Energy-aware AI-driven Framework for Edge Computing-based IoT Applications

Muhammad Zawish, *Student Member, IEEE,* Nouman Ashraf, *Member, IEEE,* Rafay Iqbal Ansari, *Senior Member, IEEE,* Steven Davy, *Member, IEEE*

***Abstract*—The significant growth in the number of Internet-of-things (IoT) devices has given impetus to the idea of edge computing for several applications. In addition, energy harvestable or wireless-powered wearable devices are envisioned to empower the edge intelligence in IoT applications. However, the intermittent energy supply and network connectivity of such devices in scenarios including remote areas and hard-to-reach regions such as in-body applications can limit the performance of edge computing-based IoT applications. Hence, deploying state-of-the-art convolutional neural networks (CNNs) on such energy-constrained devices is not feasible due to their computational cost. Existing model compression methods such as network pruning and quantization can reduce complexity, but these methods only work for fixed computational or energy requirements, which is not the case for edge devices with an intermittent energy source. In this work, we propose a pruning scheme based on deep reinforcement learning (DRL), which can compress the CNN model adaptively according to the energy dictated by the energy management policy and accuracy requirements for IoT applications. The proposed energy policy uses predictions of energy to be harvested and dictates the amount of energy that can be used by the edge device for deep learning inference. We compare the performance of our proposed approach with existing state-of-the-art CNNs and datasets using different filter-ranking criteria and pruning ratios. We observe that by using DRL-driven pruning, the convolutional layers that consume relatively higher energy are pruned more as compared to their counterparts. Thereby, our approach outperforms existing approaches by reducing energy consumption and maintaining accuracy.**

***Index Terms*—Artificial Intelligence, edge computing, energy efficiency, Internet of Things.**

I. Introduction

Fifth-generation (5G) wireless networks aim at supporting new applications and have given impetus to the evolution of the Internet-of-things (IoT). The advancement towards sixth-generation (6G) networks will be empowered by solutions where IoT devices will play a significant role [1]. The rapid growth in the number of connected devices for applications such as smart homes, smart cities and wearable devices for healthcare has led to the utilization of edge computing, especially for scenarios with strict requirements of latency, reliability and energy efficiency [2], [3]. Moreover, the artificial intelligence (AI)-based solutions will play a key role in improving the key performance indicators (KPIs) in future 6G networks [4], [5].

European Telecommunications Standard Institute (ETSI) introduced the architecture that utilizes edge computing [6], leading towards the development of edge computing-assisted applications. The idea was to delegate the computation to the edge devices instead of undertaking them at the base station (BS). As a result, the data is processed near the source hence ensuring the security and low-latency decisions by saving the energy required for wireless transfer of data to the cloud. The edge devices with computing capability are in close vicinity of the end users, which allows them to realize several applications such as smart agriculture [5]. Despite the fact that due to edge computing, wearable devices are getting more realizable and inexpensive, AI-based applications that normally need greater processing resources may overload them with more data transfers and, as a result, increased energy consumption. However, as most edge devices (due to their mobile nature) are powered by batteries, their energy resources and operational hours are limited. Similarly, if enough battery power is not ~~is~~ available for task transmission, compute performance may suffer. Thus, the computing capability and transmissions from edge devices heavily depend on the energy available at the device. This issue can be solved by using a bigger battery or charging it more frequently. The tiny size of edge or IoT devices, on the other hand, makes it difficult to equip them with bigger batteries or to recharge their batteries on a regular basis. Energy-harvesting technologies have been recognized as possible ways ~~for~~ of increasing battery life and achieving energy-autonomous systems in order to meet these difficulties [7].

In several circumstances, the energy harvesting process may be affected by unpredictable extraneous environments, such as fluctuating solar irradiance in the case of solar energy harvesting. Furthermore, energy cannot be harvested at all times (such as at night), and the pace at which energy may be generated is restricted. The edge-computing enabled IoT devices with low energy consumption may incur long delays since the energy is not completely utilized to transmit at high data rates or compute at full capability, and the devices may miss recharging opportunities owing to battery capacity restrictions. However, excessive energy usage may result in poor capability and lead to connections for brief periods of time [8]. Furthermore, total exhaustion of the battery restricts the computation of edge devices, which may be critical in

M. Zawish* and S. Davy are with Walton Institute, South East Technological University, Ireland, Emails:{muhammad.zawish, steven.davy}@waltoninstitute.ie
N. Ashraf is with Technological University Dublin, Ireland Email: {nouman.ashraf}@tudublin.ie
R. I. Ansari is with the Department of Computer and Information Sciences, Northumbria University Newcastle, UK. Email: {rafay.ansari}@northumbria.ac.uk
*Corresponding Author

particular applications. As a result, energy-neutral consumption strategies are necessary to avoid energy depletion, to ensure edge device performance for an extended period of time by balancing the aforementioned conflicting objectives, and maximizing the application performance.

To deal with the aforementioned scenario, this work focuses on energy-constrained Internet-of-things (IoT)-enabled edge devices used for monitoring, especially in scenarios with intermittent energy supply and network connectivity. It is to be noted that the proposed approach applies to both energy-harvestable IoT devices and wireless-powered IoT devices. The utilization of edge devices such as wearables or unmanned aerial vehicles (UAVs) for monitoring has led to the development of several IoT applications [9]. Such edge devices can help to monitor the environment, capture images, and video, and process the data before sending it to the cloud [10]. Fig. 1 presents several edge-AI applications for IoT.

Convolutional neural networks (CNNs) are among the foremost AI algorithms for processing the raw data generated from the edge nodes and providing valuable insights. However, their computational complexity makes them practically infeasible for the aforementioned energy-constrained applications. Thus, state-of-the-art CNNs such as VGG-16 [11], or ResNet [12] are required to be compressed according to intermittent requirements of edge devices. IoT-enabled edge nodes can be deployed either in isolated or outlying regions such as in a rural environment, or are geographically spread in an area [3], [13], [14], thereby having limited network connectivity and energy supply. As a result, preventing increased energy usage and communication costs is a complex and demanding problem that needs to be addressed. In such scenarios, there is a need for an adaptive AI-driven scheme that can compress a CNN model dynamically based on the energy availability status. Deep reinforcement learning (DRL), a category of AI, has recently gained attention for utilization in systems having dynamic requirements. Thereby, this work focuses on a framework comprising DRL-driven adaptive model compression that can be helpful in realizing energy-constrained edge-empowered IoT applications.

Recent studies on CNN compression are mainly based on filter pruning [6], [15]–[20], weight pruning [21]–[23], and quantization [22], [24]. These approaches either proposed a complex filter-ranking criteria or a joint multi-stage offline framework for reducing floating-point operations (FLOPs)/memory. These approaches are only suitable when a certain application has a fixed accuracy-complexity requirement. However, they do not cater ~~for~~ to the needs for applications of IoT, where resource requirement changes dynamically. In contrast, this work aims to propose a solution based on DRL, where any pre-trained CNN can be pruned dynamically with the best accuracy and energy trade-off. The proposed framework consists of two parts: intelligent energy management and energy-aware intelligent edge computing. The formal is based on modeling the battery of the edge device as a linear queue accommodating energy charges [25]. For this energy queue model, we use Model Predictive Control to design a control algorithm, which considers the prediction of energy to be harvested and regulates the output energy of the battery to a predetermined level [8]. This regulation of energy makes sure that the edge device always has energy available for its crucial operation. Our second algorithm takes the energy availability status parameter from the energy management system (EMS) and accuracy requirement as input and prunes the CNN model iteratively until the desired accuracy-energy requirements are met. Then, the CNN model can be executed, for instance, on an energy-constrained wearable edge device deployed on or in the body of a human or an animal. The rest of the paper is organized as follows. We review the related literature in Section II, followed by the contribution and significance of this work. Section III and IV describe the proposed approach and its performance evaluation, respectively, followed by a conclusion in Section V.

II. Related Work and Contributions

A. Compressing CNNs for Edge devices

In recent years, various techniques have been proposed to optimize CNNs for resource-constrained edge devices. For example, network pruning [15], [16], [18], [22], [24], [26], [27], quantization [22], and light-weight structure design [28] are a few techniques widely used for compressing or redesigning the CNNs. Network pruning has recently gained attention for reducing the complexity of CNNs without significantly degrading the performance. Several works propose techniques for identifying and removing unimportant weight connections to prune CNNs in an unstructured manner. For example, [21]–[23], demonstrated the significant compression on unstructured pruning of AlexNet, VGG-16, and ResNets. However, their proposed work underpins highly sparse models, which are complex to fine-tune and require additional time for hyperparameter optimization. In contrast, filter pruning was proposed to maintain the structured sparsity in CNNs. Therefore, it does not require any custom hardware or software for the execution, unlike weight pruning. For this reason, in this work, we focus on exploiting the idea of structured pruning using DRL.

B. CNN pruning schemes

Several techniques have been proposed to identify the unimportant filters in a convolutional layer, such as ℓ_1-norm [16], APoZ [15], Taylor expansion [17], and mean activation [27]. According to Hu et al. [15], eliminating those filters whose feature maps have the highest average percentage of zero activations leads to a compact model. Similarly, Li et al. [16], proposed to calculate the magnitude of filters using ℓ_1-norm, and removed filters with least ℓ_1-norm. However, these methods follow a one-shot pruning mechanism and yield a model with strict resource requirements. On the other hand, some studies proposed iterative pruning [6], [18], [19], where a model is compressed iteratively with a small pruning ratio to achieve the resource-accuracy trade-off. Keeping in view the idea of hand-crafted complex filter ranking techniques, Mittal et al. [29] suggested that pruning filters randomly can also achieve similar performance as state-of-the-art ranking-based techniques. Thereby, a ranking framework does not essentially contribute to the performance of CNNs, and it is the plasticity of CNNs which helps them to recover after fine-tuning [29].

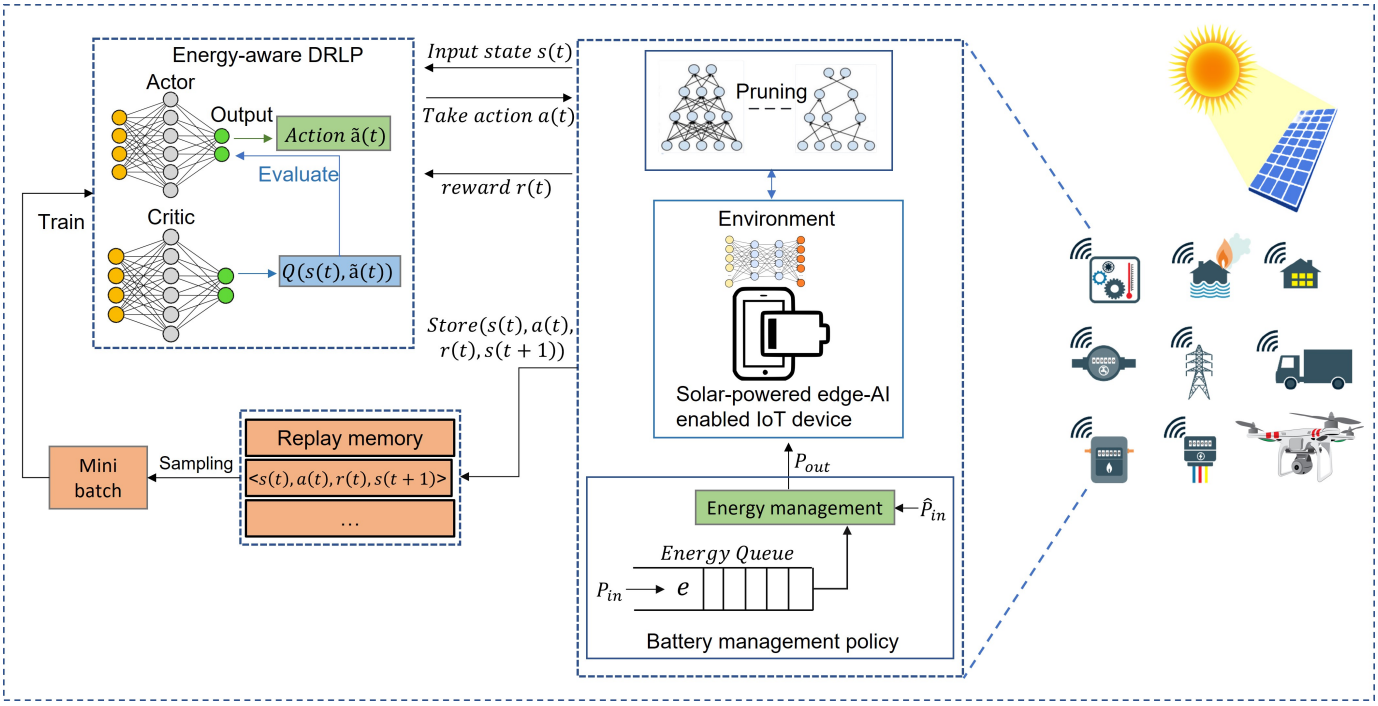

Figure 1: The proposed energy-aware DRL-driven model compression and battery managemet scheme for Edge-AI enabled IoT applications.

However, these works prune filters either from all layers or on an ad-hoc basis. As a result, this approach is not rationalized because each layer has a different level of complexity. In this work, we overcome it by compressing each layer with respect to its defined complexity i.e., energy in our case.

C. DRL based CNN compression

Recently, DRL-based techniques have emerged to automate the optimization of CNNs using policies such as Deep Deterministic Policy Gradient (DDPG), Q learning, etc. [30]–[32]. Most of these techniques are focused on generating or searching the architecture of CNNs using the agent's search space. However, there is limited work available on DRL-based pruning mechanisms. For instance, [24] recently proposed a two-stage framework based on joint optimization of CNNs using pruning and quantization. The authors use a DRL agent to sample the sparsity ratio for a certain layer based on an observation vector consisting of layer properties. However, the above works are only theoretically acceptable because they do not consider the dynamic resource requirements of mobile edge devices. In this work, we address the aforementioned problems by proposing a DRL-based automatic pruning scheme for CNNs where the model will be compressed according to the dynamic energy requirements of the device. Moreover, unlike previous studies, our DRL agent selects a particular layer for pruning based on its energy proportion with respect to the CNN's energy. The proposed technique can be particularly helpful in edge-empowered IoT applications such as healthcare and agriculture, where intermittent energy and network conditions necessitate the need for a dynamic CNN compression approach.

D. Novelty and Significance of the Proposed Solutions

In this work, we propose a DRL-based CNN model compression to enable edge-AI in IoT applications. To summarize, following are the key contributions of this work:

- We present an energy management strategy that uses the prediction of energy to be harvested and regulates the output power of the battery by using the Model predictive control method.
- We propose a novel DRL based pruning scheme (DRLP), which prunes the model iteratively to meet the dynamic energy requirements of the energy-constrained edge devices for IoT applications.
- We demonstrate the significance of the DRLP in two aspects: i) a CNN model is compressed automatically for immediate execution on edge devices based on the energy-accuracy criteria, and ii) the pruning mechanism considers the underlying energy of each layer in the CNN model and prunes a certain layer based on its proportion to overall model complexity.
- We evaluate the performance of our approach through extensive simulation experiments and compare it with state-of-the-art methods on various filter-ranking mechanisms and pruning ratios using standard CNN models and benchmark datasets. It can be observed from the results that our approach outranks other state-of-the-art methods on model pruning in terms of the energy-accuracy trade-off.
- We integrate energy management policy with our proposed DRLP method and perform extensive simulations to demonstrate the combined performance of the method by using realistic data of energy.

It is to be noted that the contribution of this work not only lies in proposing the use of energy-aware DRL for CNN compression but also in proposing a mechanism for adaptive on-demand compression of CNNs, which has not been explored in literature. In the subsequent sections, we present the methodology of our proposed framework and provide details on the energy consumption of CNNs and the energy-aware DRLP scheme. Moreover, the energy management scheme in this work is based on our previous work in [8]; however, in this work, instead of considering the case of IoT devices, we integrate this energy management strategy with edge computing-based IoT devices.

III. The Proposed Energy-Aware Scheme

The framework for the energy-aware DRL-driven model compression and battery management is shown in Fig. 1. In the proposed model, we consider energy harvesting enabled edge devices, where these edge devices harvest ambient energy from the surrounding environment. The proposed framework consists of two parts: a battery management policy and an energy-aware DRLP scheme. Our energy management policy is responsible for monitoring the intermittent energy status of a certain edge device that is empowered with battery and solar power. The EMS regulates the available energy required for the execution of a CNN model on the edge devices for a certain IoT application [33]. The proposed DRLP model then considers the status of the energy system with intermittent energy supply and provides a compressed CNN model in return for execution. In contrast to existing compression schemes, our DRLP scheme prunes a pre-trained CNN in an energy-aware manner such that each layer of CNN is pruned according to its weightage in the overall energy complexity of CNN. The DRL model needs to be pre-trained on the GPU or a cloud server before deployment on the edge device. Then, any CNN model can be compressed using the proposed DRLP on the device itself without further training/fine-tuning and made available immediately for inference on edge, unlike offloading to the cloud. It has been reported in several studies that offloading inference tasks to the cloud requires orders of magnitude more energy than local inference on energy-harvesting edge devices [13], [23]. Therefore, the proposed scheme avoids computational offloading to cloud, hence reducing the energy expenditure required for it. In subsequent sections, we first present our energy management scheme. Secondly, we analyze the energy consumption of CNNs, followed by a comprehensive description of the proposed DRLP scheme. We demonstrate the DRLP scheme with an understanding of state, action, and reward functions curated for this task. The summary of notations is presented in Table I.

Table I: Summary of Notations

Notation	Definition
$P_{in}(t)$	Rate of the energy entering the battery
$P_{out}(t)$	Rate of the energy leaving the battery
$\hat{P}_{in}(t)$	Predicted rate of the energy to be harvested
M	Number of samples in the prediction horizon
$e(t)$	Energy stored in the battery
M_{FLOPs}	Total FLOPs for CNN model M with K $conv$ layers
M_{Memory}	Total memory for CNN model M with K $conv$ layers
M_{Energy}	Total energy for CNN model M with K $conv$ layers
C_{in}	Number of input channels
C_{out}	Number of input channels
Ω^2	Size of filters in the output layer
S_{out}	Size of feature maps in the output layer
E_{FLOP}	Energy required for a single FLOP
E_{access}	Energy required for DRAM access
P_r	Pruning ratio

A. Energy management of energy harvestable edge device

In this section, we present our energy management framework for edge-enabled IoT devices. We model a battery of edge-enabled IoT devices as a queue that stores energy packets. To represent the energy queue, we adopt the queuing dynamics model shown below. The model relies on fluid flow principles.

$$e(t) = P_{in}(t) - P_{out}(t). \tag{1}$$

where $P_{in}(t)$ denotes the rate of energy entering the battery, $e(t)$ represents the energy stored in the battery queue, and P_{out} denotes the rate of energy exiting the battery. The energy P_{out} is used for edge-enabled IoT device operation and affects the computation capability of the edge device. Our goal is to regulate the state of the above-defined queue or battery to a target level by controlling P_{out}. The control of the battery to a predetermined reference value ensures the device's continuous lifespan. In other words, P_{out} is controlled by pruning the CNN model. To regulate the battery to a predefined reference value, we use the Model Predictive control to dictate the value of P_{out} at every instant. We formulate the problem as the following optimization problem.

$$P = \max_{P_{out}(k)} \quad \min_{k=1\ldots M} \quad P_{out}(k). \tag{2}$$

subject to:

$$e(k+1) = e(k) + P_{in}(k) - P_{out}(k) \quad \text{for} \quad k = 1\ldots M. \tag{3}$$

$$P_{in}(k) = \hat{P}_{in}(k) \quad \text{for} \quad k = 1\ldots M. \tag{4}$$

$$e(min) < e(k) < e(max) \quad \text{for} \quad k = 1\ldots M. \tag{5}$$

$$P_{out}(k) > 0 \quad \text{for} \quad k = 1\ldots M. \tag{6}$$

where M in the number of samples in prediction horizon, $P_{in}(k)$ and P_{out} (k) are the corresponding discrete time variables for $P_{in}(t)$ and $P_{out}(t)$. $e(min)$ and $e(max)$ are the capacity constraints of the battery. It is to be noted that equation (3) represents the discrete time version of the battery model presented in equation (1) and the constraint in equation (4) defines the prediction of the energy to be harvested. Constraints in equation (5) are the bounds on minimum and maximum energy storage level of the battery such that charge stored in the battery can never be negative and always bounded by a maximum capacity value. Similarly, constraint in equation (6) makes sure that energy leaving the battery is always positive and battery supplies energy to the device.

B. Energy consumption of CNNs

A standard CNN is composed of multiple convolutional and fully connected layers piled up in a vanilla shape. Among all layers of a CNN, convolutional layers perform the bulk of the computations, 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-dimensional tensor: $W \in \mathbb{R}^{C\times X\times Y\times F}$, where X and Y represent the kernel's spatial dimensions, while C and F 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 dimension of U and V, then the output feature map f with the spatial dimension (x, y) can be calculated using Eq. 7:

$$T(f,x,y) = \sum_{c=1}^{C}\sum_{x'=1}^{X}\sum_{y'=1}^{Y} I(c,x-x',y-y')\cdot W(c,x',y',f). \quad (7)$$

Thereby, for a CNN model M with K convolutional layers, the FLOPs can be calculated using Eq. 8:

$$M_{FLOPs} = \sum_{k=1}^{K}[C_{in}^{k} \times (\Omega^{k})^2 \times C_{out}^{k} \times S_{out}^{k}]. \quad (8)$$

Similarly, the memory consumption for a CNN model M with K convolutional layers can be calculated using Eq. 9:

$$M_{Memory} = \sum_{k=1}^{K}[C_{in}^{k} \times (\Omega^{k})^2 \times C_{out}^{k} \times 4]. \quad (9)$$

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 accessing the data and energy consumption in executing arithmetic operations. The former is a product of energy required for DRAM access (referred as E_{access}) in a 45nm CMOS, i.e., 640pJ [23] and model memory, while the latter is a product of energy required for a single FLOP (referred as E_{FLOP}), i.e., 2.3pJ [23] and model's total FLOPs. Thus, the energy consumption for a CNN model M with K convolutional layers can be calculated using Eq. 10:

$$M_{Energy} = \sum_{k=1}^{K}[(M_{Memory} \times E_{access}) + (M_{FLOPs} \times E_{FLOP})]. \quad (10)$$

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 M_{Energy} denotes the model's total FLOPs, memory and energy. The M_{FLOPs}^{k}, M_{Memory}^{k}, and M_{Energy}^{k} denotes the k-th layer's FLOPs, memory and energy, respectively. In this regard, we describe our pruning scheme in the subsequent section concerning the M_{Energy}^{k} and M_{Energy}.

C. DRLP: Deep Reinforcement Learning based Pruning

In this section, we describe our pruning scheme based on the actor-critic DRL approach. Unlike previous approaches, where the focus has been to obtain the pruning rate for each layer using DRL, we propose pruning of a particular layer based on its energy consumption using DRL. Hence, this approach is termed energy-aware pruning because it automatically caters to the individual layer's complexity instead of manually selecting a layer as done in existing methods. We leverage the actor-critic DRL approach that learns the optimal policy through deep neural networks (DNNs) and solves the layer-pruning problem with continuous action spaces. The actor-critic approach utilizes 2 DNNs; one serves as an actor network, while the other serves as a critic network. The actor network learns to take actions based on the policy gradient method and updates the actions given the evaluation from the critic network.

1) Actor, Critic, and Replay memory: In our case, the actor DNN is responsible ~~to~~ for producing labels on its output layer, which are modeled in continuous action space. The output on each node is quantized to either 0 or 1 for the corresponding layer, and the total output nodes are equal to the K convolutional layers for a particular CNN. The actions are generated using a parametrized function $\pi(s;\vartheta)$, where ϑ denotes the weight matrix of DNN. As the objective is to maximize the potential discounted reward $J(\pi)$, thus the weight ϑ is updated according to the direction of the objective function, $\nabla_{\vartheta} J(\pi_{\vartheta})$ [34]

$$\nabla_{\vartheta} J = \frac{\partial J}{\partial \pi}\frac{\partial \pi}{\partial \vartheta} = \nabla_{\tilde{a}} Q(s,\tilde{a};\theta)\nabla_{\vartheta}\pi(s), \quad (11)$$

where $Q(s,\tilde{a};\theta)$ is produced by critic which denotes the Q value for each state and action pair, θ is the weight matrix of the critic's DNN. Following equation is used to update the parameter ϑ for actor DNN:

$$\vartheta \leftarrow \vartheta + \alpha_a \nabla_{\vartheta} J, \quad (12)$$

where α_a denotes the learning rate and is one of the hyper-parameters of the actor. Using Eq. 12, the local optimum for the objective function $J(\pi)$ can be achieved over a number of iterative parameter updates. Meanwhile, the critic DNN keeps evaluating the actor's actions by monitoring the state-action Q value function $Q(s,\tilde{a};\theta)$ that is responsible for updating the actor DNN's weight θ as shown in Eq. 11 and 12. The main objective of critic DNN is to minimize the temporal difference error $\delta(t)$ [35]

$$\delta(t) = -c(t+1) + \gamma Q(s(t+1),\tilde{a}(t+1);\theta) - Q(s(t),\tilde{a}(t);\theta), \quad (13)$$

where γ is the discount factor which defines the significance of future rewards. Similar to above equations, θ for critic DNN is updated in the direction of its temporal difference objective:

$$\theta \leftarrow \theta + \alpha_c \nabla_{\theta}\delta(t). \quad (14)$$

where α_c acts as a learning rate hyperparameter for critic DNN. Moreover, during the training of actor and critic DNNs, training samples are stored in a finite-sized first-in-first-out (FIFO) based replay memory. At each time epoch t, training samples $(s(t), a(t), c(t), s(t+1))$ are stored in this cache once the actions are made by the actor. Simultaneously, a minibatch is sampled from the replay memory at each time epoch to update weights ϑ, and θ are updated to train the DNNs of actor and critic.

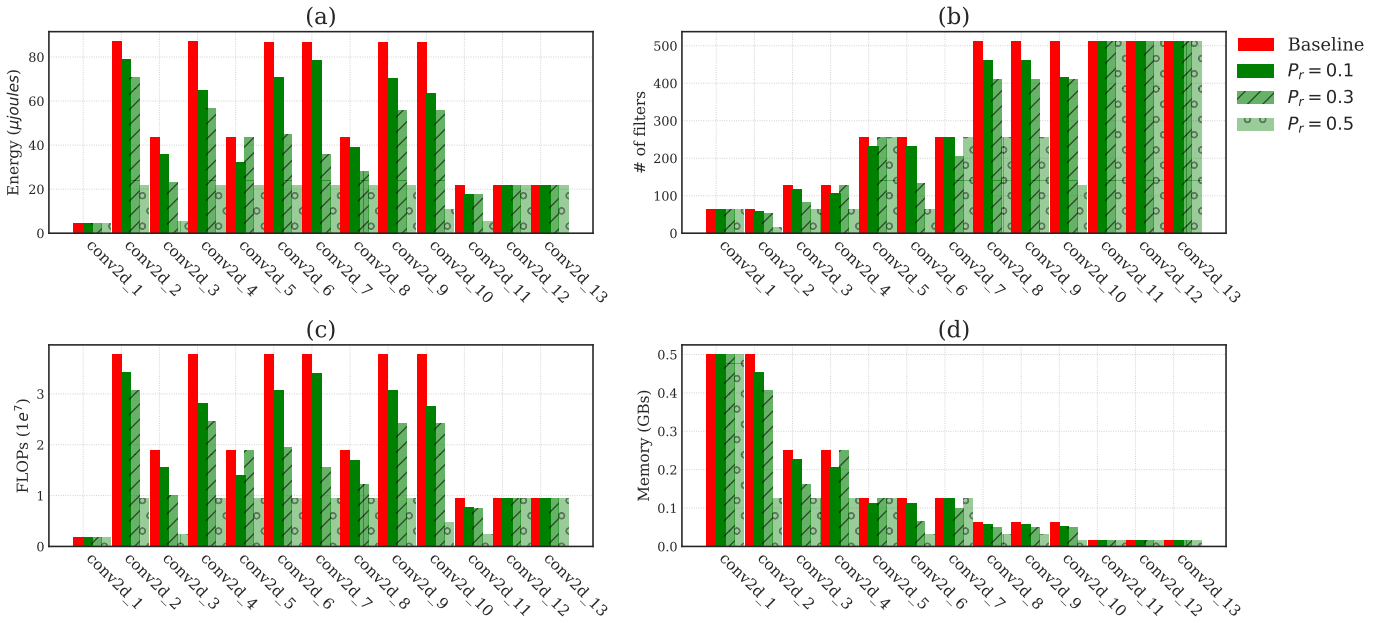

Figure 2: (a) energy consumption, (b) filters, (c) FLOPs, and (d) memory over 13 conv layers of VGG-16.

2) State, Action, and Reward: The proposed agent interacts with the environments, takes the state as an input, and produces action as an output that maximizes the reward. In our case, the environment is a CNN model. Thus, at a given time instant t, the proposed actor-critic DRL model takes state space $s(t)$ from the environment. After observing the environment at a time unit t, the DRL model takes the following set of parameters as a part of the state:

$$s(t) = \{[I_k, Y_k, Z_k, M^k_{Energy}, R_k], [I_{k+1}, Y_{k+1}, Z_{k+1}, M^{k+1}_{Energy}, R_{k+1}], ..., [I_K, Y_K, Z_K, M^K_{Energy}, R_K]\}_t. \quad (15)$$

where I_k, Y_k, and Z_k denote the index, input channels, and output channels of layer k, respectively. M^k_{Energy} is the current energy status of layer k, while R_k is the energy proportion of layer k in the overall CNN with K layers. The core idea behind providing these features of individual layers is to assist the agent in differentiating one convolutional layer from the other. Moreover, M^k_{Energy} and R_k particularly enhance the agent's ability to select a layer based on its underlying energy contribution to the model complexity.

Given the state space, the agent provides an action based on the policy defined above. The agent is required to decide whether a particular layer (k) should be selected for pruning or not in each iteration (i.e., $a_k \in \{0, 1\}$). Since there are 2^K possibilities of selection strategies, thus the number of output nodes in actor DNN can be calculated as 2^K. Thereby, the complexity of actor DNN exponentially increases with the number of layers (K) in a particular CNN. To overcome this, we model the actions in a scalar continuous action space which can be represented as $\tilde{a}_k \in [0, 1]$. Then, the selection decision for a certain layer is quantized to 0 or 1.

Finally, we model our reward function, which is inspired from [30] to evaluate the performance of the actor-critic compression policy. In [30], the reward function is synthesized in a way that it offers an incentive for not only maintaining the accuracy but also reducing the FLOPs or model size. However, the alternative reward function $reward = -Error$ aggressively compresses the model without incentivizing the complexity reduction. Moreover, it has been empirically observed that $Accuracy$ is directly proportional to the M_{Energy}. Thus, in our case, we model the reward function to incentivise both factors: $reward = -Error \cdot log(M_{Energy})$. This reward function is constrained to not only $-Error$, but also the M_{Energy}. Hence, the model is compressed, taking care of the dynamic accuracy and energy budget required for a certain IoT application.

IV. Experiments and Results

In this section, we first discuss the experimental setup and analyze the impact of the DRLP scheme on energy, FLOPs, memory, and filters in each convolutional layer for different pruning ratios. Next, we evaluate the performance of our proposed DRLP model in terms of accuracy and energy trade-offs over several iterations and pruning ratios with 3 filter pruning mechanisms, i.e., ℓ_1-norm, Taylor expansion, and random. Lastly, we compare our model with state-of-the-art pruning techniques on different datasets and models.

A. Experimental Setup

We train our actor and critic networks with a learning rate of 1×10^{-3} and batch size of 100 on approximately 5000 episodes. Both DNNs are initialized with 3 fully-connected hidden layers having sigmoid as an activation function. The input layer has a dimension equal to the number dimension of input states, while the output layer has a dimension equal to the number of layers in a CNN under observation. We perform all experiments using Keras deep learning API with Tensorflow as the backend. We pre-trained the above models on an Nvidia

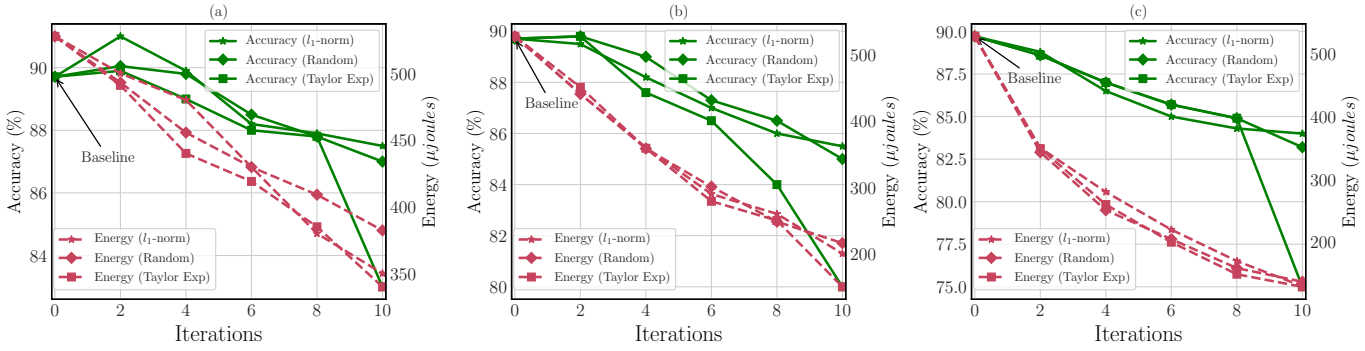

Figure 3: Energy-accuracy trade-off on 10 pruning iterations for AlexNet on CIFAR-100.

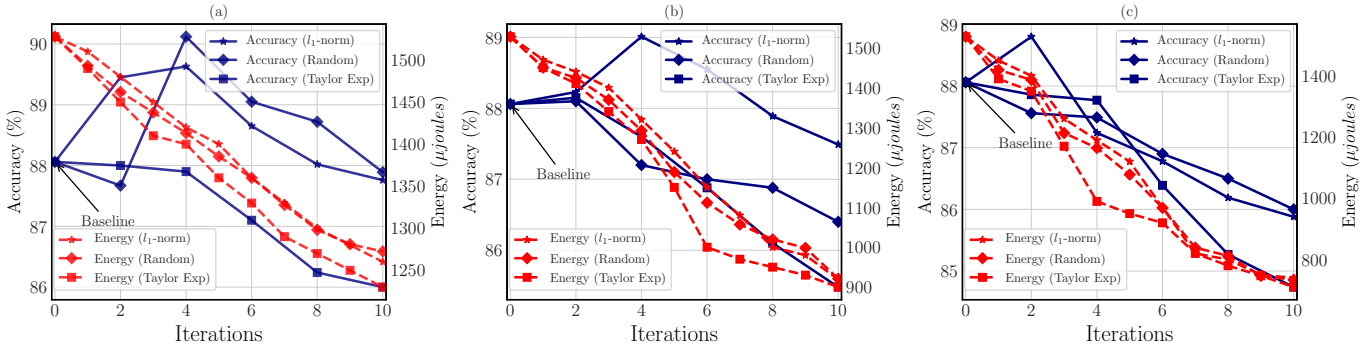

Figure 4: Energy-accuracy trade-off on 10 pruning iterations for VGG-16 on CIFAR-10.

Tesla K20 GPU machine with 8GB of memory having Ubuntu 18.04 OS with Tensorflow version 1.15, Keras version 2.2.4, and Python version 3.6.12. All models are trained with data augmentation and follow the setup described in [18]. In the following subsections, we mention some of the prominent CNN models and datasets which are considered to evaluate the proposed DRLP scheme. However, the approach can be extended to more similar CNNs and datasets.

1) DNN Models: We performed experiments on AlexNet, VGG-16, ResNet-18, and ResNet-32 which are usually considered the standard CNNs for benchmarking model compression approaches. AlexNet and VGG-16 were originally proposed for the ImageNet dataset [11], [36], but several studies have reported their promising performance on CIFAR-100 and CIFAR-10 datasets. The topology of AlexNet is composed of 5 convolutional layers, followed by 3 fully connected layers. In comparison, VGG-16 is a slightly wider and deeper model consisting of 13 convolutional and 3 fully connected layers stacked on top of each other. In both models, ReLu activation is used in all layers except the last layer, which involves softmax as an activation function and represents the number of classes in a specific dataset, e.g., 100 for CIFAR-100. We also consider ResNet-18 and ResNet-32, which are among the prominent models of the ResNet family. In general, both ResNet-18 and ResNet-32 begin with a convolutional layer followed by several parallel branches of layers along with skip connections and a fully connected layer in the end. However, ResNet-18 consists of 8 residual units, and ResNet-32 consists of 15 units, and each residual unit is made up of 2 convolutional layers.

2) Datasets: We use 2 datasets from CIFAR for the evaluation, namely CIFAR-10 and CIFAR-100, which are the standard datasets for benchmarking pruning techniques. Both datasets consist of 60,000 digital images of size 32x32, with 50,000 images as a training set and 10,000 images as a test set. In CIFAR-10, there are 10 classes, and each class has 5000 training and 1000 test images, while CIFAR-100 has 100 distinct classes, and for each class, there are 500 train images and 100 test images. In the subsequent, we present the performance evaluation of our proposed DRLP scheme.

B. Performance Evaluation

The aim of our DRLP scheme is to automatically produce a compressed model given the dynamic energy requirements of edge devices in IoT. Hence, to validate the objective of our DRL model in terms of energy-aware pruning, we evaluate it on VGG-16 on CIFAR-10, as shown in Fig. 2. We perform pruning up to 10 iterations with pruning ratios $P_r = \{0.1, 0.3, 0.5\}$. Fig. 2 (a)-(d) shows the layer-wise energy, filters, FLOPs, and memory reduction for VGG-16, respectively. The x-axis shows the indices of the convolutional layers in the corresponding CNN, while the y-axis shows the (a) energy consumption, (b) number of filters, (c) FLOP count, and (d) memory consumption for each layer.

In contrast to typical pruning methods where layers are pruned according to the proportion of filters they carry, our proposed model prunes a certain layer based on the amount of energy it consumes. More importantly, a layer with more filters

does not necessarily consume more energy. In fact, the energy consumption of each layer is constituted of not only the FLOP counts but also the memory access. Therefore, the energy required for accessing data from the DRAM for a certain operation is higher than the energy required for the operation itself. As a result, DRAM access for both feature maps and filter weights dominates the overall energy consumption of a CNN. It can be observed in Fig. 2 that a few layers consume the same amount of energy regardless of an unequal number of filters. This inconsistent distribution of energy and filters over layers is due to the impact of intermediate feature maps among layers. For instance, there is a slight reduction in the energy of $conv\_2d\_11$ despite the negligible reduction in filters. This type of scenario justifies the aforementioned arguments as energy is reduced only due to the reduction in FLOPs of the corresponding layer, which is a result of the change in feature maps from the preceding layer.

We also evaluate the DRL model with different filter importance measure techniques. The proposed model is able to work with a wide range of filter ranking techniques; however, we evaluate it on following 3 state-of-the-art procedures.

ℓ_1-norm [16]. In this method, the relative importance of a certain filter F in each layer is measured by summing its absolute weights $\sum \mid F \mid$ which results in its ℓ_1-norm ($\|F\|_1$). In the next step, filters are sorted according to ℓ_1-norm value, and a percentage of filters with the smallest value are pruned.

Taylor Expansion [17]. For a training dataset D and loss function C, this method aims to prune a model in such a way that the change in loss $\mid \Delta C \mid$ is minimal once the model has maximum non-zero feature maps. The $\mid \Delta C \mid$ after eliminating $h_{i,j}$ feature maps (i.e. obtained from parameters i, j) can be calculated using Eq. 16

$$\mid \Delta C(h_{i,j}) \mid = \mid C(D, h_{i,j} = 0) - C(D, h_{i,j}) \mid, \qquad (16)$$

where $C(D, h_{i,j})$ and $C(D, h_{i,j} = 0)$ are losses with and without considering feature maps $h_{i,j}$, respectively. The loss with $h_{i,j}$ included can be calculated with

$$C(D, h_{i,j} = 0) = C(D, h_{i,j}) - \frac{\delta C}{\delta h_{i,j}} h_{i,j} + R_1(h_{i,j} = 0),$$

where $R_1(h_{i,j} = 0)$ is the first-degree remainder of Taylor expansion. We can ignore R_1, since this method considers only the first degree estimation, thus, Eq. 16 can be transformed as:

$$\mid \Delta C(h_{i,j}) \mid = \mid C(D, h_{i,j}) - \frac{\delta C}{\delta h_{i,j}} h_{i,j} - C(D, h_{i,j}) \mid = \mid \frac{\delta C}{\delta h_{i,j}} h_{i,j}$$

where $\mid \Delta C(h_{i,j}) \mid$ denotes the importance of filter $w_{i,j}$ associated with feature map $h_{i,j}$, and both $\frac{\delta C}{\delta h_{i,j}}$ and δC can be obtained during the feed-forward and back-propagation of CNN by providing sufficient samples.

Random Pruning [29]. In this approach, filters are neither ranked nor hold any kind of importance metric to get eliminated from the layers. However, as opposed to ranking-based techniques, this method encourages randomly pruning a required percentage of filters from any layer without additional calculations.

We evaluate the effectiveness of aforementioned approaches on our compression scheme by pruning the AlexNet and VGG-16 up to 10 iterations with pruning ratios $P_r = \{0.1, 0.3, 0.5\}$ as shown in Fig. 3 and 4 respectively. For Fig. 3(a), after soft pruning with $P_r = 0.1$, there is 33.6%, 27.5%, and 35.5% energy reduction and 2.4%, 3%, and 7.4% accuracy loss using ℓ_1-norm, random, and Taylor expansion approaches, respectively. Similarly, in Fig. 3 (c), with a relatively higher pruning ratio $P_r = 0.5$, we observe 75.3%, 74%, and 75.5% energy reduction and 6.3%, 7.2%, and 16.3% accuracy loss using ℓ_1-norm, random, and Taylor expansion approaches, respectively. This behavior is consistent for the VGG-16 as well, which can be seen in Fig. 4 (a)-(c). For the proposed scenario, it is obvious that ℓ_1-norm and random schemes are efficient, and Taylor expansion failed to minimize the accuracy loss. However, ℓ_1-norm involves some calculations that essentially increase the compression scheme's overall computation time. In contrast, randomly pruning does not incur additional computations but still performs consistently with ℓ_1-norm. Therefore, it can be concluded that the proposed DRL model should incorporate either random pruning or ℓ_1-norm (if latency can be tolerated) to efficiently trade-off between energy and accuracy.

C. Comparison with state-of-the-art

We compare our approach with state-of-the-art pruning approaches on 2 types of networks; simple CNNs (AlexNet on CIFAR-100 and VGG-16 on CIFAR-10) and Residual networks (ResNet-18 on CIFAR-10 and ResNet-32 on CIFAR-100). We use the same environment, pre-processing, and hyper-parameter tuning to get the baseline accuracy and energy. To reproduce the [6], [18], [19], we follow the given guidelines and prune the model without fine-tuning it to synchronize with the proposed scheme. Thus, the accuracy can be different from the one mentioned in the original papers, as they all fine-tune the networks once they are pruned, which is not the case in our scenario.

1) VGG-16 on CIFAR-10: As shown in Fig. 5 (a), the baseline accuracy for VGG is 88.06%, and the original model consumes approximately 1520 $\mu joules$. Our approach achieves the best tradeoff between loss in accuracy and energy reduction with ℓ_1-norm and $P_r = 0.3$ by reducing 40% energy and losing only 0.6% accuracy. However, [6], [18] reduce relatively more energy but fail to gain any improvement in accuracy. Moreover, [19] loses 1.2% accuracy, which is relatively lesser than [6], [18] but can only reduce 11% energy.

2) AlexNet on CIFAR-100: The baseline accuracy for AlexNet is 89.71%, and the uncompressed model consumes approximately 527 $\mu joules$ as shown in Fig. 6 (a). [6] is able to reduce 72% of model energy aggressively as compared to [18], [19], but in return, it loses 6.5% of accuracy. In contrast, our approach with ℓ_1-norm and $P_r = 0.5$ is able to reduce 75.3% of energy with only 6.3% of accuracy drop. Similarly, with random pruning and $P_r = 0.1$, our approach performs better than [19] by reducing 27.5% of energy with a minimal drop of 3% in accuracy.

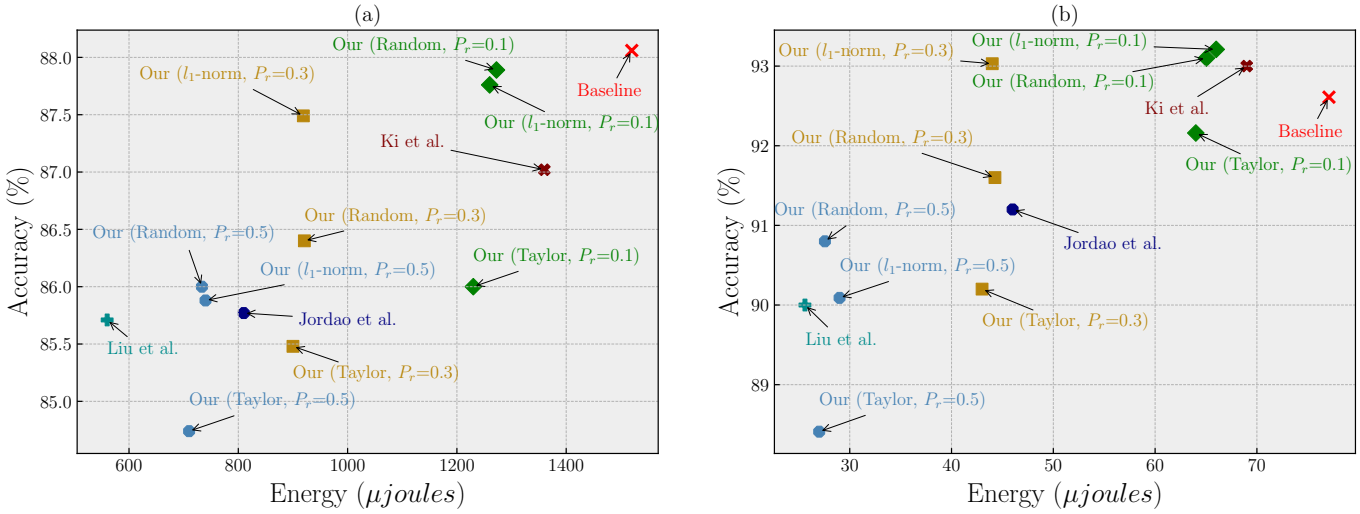

Figure 5: Comparison of DRLP with state-of-the-art on (a) VGG-16-CIFAR-10, and (b) ResNet-18-CIFAR-10.

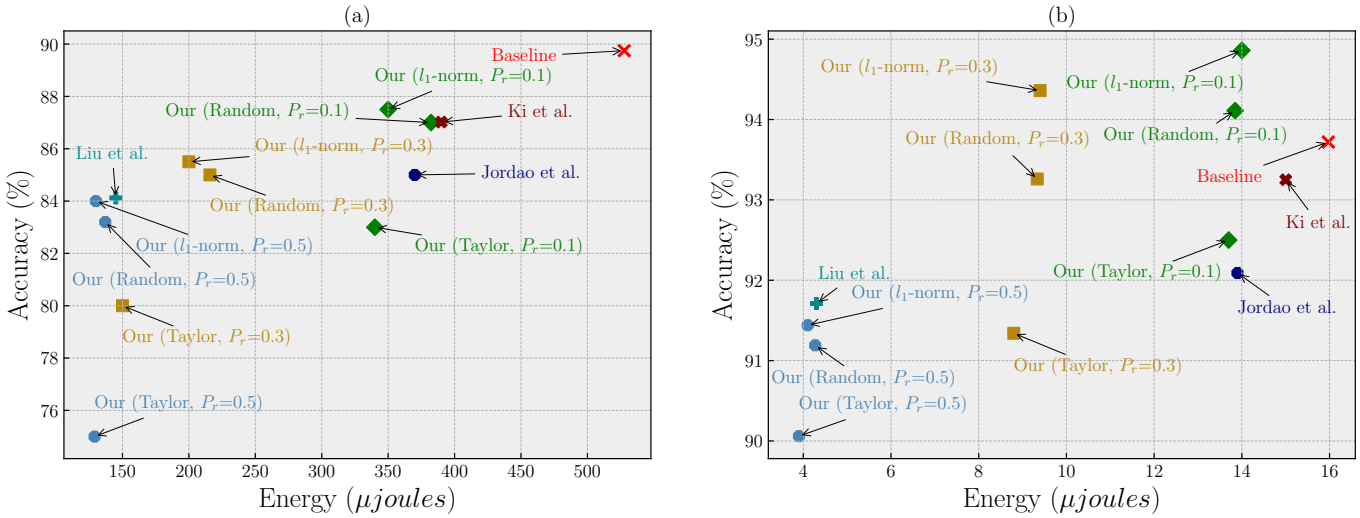

Figure 6: Comparison of DRLP with state-of-the-art on (a) AlexNet-CIFAR-100, and (b) ResNet-32-CIFAR-100.

3) ResNets on CIFAR-10 and CIFAR-100: We choose two well-known models from the ResNet family, i.e., ResNet-18 and ResNet-32, trained on CIFAR-10 and CIFAR-100, respectively. In general, pruning ResNets is challenging due to their architecture comprising several parallel branches and skip connections. Therefore, when dealing with such complex architectures, there are some requirements to adhere to as a result of inter-layer dependencies.

For this reason, our DRL model does not select any layer with shortcut connections for pruning in order to maintain the structure in ResNets. The baseline ResNet-18 model has 92.61% accuracy and it consumes approximately 77 $\mu joules$ as shown in Fig. 5 (b). For ResNet-18, our approach achieves best trade-off by gaining 0.45% of accuracy and reducing 43% of energy using ℓ_1-norm with P_r = 0.3. On the other hand, among state-of-the-art approaches, only [19] is able to gain 0.4% of accuracy but with minimal energy reduction. Moreover, the original ResNet-32 model has 93.72% accuracy and consumes approximately 16 $\mu joules$ of energy, as shown in Fig. 6 (b).

Similarly, for ResNet-32, the best trade-off is achieved by our model with 0.7% accuracy gain and 40% energy loss using ℓ_1-norm with P_r = 0.3. In contrast, none of the state-of-the-art approaches could gain accuracy after reducing the energy. However, [6] is able to reduce the significant energy for both ResNet-18 and ResNet-32 but slightly lower than ours with random pruning and P_r = 0.5. It is clear that at a certain stage, our approach loses more accuracy by trading off the energy consumption, which is consistent with state-of-the-art studies. Therefore, as opposed to state-of-the-art, our approach can be used with different ranking criteria and pruning ratios to achieve the desired level of accuracy and complexity. For example, based on the above discussion, if a certain application can not tolerate a significant loss in accuracy but needs a reduction in energy, then either ℓ_1-norm or random criteria can be used with a small P_r such as 0.1 or 0.2. Alternatively,

suppose the requirement of the application is to lower the energy footprint without taking care of accuracy. In that case, the aggressive pruning can be done using relatively larger P_r such as 0.4 or 0.5 with suitable ranking criteria such as ℓ_1-norm/random in our case.

D. Performance Evaluation of Proposed DRLP Scheme in Presence of EMS

In this section, we evaluate the performance of the proposed DRLP in the presence of our energy management scheme. In particular, we demonstrate that our proposed energy-aware DRLP algorithm is efficient in adjusting the CNN model based on the time-varying energy constraints which arise due to the intermittent nature of harvestable energy. As discussed in the previous sections, for energy management, we model the battery of the IoT device as a linear buffer, which stores packets of the harvested energy. For the simulations, we are using realistic data for solar irradiance [7]. We use the Markov chain to predict the amount of energy to be harvested [7]. Fig. 7 depicts the energy predictions for 48 hours, where 0 *hour* represents midnight. Based on these energy predictions, our MPC-based energy management controller regulates the output energy of the battery (P_{out}), which is used for the operation of IoT devices, including the CNN model inferences. Fig. 7 shows the time evolution of P_{out}, which was dictated by our energy management scheme and the corresponding residual energy of the battery. It can be observed that based on the harvestable energy predictions; our scheme is able to regulate the P_{out} in such a way that residual energy (State of charge: SOC) in the battery is always above a predefined reference value even for the time instants when there is no harvestable energy available. This reference level was set to $2J$ in our simulations. Moreover, P_{out} is always greater than zero, as shown in the magnifying window of Fig. 7. This P_{out} is the amount of energy that is utilized for CNN-based inferences. As shown in Fig. 7, the P_{out} is time-varying in nature, which will affect the performance of the CNN model while performing the inference tasks. More precisely, for the particular time instance where P_{out} is below the energy required for the CNN model, it will fail to perform the given inference tasks. Therefore, to overcome this, our proposed energy-aware DRLP scheme is able to compress the model so it can adapt according to the time-varying nature of P_{out} at the cost of accuracy. To demonstrate the effectiveness of the proposed approach, we use the VGG-16 as the baseline CNN model, which is used to perform the inference tasks. The DRLP scheme prunes the VGG-16 model iteratively at every time instance until the energy required for a certain number of inferences is less than P_{out}. In Fig. 8, we show the trade-off between energy and accuracy of VGG-16 over the 10, 50, and 100 inference tasks. It can be seen that the energy required for performing 10 inference tasks by VGG-16 does not exceed the P_{out}; hence the accuracy remains unaffected. In contrast, the energy required for 50 and 100 inference tasks exceeds the P_{out} at time instances where the harvested energy is low. As a result, the number of inference tasks can be reduced at time instances where P_{out} is low and can be increased at time instances where P_{out} is high.

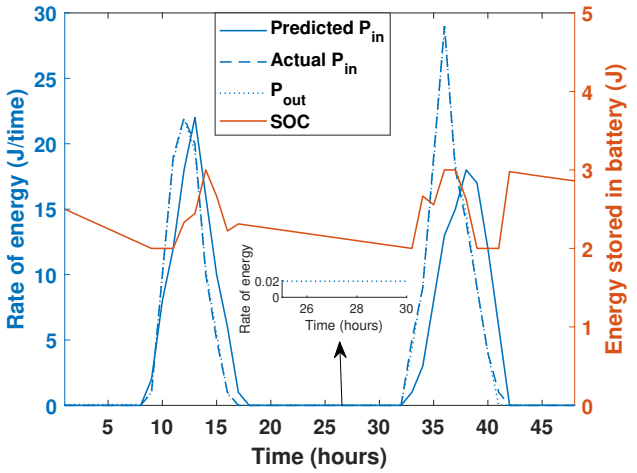

Figure 7: Performance of energy management scheme

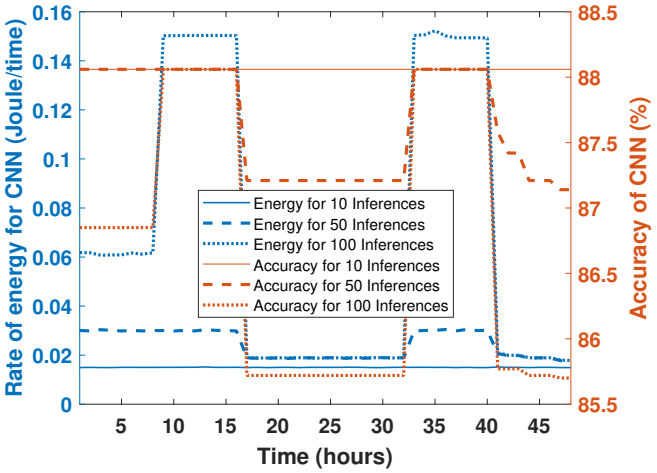

Figure 8: Energy-accuracy trade-off for VGG-16 over 10, 50 and 100 inferences based on P_{out}

V. CONCLUSION

Edge computing is emerging as a prominent technique for several IoT applications. However, edge devices that are responsible for the execution of tasks are usually resource-constrained, hence cannot afford the deployment of raw CNN models. Therefore, this motivates a need for compressing the CNNs before deployment. The existing techniques to reduce CNNs complexity work on strict resource requirements and are not suitable in environments with intermittent energy supply. In this work, we considered an energy-aware AI-driven framework for edge-computing based IoT applications. The proposed DRLP scheme can adapt according to the availability of energy. It can compress the model on the device dynamically to provide the best energy-accuracy trade-off. Our results showed that the proposed DRLP outranks other state-of-the-art approaches on model pruning in terms of energy-accuracy trade-off. In the future, we intend to consider the combined performance of our proposed scheme in the presence of an EMS.

ACKNOWLEDGMENT

This research was supported by Science Foundation Ireland and the Department of Agriculture, Food and Marine on behalf of the Government of Ireland VistaMilk research centre under the grant 16/RC/3835.

REFERENCES

[1] S. Liao, J. Wu, J. Li, and K. Konstantin, "Information-centric massive iot-based ubiquitous connected vr/ar in 6g: A proposed caching consensus approach," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5172–5184, 2021.

[2] M. Wang, L. Zhu, L. T. Yang, M. Lin, X. Deng, and L. Yi, "Offloading-assisted energy-balanced iot edge node relocation for confident information coverage," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4482–4490, 2019.

[3] J. Chen, W. Wang, Y. Zhou, S. H. Ahmed, and W. Wei, "Exploiting 5g and blockchain for medical applications of drones," *IEEE Network*, vol. 35, no. 1, pp. 30–36, 2021.

[4] H. Sami, H. Otrok, J. Bentahar, and A. Mourad, "Ai-based resource provisioning of ioe services in 6g: A deep reinforcement learning approach," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3527–3540, 2021.

[5] M. Zawish, N. Ashraf, R. I. Ansari, S. Davy, H. K. Qureshi, N. Aslam, and S. A. Hassan, "Towards on-device ai and blockchain for 6g enabled agricultural supply-chain management," *arXiv preprint arXiv:2203.06465*, 2022.

[6] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[7] N. Ashraf, M. Faizan, W. Asif, H. K. Qureshi, A. Iqbal, and M. Lestas, "Energy management in harvesting enabled sensing nodes: Prediction and control," *Journal of Network and Computer Applications*, vol. 132, pp. 104–117, 2019.

[8] N. Ashraf, A. Hasan, H. K. Qureshi, and M. Lestas, "Combined data rate and energy management in harvesting enabled tactile iot sensing devices," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3006–3015, 2019.

[9] O. Salem, K. Alsubhi, A. Shaafi, M. Gheryani, A. Mehaoua, and R. Boutaba, "Man in the middle attack mitigation in internet of medical things," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

[10] X. Yuan, J. Chen, K. Zhang, Y. Wu, and T. Yang, "A stable ai-based binary and multiple class heart disease prediction model for iomt," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2021.

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

[12] 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.

[13] 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.

[14] F. A. Dharejo, M. Zawish, Y. Zhou, S. Davy, K. Dev, S. A. Khowaja, Y. Fu, and N. M. F. Qureshi, "Fuzzyact: A fuzzy-based framework for temporal activity recognition in iot applications using rnn and 3d-dwt," *IEEE Transactions on Fuzzy Systems*, 2022.

[15] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

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

[17] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings*, 2019.

[18] A. Jordao, F. Yamada, and W. R. Schwartz, "Deep network compression based on partial least squares," *Neurocomputing*, vol. 406, pp. 234–243, 2020.

[19] S.-K. Yeom, P. Seegerer, S. Lapuschkin, A. Binder, S. Wiedemann, K.-R. Müller, and W. Samek, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognition*, vol. 115, p. 107899, 2021.

[20] M. Zawish, S. Davy, and L. Abraham, "Complexity-driven cnn compression for resource-constrained edge ai," *arXiv preprint arXiv:2208.12816*, 2022.

[21] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[22] 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.

[23] 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.

[24] H. Zhan, W.-M. Lin, and Y. Cao, "Deep model compression via two-stage deep reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 238–254.

[25] N. Ashraf, N. Christofides, and M. Lestas, "Combined data rate and energy management in wireless sensor networks with energy harvesting capability," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6717–6722.

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

[27] A. Polyak and L. Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015.

[28] 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.

[29] 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.

[30] 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.

[31] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Blockqnn: Efficient block-wise neural network architecture generation," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[32] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[33] N. Ashraf, A. Hasan, H. K. Qureshi, and M. Lestas, "Combined data rate and energy management in harvesting enabled tactile iot sensing devices," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3006–3015, 2019.

[34] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[35] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE transactions on automatic control*, vol. 42, no. 5, pp. 674–690, 1997.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.