaluated at point a, and $R_p(x)$ is the p-th order remainder. Approximating $C(D, h_i = 0)$ with a first-order Taylor polynomial near $h_i = 0$, the equation is :

$$C(D, h_i = 0) = C(D, h_i) - \frac{\delta C}{\delta h_i} h_i + R_1(h_i = 0) \tag{4.5}$$

The first-order remainder $R_1(h_i = 0)$ can be neglected due to its significant computation and the influence of ReLU activation, which encourages a smaller second-order term.

By substituting into the cost change equation and ignoring the remainder, the Taylor Expansion criterion Θ_{TE} for pruning is defined. This criterion prunes parameters with almost flat gradients of the cost function with respect to the feature map h_i. It necessitates accumulating the product of the activation and the gradient of the cost function with respect to the activation. For a multi-variate output such as a feature map averaged over a minibatch is given as:

$$\Theta_{TE}(z_l^{(k)}) = \frac{1}{M} \sum_m \frac{\delta C}{\delta z_l^{(k),m}} z_l^{(k),m} \tag{4.6}$$

. This method, based on the Taylor expansion, offers a nuanced and computationally informed approach to reducing the complexity of CNNs.

- **Random Pruning** [93]: This technique employs a random approach to prune a specified percentage of filters from any layer. It bypasses the need for pre-calculating filter importance, providing a simple and unbiased method for network pruning.

4.2.1 Comparison of Theoretical Energy with Hardware-Based Energy

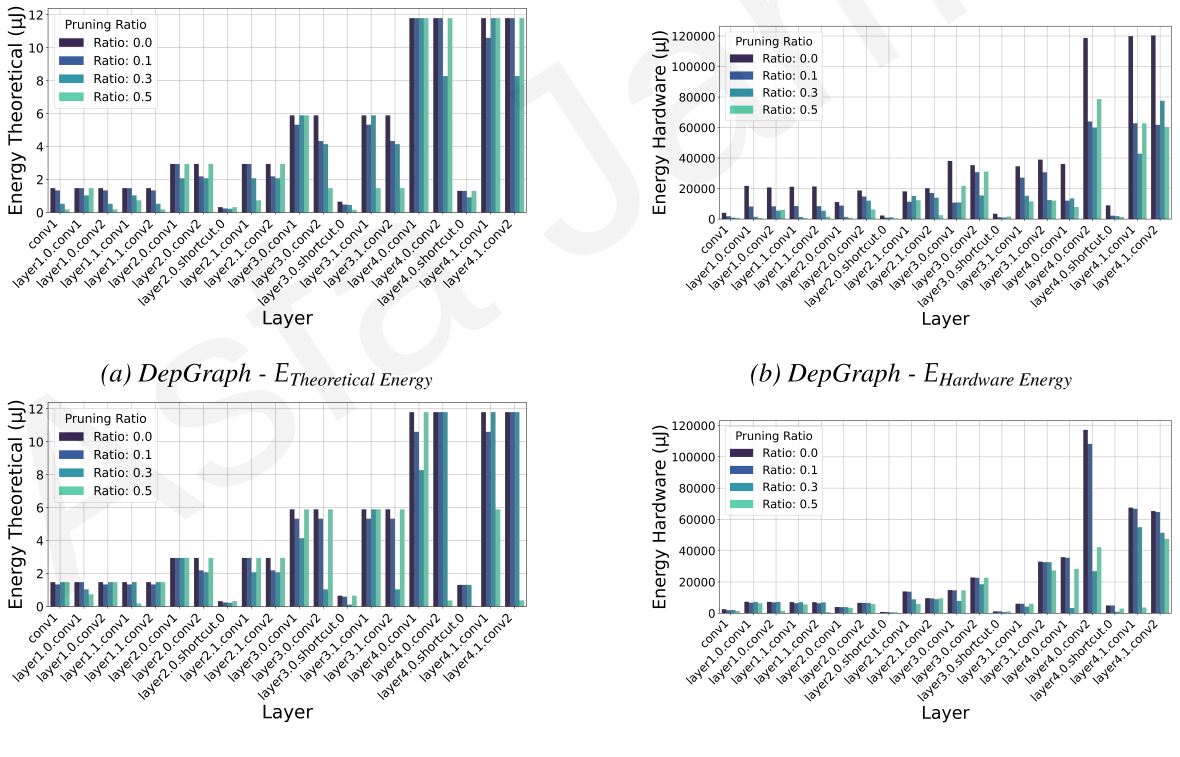
In this experimental evaluation, a comparative analysis between the evaluation of the theoretical energy ($E_{\text{Theoretical Energy}}$) and hardware energy ($E_{\text{Hardware Energy}}$) for each layer of "ResNet-18" CNN model is performed for both the DepGraph (baseline pruning framework) and the proposed DepGraphRL framework. Pruning of baseline CNN models is performed up to ten iterations with pruning ratios $Pr = \{0.1, 0.3, 0.5\}$. Further, all the above-defined filter ranking techniques, (i.e., l_1-norm, Random, and Taylor Expansion) are also considered. The results are illustrated via bar plot in Fig. 4.1, 4.2, 4.3.

In these results, it can be observed that the baseline pruning framework randomly selects layers to prune irrespective of energy consumption. Though it leads to a reduction in the overall energy consumption of the CNN model, however, it does not guarantee an optimal trade-off with target accuracy as pruning decisions are randomly assigned. In the case of the proposed framework, as shown in Fig. 4.1, 4.2, 4.3, as the DepGraphRL considers energy consumption as well as its impact of layer on accuracy of a CNN model, it can be observed that few layers with lower energy consumption are pruned instead of layers having higher energy consumption. This, subsequently also has an impact on target accuracy (a scenario discussed in the next subsection). Furthermore, it is evident from experiments that $E_{\text{Theoretical Energy}}$ is considerably lower than $E_{\text{Hardware Energy}}$. Also, the reduction in $E_{\text{Theoretical Energy}}$ of each layer not necessarily mean the same amount of energy reduction for $E_{\text{Hardware Energy}}$.

From both evaluation matrices (i.e., $E_{\text{Theoretical Energy}}$ and $E_{\text{Hardware Energy}}$), it can also be seen that the greater the pruning ratio, the more there is the reduction in energy consumption. Also, based on evaluation results, it can be seen that the DepGraphRL framework takes different pruning actions (selection of layers) for pruning considering different pruning ratios.

When comparing different filter ranking techniques, it is evident that DepGraphRL undertakes decisions which take into account different filter pruning techniques, showcasing its ability to dynamically adjust its energy-aware pruning approach due to its DRL-based A2C agent.

The overall impact on energy consumption and accuracy based on pruning decisions are further mentioned in the next sections.



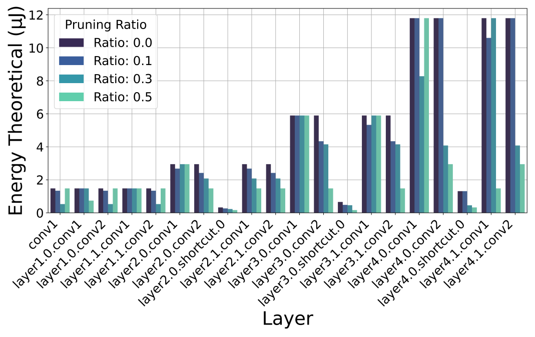
(a) DepGraph - $E_{Theoretical\ Energy}$

(b) DepGraph - $E_{Hardware\ Energy}$

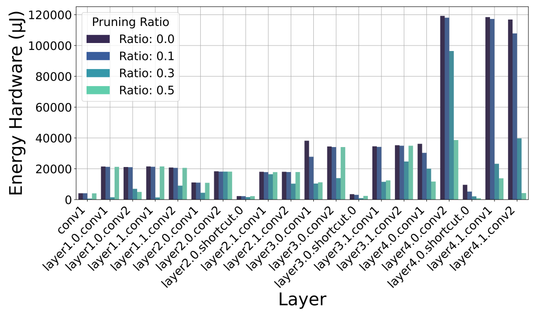
(c) DepGraphRL - $E_{Theoretical\ Energy}$

(d) DepGraphRL - $E_{Hardware\ Energy}$

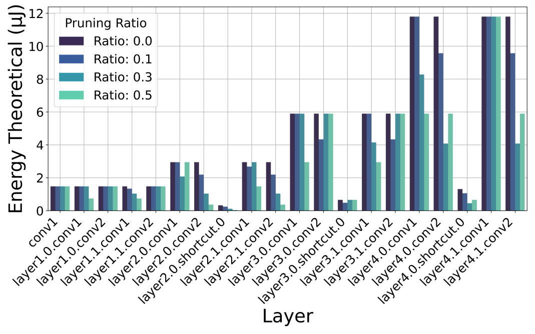
Figure 4.1: A comparative analysis between the evaluation of the theoretical energy ($E_{Theoretical\ Energy}$) and hardware energy ($E_{Hardware\ Energy}$) for the "ResNet-18" CNN model, employing an $l_1 - norm$ filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework.



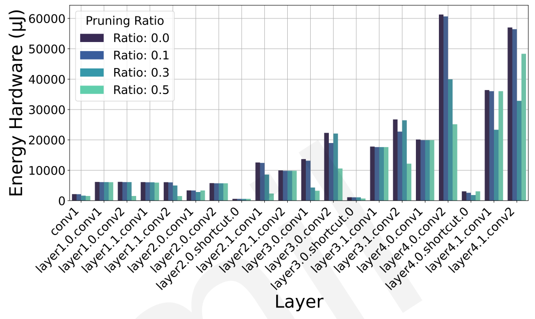
(a) DepGraph - $E_{Theoretical\ Energy}$



(b) DepGraph - $E_{Hardware\ Energy}$



(c) DepGraphRL - $E_{Theoretical\ Energy}$



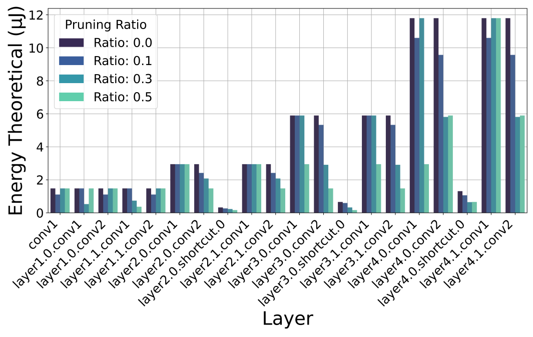
(d) DepGraphRL - $E_{Hardware\ Energy}$

Figure 4.2: A comparative analysis between the evaluation of the theoretical energy ($E_{Theoretical\ Energy}$) and hardware energy ($E_{Hardware\ Energy}$) for the "ResNet-18" CNN model, employing "Random Pruning" filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework.

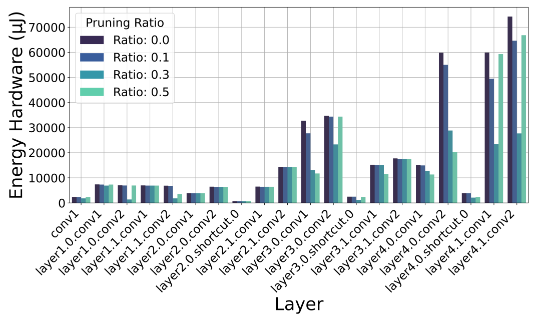
4.2.2 Impact of Number of Filters on Pruning Decisions

Fig. 4.4, it is evident for all pruning ratios and filter ranking techniques, the decisions taken by the DepGraphRL framework to prune a layer based on energy doesn't necessarily means that the layer has more number of filters. Thus, it can be inferred from the results, that the larger number of filters doesn't reflect that layer contributes more towards the overall energy consumption of the model and has a linear impact on the accuracy of the model.

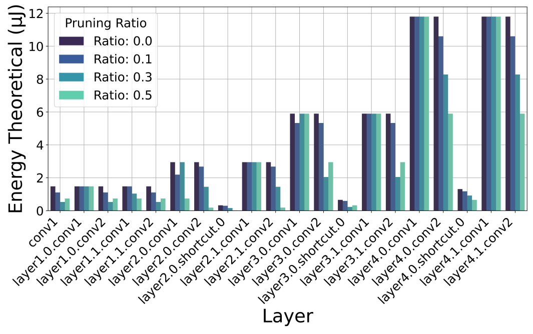
From the presented results in this scenario, it can be seen that the number of filters pruned for each pruning ratio can be different for the same layer owing to its overall contribution to the energy and accuracy of the model. For different filter ranking techniques, the pruned filters are different which demonstrates that filter ranking techniques also contribute to energy as well as the accuracy of the pruned model.



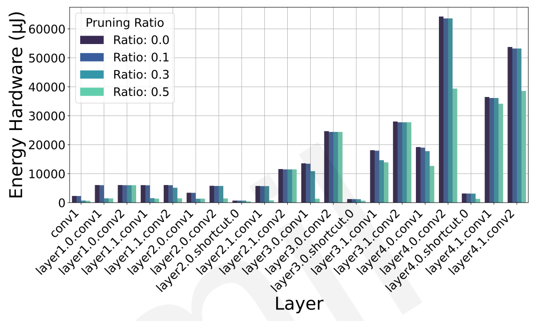
(a) DepGraph - $E_{Theoretical\ Energy}$



(b) DepGraph - $E_{Hardware\ Energy}$



(c) DepGraphRL - $E_{Theoretical\ Energy}$



(d) DepGraphRL - $E_{Hardware\ Energy}$

Figure 4.3: A comparative analysis between the evaluation of the theoretical energy ($E_{Theoretical\ Energy}$) and hardware energy ($E_{Hardware\ Energy}$) for the "ResNet-18" CNN model, employing "Taylor Expansion" filter ranking, and taking into account both the DepGraph and the proposed DepGraphRL framework.

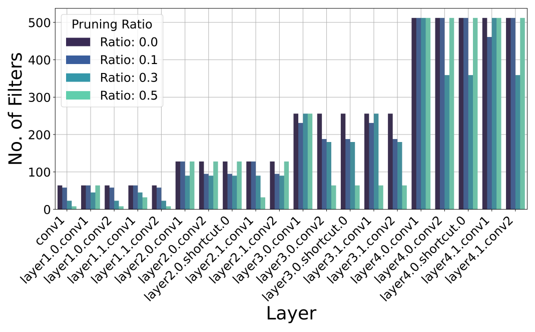
4.2.3 Energy-Accuracy Comparison

In these experiments, the CNN models "ResNet-18" and "MobileNet" were pruned for 10 iterations with pruning ratios of 0.1, 0.3, and 0.5, considering all filter ranking techniques mentioned earlier.

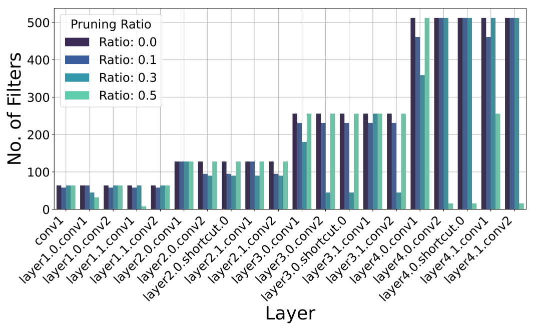
ResNet-18 CNN Model [90]:

The results, depicted in Figs. 4.5, 4.6, and 4.7, show the accuracy and energy consumption (i.e., $E_{\text{Theoretical Energy}}$ and $E_{\text{Hardware Energy}}$) for each iteration in ResNet-18 CNN model. The accuracy of each pruned model was calculated after re-training for 1 epoch. The baseline accuracy of the ResNet-18 CNN model, after training for 200 epochs, is 84.35%, with $E_{\text{Theoretical Energy}}$ being 0.033J and $E_{\text{Hardware Energy}}$ being 0.70J.

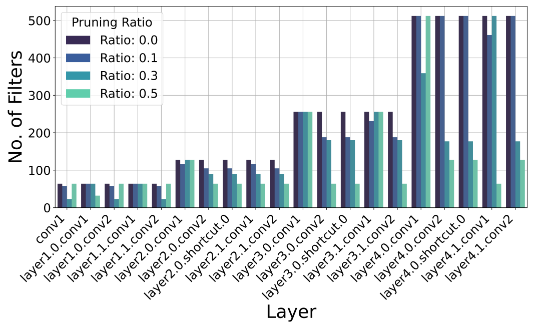
First, let's discuss the results for pruning ratio 0.1 that are shown in Fig. 4.5. Here, after 10 iterations using l_1-norm filter ranking technique with DepGraphRL framework, the pruned model achieved an accuracy of 92.46% with a drop in $E_{\text{Theoretical Energy}}$ of 11.7% from



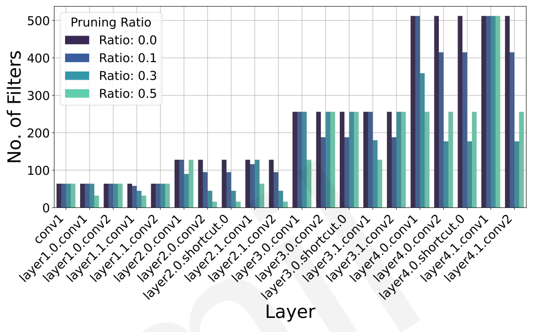
(a) DepGraph - $l_1 - norm$



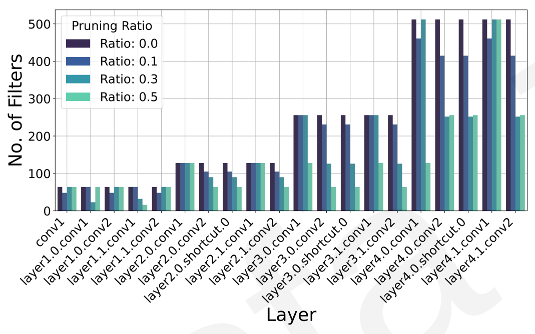
(b) DepGraphRL - $l_1 - norm$



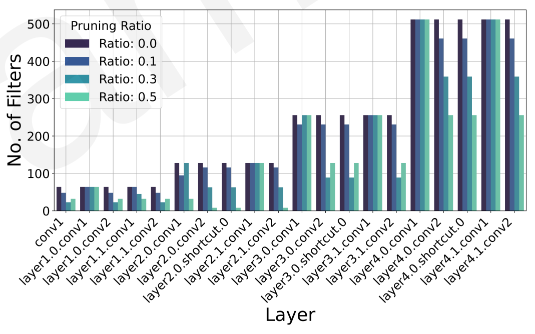
(c) DepGraph - Random Pruning



(d) DepGraphRL - Random Pruning



(e) DepGraph - Taylor Expansion

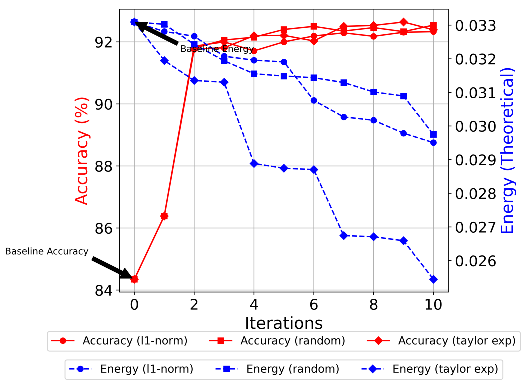


(f) DepGraphRL - Taylor Expansion

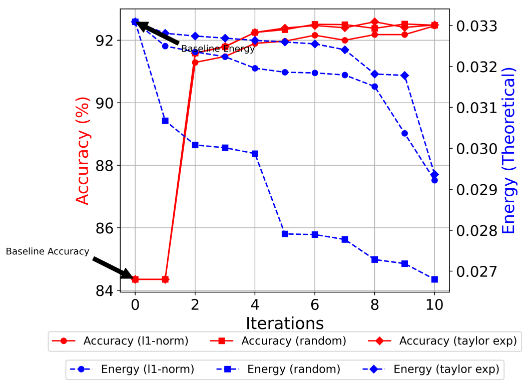
Figure 4.4: A comparative analysis in the context of a number of filters between $l_1 - norm$, Random Pruning, and Taylor expansion for the pruning of "ResNet-18" CNN model considering both the DepGraph and the proposed DepGraphRL framework.

the baseline theoretical energy and a drop in $E_{\text{Hardware Energy}}$ of 55.71% from the baseline hardware-based energy. In contrast, using the baseline pruning framework (DepGraph) with the same pruning ratio and filter ranking technique, the pruned model reached an accuracy of 92.33%, with a reduction in $E_{\text{Theoretical Energy}}$ of approximately 10.84% from the baseline and a reduction in $E_{\text{Hardware Energy}}$ of 54.06% from the baseline.

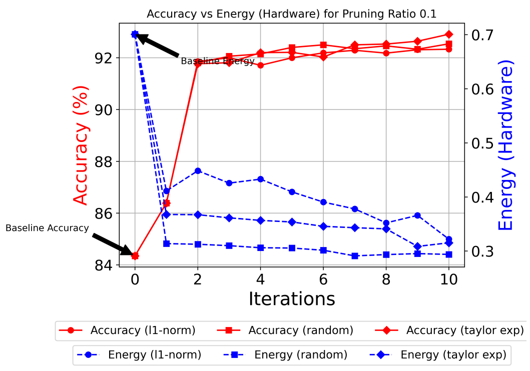
Considering random filter ranking after 10 iterations with a pruning ratio of 0.1 using the DepGraphRL framework, the pruned model attained an accuracy of 92.48%. This corresponds to a decrease in $E_{\text{Theoretical Energy}}$ of 18.99% from the baseline theoretical energy and a decrease in $E_{\text{Hardware Energy}}$ of 58.67% from the baseline hardware-based energy. Conversely,



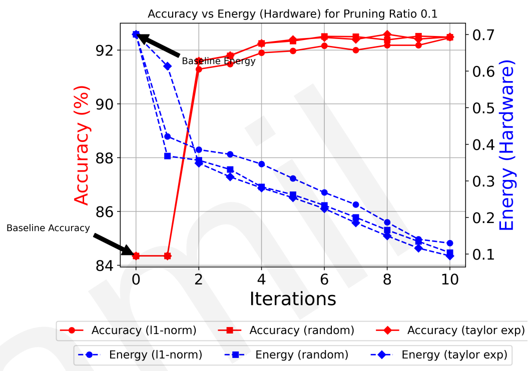
(a) $E_{Theoretical\ Energy}$ in DepGraph with $P_r = 0.1$



(b) $E_{Theoretical\ Energy}$ in DepGraphRL with $P_r = 0.1$



(c) $E_{Hardware\ Energy}$ in DepGraph with $P_r = 0.1$



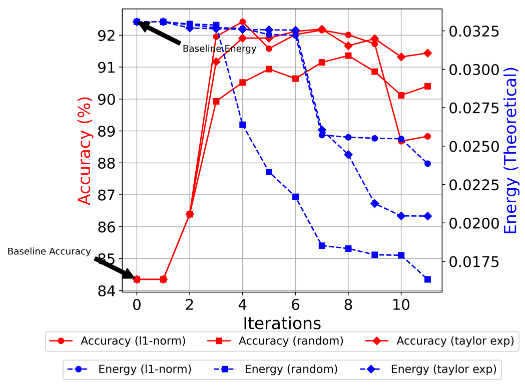
(d) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.1$

Figure 4.5: Tradeoff between energy (J) and accuracy (%) over ten pruning cycles at a pruning ratio of 0.1 for the ResNet-18 convolutional neural network model applied to the CIFAR-10 dataset.

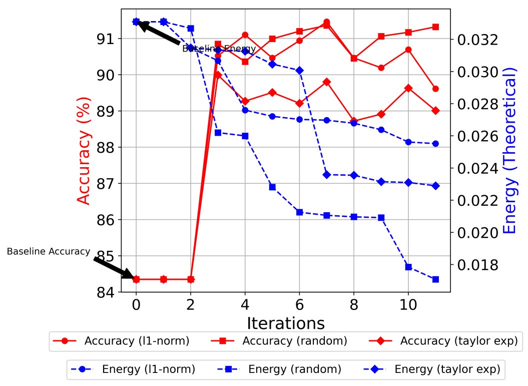
using the baseline pruning framework (DepGraph) under the same conditions, the pruned model achieved an accuracy of 92.54%, with a reduction in $E_{\text{Theoretical Energy}}$ of 10.095% from the baseline and a reduction in $E_{\text{Hardware Energy}}$ of 58.19% from the baseline.

For the Taylor Expansion ranking method after 10 iterations with a pruning ratio of 0.1 using the DepGraphRL framework, the pruned model reached an accuracy of 92.49%. This reflects a decrease in $E_{\text{Theoretical Energy}}$ of 11.21% from the baseline theoretical energy and a decrease in $E_{\text{Hardware Energy}}$ of 56.14% from the baseline hardware-based energy. In contrast, with the same ranking method and using the baseline pruning framework (DepGraph), the pruned model achieved an accuracy of 92.48%, with a decrease in $E_{\text{Theoretical Energy}}$ of approximately 20.42% from the baseline and a decrease in $E_{\text{Hardware Energy}}$ of 55.43% from the baseline.

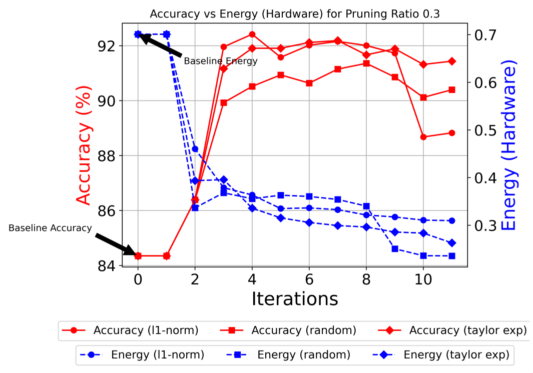
Overall in the above results with a pruning ratio of 0.1, the performance of the baseline pruning scheme (DepGraph) is relatively comparable with the proposed framework, where DepgraphRL shows slight better performance interms on accuracy as well energy. Taylor explansion performs the best in terms of accuracy-energy trade-off as a filter ranking technique



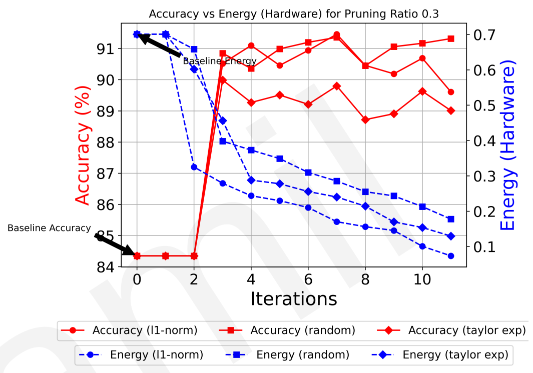
(a) $E_{Theoretical\ Energy}$ in DepGraph with $P_r = 0.3$



(b) $E_{Theoretical\ Energy}$ in DepGraphRL with $P_r = 0.3$



(c) $E_{Hardware\ Energy}$ in DepGraph with $P_r = 0.3$



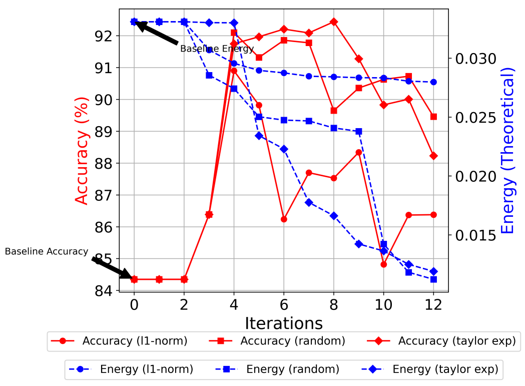
(d) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.3$

Figure 4.6: Tradeoff between energy (J) and accuracy (%) across ten iterations of pruning at a 0.3 ratio for the ResNet-18 CNN model applied to the CIFAR-10 dataset.

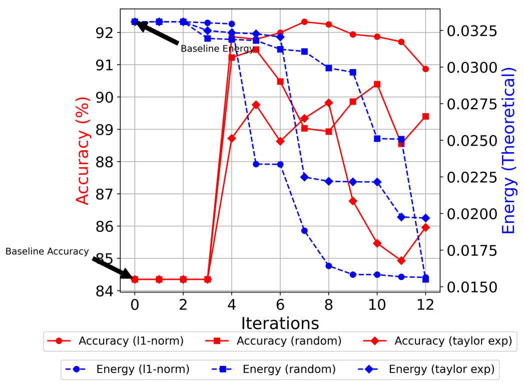
for this pruning ratio. The reason for this behaviour is the fact that the pruning ratio of 0.1 is very small. Therefore, the thesis also presents the result with a larger pruning ratio which is discussed ahead.

In the case of a pruning ratio of 0.3, it is evident from the results that the combination of random filtering technique with DepGraphRL achieves the highest accuracy and least $E_{\text{Theoretical Energy}}$ energy consumption when compared with the baseline pruning scheme (DepGraph) and all other filter pruning techniques. For the case of the $E_{\text{Hardware Energy}}$, the combination of the random filtering technique with DepGraphRL achieves higher accuracy, however, the combination of random filter ranking technique with DepGraphRL has the most reduction in energy consumption as compared to other combinations of pruning frameworks.

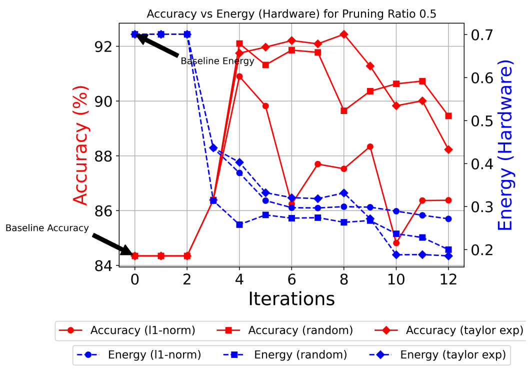
Now shift towards the results with a pruning ratio 0.5 that are shown in Fig. 4.7. Here, after 10 iterations using l_1-norm filter ranking technique with DepGraphRL framework, the pruned model achieved an accuracy of 90.87%. This represents a drop in $E_{\text{Theoretical Energy}}$ of 52.72% from the baseline theoretical energy and a drop in $E_{\text{Hardware Energy}}$ of 61.36% from



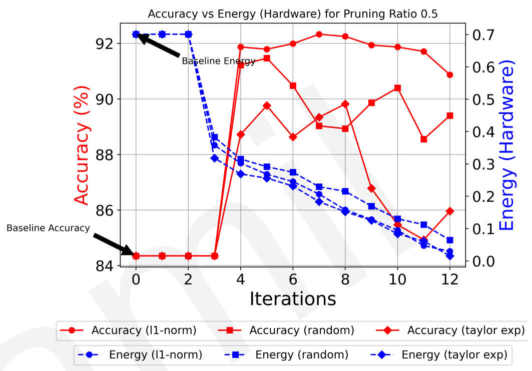
(a) $E_{Theoretical\ Energy}$ in DepGraph with $P_r = 0.5$



(b) $E_{Theoretical\ Energy}$ in DepGraphRL with $P_r = 0.5$



(c) $E_{Hardware\ Energy}$ in DepGraph with $P_r = 0.5$



(d) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.5$

Figure 4.7: Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning at a ratio of 0.5 in the ResNet-18 CNN model, applied to the CIFAR-10 dataset.

the baseline hardware-based energy. Considering random filter ranking after 10 iterations with a pruning ratio of 0.5 using the DepGraphRL framework, the pruned model attained an accuracy of 89.4%. This corresponds to a decrease in $E_{\text{Theoretical Energy}}$ of 53.085% from the baseline theoretical energy and a decrease in $E_{\text{Hardware Energy}}$ of 71.50% from the baseline hardware-based energy. For the Taylor Expansion ranking method after 10 iterations with a pruning ratio of 0.5 using the DepGraphRL framework, the pruned model reached an accuracy of 85.96%. This reflects a decrease in $E_{\text{Theoretical Energy}}$ of 40.51% from the baseline theoretical energy and a decrease in $E_{\text{Hardware Energy}}$ of 73.67% from the baseline hardware-based energy.

Considering the pruning ratio of 0.5 in the above results, it is evident from the results that the combination of l_1-norm with DepGraphRL best energy accuracy tradeoff for $E_{\text{Theoretical Energy}}$ when compared with the baseline pruning scheme (DepGraph) and all other filter pruning techniques. For the case of the $E_{\text{Hardware Energy}}$, the combination of l_1-norm with DepGraphRL achieves better energy accuracy tradoff compared to others, however the combination of random filter ranking technique with DepGraphRL provides comparable reduction in energy consumption as compared to other combinations of pruning frameworks.

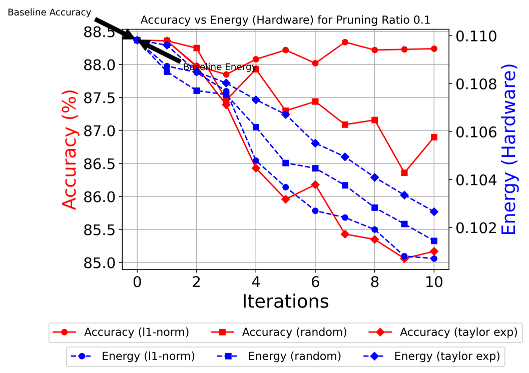
The summary of above presented results is that the random and l_1 as compared to Taylor expansion filter ranking method often provides a good balance between energy consumption and accuracy across both theoretical and hardware types. Taylor's method, while sometimes achieving a better accuracy, often does so at the cost of higher loss of accuracy. In the context of different pruning ratios, it is observed from results presented in Fig. 4.5, 4.6, and 4.7 that with increasing pruning ratio (starting from 0.1 till 0.5), the energy consumption tends to decrease for the theoretical type as well as for the hardware type. Also, the accuracy tends to decrease with higher pruning ratios for both energy types. Lastly, the DepGraphRL-based energy-aware pruning performs significantly better than the baseline scheme in terms of energy-accuracy tradeoff.

MobileNet [91]:

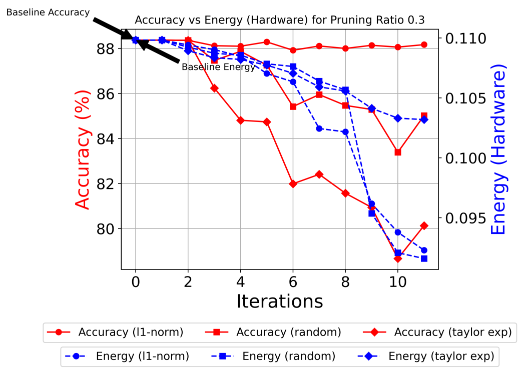
For the case of the MobileNet CNN model, the results are presented in Fig. 4.8 which are generated while considering DepGraphRL as a pruning framework. Here, the results are only illustrated for $E_{\text{Hardware Energy}}$. As can be observed from these plots, the highest achieved accuracy of 88.24% with the combination of l_1 filter ranking technique in DepGraphRL at pruning ratio 0.1. Here, the energy drop is 7.27%. For pruning ratio 0.3, the best energy accuracy tradeoff is achieved using l_1 filter ranking technique with an accuracy of 88.17% and an energy drop of 15.63%. For pruning ratio 0.5, the best energy accuracy tradeoff is achieved using the l_1 filter ranking technique with an accuracy of 85.26% and an energy drop of 39.09%. Overall, the l_1 filter ranking technique provides the best energy-accuracy tradeoff across all pruning ratios.

VGG-19 [92]:

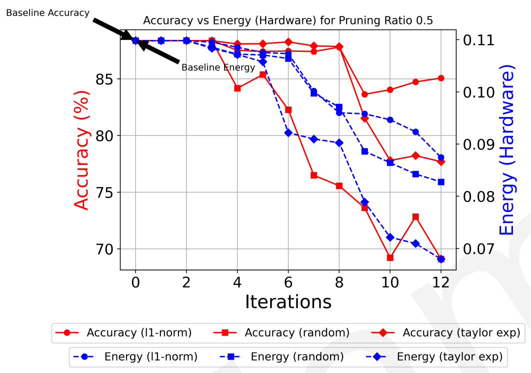
For the case of the VGG CNN model, the results are presented in Fig. 4.9 which are generated while considering DepGraphRL as a pruning framework. Here, the results are only illustrated for $E_{\text{Hardware Energy}}$. As can be observed from these plots, the highest achieved accuracy of 93.93% with an energy drop of 21.62% is obtained with the combination of taylor expansion filter ranking technique in DepGraphRL at pruning ratio 0.1. In the case of pruning ratio 0.3, the best accuracy energy trade-off is obtained with a taylor expansion ranking technique where accuracy is 92.39% and energy drop of 23.27%. Further, for a pruning ra-



(a) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.1$



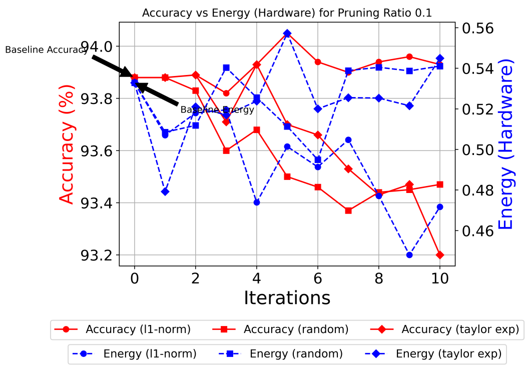
(b) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.3$



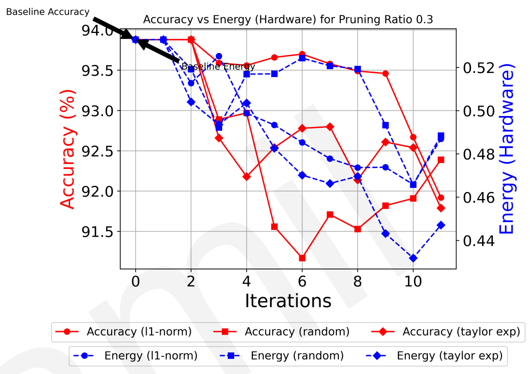
(c) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.5$

Figure 4.8: Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning for MobileNet applied to the CIFAR-10 dataset.

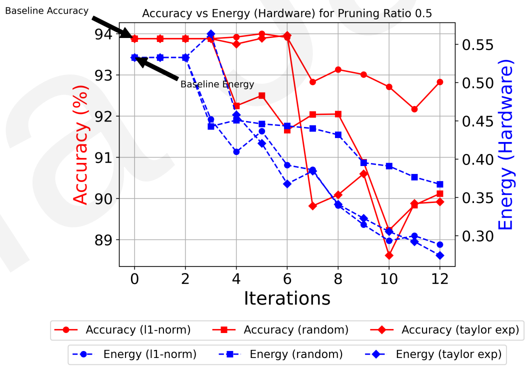
tio of 0.5, the best accuracy energy trade-off is obtained with the l_1 filter ranking technique where accuracy is 92.83% and energy drop of 45.86%.



(a) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.1$



(b) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.3$



(c) $E_{Hardware\ Energy}$ in DepGraphRL with $P_r = 0.5$

Figure 4.9: Tradeoff between energy (J) and accuracy (%) over ten iterations of pruning for VGG applied to the CIFAR-10 dataset.

Chapter 5

Conclusion and Future Work

In the era of IoT development, which extensively utilizes Deep Learning and Machine Learning techniques, two primary challenges are evident: computational power and energy efficiency. The integration of cloud computing seemed a promising solution, addressing these concerns, but it brought its own set of obstacles, notably latency and connectivity issues. A promising direction is the shift towards Edge-IoT devices, however, they demand efficient energy management to run computationally extensive neural networks on the edge.

One way to achieve this is to deploy efficient energy-aware, less structurally complex neural network models on Edge-IoT devices through network pruning. Still, this deployment comes with various constraints, i.e. utilizing empirical measurements for an accurate estimate of energy consumption in practical applications, the lack of generalizability across various neural network architectures and the complexities introduced by structural coupling in traditional pruning processes. Another constraint is the optimization challenges in pruning CNN models for Edge-IoT devices, considering the multifaceted nature of variables within a dynamic environment, such as layer count, energy consumption, and accuracy requirements.

To resolve these constraints, this thesis proposes a novel DepGraphRL framework to optimise the CNN models on Edge-IoT devices. The proposed framework resolves these constraints by utilizing DRL to provide an energy-aware, fully automated, and generalizable layer-wise pruning technique for neural networks by employing DepGraph for structural pruning and empirical methods for energy consumption calculation of CNNs at edge devices. Based on the Proposed DepGraphRL framework for different pruning ratios and feature importance ranking techniques on different neural network architectures like ResNet-18, VGG-19, and MobileNet, it is evident that the proposed framework reduces overall energy consumption while obtaining accuracy comparable to the baseline neural network models. This shows the effectiveness of the proposed energy-aware framework in optimizing the performance of neural networks in energy-constraint environments by providing energy-efficient pruned architectures of baseline CNN models with comparable accuracy to the baseline.

Future enhancements to this framework could be achieved by assessing its performance on various hardware platforms, and using different methods for measuring energy consumption. Additionally, applying the framework in real-world IoT environments would provide valuable insights. This would involve deploying the optimized models in diverse IoT scenarios such as smart cities, healthcare monitoring, and industrial automation, to evaluate practical challenges and performance in actual conditions.

Exploring the integration of DepGraphRL with other model compression techniques, like quantization or knowledge distillation, may lead to more efficient models without compromising performance. Improving the accuracy of empirical energy measurement methods, to reflect a range of operational conditions, would enhance the framework's practical application. Finally, adapting the optimization process to meet specific user requirements, such as particular accuracy or latency needs, could make the framework more versatile for various applications.

Bibliography

[1] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, “Depgraph: Towards any structural pruning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 091–16 101.

[2] K. Rose, S. Eldridge, and L. Chapin, “The internet of things: An overview,” *The internet society (ISOC)*, vol. 80, pp. 1–50, 2015.

[3] S. Li, L. D. Xu, and S. Zhao, “The internet of things: a survey,” *Information systems frontiers*, vol. 17, pp. 243–259, 2015.

[4] R. M. Dijkman, B. Sprenkels, T. Peeters, and A. Janssen, “Business models for the internet of things,” *International Journal of Information Management*, vol. 35, no. 6, pp. 672–678, 2015.

[5] Z. Mahmood, *The internet of things in the industrial sector*. Springer, 2019.

[6] L. Georgios, S. Kerstin, and A. Theofylaktos, “Internet of things in the context of industry 4.0: An overview,” 2019.

[7] S. Fang, L. Da Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan, and Z. Liu, “An integrated system for regional environmental monitoring and management based on internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1596–1605, 2014.

[8] H. Li, S. Liu, Q. Duan, and W. Li, “Application of multi-sensor image fusion of internet of things in image processing,” *Ieee Access*, vol. 6, pp. 50 776–50 787, 2018.

[9] B. L. R. Stojkoska and K. V. Trivodaliev, “A review of internet of things for smart home: Challenges and solutions,” *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.

[10] A. Ghasempour, “Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges,” *Inventions*, vol. 4, no. 1, p. 22, 2019.

[11] V. P. Kour and S. Arora, “Recent developments of the internet of things in agriculture: a survey,” *Ieee Access*, vol. 8, pp. 129 924–129 957, 2020.

[12] S. Singh and N. Singh, “Internet of things (iot): Security challenges, business opportunities & reference architecture for e-commerce,” in *2015 International conference on green computing and internet of things (ICGCIoT)*. Ieee, 2015, pp. 1577–1581.

[13] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[14] F. Shi, H. Ning, W. Huangfu, F. Zhang, D. Wei, T. Hong, and M. Daneshmand, “Recent progress on the convergence of the internet of things and artificial intelligence,” *IEEE Network*, vol. 34, no. 5, pp. 8–15, 2020.

[15] F. Firouzi, B. Farahani, and A. Marinšek, “The convergence and interplay of edge, fog, and cloud in the ai-driven internet of things (iot),” *Information Systems*, vol. 107, p. 101840, 2022.

[16] E. Mohamed, “The relation of artificial intelligence with internet of things: A survey,” *Journal of Cybersecurity and Information Management*, vol. 1, no. 1, pp. 30–24, 2020.

[17] A. Ghosh, D. Chakraborty, and A. Law, “Artificial intelligence in internet of things,” *CAAI Transactions on Intelligence Technology*, vol. 3, no. 4, pp. 208–218, 2018.

[18] A. Sehgal, V. Perelman, S. Kuryla, and J. Schonwalder, “Management of resource constrained devices in the internet of things,” *IEEE Communications Magazine*, vol. 50, no. 12, pp. 144–149, 2012.

[19] W. Ejaz, M. Naeem, A. Shahid, A. Anpalagan, and M. Jo, “Efficient energy management for the internet of things in smart cities,” *IEEE Communications magazine*, vol. 55, no. 1, pp. 84–91, 2017.

[20] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, “Iot and cloud computing issues, challenges and opportunities: A review,” *Qubahan Academic Journal*, vol. 1, no. 2, pp. 1–7, 2021.

[21] M. Díaz, C. Martín, and B. Rubio, “State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing,” *Journal of Network and Computer applications*, vol. 67, pp. 99–117, 2016.

[22] F. S. Abkenar, P. Ramezani, S. Iranmanesh, S. Murali, D. Chulerttiyawong, X. Wan, A. Jamalipour, and R. Raad, “A survey on mobility of edge computing networks in iot: State-of-the-art, architectures, and challenges,” *IEEE Communications Surveys & Tutorials*, 2022.

[23] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: analysis, applications, and prospects,” *IEEE transactions on neural networks and learning systems*, 2021.

[24] S. Sakib, N. Ahmed, A. J. Kabir, and H. Ahmed, “An overview of convolutional neural network: Its architecture and applications,” 2019.

[25] T. Guo, J. Dong, H. Li, and Y. Gao, “Simple convolutional neural network on image classification,” in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, 2017, pp. 721–724.

[26] A. V. Abraham, P. Sasidharan, S. S. Tejas, M. Manohara, R. Muthu, and R. C. Naidu, “Predicting Energy Consumption Using LSTM and CNN Deep Learning Algorithm,” in *2022 7th International Conference on Environment Friendly Energies and Applications (EFEA)*. IEEE, dec 14 2022.

[27] R. Xie, X. Jia, L. Wang, and K. Wu, “Energy Efficiency Enhancement for CNN-based Deep Mobile Sensing,” *IEEE Wireless Communications*, vol. 26, no. 3, pp. 161–167, 6 2019.

[28] W. Jiang, H. Yu, J. Zhang, J. Wu, S. Luo, and Y. Ha, “Optimizing energy efficiency of CNN-based object detection with dynamic voltage and frequency scaling,” *Journal of Semiconductors*, vol. 41, no. 2, p. 022406, feb 1 2020.

[29] T.-J. Yang, Y.-H. Chen, and V. Sze, “Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 7 2017.

[30] Yue Wang, Ziyu Jiang, Xiaohan Chen, Pengfei Xu, Yang Zhao, Yingyan Lin, and Zhangyang Wang, "E2-Train: Training State-of-the-art CNNs with Over 80

[31] K. Gaur and S. Kumar Singh, "Cnn-Bi-LSTM Based Household Energy Consumption Prediction," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, may 13 2021.

[32] M. Zawish, "Energy-aware ai-driven framework for edge-computing-based iot applications," *IEEE Internet of Things Journal*, 2023.

[33] ——, "Complexity-driven cnn compression for resource-constrained edge ai," *arXiv preprint*, 2022.

[34] M. Zawish, N. Ashraf, R. I. Ansari, and S. Davy, "Energy-Aware AI-Driven Framework for Edge-Computing-Based IoT Applications," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5013–5023, mar 15 2023.

[35] C. Yao, W. Liu, W. Tang, J. Guo, S. Hu, Y. Lu, and W. Jiang, "Evaluating and analyzing the energy efficiency of CNN inference on highperformance GPU," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 6, oct 21 2020.

[36] A. Abdelaziz, V. Santos, and M. S. Dias, "Convolutional Neural Network With Genetic Algorithm for Predicting Energy Consumption in Public Buildings," *IEEE Access*, vol. 11, pp. 64 049–64 069, 2023.

[37] A. Sanchez-Flores, L. Alvarez, and B. Alorda-Ladaria, "A review of CNN accelerators for embedded systems based on RISC-V," in *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, aug 1 2022.

[38] A. Yoosefi and M. Kargahi, "Improving Energy-Efficiency of CNNs via Prediction of Reducible Convolutions for Energy-Constrained IoT Devices," in *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE, 6 2020.

[39] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the Energy Efficiency of Deep Convolutional Neural Networks on CPUs and GPUs," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*. IEEE, 10 2016.

[40] Boyu Zhang, A. Davoodi, and Y. Hu, "Efficient Inference of CNNs via Channel Pruning," *ArXiv*, 2019.

[41] N. Tian, Y. Liu, W. Wang, and D. Meng, "Energy-saving CNN with Clustering Channel Pruning," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 18 2021.

[42] Hao Li, Asim Kadav, Igor Durdanovic, H. Samet, and H. Graf, "Pruning Filters for Efficient ConvNets," *International Conference on Learning Representations*, 2016.

[43] Gianlorenzo D'angelo, Mattia D'emidio, and D. Frigioni, "Pruning the Computation of Distributed Shortest Paths in Power-law Networks," *Informatica*, 2013.

[44] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[45] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.

[46] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.

[47] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *International Conference on Learning Representations*, 2018.

[48] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “Amc: Automl for model compression and acceleration on mobile devices,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 784–800.

[49] H. Wang, Q. Zhang, Y. Wang, L. Yu, and H. Hu, “Structured Pruning for Efficient ConvNets via Incremental Regularization,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 7 2019.

[50] G. Krishnan, Y. Ma, and Y. Cao, “Small-world-based Structural Pruning for Efficient FPGA Inference of Deep Neural Networks,” in *2020 IEEE 15th International Conference on Solid-State & amp; Integrated Circuit Technology (ICSICT)*. IEEE, nov 3 2020.

[51] Z. Xu, J. Sun, Y. Liu, and G. Sun, “An Efficient Channel-level Pruning for CNNs without Fine-tuning,” in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 18 2021.

[52] W. Wang and L. Zhu, “Channel Pruning for Efficient Convolution Neural Networks,” *Journal of Physics: Conference Series*, vol. 1302, no. 2, p. 022073, aug 1 2019.

[53] T. Jeong, E. Ghasemi, J. Tuyls, E. Delaye, and A. Sirasao, “Neural network pruning and hardware acceleration,” in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 12 2020.

[54] Xuanyi Dong and Yi Yang, “Network Pruning via Transformable Architecture Search,” *Neural Information Processing Systems*, 2019.

[55] S. Yu, A. Mazaheri, and A. Jannesari, “Auto Graph Encoder-Decoder for Neural Network Pruning,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 10 2021.

[56] Y. Aflalo, Asaf Noy, Ming Lin, Itamar Friedman, and Lihi Zelnik-Manor, “Knapsack Pruning with Inner Distillation,” *arXiv.org*, 2020.

[57] Dongsoo Lee, Se Jung Kwon, Byeongwook Kim, Parichay Kapoor, and Gu-Yeon Wei, “Network Pruning for Low-Rank Binary Indexing,” *arXiv.org*, 2019.

[58] N. T. Siebel, J. Botel, and G. Sommer, “Efficient neural network pruning during neuro-evolution,” in *2009 International Joint Conference on Neural Networks*. IEEE, 6 2009.

[59] I. Partalas, G. Tsoumakas, and I. Vlahavas, “Pruning an ensemble of classifiers via reinforcement learning,” *Neurocomputing*, vol. 72, no. 7-9, pp. 1900–1909, 3 2009.

[60] B. Bencsik and M. Szemenyei, “Efficient Neural Network Pruning Using Model-Based Reinforcement Learning,” in *2022 International Symposium on Measurement and Control in Robotics (ISMCR)*. IEEE, sep 28 2022.

[61] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou, “Runtime Neural Pruning,” *Neural Information Processing Systems*, 2017.

[62] S. Malik, M. U. Haider, O. Iqbal, and M. Taj, “Neural Network Pruning Through Constrained Reinforcement Learning,” in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, aug 21 2022.

[63] Z. Wang and C. Li, “Channel Pruning via Lookahead Search Guided Reinforcement Learning,” in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1 2022.

[64] C. Omlin and C. Giles, “Pruning recurrent neural networks for improved generalization performance,” in *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*. IEEE.

[65] J. Fontaine, A. Shahid, B. Van Herbruggen, and E. De Poorter, “Impact of Embedded Deep Learning Optimizations for Inference in Wireless IoT Use Cases,” *IEEE Internet of Things Magazine*, vol. 5, no. 4, pp. 86–91, 12 2022.

[66] L. Ye, Z. Wang, Y. Liu, P. Chen, H. Li, H. Zhang, M. Wu, W. He, L. Shen, Y. Zhang, Z. Tan, Y. Wang, and R. Huang, “The Challenges and Emerging Technologies for Low-Power Artificial Intelligence IoT Systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 12, pp. 4821–4834, 12 2021.

[67] M. M. Alenazi, B. A. Yosuf, T. El-Gorashi, and J. M. H. Elmirghani, “Energy Efficient Neural Network Embedding in IoT over Passive Optical Networks,” in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*. IEEE, 7 2020.

[68] S. Govindaraj and S. N. Deepa, “Network Energy Optimization of IOTs in Wireless Sensor Networks Using Capsule Neural Network Learning Model,” *Wireless Personal Communications*, vol. 115, no. 3, pp. 2415–2436, aug 5 2020.

[69] J. Zhang, “Real-time detection of energy consumption of IoT network nodes based on artificial intelligence,” *Computer Communications*, vol. 153, pp. 188–195, 3 2020.

[70] Elliot J. Crowley, Jack Turner, A. Storkey, and M. O’Boyle, “A Closer Look at Structured Pruning for Neural Network Compression,” 2018.

[71] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, “Towards Optimal Structured CNN Pruning via Generative Adversarial Learning,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2019.

[72] A. Bragagnolo, E. Tartaglione, A. Fiandrotti, and M. Grangetto, “On the role of structured pruning for neural network compression,” in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3527–3531.

[73] Z. Wang, C. Li, and X. Wang, “Convolutional Neural Network Pruning with Structural Redundancy Reduction,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2021.

[74] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and J. Kautz, “Pruning Convolutional Neural Networks for Resource Efficient Inference,” *International Conference on Learning Representations*, 2016.

[75] L. Guerra and T. Drummond, “Automatic Pruning for Quantized Neural Networks,” in *2021 Digital Image Computing: Techniques and Applications (DICTA).* IEEE, 11 2021.

[76] M. Augasta and T. Kathirvalavakumar, “Pruning algorithms of neural networks — a comparative study,” *Open Computer Science*, vol. 3, no. 3, jan 1 2013.

[77] M. A. Costa, A. P. Braga, and B. R. de Menezes, “Constructive and pruning methods for neural network design,” in *VII Brazilian Symposium on Neural Networks, 2002. SBRN 2002. Proceedings.* IEEE, 2002, pp. 49–54.

[78] R. Setiono and A. Gaweda, “Neural network pruning for function approximation,” in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium.* IEEE, 2000.

[79] A. Sankar and R. J. Mammone, “Optimal pruning of neural tree networks for improved generalization,” in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. 2. IEEE, 1991, pp. 219–224.

[80] M. H. Uddin and S. Baidya, “Optimizing neural network efficiency with hybrid magnitude-based and node pruning for energy-efficient computing in iot,” *ACM Transactions*, 2023.

[81] X. Ma, G. Yuan, S. Lin, Z. Li, H. Sun, and Y. Wang, “Resnet can be pruned 60×: Introducing network purification and unused path removal (p-rm) after weight pruning,” in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH).* IEEE, 2019, pp. 1–2.

[82] X. He, W. Lu, K. Liu, G. Yan, and X. Zhang, “A quantitative exploration of collaborative pruning and approximation computing towards energy efficient neural networks,” *IEEE Design & Test*, vol. 37, no. 1, pp. 36–45, 2019.

[83] Z. Montazeri and T. Niknam, “Energy carriers management based on energy consumption,” in *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI).* IEEE, 12 2017.

[84] E. Ghazisaeedi and C. Huang, “Energy-aware node and link reconfiguration for virtualized network environments,” *Computer Networks*, vol. 93, pp. 460–479, 12 2015.

[85] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, “Metapruning: Meta Learning for Automatic Neural Network Channel Pruning,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV).* IEEE, 10 2019.

[86] M. Zawish, N. Ashraf, R. I. Ansari, and S. Davy, “Energy-aware ai-driven framework for edge-computing-based iot applications,” *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5013–5023, 2022.

[87] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “Eie: Efficient inference engine on compressed deep neural network,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[88] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1389–1397.

[89] G. Gobieski, B. Lucia, and N. Beckmann, “Intelligence beyond the edge: Inference on intermittent embedded systems,” in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 199–213.

[90] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[91] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[92] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[93] D. Mittal, S. Bhardwaj, M. M. Khapra, and B. Ravindran, “Studying the plasticity in deep convolutional neural networks using random pruning,” *Machine Vision and Applications*, vol. 30, no. 2, pp. 203–216, 2019.