Performance-aware NILM model optimization for edge deployment

Stavros Sykiotis, Sotirios Athanasoulias, Maria Kaselimi, Anastasios Doulamis, *Member, IEEE,* Nikolaos Doulamis, *Member, IEEE,* Lina Stankovic, *Senior Member, IEEE,* Vladimir Stankovic *Senior Member, IEEE*

Abstract

Non-Intrusive Load Monitoring (NILM) describes the extraction of the individual consumption pattern of a domestic appliance from the aggregated household consumption. Nowadays, the NILM research focus is shifted towards practical NILM applications, such as edge deployment, to accelerate the transition towards a greener energy future. NILM applications at the edge eliminate privacy concerns and data transmission-related problems. However, edge resource restrictions pose additional challenges to NILM. NILM approaches are usually not designed to run on edge devices with limited computational capacity and therefore model optimization is required for better resource management. Recent works have started investigating NILM model optimization, but they utilize compression approaches arbitrarily, without considering the trade-off between model performance and computational cost. In this work, we present a NILM model optimization framework for edge deployment. The proposed edge optimization engine optimizes a NILM model for edge deployment depending on the edge device's limitations and includes a novel performance-aware algorithm to reduce the model's computational complexity. We validate our methodology on three edge application scenarios for four domestic appliances and four model architectures. Experimental results demonstrate that the proposed optimization approach can lead up to 36.3% average reduction of model computational complexity and 75% reduction of storage requirements.

Index Terms

Edge Inference, Non-Intrusive Load Monitoring, Quantization, Pruning, Optimization, Resource Management, Green Computing

I. Introduction

Non-Intrusive Load Monitoring (NILM) refers to the process of analyzing the aggregated energy consumption of a residential building to infer the individual consumption pattern of domestic appliances [1]. In recent years, NILM approaches have transversed from statistical analysis methods to deep learning techniques due to their superior performance capabilities. However, most deep learning NILM approaches are designed to be deployed in a central server, instead of performing inference on the edge, due to the increased computational needs [2]–[4]. This design methodology assumes data transfer from the data source, i.e. the domestic house, to an external entity, and impacts the wider deployment scalability of NILM frameworks. Central data storage has increased costs for the service provider, since the accumulation of large amounts of data requires an expanded storage infrastructure. In addition, performing inference centrally usually requires more computational resources, thus increasing the energy required to run the service and increasing the carbon footprint of the solution. Finally, apart from the heavy reliance on a stable internet connection for data transmission, privacy concerns arise, since sensitive customer information can be inferred [5]. It can therefore be argued that a transition to deploying NILM algorithms on the edge (i.e. at each domestic house equipped with a smart meter and a device with restricted processing power) is a more attractive solution that alleviates the issues of central data processing.

A. Our Contribution

In this study, we propose a performance-aware NILM optimization framework for edge deployment that takes into account the edge device characteristics. Our approach considers multiple hardware limitations and, depending on the deployment scenario, employs a different model optimization technique to efficiently preserve the limited edge device resources, resulting in an efficient resource management scheme. The basic contributions of our work are summarized below:

- **NILM green computing edge-inference framework**: We propose an edge inference framework for NILM, that utilizes multiple model optimization techniques, taking into account the edge device hardware characteristics to enable an efficient edge green computing scheme.
- **Model optimization metric**: We introduce Pruning Gain, an objective model optimization metric for NILM algorithms that describes the trade-off between model performance and computational complexity.
- **Performance-aware NILM model edge optimization**: We present performance-aware pruning, an iterative algorithm to determine which model parameters can be removed from the network without severely impacting model performance.
- **Application specific NILM model edge optimization**: We explore the impact of model optimization on various NILM techniques (CNN, LSTM, Transformers) for different appliances and we experimentally prove that, depending on the application scenario, a different level of model optimization for resource management is tolerable from a model performance perspective.

The rest of the paper is organized as follows. In Section II, we present an overview of the existing work for deploying NILM algorithms on edge devices. Section III mathematically formulates the problem of performance-aware NILM model optimization, whereas Section IV describes in detail the proposed NILM edge optimization framework. Finally, Section V presents the experimental setup and results, while Section VI summarizes the main outcomes of the paper and potential future steps.

II. Related Work

Since its official problem formutation [6], NILM has received increasing research interest, backed by the expanded availability of smart meter data. Earlier NILM approaches were based on signal processing techniques, such as Graph Signal Processing [7], [8] and Hidden Markov Models [9], [10]. Since 2015, the research focus has shifted towards utilizing deep learning techniques on low-frequency data, with models such as Denoising Autoencoders [11], [12], Recurrent Neural Networks [13], [14] and Convolutional Neural Networks [15]–[17] being successfully applied for energy disaggregation. Due to the rapid advancements in deep learning, state-of-the-art network architectures such as U-net [18], Generative Adversarial Networks [19], [20] and Transformers [21], [22] have been employed to advance NILM.

Recently, progress has been made towards the deployment of NILM and other energy-related applications on edge devices, either as part of a Home Energy Management System (HEMS) [23] or as standalone applications [24]. NILM edge inference does not require the transmission of data to an external server, and, therefore, alleviates the aforementioned issues of central data processing. Approaches to deploy NILM models on the edge have been proposed, both on embedded computers, such as Raspberry Pi, and on more resource-constrained devices. Deployment on a Raspberry Pi has been proposed [25], [26], but the deployed models either require additional metadata, such as room occupancy, or utilize high-frequency features for energy disaggregation, which increases data acquisition costs. In addition, NILM models on more resource-constrained devices, such as microcontrollers [27], [28] and FPGA [29] have been also been proposed, but the respective models only consider appliance state classification instead of regression and require high frequency data to operate.

Despite the great success of deep learning in diverse applications, neural networks often possess a vast number of parameters, leading to significant challenges in deploying deep learning systems to a resource limited device [30], [31]. The deployment of sensing devices with higher computational power has been investigated [32], [33], but the devices have high cost and high power demands, thus making them impractical for commercialization [27]. Therefore, edge inference requires compression and optimization of NILM deep learning models, to account for the limited computational resources. Quantization, parameter pruning, low-rank factorization and knowledge distillation [34], as well as combinations of one or more techniques [35] are the main approaches employed in the literature. Even though NILM-related deep learning applications have utilized state-of-the-art architectures [22], [36], [37], research on the constraints and methodology for deploying NILM deep learning models on edge devices remains limited. In [38], the quantization of a sequence-to-point (seq2point) convolutional neural network (CNN) [17] from 32-bit float model weights to 8-bit integer weights is applied. The application of multiple pruning approaches on the same seq2point [17] model has also been investigated [39], and the methods have been tested on 2 appliances from the REFIT [40] dataset. Finally, [41] explores model compression of a multi-class seq2point CNN using pruning and tensor decomposition, while the evaluation is performed for 3 appliances from the REDD dataset [42].

Even though the aforementioned works can be considered as an initial entry-point towards low-frequency ($\leq$ 1Hz) NILM inference on edge devices, there are several limitations. First, these papers [38], [39], [41] do not take into consideration the hardware characteristics of edge devices. This can be an issue in quantization approaches, where some quantization protocols are applicable only to specific chip architectures. Second, all papers employ compression approaches on a specific model architecture (seq2point CNN). Seq2point models are less computationally efficient than sequence-to-sequence models (seq2seq), since they produce only one timepoint prediction instead of a whole window in the testing phase. As a result, significantly more forward pass iterations are required to produce the same number of outputs seq2seq models, which leads to a noteworthy increase in energy consumption. In addition, the effects of model compression on different model architectures, such as recurrent neural networks or Transformers have not been investigated. Furthermore, the works that investigate more than one model compression strategy do not explore the impact of their combination on model performance, which can results in the optimization of different model aspects. Finally, pruning is applied on an arbitrary basis and no framework has been proposed to interconnect performance loss after compression with model complexity. A summary of the aforementioned limitations of existing literature can be found in Table I.

TABLE I
SUMMARY OF EXISTING LITERATURE FOR EDGE NILM

Work	**Deployment Device**	**Model**	**Compression Approach**	**Limitations**
Uttama et. al. [25]	Raspberry Pi	Combinatorial optimization	state complexity reduction	requires additional metadata (room occupancy information)
Xu et. al. [26]	Raspberry Pi	Support Vector Machine	Feature reduction	requires high frequency data
Tabanelli et. al. [27]	Microcontroller	Random Forest	Feature reduction	-Requires high frequency data -event-based NILM (classification)
Tabanelli et. al. [28]	Microcontroller	Random Forest	Feature reduction	
Hernandez et. al. [29]	FPGA	Hardware-oriented	-	
Ahmed et.al. [38]	-	seq2point CNN	Quantization	- Hardware characteristics are not taken into account -Seq2point requires more forward passes than seq2seq; computationally intensive - Only 1 model architecture is considered - Different model compression approaches are not jointly investigated - Compression is conducted arbitrarily and not connected to performance loss
Barber et.al.[39]	-	seq2point CNN	Pruning	
Kukunuri et.al.[41]	-	seq2point CNN	Pruning Tensor decomposition	

III. Problem Statement

Under a NILM framework [6], we can assume that the aggregate consumption signal x of a domestic house with M operational appliances, at any time point t, equals to the sum of the individual appliance consumption loads $y_i, i = 1...M$, plus a noise term ϵ [43]:

$$x(t) = \sum_{i=1}^{M} y_i(t) + \epsilon(t) \tag{1}$$

To extract the consumption signal of a selected appliance $a \in \{1, ..., M\}$, NILM approaches are designed to filter out all non-relevant appliance consumption signals $y_i \ \forall i \neq a$. Depending on the chosen appliance a, the power signal y_a may showcase different statistical characteristics in terms of peaks, sparsity, as well as duration of appliance activations, which is defined as the consecutive time interval that the appliance is turned on. As a result, not only may different model architectures have different sensitivity to model optimization approaches, but also the same model, trained to disaggregate different appliances, may showcase different behavior related to compression.

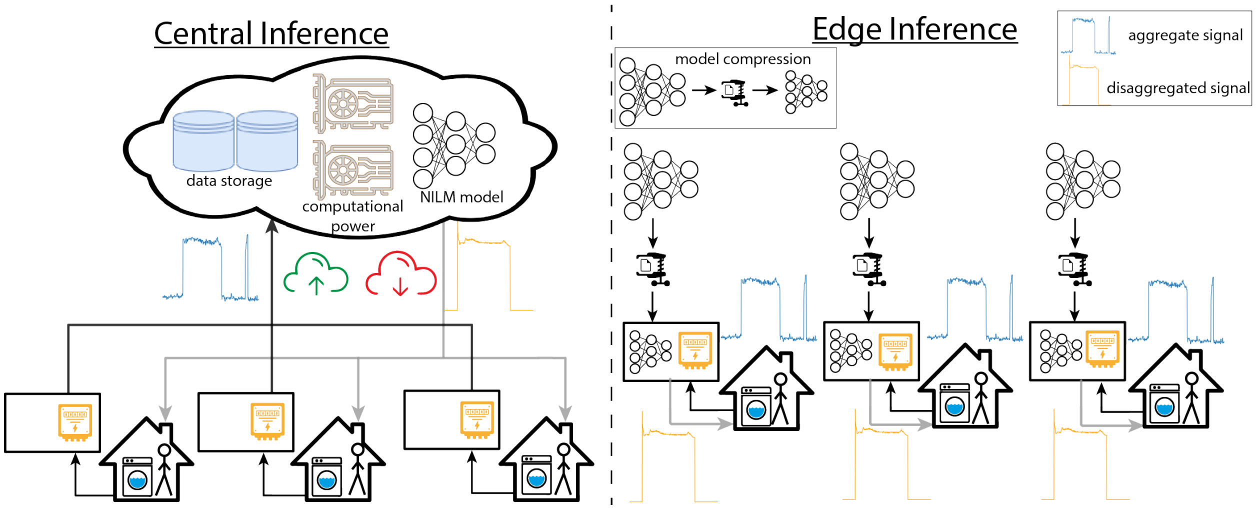


Fig. 1. Overview of the required infrastructure setup to perform inference centrally vs on the edge. Central inference requires upload and download of consumption data to a central processing entity, as well as increased data storage capacity and computational power. On the contrary, performing inference on the edge devices alleviates these limitations and only requires the compression of the models and their deployment on the edge device, while data exchange takes place only between the edge device and the domestic house.

It can therefore be argued that the optimization strategy must be bound to model performance, with the goal of finding an equilibrium between model complexity and performance loss.

Two different NILM infrastructure setups for central and edge device deployment are illustrated in Figure 1. Even though performing inference centrally theoretically allows for the model to utilize larger amounts of computational power, it can be easily seen that the complexity and drawbacks of such an approach are significant. On the other hand, performing inference on edge devices alleviates the need to transmit and receive data to an external server, with the only limitation being the fact that the models need to be compressed and optimized to run on resource constrained devices.

To mathematically formulate the aforementioned approach, let a NILM model $f(x;w) : \mathcal{X} \rightarrow \mathcal{Y}, w \in \mathcal{R}^N$ have a performance P_f. Our goal is to obtain a lower-dimensionality model $h(x;\theta) : \mathcal{X} \rightarrow \mathcal{Y}$, where θ is some transformation of w, i.e. $\theta = \mathcal{T}(w), \theta \in \mathcal{R}^C, C < N$, to perform the same task with performance P_h. In other words, we are trying to minimize the following function:

$$\mathcal{L} = min(|f(x;w) - h(x;\theta)|) \tag{2}$$

However, Equation 2 does not take into consideration the performance loss that occurs as a consequence of model optimization. The model should be optimized to match the deployment criteria only to the level that the performance loss is acceptable. Therefore, Equation 2 needs to be constrained with the condition that performance loss must not fall below a tolerance threshold δ. Therefore, a performance-aware model compression framework can be written as:

$$\mathcal{L}_{pa} = min(|f(x;w) - h(x;\theta)|) \;\; s.t. \;\; P_f - P_h < \delta \tag{3}$$

IV. A GREEN EDGE RESOURCE MANAGEMENT FRAMEWORK FOR NILM

Our proposed green computing framework for NILM model edge optimization is illustrated in Figure 2. The backbone of our approach is the edge optimization engine, which is responsible for the optimization of a NILM model depending on the edge deployment requirements. Since resource limitations of the edge device may vary, the optimization engine first receives the edge device characteristics, as well as any additional restrictions imposed by the user. Then, the trained NILM model to be deployed is analyzed and an optimization strategy is set. The optimization strategy can either be static to reduce the model's storage requirements through model quantization, or performance-aware to apply complexity reduction trough weight pruning. Performance-aware optimization is defined as the removal of insignificant model weights not arbitrarily, but by taking into consideration the respective impact on model performance. In this case, complexity reduction is performed incrementally, until the the edge deployment requirements are met, under the condition that the trade-off between performance loss and complexity reduction is satisfactory.

An overview of the optimization approaches employed for memory and complexity reduction is depicted in Figure 3. The following sections provide a detailed description of these techniques, as well as the proposed performance-aware iterative complexity reduction scheme.

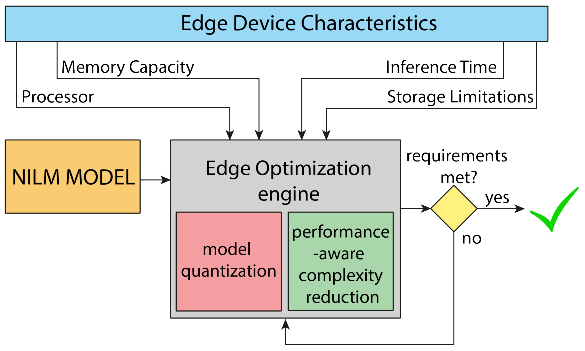


Fig. 2. High-level overview of the proposed NILM edge optimization framework

A. Model weights quantization

Model quantization refers to the process where the model's weight type is changed to a lower numerical precision to limit the storage and memory space required for the model. In essence, quantization can be formulated as a irreversible mapping function $\mathcal{Q} : w \in \mathcal{R}^N \rightarrow w' \in \mathcal{R}^Z$, $Z \subseteq N$ that maps the model weights w, stored in a floating point format, to an integer representation w'. The value range of w is divided into bins and each value w_i is mapped to the integer representing the corresponding bin.

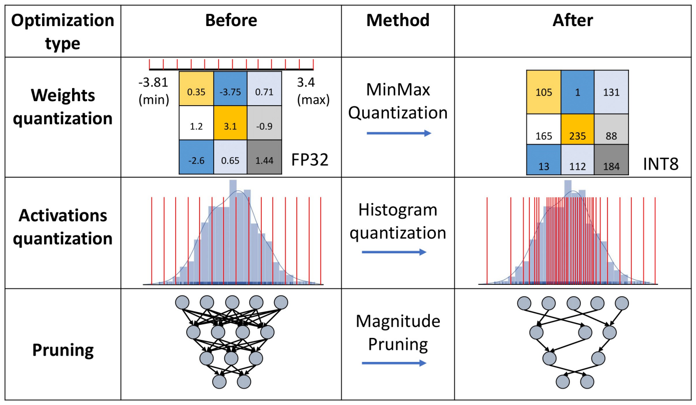


Fig. 3. Overview of the model optimization methods adopted in this study. We explore model quantization by performing MinMax quantization of the model weights, as well as histogram quantization for the activation function outputs to minimize performance loss. We also integrate magnitude pruning in our approach to remove weights with small L1-norm that contribute minimally to the model's predictions.

Quantization is either executed post-training, meaning that an already trained model is compressed, or during training, in the sense that the quantized version of the model is taken into account when the model is trained (quantization-aware training). In this work, we focus on post-training model quantization, and apply a calibration phase on an indicative dataset, during which the quantization parameters are fine-tuned, resulting in a more accurate representation of the initial model weights w. This additional calibration step also allows for the quantization of activation function outputs.

To quantize the models, we quantize both model weights and activation outputs to further avoid floating point multiplication operations [44]. Since the activation outputs are fed to the next layer, a more sensitive quantization approach is required to minimize model performance degradation. Therefore, we have opted for min-max uniform quantization for model weights and histogram quantization for the activation outputs, where the activation values are recorded and a different range per bin is assigned, depending on the corresponding probability distribution. An illustration of the different quantization approaches can be seen in Figure 3.

B. *Model complexity reduction*

An alternative methodology for optimizing a deep learning model is the removal of synaptic connections between model layers. This process, which is commonly referred to in the literature as model pruning, assumes that a deep learning network is over-parameterized and incorporates a subnetwork that contains most of the information [45]. In other words, model pruning is an approach to transform a model's weights $w \in \mathcal{R}^N$ to a lower dimensionality representation $w^{'} \in \mathcal{R}^M, M < N$ by removing non-informative model connections.

Different techniques on how to optimally remove model connections with minimal information loss have been proposed in the literature. Similar to quantization, pruning can either be applied post-training [46] or in a compression-aware training scheme [47]. The removal of weights is performed either on the overall set of model weights, or by eliminating predetermined architectural blocks, such as convolutional filters [48]. In addition, different pruning approaches remove weights by evaluating different metrics, such weights magnitude, gradients magnitude, intra-layer mutual information, or even by introducing a learnable pruning threshold [49]–[51].

In this work, we implement magnitude pruning, and remove the model connections with the smallest contribution to the model output. Let $w = \{w_i \ \forall i = 1, ..., N\} \in \mathcal{R}^N$ be a vector containing all model parameters. Then the magnitude-pruned vector w' is expressed in Equation 4:

$$w' \subset w, w' := \left\{ w_i \in w : \frac{F(\|w_i\|)}{\sum_{j=1}^{N} F(\|w_j\|)} < p_{thres} \right\} \tag{4}$$

$F(\|w_i\|)$ signifies the cumulative distribution function of weight magnitudes. In other words, after magnitude pruning we only keep the weights with the highest $1 - p_{thres}\%$ magnitudes and discard the rest. Even though magnitude pruning is usually executed only once in post-training pruning, in the next section we present an iterative variation that calculates the optimal p_{thres} bound to the resulting model performance.

C. Iterative performance-aware green resource management algorithm

Magnitude pruning removes a percentage of a model's lowest L^1-norm connections, according to a specified threshold p_{thres}. However, finding the optimal pruning threshold p_{opt} that represents the optimal tradeoff between model complexity and performance is often a tedious procedure that requires multiple experimentations, whose evaluation is, in most cases, subjective. Therefore, we propose Performance-Aware Optimized Pruning (PAOP), an iterative algorithm to determine the optimal pruning threshold for NILM models. Optimality must be bound in terms of performance, as stated Equation 3. Consequently, finding the optimal pruning threshold p_{opt} requires an objective metric that incorporates both the performance degradation of the reduced model and the gain in terms of parameter reduction. Therefore, the metrics utilized for model performance evaluation need to be first defined. Since seq2seq disaggregation is primarily a regression task, and secondarily a classification task, we record three widely used metrics for model evaluation, namely Mean Absolute Error (MAE) and Mean Relative Error (MRE) for regression evaluation and F1-score for classification performance, as presented in Equation 5.

$$\begin{aligned} MAE &= \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i| \\ MRE &= \frac{1}{max(y)}\sum_{i=1}^{N} |\hat{y}_i - y_i| \\ F1 &= \frac{TP}{TP + \frac{1}{2}(FP + FN)} \end{aligned} \tag{5}$$

where y and $\hat{y}$ are the original and the predicted appliance consumption load respectively, and TP, FP and FN stand for the True Positive, False Positive and False Negative classified time instances in the predicted signature.

For different pruning thresholds p_{thres}, the model performance on the test set will change. At the same time, each metric should not be evaluated independently. Instead, all metrics should be combined in a single term. Taking into consideration all the aforementioned considerations, we propose the Pruning Gain metric (PG) to quantify the tradeoff between model complexity and performance, which is formulated in Equation 6.

$$PG = \frac{MAE_b}{MAE_p}\frac{MRE_b}{MRE_p}\frac{F1_p}{F1_b}\frac{N_{param,b}}{N_{param,p}} \tag{6}$$

Pruning Gain measures the pruning-related change in a given metric as the ratio of the baseline performance of the model to the performance of the model after pruning. For metrics where a lower score is better, the terms of the ratio need to be reverted (baseline/pruned). For each metric, we record the ratio of baseline model performance (subscript b) and the model performance after pruning (subscript p), and multiply it by the ratio of change in the number of parameters of the original model and the redacted version. The idea behind PG is to combine the increase or decrease of the metrics recorded to evaluate model performance with the reduction in model size in a multiplicative way. This approach was selected to emphasize the sensitivity of changes in model performance, as an averaging operation of the individual terms would lead to the phenomenon where a positive change in one metric may envelop negative changes in the other ones. Even tough the separate metrics are in different scales, the ratio of each metric captures the relative change between the baseline and the pruned version, which regularizes each ratio separately. No change results in a ratio of 1. A PG score greater than 1 means that the performance loss from removing model weights is beneficial, whereas a score smaller than 1 signifies that the performance drop was more significant than the model compression achieved. Therefore, the proposed metric captures the relative changes between the metrics and can be used to decide whether the impact of pruning on the NILM model was negligible or not.

Utilizing PG, we are now able to perform iterative magnitude pruning to optimally compress a NILM model. To take the hardware characterstics of the edge device into account, the expert needs to define computational cost goals, depending on the deployment scenario. Then, iterative model optimization can begin. First, we define a selected range of pruning threshold percentages $[0, p_{max}]$ that should be taken into consideration, as well as an iteration step p_{step}. Then, for each pruning threshold *p*, we calculate the performance metrics, as well as the Pruning Gain PG. If Pruning Gain is, for the given pruning percentage, higher than 1, then we assume that the reduction of the weights dimensionality was beneficial and that the model can be further compressed, in which case we increment the pruning threshold with $p_{step}\%$. We continue the aforementioned loop until the Pruning Gain falls below 1, where the iteration stops. If the computational cost goals were met, then the previous pruning percentage *p* is selected as the optimal pruning for the given model. Otherwise, the model is not deployable on the edge device. The iterative algorithm is summarized in Algorithm 1.

V. Experimental setup and results

A. Experimental Setup

The methodology to optimize a model for edge inference should depend on the application scenario and the hardware limitations inherent to the edge device. The deployment of NILM models on the edge can be achieved by connecting a smart meter that records the aggregate consumption with a Raspberry Pi 3 Model b single-board computer. Raspberry Pi is one of the most popular edge devices in IoT systems and are commonly used as a gateway to enable the deployment of AI applications in real-world settings [52]. Therefore, we have designed

Algorithm 1 Performance-aware green resource management algorithm

```
1: $cost_{goal}$: Expert-defined computational cost goals (MFLOPs)
2: p : pruning percentage
3: Define $p_{max}$, $p_{step}$,$p_{opt}$
4: for p in range (0, $p_{step}$, $p_{max}$) do
5:     Calculate performance metrics (MAE, MRE, F1)
6:     Calculate pruning gain PG
7:     if $PG > 1$ then
8:         $p_{opt}$ = p
9:     else if $PG < 1$ then
10:        Calculate $cost_{new}$
11:        if $cost_{new} \leq cost_{goal}$ then
12:            break, optimal model found
13:        else
14:            break, model is not deployable
15:        end if
16:    end if
17: end for
```

TABLE II
TECHNICAL SPECIFICATIONS OF A RASPBERRY PI 3 MODEL B

Architecture	ARM
Processing power	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
Memory size	1GB RAM
Connectivity	Ethernet, WLAN
Storage	SD Card

our methodology and experiments to use a Raspberry Pi 3 as the edge device. Raspberry Pi's run on an ARM architecture and have limited storage space and computational power, but are easy to install and use. Their hardware characteristics can be found in Table II.

The architecture to deploy the optimized models on the edge device is depicted in Figure 4. The edge solution consists of 3 different services responsible for data collection, and the NILM inference service, which processes the collected data and produces the disaggregation results. The components of the data collection process are described below:

- **Z-Wave JS UI** is an open source dockerized service which communicates with the aggregate consumption smart meter through Z-wave protocol and forwards the collected data to the ZWave-service through the mqtt protocol.

- **Z-wave-service** is a custom service which receives the collected data from the Z-Wave JS UI through mqtt protocol and forwards them to the data broker service through an API.
- **DataBroker-service** is responsible for receiving the collected data from Z-wave service and communicates with the PostgreSQL database. DataBroker service is also responsible to update (save and delete) the collected data to the existing database.

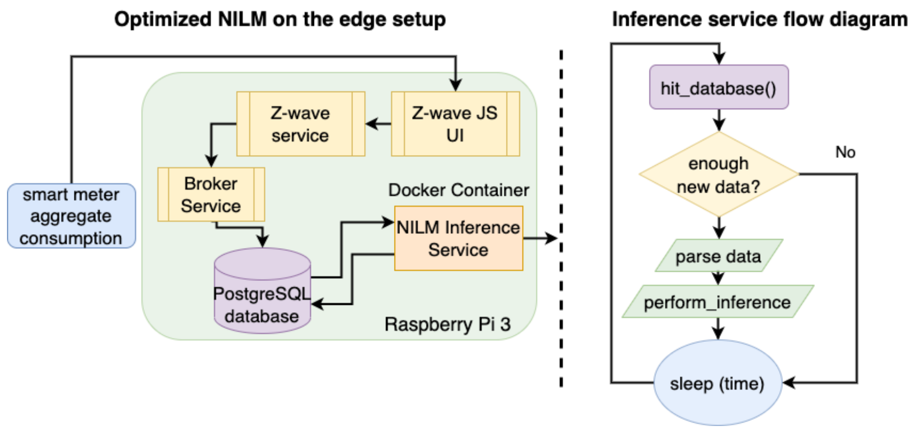


Fig. 4. Overview of the proposed edge NILM deployment architecture. The edge setup consists of services for data collection and NILM inference (left), which performs disaggregation once enough data are collected (right).

The **NILM inference service** is included in a Docker container that runs continuously on the edge device. This service communicates directly with the database after a specified time interval and checks if enough data are collected to produce the disaggregation results.

Depending on the processor's architecture, there are two main quantization backend libraries that can be used, namely FGBEMM [53] and QNNPACK [54]. The term backend refers to reduced precision tensor matrix math libraries which are utilized during model compression. FBGEMM can be used to quantize a model to run on x86 architectures, while QNNPACK supports ARM processor architectures. Since the Raspberry Pi processing unit is based on an ARM architecture, we have chosen QNNPACK as the quantization backend.

Model architecture for the four NILM models that were used to evaluate the impact of model compression on model performance and complexity. The upper left subfigure describes a convolutional neural network [18], whereas the next two subfigures correspond to reccurent architectures with different gating mechanisms (LSTM [57], GRU [58]). Finally, the lower right subfigure presents the Transformer-based architecture [24]

To evaluate our approach, we conducted experiments on different appliances from UK-Dale [57] and REDD [42] datasets. Both datasets consist of aggregate and appliance level energy consumption measurements from five different houses in the United Kingdom and six different houses in the Unites States respectively. UK-Dale was generated at a sample rate of 1 Hz for the aggregate and 1/6 Hz for individual appliances while REDD was monitored at a sample rate of 1 Hz for the aggregate consumption and 1Hz for the plug-level data. The data were resampled at a sample rate of 1/6 Hz and pose the appliance characteristics described in Table III. The models were tested on unseen data from houses not included in the training set, as shown in Figure 6. The reason for

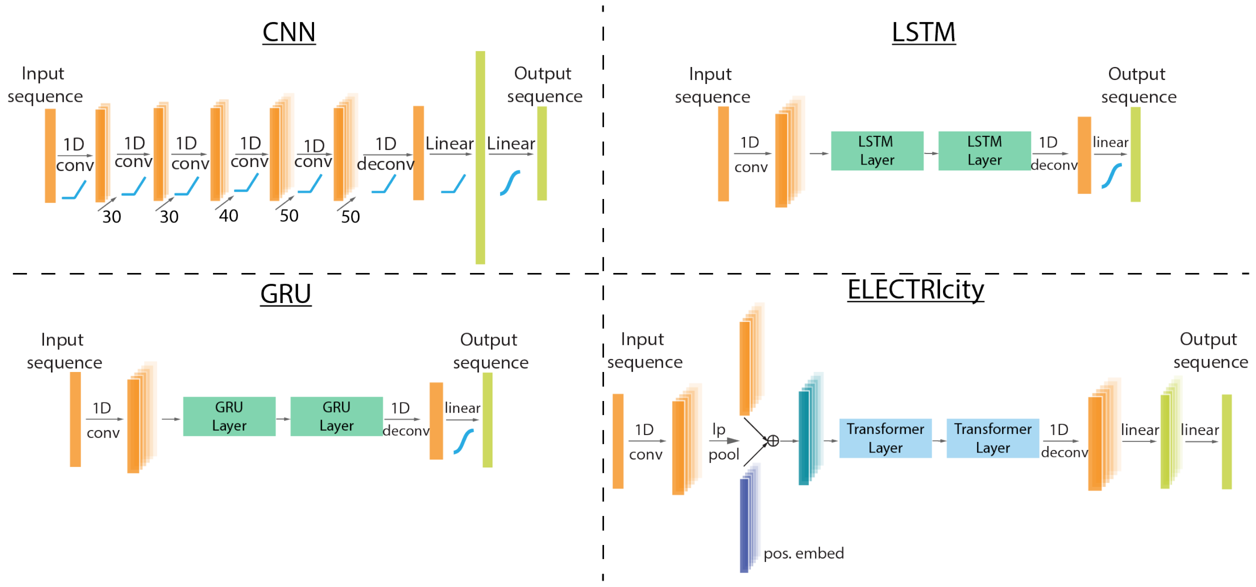


Fig. 5. Model architecture for the four NILM models that were used to evaluate the impact of model compression on model performance and complexity. The upper left subfigure describes a convolutional neural network [17], whereas the next two subfigures correspond to reccurent architectures with different gating mechanisms (LSTM [55], GRU [56]). Finally, the lower right subfigure presents the Transformer-based architecture [22].

testing the models on a house not used in the training set is due to the core concept of NILM; if a house has smart meter data to record appliance consumption, there is no point in deploying a NILM algorithm to infer them, since they are already available to the consumer. Therefore, the proposed approach is to perform NILM on smart meter aggregate readings from a house using pre-trained models, for which ground truth in terms of submetering was available for training on a centralized server, e.g. using publicly available datasets. In UK-Dale we focused on 4 appliances (washer, kettle, fridge, dishwasher), while in REDD on appliances (microwave, washer, dishwasher). The set of appliances selected represent single state and multi-state appliances, with variable load fluctuations.

TABLE III
APPLIANCE CHARACTERISTICS FOR UK-DALE AND REDD DATASETS.

Dataset	Appliance	Max Limit [W]	On Thresh. [W]	Min. On Duration [s]	Min. Off Duration [s]
UK-Dale	Kettle	3100	2000	12	0
	Washer	2500	20	1800	160
	Fridge	300	50	60	12
	Dishwasher	2500	10	1800	1800
REDD	Microwave	1800	200	12	30
	Dishwasher	1200	10	1800	1800
	Washer-Dryer	500	20	1800	160

To diversify our experimental evaluation and test the generalization capabilities of our performance-aware edge

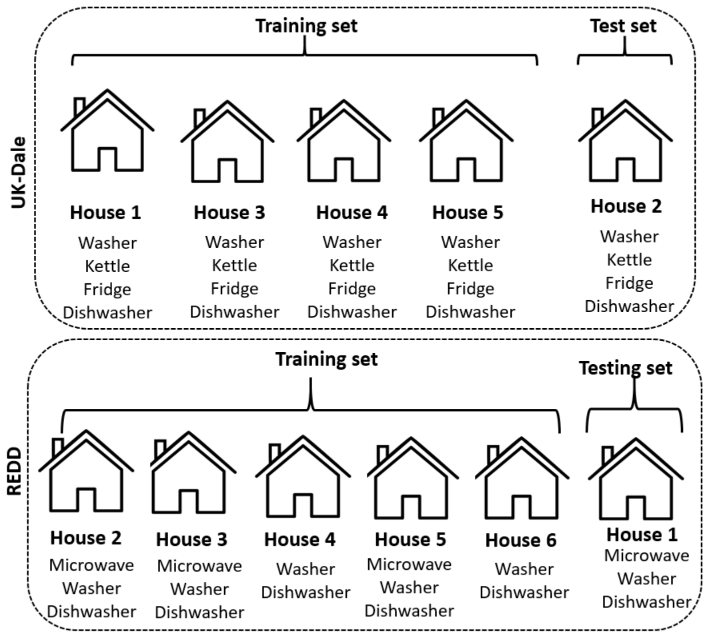


Fig. 6. Train-test split for UK-Dale and REDD datasets. The models were tested on unseen houses non included in the training set. In UK-Dale, houses 1,3,4 and 5 were used for training and house 2 for testing, while in REDD house 2,3,4,5 and 6 were included in the training set and house 1 was kept for model evaluation.

inference optimization framework, the models are based on different architectural philosophies. In particular, one convolutional neural network [17], 2 recurrent architectures (LSTM [55], GRU [56]) and a Transformer-based model [22] were chosen for the evaluation of our approach, and their architectural representations are illustrated in Figure 5. Even though all models initially employ a 1-d convolutional filter for feature extractions, the intermediate part of the model structure varies significantly. The models were purposely trained and evaluated on unbalanced data. The reason for not balancing the dataset is that, to mitigate the negative aspects of central data storage, model training should take place in a federated manner on edge devices, where the possibility of data balancing is limited by hardware constraints. We envision our work as part of a wider NILM framework that enables the transition from central data processing to all computations occurring on the edge to increase the privacy of customers.

In our analysis, we diversify between three edge deployment scenarios, based on different edge device limitations. In the first, the edge device has limited storing capacity, and the edge optimization engine employs model quantization to limit the required storage space of the model. In the second scenario, the limitation is based on the edge device's processing power, and we optimize the models with Performance-Aware Optimized Pruning (PAOP) to reduce their computational complexity. Finally, we investigate the optimization scenario where the edge devices has limited both storage space and computational power. We apply a combination of performance-aware optimized pruning to reduce the number of floating point operations during a forward pass, followed by weight quantization to reduce storage requirements. We call the combination of both techniques Performance-Aware Pruning and Quantization (PAOPQ). In the utilization of the proposed performance-aware schemes, the model complexity reduction ranged between [0, 70] %, with an increment step of 5%. All optimization experiments were performed on an Apple Macbook M1 Pro to take advantage of the ARM CPU architecture to accurately simulate the deployment of the aforementioned models on a real-world setting with Raspberry Pi edge devices.

TABLE IV
SIZE ON DISK BEFORE AND AFTER MODEL QUANTIZATION

Model	Size on disk (MB)	
	Original	Quantized
CNN	4.0	0.98
LSTM	4.6	1.15
GRU	3.6	0.90
ELECTRIcity	7.8	2.89

B. Results

1) Scenario 1: Limited storage capacity: The first step in our analysis is to examine how the aforementioned models were impacted by weight quantization. As can be seen in TableIV, quantization of model weights leads to a significant 75% reduction in the size required to store the model on the disk. At the same time, the effect of quantization on the disaggregation performance, which is presented in Table V and Table VI, varies across model architectures and appliances. In UK-Dale results, recurrent neural networks (LSTM, GRU) showcase minimal performance degradation when disaggregating the kettle and fridge, with a performance reduction less than 0.5% for all metrics. However, they are sensitive to weights quantization for appliances with sparse and long appliance activations, such as the washing machine and the dishwasher. The CNN model has a small performance loss consistent across appliances, while the Transformer-based model (ELECTRIcity) is robust to quantization, showcasing a minimal performance degradation averaging across all appliances (-0.01% MAE, -1.09% MRE and -0.29% F1). The effects of quantization presented in UK-Dale are very similar when the quantized models are evaluated on REDD dataset. Recurent as well as Transformer architectures present a minimal performance degradation across all the tested appliances. In some cases, quantization could also lead to a slight improvement on disaggregation results as it happens in LSTM, GRU and Electricity models on microwave appliance as well as on Electricity model on washer with an average improvement of 10.80% MAE, 8.57% MRE and 2.2% F1.

2) Scenario 2: Limited processing power: Next we would like to evaluate how the models are affected by PAOP. Applying the proposed iterative algorithm to find the optimal pruning threshold, the average number of model parameters can be decreased by 40.93 % in UK-Dale and by 40 % in REDD dataset. The optimal pruning threshold for each model and appliance, as well as the number of baseline parameters are illustrated in Table VII. By comparing the obtained optimal pruning threshold with the performance metrics, as described in Table V, we observe that, in cases where the baseline model does not perform well, indicated by the low F1 score and the high MRE, our algorithm concludes to suggest the highest pruning percentage p_{max}, whereas in cases where the model performs well, the suggested optimal pruning threshold coincides with a plausible value close to the average.

TABLE V
PERFORMANCE METRICS OF OPTIMIZED MODELS -UK-DALE

Appliance	Model	Approach	MAE	MRE	F1	Appliance	Model	Approach	MAE	MRE	F1
Kettle	CNN	Baseline	7.03	0.0025	0.89	Washer	CNN	Baseline	10.45	0.02	0.58
		Quantized	7.55	0.0026	0.88			Quantized	13.64	0.02	0.56
		PAOP	8.71	0.0030	0.85			PAOP	9.77	0.02	0.59
		PAOPQ	8.97	0.0031	0.85			PAOPQ	12.43	0.02	0.57
	LSTM	Baseline	12.18	0.0043	0.80		LSTM	Baseline	21.88	0.05	0.24
		Quantized	12.20	0.0043	0.79			Quantized	23.51	0.11	0.13
		PAOP	13.74	0.0048	0.76			PAOP	14.59	0.10	0.12
		PAOPQ	13.74	0.0048	0.76			PAOPQ	23.51	0.11	0.13
	GRU	Baseline	12.93	0.0046	0.79		GRU	Baseline	19.29	0.05	0.24
		Quantized	12.93	0.0046	0.79			Quantized	20.02	0.08	0.18
		PAOP	14.08	0.0050	0.77			PAOP	9.38	0.05	0.10
		PAOPQ	14.08	0.0050	0.77			PAOPQ	20.01	0.08	0.18
	ELECTRIcity	Baseline	9.26	0.0032	0.94		ELECTRIcity	Baseline	3.65	0.01	0.85
		Quantized	9.26	0.0032	0.94			Quantized	3.66	0.01	0.84
		PAOP	10.17	0.0036	0.92			PAOP	5.21	0.01	0.73
		PAOPQ	10.18	0.0036	0.92			PAOPQ	4.58	0.01	0.76
Fridge	CNN	Baseline	31.86	0.78	0.64	Dishwasher	CNN	Baseline	41.29	0.04	0.09
		Quantized	34.45	0.75	0.62			Quantized	41.30	0.04	0.09
		PAOP	37.41	0.63	0.65			PAOP	41.38	0.05	0.08
		PAOPQ	39.45	0.63	0.64			PAOPQ	41.37	0.05	0.08
	LSTM	Baseline	32.85	0.82	0.63		LSTM	Baseline	31.25	0.03	0.65
		Quantized	33.07	0.82	0.62			Quantized	32.74	0.13	0.28
		PAOP	35.72	0.86	0.54			PAOP	33.01	0.04	0.57
		PAOPQ	35.95	0.87	0.54			PAOPQ	32.71	0.13	0.28
	GRU	Baseline	31.32	0.80	0.67		GRU	Baseline	31.43	0.03	0.62
		Quantized	31.44	0.80	0.66			Quantized	31.53	0.05	0.53
		PAOP	34.21	0.84	0.59			PAOP	39.75	0.04	0.45
		PAOPQ	34.38	0.84	0.58			PAOPQ	31.55	0.05	0.53
	ELECTRIcity	Baseline	23.10	0.71	0.80		ELECTRIcity	Baseline	18.96	0.03	0.82
		Quantized	23.08	0.71	0.80			Quantized	18.93	0.03	0.81
		PAOP	27.61	0.76	0.73			PAOP	24.75	0.03	0.79
		PAOPQ	27.52	0.76	0.73			PAOPQ	24.40	0.04	0.71

An indicative example on this finding is illustrated during pruning of the LSTM model for the disaggregation of the washing machine in UK-Dale dataset. Since the baseline performance is suboptimal, the ratio of baseline performance to performance after pruning is very sensitive to change and, even though the MRE rises by 4.3% in absolute value, the relative change is -83.22%. At the same time, however, the MAE is 33.28% better than the baseline, which can be explained by the fact that artefacts in the predicted appliance signature no longer being produced, and, multiplied with the ratio of model parameter reduction, leads to a positive Pruning Gain value. In the example of ELECTRIcity for the dishwasher appliance, we observe that 35 % of the model weights are removed without notable affecting the model's disaggregation performance. Overall, it can be concluded that the utilization

TABLE VI
PERFORMANCE METRICS OF OPTIMIZED MODELS-REDD

Appliance	Model	Approach	MAE	MRE	F1
Microwave	CNN	Baseline	17.49	0.0558	0.38
		Quantized	17.29	0.0558	0.37
		PAOP	18.16	0.0562	0.21
		PAOPQ	17.27	0.0557	0.37
	LSTM	Baseline	34.86	0.0923	0.31
		Quantized	19.69	0.0622	0.33
		PAOP	35.03	0.1065	0.28
		PAOPQ	17.04	0.3156	0.03
	GRU	Baseline	19.35	0.0618	0.37
		Quantized	19.35	0.0618	0.38
		PAOP	18.84	0.0588	0.24
		PAOPQ	5.44	0.0348	0.18
	ELECTRIcity	Baseline	17.45	0.0562	0.42
		Quantized	17.46	0.0562	0.43
		PAOP	20.23	0.0667	0.52
		PAOPQ	11.16	0.0198	0.29
Washer-Dryer	CNN	Baseline	5.83	0.0283	0.24
		Quantized	3.83	0.0125	0.00
		PAOP	5.92	0.0346	0.22
		PAOPQ	3.83	0.0125	0.12
	LSTM	Baseline	6.17	0.0288	0.21
		Quantized	6.19	0.0289	0.21
		PAOP	6.16	0.0287	0.20
		PAOPQ	6.20	0.0299	0.20
	GRU	Baseline	5.38	0.0342	0.19
		Quantized	5.39	0.0344	0.18
		PAOP	5.61	0.0348	0.16
		PAOPQ	5.39	0.0344	0.18
	ELECTRIcity	Baseline	15.98	0.0204	0.31
		Quantized	15.98	0.0208	0.32
		PAOP	11.17	0.0197	0.29
		PAOPQ	11.16	0.0198	0.29
Dishwasher	CNN	Baseline	30.23	0.0784	0.12
		Quantized	28.65	0.0768	0.12
		PAOP	30.88	0.0810	0.06
		PAOPQ	29.54	0.0745	0.06
	LSTM	Baseline	34.86	0.0923	0.31
		Quantized	34.83	0.0938	0.30
		PAOP	35.03	0.1065	0.28
		PAOPQ	34.98	0.1066	0.27
	GRU	Baseline	49.33	0.0942	0.30
		Quantized	49.37	0.0948	0.30
		PAOP	48.37	0.0966	0.30
		PAOPQ	49.04	0.0991	0.29
	ELECTRIcity	Baseline	17.35	0.0498	0.63
		Quantized	17.36	0.0496	0.61
		PAOP	11.17	0.0197	0.29
		PAOPQ	21.67	0.048	0.57

of our performance-aware model compression strategy can reduce the computational complexity of a NILM model without significantly affecting its performance. The complexity reduction is validated through the reduced number

TABLE VII
BASELINE MODEL PARAMETERS AND OPTIMAL PRUNING THRESHOLD, AS OBTAINED BY APPLYING ALGORITHM 1

Dataset	**Appliance**	**Model**	**Baseline Parameters N**	**Optimal pruning threshold p_{opt} (%)**		**MFLOPs**		
				PAOP	*PAOPQ*	*Baseline*	*PAOP*	*PAOPQ*
UK-Dale	Kettle	CNN	996595	40	5	18.33	13.75	17.41
		LSTM	1141777	30	30	16.71	12.50	12.50
		GRU	887569	25	25	12.44	10.17	10.17
		ELECTRIcity	1938433	25	25	586.70	452.08	452.08
	Fridge	CNN	996595	70	70	18.33	10.00	10.00
		LSTM	1141777	30	30	16.71	11.67	11.67
		GRU	887569	25	25	12.44	8.67	8.67
		ELECTRIcity	1938433	40	40	586.70	376.43	376.43
	Washer	CNN	996595	10	5	183.3	15.79	17.41
		LSTM	1141777	60	5	16.71	7.50	15.86
		GRU	887569	70	5	12.44	3.72	11.94
		ELECTRIcity	1938433	60	50	586.70	268.18	283.73
	Dishwasher	CNN	996595	30	30	18.33	15.20	15.20
		LSTM	1141777	35	5	16.71	10.80	15.86
		GRU	887569	70	5	12.44	3.71	11.94
		ELECTRIcity	1938433	35	55	586.70	398.13	309.48
Redd	Microwave	CNN	996595	70	5	18.33	10.45	17.82
		LSTM	1141777	5	30	16.71	15.73	11.79
		GRU	887569	45	45	12.44	7.60	7.60
		ELECTRIcity	1938433	70	70	586.70	220.04	220.04
	Washer-Dryer	CNN	996595	60	5	18.33	11.65	17.80
		LSTM	1141777	5	5	16.71	15.72	15.72
		GRU	887569	70	5	12.44	3.85	11.81
		ELECTRIcity	1938433	35	35	586.70	402.62	402.62
	Dishwasher	CNN	996595	60	60	18.33	11.94	11.94
		LSTM	1141777	5	5	16.71	15.73	15.73
		GRU	887569	20	15	12.44	10.40	10.91
		ELECTRIcity	1938433	35	35	586.70	403.27	403.27

of floating point operations (FLOPs) required to perform a forward pass, as can be seen in Table VII. On average, PAOP reduces the FLOPs of a NILM model by 36.3% in UK-Dale and by 31.8% in REDD.

3) Scenario 3: Limited storage capacity and processing power: The last experiment that was conducted was the combination of both aforementioned model compression approaches (PAOPQ). To calculate the optimal pruning threshold in this case, the model performance was evaluated after both schemes were applied. Therefore, the optimal threshold obtained is different than in the case of pruning (see Table VII). It can be easily noticed that the combination of both techniques tolerates significantly lower pruning percentages for most models. On average, the optimal pruning threshold is 37.4% lower in UK-Dale and 26.25% in REDD, compared to when weights quantization is not utilized. Therefore, it can be concluded that, without the proposed performance aware optimization scheme,

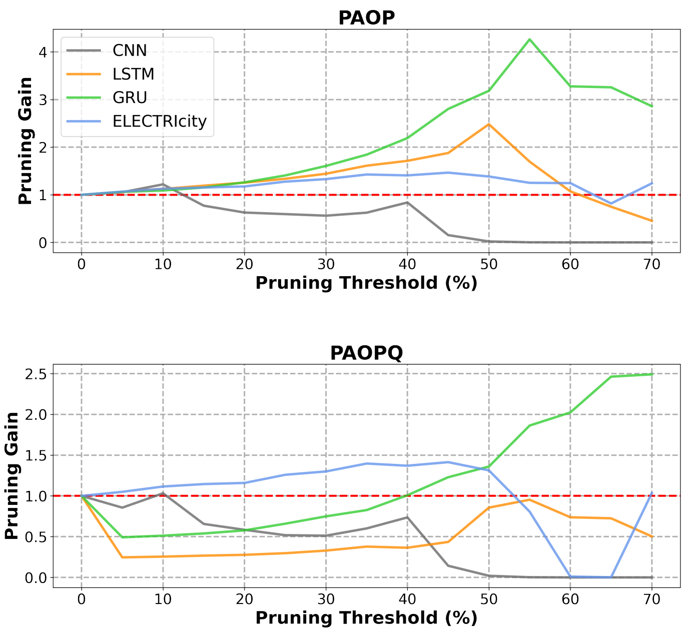


Fig. 7. Pruning Gain plot over different pruning thresholds during the application of proposed iterative Algorithm 1 on the washing machine when the compression method is pruning (upper) vs pruning & quantization (lower). We observe that the curves of the pruning gain are significantly different depending on the chosen model compression approach and that, if the model does not have a strong baseline performance (GRU), the Pruning Gain is gradually increasing with further parameter reduction.

the performance degradation of the models would be significantly higher. The integration of both techniques in our scheme results in 75% less size on disk for both UK-Dale and REDD and, on average, 25.62% less model parameters and 22% less FLOPs for UK-Dale and 21.51% less model parameters and 21.09% less FLOPs for REDD, thus reducing both storage requirements and model computational complexity. Another interesting finding is that the optimal thresholds for ELECTRIcity remain the same as in the case of applying only parameter pruning and, in the case of dishwasher, the model tolerates a higher pruning percentage, meaning that Transformer-based architectures are more robust to model compression than convolution-based and recurrent modeling approaches.

An illustration of the Pruning Gain metric values during the application of the algorithm to find the optimal pruning threshold p_{thres} is illustrated in Figure 7. The upper part of the figure showcases the Pruning Gain distribution when only model pruning is applied, while the lower part demonstrates the distribution when both pruning and quantization are selected. The difference in both plots validates the observation that model performance is more sensitive to combining both compression approaches and that the proposed metric can accurately quantify the tradeoff between model performance and computational complexity.

Finally, we recorded the CO2 emissions of each optimized model through the CodeCarbon Python library, and the results are presented in Table VIII. All optimization approaches reduce CO_2 emissions by $\approx 17\%$. The only exception can be found for recurrent architectures (LSTM, GRU), where the CO_2 emissions are higher when quantization is involved. This can be explained by the increased complexity of recurrent layer computations, where multiple activation functions are utilized inside each memory cell. Since the outputs of activation functions are

TABLE VIII

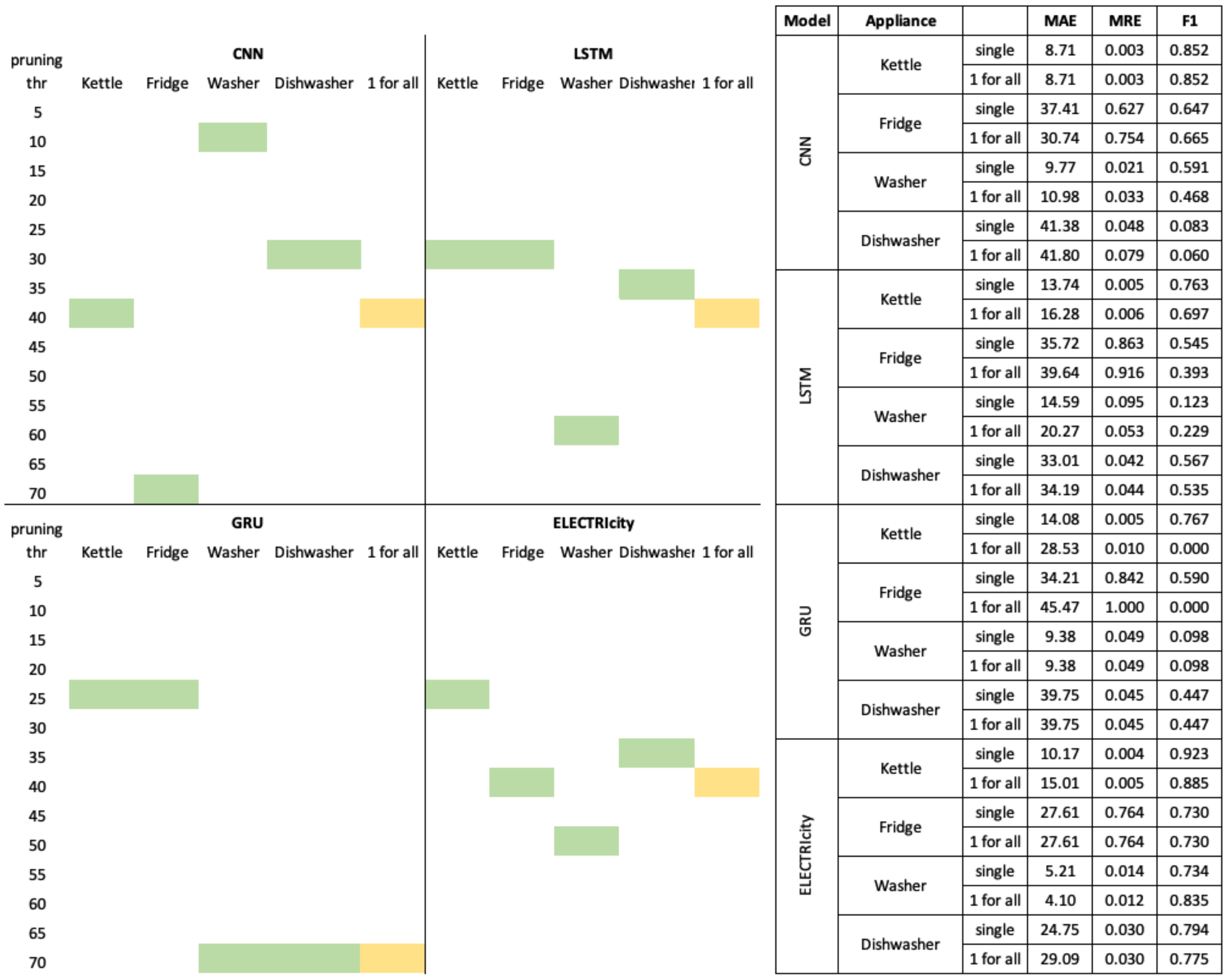
CO2 EMISSIONS[G] FOR THE BASELINE MODEL AND EACH OPTIMIZATION METHOD

			CO2 Emissions[g] of each model optimized for edge deployment			
Dataset	**Appliance**	**Model**	**Baseline**	**Quantization**	**PAOP**	**PAOPQ**
UK-Dale	Kettle	CNN	0.072	0.061	0.071	0.061
		LSTM	0.083	0.178	0.081	0.177
		GRU	0.070	0.185	0.068	0.186
		ELECTRIcity	1.240	1.059	1.209	1.068
	Fridge	CNN	0.060	0.049	0.059	0.049
		LSTM	0.068	0.145	0.065	0.145
		GRU	0.057	0.151	0.053	0.150
		ELECTRIcity	1.246	1.014	1.134	1.025
	Washing Machine	CNN	0.060	0.050	0.057	0.050
		LSTM	0.067	0.145	0.065	0.145
		GRU	0.056	0.150	0.055	0.150
		ELECTRIcity	1.257	1.016	1.157	1.014
	Dishwasher	CNN	0.060	0.049	0.059	0.049
		LSTM	0.067	0.144	0.065	0.145
		GRU	0.056	0.150	0.055	0.150
		ELECTRIcity	1.235	1.030	1.187	1.052
Redd	Microwave	CNN	0.003	0.001	0.003	0.002
		LSTM	0.002	0.002	0.002	0.002
		GRU	0.002	0.001	0.002	0.001
		ELECTRIcity	0.051	0.032	0.051	0.032
	Washer-Dryer	CNN	0.003	0.001	0.003	0.001
		LSTM	0.002	0.002	0.002	0.002
		GRU	0.002	0.001	0.002	0.001
		ELECTRIcity	0.051	0.032	0.051	0.032
	Dishwasher	CNN	0.003	0.001	0.003	0.002
		LSTM	0.002	0.002	0.002	0.002
		GRU	0.002	0.001	0.002	0.001
		ELECTRIcity	0.051	0.032	0.051	0.032

dynamically quantized during inference, the increased energy needs to perform the quantization is justifiable.

C. Discussion

As already mentioned, our approach suffers from certain limitations. First, we have observed that models that do not showcase good baseline performance tend to be overpruned by our proposed iterative algorithm during the search for the optimal pruning threshold. Even though such models should not be deployed to perform inference, as the insights that the consumer will get regarding energy consumption will not be accurate, our approach should still take into consideration such cases. The second limitation concerns the fact that, in our approach, the models are optimized for each different appliance. Therefore, to perform energy disaggregation for multiple appliances, the deployment of multiple NILM models is required. However, we have experimented with model optimization for all



pruning	CNN					LSTM				
thr	Kettle	Fridge	Washer	Dishwasher	1 for all	Kettle	Fridge	Washer	Dishwasher	1 for all
5										
10			🟩							
15										
20										
25										
30				🟩		🟩	🟩			
35									🟩	
40	🟩				🟨					🟨
45										
50										
55										
60								🟩		
65										
70		🟩								

pruning	GRU					ELECTRIcity				
thr	Kettle	Fridge	Washer	Dishwasher	1 for all	Kettle	Fridge	Washer	Dishwasher	1 for all
5										
10										
15										
20										
25	🟩	🟩				🟩				
30										
35									🟩	
40							🟩			🟨
45										
50								🟩		
55										
60										
65										
70			🟩	🟩	🟨					

Model	Appliance		MAE	MRE	F1
CNN	Kettle	single	8.71	0.003	0.852
		1 for all	8.71	0.003	0.852
	Fridge	single	37.41	0.627	0.647
		1 for all	30.74	0.754	0.665
	Washer	single	9.77	0.021	0.591
		1 for all	10.98	0.033	0.468
	Dishwasher	single	41.38	0.048	0.083
		1 for all	41.80	0.079	0.060
LSTM	Kettle	single	13.74	0.005	0.763
		1 for all	16.28	0.006	0.697
	Fridge	single	35.72	0.863	0.545
		1 for all	39.64	0.916	0.393
	Washer	single	14.59	0.095	0.123
		1 for all	20.27	0.053	0.229
	Dishwasher	single	33.01	0.042	0.567
		1 for all	34.19	0.044	0.535
GRU	Kettle	single	14.08	0.005	0.767
		1 for all	28.53	0.010	0.000
	Fridge	single	34.21	0.842	0.590
		1 for all	45.47	1.000	0.000
	Washer	single	9.38	0.049	0.098
		1 for all	9.38	0.049	0.098
	Dishwasher	single	39.75	0.045	0.447
		1 for all	39.75	0.045	0.447
ELECTRIcity	Kettle	single	10.17	0.004	0.923
		1 for all	15.01	0.005	0.885
	Fridge	single	27.61	0.764	0.730
		1 for all	27.61	0.764	0.730
	Washer	single	5.21	0.014	0.734
		1 for all	4.10	0.012	0.835
	Dishwasher	single	24.75	0.030	0.794
		1 for all	29.09	0.030	0.775

Fig. 8. Comparison of optimal pruning threshold and performance between optimizing the models for each individual appliance (green) vs jointly (yellow). Optimizing the models on the joint set of appliances (1 for all) leads to subpar optimization.

appliances simultaneously, as shown in Figure 8, and have found that optimizing the model for all appliances at the same time leads to over/underpruning and impacts the achievable performance. Averaging across all appliances and models, this approach would lead to a performance loss of 8.92% MAE, 7.32% MRE and 12.20% F1.

VI. Conclusions and future work

In this work, we have proposed an efficient, performance-aware model optimization framework for edge deployment of NILM models that takes into account the edge device characteristics. We have explored three different deployment limitations, for which optimization of different model aspects is required. Additionally, we proposed an objective model optimization metric and a performance-aware model complexity reduction algorithm that constrains model optimization on performance loss. Experimental results validate that our proposed method to bind model performance with model compression, instead of performing it arbitrarily, allows for the combined utilization of more than one compression approaches on the same model without significantly affecting model performance, thus enabling the efficient deployment of NILM models on edge devices.

In future work, we would like to implement further techniques, such as knowledge distillation or tensor decomposition, in our performance-aware compression scheme. Further methods for weight quantization, as well as different magnitude pruning approaches (gradient based magnitude pruning, information based pruning) and structured pruning will also be evaluated. We would also like to test our approach on different model architectures, assess the difference on models trained with balanced datasets versus unbalanced datasets, and adapt our proposed iterative scheme to account for optimal model compression of models with subpar disaggregation performance. In addition, we plan to utilize recent advancements in sparse matrix computation on edge devices to maximize the optimization potential of our methods [58]. Finally, our methodology has been structured around the limitations of a real-world scenario, and we would like to deploy the compress models on Raspberry Pi devices, connected with house smart meters, to evaluate whether the simulation experiments that we have conducted are translated to real-world conditions in the same way.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 955422.

References

[1] M. Kaselimi, E. Protopapadakis, A. Voulodimos, N. Doulamis, and A. Doulamis, "Towards Trustworthy Energy Disaggregation: A Review of Challenges, Methods, and Perspectives for Non-Intrusive Load Monitoring," *Sensors*, vol. 22, no. 15, 2022. DOI: 10.3390/s22155872.

[2] M. A. Mengistu, A. A. Girmay, C. Camarda, A. Acquaviva, and E. Patti, "A cloud-based on-line disaggregation algorithm for home appliance loads," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3430–3439, 2018.

[3] M. W. Asres, L. Ardito, and E. Patti, "Computational cost analysis and data-driven predictive modeling of cloud-based online NILM algorithm," *IEEE Transactions on Cloud Computing*, 2021.

[4] D. Ibaseta, A. Garcia, M. Alvarez, *et al.*, "Monitoring and control of energy consumption in buildings using WoT: A novel approach for smart retrofit," *Sustainable Cities and Society*, vol. 65, p. 102 637, 2021.

[5] Y.-L. Lee, P.-K. Tsung, and M. Wu, "Techology trend of edge AI," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, Hsinchu, Taiwan: IEEE, 2018, pp. 1–2. DOI: 10.1109/VLSI-DAT.2018.8373244.

[6] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[7] K. He, L. Stankovic, J. Liao, and V. Stankovic, "Non-intrusive load disaggregation using graph signal processing," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1739–1747, 2016.

[8] B. Zhao, K. He, L. Stankovic, and V. Stankovic, “Improving event-based non-intrusive load monitoring using graph signal processing,” *IEEE Access*, vol. 6, pp. 53 944–53 959, 2018.

[9] J. Z. Kolter and T. Jaakkola, “Approximate inference in additive factorial hmms with application to energy disaggregation,” in *Artificial intelligence and statistics*, PMLR, 2012, pp. 1472–1482.

[10] D. Bajovic, K. He, L. Stankovic, D. Vukobratovic, and V. Stankovic, “Optimal detection and error exponents for hidden semi-Markov models,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 5, pp. 1077–1092, 2018.

[11] J. Kelly and W. Knottenbelt, “Neural nilm: Deep neural networks applied to energy disaggregation,” in *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, 2015, pp. 55–64.

[12] R. Bonfigli, A. Felicetti, E. Principi, M. Fagiani, S. Squartini, and F. Piazza, “Denoising autoencoders for non-intrusive load monitoring: improvements and comparative evaluation,” *Energy and Buildings*, vol. 158, pp. 1461–1474, 2018.

[13] D. Murray, L. Stankovic, V. Stankovic, S. Lulic, and S. Sladojevic, “Transferability of neural network approaches for low-rate energy disaggregation,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8330–8334.

[14] J. Kim, T.-T.-H. Le, and H. Kim, “Nonintrusive load monitoring based on advanced deep learning and novel signature,” *Computational intelligence and neuroscience*, vol. 2017, 2017.

[15] M. Kaselimi, E. Protopapadakis, A. Voulodimos, N. Doulamis, and A. Doulamis, “Multi-channel recurrent convolutional neural networks for energy disaggregation,” *IEEE Access*, vol. 7, pp. 81 047–81 056, 2019.

[16] A. Harell, S. Makonin, and I. V. Bajic, “Wavenilm: A causal neural network for power disaggregation from the complex power signal,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 8335–8339.

[17] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, “Sequence-to-Point Learning with Neural Networks for Non-Intrusive Load Monitoring,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI’18/IAAI’18/EAAI’18, New Orleans, Louisiana, USA, 2018.

[18] A. Faustine, L. Pereira, H. Bousbiat, and S. Kulkarni, “UNet-NILM: A deep neural network for multi-tasks appliances state detection and power estimation in NILM,” in *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*, 2020, pp. 84–88.

[19] K. Bao, K. Ibrahimov, M. Wagner, and H. Schmeck, “Enhancing neural non-intrusive load monitoring with generative adversarial networks,” *Energy Informatics*, vol. 1, no. 1, pp. 295–302, 2018.

[20] M. Kaselimi, A. Voulodimos, E. Protopapadakis, N. Doulamis, and A. Doulamis, “Energan: A generative adversarial network for energy disaggregation,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 1578–1582.

[21] Z. Yue, C. R. Witzig, D. Jorde, and H.-A. Jacobsen, “Bert4nilm: A bidirectional transformer model for non-intrusive load monitoring,” in *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*, 2020, pp. 89–93.

[22] S. Sykiotis, M. Kaselimi, A. Doulamis, and N. Doulamis, “ELECTRIcity: An Efficient Transformer for Non-Intrusive Load Monitoring,” *Sensors*, vol. 22, no. 8, 2022. DOI: 10.3390/s22082926.

[23] C. Xia, W. Li, X. Chang, F. C. Delicato, T. Yang, and A. Y. Zomaya, “Edge-based energy management for smart homes,” in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, IEEE, 2018, pp. 849–856.

[24] S. R. Tito, S. Soltic, B. Parkash, *et al.*, “Is Edge Computing the Answer for Smart Building Energy Management System?” In *2021 International Conference on Technology and Policy in Energy and Electric Power (ICT-PEP)*, IEEE, 2021, pp. 378–383.

[25] A. S. Uttama Nambi, A. Reyes Lua, and V. R. Prasad, “Loced: Location-aware energy disaggregation framework,” in *Proceedings of the 2nd acm international conference on embedded systems for energy-efficient built environments*, 2015, pp. 45–54.

[26] Q. Xu, Y. Liu, and K. Luan, “Edge-Based NILM System with MDMR Filter-Based Feature Selection,” in *2022 IEEE 5th International Electrical and Energy Conference (CIEEC)*, IEEE, 2022, pp. 5015–5020.

[27] E. Tabanelli, D. Brunelli, A. Acquaviva, and L. Benini, “Trimming Feature Extraction and Inference for MCU-based Edge NILM: A Systematic Approach,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 943–952, 2021.

[28] E. Tabanelli, D. Brunelli, and L. Benini, “A feature reduction strategy for enabling lightweight non-intrusive load monitoring on edge devices,” in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, IEEE, 2020, pp. 805–810.

[29] A. Hernandez, R. Nieto, D. Fuentes, and J. Urena, “Design of a SoC Architecture for the Edge Computing of NILM Techniques,” in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, IEEE, 2020, pp. 1–6.

[30] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018. DOI: 10.1109/MSP.2017.2765695.

[31] A. Alqahtani, X. Xie, and M. W. Jones, “Literature Review of Deep Network Compression,” in *Informatics*, vol. 8, Basel, Switzerland: Multidisciplinary Digital Publishing Institute, p. 77. DOI: 10.3390/informatics8040077.

[32] H.-H. Chang, P. W. Wiratha, and N. Chen, “A non-intrusive load monitoring system using an embedded system for applications to unbalanced residential distribution systems,” *Energy Procedia*, vol. 61, pp. 146–150, 2014.

[33] T. Sirojan, T. Phung, and E. Ambikairajah, “Intelligent edge analytics for load identification in smart meters,” in *2017 IEEE Innovative Smart Grid Technologies-Asia (ISGT-Asia)*, IEEE, 2017, pp. 1–5.

[34] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A Survey of Model Compression and Acceleration for Deep Neural Networks," *Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding*, vol. 35, 2017. DOI: 10.48550/ARXIV.1710.09282.

[35] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., Caribe Hilton, San Juan, Puerto Rico: ICLR, 2016.

[36] M. Kaselimi, N. Doulamis, A. Doulamis, A. Voulodimos, and E. Protopapadakis, "Bayesian-optimized Bidirectional LSTM Regression Model for Non-intrusive Load Monitoring," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK: IEEE, 2019, pp. 2747–2751. DOI: 10.1109/ICASSP.2019.8683110.

[37] S. Athanasoulias, S. Sykiotis, M. Kaselimi, E. Protopapadakis, and N. Ipiotis, "A First Approach Using Graph Neural Networks on Non-Intrusive-Load-Monitoring," in *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '22, Corfu, Greece: Association for Computing Machinery, 2022, pp. 601–607. DOI: 10.1145/3529190.3534722.

[38] S. Ahmed and M. Bons, "Edge Computed NILM: A Phone-Based Implementation Using MobileNet Compressed by Tensorflow Lite," pp. 44–48, 2020.

[39] J. Barber, H. Cuayahuitl, M. Zhong, and W. Luan, "Lightweight Non-Intrusive Load Monitoring Employing Pruned Sequence-to-Point Learning," in *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*. New York, NY, USA: Association for Computing Machinery, 2020, vol. 1, pp. 11–15. DOI: https://doi.org/10.1145/3427771.3427845.

[40] D. Murray, L. Stankovic, and V. Stankovic, "An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study," *Scientific Data*, vol. 4, p. 160 122, Jan. 2017. DOI: 10.1038/sdata.2016.122.

[41] R. Kukunuri, A. Aglawe, J. Chauhan, *et al.*, "EdgeNILM: Towards NILM on Edge Devices," in *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '20, Virtual Event, Japan: Association for Computing Machinery, 2020, pp. 90–99. DOI: 10.1145/3408308.3427977.

[42] J. Kolter and M. Johnson, "REDD: A Public Data Set for Energy Disaggregation Research," in *IN SUSTKDD*, vol. 25, Jan. 2011.

[43] H. Liu, C. Liu, L. Tian, H. Zhao, and J. Liu, "Non-intrusive Load Disaggregation Based on Deep Learning and Multi-feature Fusion," in *2021 3rd International Conference on Smart Power Internet Energy Systems (SPIES)*, Shanghai, China: IEEE, 2021, pp. 210–215. DOI: 10.1109/SPIES52282.2021.9633819.

[44] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[45] J. Frankle and M. Carbin, “The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks,” in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, New Orleans, LA, USA: OpenReview.net, 2019. [Online]. Available: https://openreview.net/forum?id=rJl-b3RcF7.

[46] S. Han, J. Pool, J. Tran, and W. J. Dally, “Learning Both Weights and Connections for Efficient Neural Networks,” in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’15, Montreal, Canada: MIT Press, 2015, pp. 1135–1143.

[47] Y. Wang, X. Zhang, L. Xie, *et al.*, “Pruning from scratch,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, New York Hilton Midtown, New York, USA: AAAI Press, 2020, pp. 12 273–12 280.

[48] 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*, Venice, Italy: IEEE, 2017, pp. 2736–2744.

[49] X. Liu, W. Xia, and Z. Fan, “A Deep Neural Network Pruning Method Based on Gradient L1-norm,” in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, IEEE, Chengdu , China: ICCC 2020, 2020, pp. 2070–2074.

[50] C. Fan, J. Li, T. Zhang, *et al.*, “Layer-wise Model Pruning based on Mutual Information,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3079–3090. DOI: 10.18653/v1/2021.emnlp-main.246.

[51] K. Azarian, Y. Bhalgat, J. Lee, and T. Blankevoort, “Learned threshold pruning,” *arXiv preprint arXiv:2003.00075*, vol. 1, 2020.

[52] A. Glória, F. Cercas, and N. Souto, “Design and implementation of an IoT gateway to create smart environments,” *Procedia Computer Science*, vol. 109, pp. 568–575, 2017, 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal. DOI: https://doi.org/10.1016/j.procs.2017.05.343.

[53] D. S. Khudia, J. Huang, P. Basu, *et al.*, “FBGEMM: Enabling High-Performance Low-Precision Deep Learning Inference,” *CoRR*, vol. abs/2101.05615, p. 5, 2021. arXiv: 2101.05615.

[54] M. Dukhan, Y. Wu, and H. Lu, *QNNPACK: Open source library for optimized mobile deep learning*, https://code.fb.com/ml-applications/qnnpack/, 2018.

[55] M. Kaselimi, N. Doulamis, A. Voulodimos, E. Protopapadakis, and A. Doulamis, “Context aware energy disaggregation using adaptive bidirectional LSTM models,” *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3054–3067, 2020.

[56] H. Rafiq, H. Zhang, H. Li, and M. K. Ochani, “Regularized LSTM Based Deep Learning Model: First Step towards Real-Time Non-Intrusive Load Monitoring,” *2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, vol. 11, no. 5, pp. 234–239, 2018.

[57] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, vol. 2, Mar. 2015. DOI: 10.1038/sdata.2015.7.

[58] Y. Jiang, S. Wang, V. Valls, *et al.*, "Model Pruning Enables Efficient Federated Learning on Edge Devices," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022. DOI: 10.1109/TNNLS.2022.3166101.