Optimizing the IoT Performance: A Case Study on Pruning a Distributed CNN

Eiraj Saqib*, Isaac Sánchez Leal*, Irida Shallari*, Axel Jantsch*†, Silvia Krug *‡, Mattias O'Nils*
* Mid Sweden University, Sundsvall, Sweden
Email: first.last@miun.se
†TU Wien, Vienna, Austria
Email: axel.jantsch@tuwien.ac.at
‡IMMS Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH (IMMS GmbH), Ilmenau, Germany
Email: silvia.krug@imms.de

***Abstract*—Implementing Convolutional Neural Networks (CNN) based computer vision algorithms in Internet of Things (IoT) sensor nodes can be difficult due to strict computational, memory, and latency constraints. To address these challenges, researchers have utilized techniques such as quantization, pruning, and model partitioning. Partitioning the CNN reduces the computational burden on an individual node, but the overall system computational load remains constant. Additionally, communication energy is also incurred. To understand the effect of partitioning and pruning on energy and latency, we conducted a case study using a feet detection application realized with Tiny Yolo-v3 on a 12th Gen Intel CPU with NVIDIA GeForce RTX 3090 GPU. After partitioning the CNN between the sequential layers, we apply quantization, pruning, and compression and study the effects on energy and latency. We analyze the extent to which computational tasks, data, and latency can be reduced while maintaining a high level of accuracy. After achieving this reduction, we offloaded the remaining partitioned model to the edge node. We found that over 90% computation reduction and over 99% data transmission reduction are possible while maintaining mean average precision above 95%. This results in up to 17x energy savings and up to 5.2x performance speed-up.**

***Index Terms*—CNN, IoT, Pruning, Quantization, Partitioning, Tiny YOLO-v3.**

I. Introduction

In recent years, the field of computer vision (CV) has experienced rapid advancements, particularly with the development of Convolutional Neural Networks (CNNs). These state-of-the-art algorithms have been successfully applied to various tasks, from simple image classification challenges, such as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1] to more complex applications like object detection [2], semantic segmentation [3], style transfer [4], and many others.

The simultaneous development of artificial intelligence and the Internet of Everything (IoE) fosters innovation by interconnecting people, devices, and data. To maximize this potential, distributed execution of CNNs across IoE devices is essential, requiring decentralized data and smarter devices capable of interacting with their environment. Consequently, there is a pressing need to adapt powerful CNN algorithms for deployment on a variety of devices, ranging from sensor nodes in industrial environments to wearable and consumer electronics.

To address the challenge of implementing CNNs in the Internet of Things (IoT) nodes, researchers have designed lightweight models, such as YOLO, Tiny YOLO, MobileNet, etc, specifically for resource-constrained devices. These models can be further optimized using various techniques, such as quantization [5], pruning [6], and knowledge distillation [7]. The primary objective of these techniques is to minimize the model size and computational cost, enabling more efficient deployment on IoT devices with limited resources and processing capabilities. An alternative to fully implementing models within a single node is model partitioning, which distributes the computational and memory workload across multiple entities. Although this method reduces the workload and energy consumption for individual nodes, it does not necessarily decrease the overall system energy consumption. Additionally, partitioning introduces communication costs due to the substantial inter-layer data in CNN models. Existing techniques focus on either partitioning or shrinking the model, but their combination remains unexplored. Merging these approaches is expected to yield synergistic benefits, leading to improved performance and efficiency for deep learning-based computer vision applications.

In this paper, optimizing node energy consumption and minimization of overall latency due to data and computation reduction is studied. In this work, Tiny Yolo-v3 (TY3) is used as a case study model and combines quantization and pruning before offloading part of the partitioned CNN model to an edge device. Drawing upon the caregiver tracking design case of Vilar et al. [8], for feet detection using MIUN-Feet dataset [9], we explore the design space by examining different partitioning points in the CNN model.

The primary research gap this paper aims to address is to identify an optimal partition point that incorporates the impact of pruning, which minimizes the combined computation and communication node energy consumption while maintaining a predefined level of detection accuracy. Through this case study, the aim is to demonstrate the potential for energy and latency optimization for CNN deployment in IoT nodes, opening up new avenues for future research and development.

II. Related work

The main constraints in the design of an IoT node, such as processing power, memory, and energy consumption, are defined by the stringent requirements for a small form factor and a long battery lifetime. In contrast with this, there are CV-based applications that rely on a heavy computational workload on large volumes of data. Consequently, optimizing them is an active research topic, with various techniques and models being developed for the optimization of CNN models.

Quantization-based CNN optimization techniques, such as scalar quantization [10], vector quantization [11], and quantization mimic [12], aim to reduce memory footprint and computational demands while maintaining accuracy. These methods involve quantizing of weights, of filter kernels, using integer-only arithmetic [13], and leveraging knowledge distillation. Quantization of CNNs may result in information loss and reduced accuracy; while retraining can recover some accuracy, lower-precision methods like binary quantization still cause significant accuracy drops, as evidenced by over 10% decrease in [5].

Pruning, as opposed to quantization, focuses on reducing memory and computational requirements by eliminating redundant or less significant connections, weights, or neurons within neural networks. Hao Li et al. [6] developed iterative magnitude pruning, which utilizes a regularization term during fine-tuning to minimize computation costs and memory requirements while maintaining accuracy. Molchanov et al. [14] proposed variational dropout, resulting in sparse solutions in both convolutional and fully connected layers by pruning unnecessary weights. Addressing the presence of redundant filters, weights, neurons, and connections that have little impact on overall accuracy, researchers like Yang et al. [15] and Han et al. [16] have contributed to overcoming computation and memory constraints through methods such as NetAdapt.However, pruning alone does not comprehensively address optimization aspects like data representation, workload distribution, or communication overhead reduction, necessitating the exploration of combined approaches with techniques like model partitioning and quantization.

Building on the ideas of pruning and quantization, another approach to lessen the computational load on IoT devices without sacrificing accuracy is the distributed execution of CNN models. In distributed systems, the CNN model is spread across various IoT devices in accordance with the memory and computational constraints of the individual devices. MoDNN [17] tackles fully distributed inference by distributing a CNN across multiple mobile phones connected via a wireless network, including both input/output and weight data. However, it does not consider communication between fully-connected layers, neglects weight-intensive convolutional layers, and requires synchronization between devices after each layer. Shallari et al. [18] introduced the concept of intelligence partitioning for constrained IoT devices, where workload partitioning between node and cloud was determined by analyzing the interplay between computation and communication. This technique can be applied to CNN models also.

Partitioning a CNN model can result in two parts of the model that are to be executed at two locations, requiring the data to be transferred between them, however, depending upon the partitioning point, the amount of data can be quite large and hence introduces a communication overhead. To mitigate this, quantization ranging from 32-bit width to 8 bits such as in, [19] have been applied. The quantization here reduces the amount of information to be transferred via the communication channel. Expanding on previous research, Isaac et al. [20] explored deep quantization by applying a range of 8-bit down to 1-bit quantization techniques. Additionally, they employed packing and compression methods to further minimize the data transfer between IoT devices and edge devices. Another full distribution is achieved with layer pipelining [21]. However, this method is not able to evenly distribute the memory demand for typical models.

With additional constraints, the communication overhead could be reduced further by the approaches presented in this work. To the best of our knowledge, there has not been much work done on reducing the data in distributed IoT systems, with a focus on reducing processing and communication overhead, hence in this work, using TY3 as a case study model, we introduce the idea of pruning the interlayer feature maps after quantizing the data and exploring the optimal partitioning point between an IoT node and edge node.

III. Methodology

The proposed method aims to optimize the energy consumption of a smart IoT node by means of quantization and pruning within the node offloading perspective. This work focuses on the computational and communication aspects as key components of the node energy consumption, as represented in Eq. 1. The sensing energy consumption is omitted from this equation, as it remains constant, regardless of the different configurations of node offloading.

$$E_{Node} = \sum_{i=1}^{j} (E_P(t_i, P)) + E_C(c_j, C) \tag{1}$$

In Eq. 1, i represents a specific CNN layer, where t_i is the computational task in that layer and c_j is the data to be communicated through the communication channel C at the partitioning point j. E_P and E_C represent the energy functions required for processing and communication respectively, with P denoting the characteristics of the processing platform. Partition points for the TY3 architecture were explored, with a focus on layers that feature sequential data paths, aiming to reduce additional data transfers occurring within the parallel components of the TY3 network. The partitioning points for TY3 are depicted in Fig1.

For each partitioning point, the model is split into two parts: $M_{1..j}$, which undergoes pruning and is processed on the IoT node, and $M_{j+1..N}$, which is processed on the edge node. Following the work in [20], the two models are combined through an interface I (highlighted in gray in Fig. 2), where

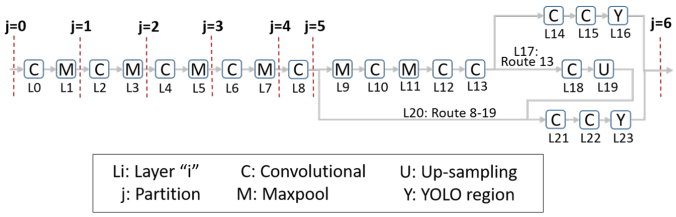

Fig. 1. Abstract model highlighting the sequential data path of the Tiny YOLO-v3 architecture and the resulting partitioning points according to [20].

quantization operations are carried out. The result is a transformed CNN model, denoted as M_p as shown in Eq. 2.

$$M_p = M_{1..j} \cup I \cup M_{j+1..N} \tag{2}$$

Equation 3 describes an optimization problem in which the goal is to minimize the energy terms, subject to a constraint on the mean Average Precision (mAP) metric in Eq. 4.

$$\min_{(t_{1..j}),\, c_j} \sum_{i=1}^{j} (E_P(t_i, P)) + E_C(c_j, C) \tag{3}$$

Equation 4, represents the constraint for the optimization problem, where the threshold th defines the acceptable reduction in mAP during the optimization of the energy terms in Eq. 3.

$$mAP(M_p) \geq mAP(M) - th. \tag{4}$$

A. Quantization

In previous work, the effects of quantization from 8-bit down to 1-bit representation were analyzed and proved that 1-bit quantization can retain a high detection accuracy for the given model [20]. In this work, we quantize the data to a 1-bit precision level using the normalization formula in Eq. 5,

$$d_j^b = \lfloor (f_j - \mu) \,/\, max \rfloor \qquad \text{with } b = 1. \tag{5}$$

where d_j^b is the quantized data at a partition point j and b the bit-precision which in this case is 1. f_j is the floating point representation of the data at the output of the partitioning point j, μ is the mean pixel value of the entire dataset, while max is the maximum value of the dynamic range.

$$f_j' = \left(d_j^b \times max\right) + \mu \quad where, b = 1. \tag{6}$$

It is important to note that in an IoT device, the data is quantized at the partition point. Therefore, when this data is sent to the edge device, it needs to be converted from quantized representation to floating point representation (f_j'). The process of converting back to floating point representation is defined by Eq. 6. It is worth mentioning that f_j' may not be equal to f_j because the quantization process results in changes in the values. This quantization and de-quantization process is represented in Eq. 2 by the interface I.

B. Pruning

The pruning in the IoT node is done in two steps. In the first step, the data sent between the IoT node and the edge node c_j over interface I is minimized, by pruning layer j denoted as γ_j.

$$\min_{(\gamma_j)} [c_j(\gamma_j)] \tag{7}$$

The constraint applied to Eq. 7 is given by:

$$mAP(M_p') \geq mAP(M) - th \tag{8}$$

where, M_p' represents the model M_p with all layers up to the pruned layer j.

Removing the filters in the last node layer will leave many redundant filters in the previous layers that produce input to the pruned layer γ_j, giving the opportunity to reduce the overall computational processing task. Thus, in the next step, the total processing task of the model is minimized, up until the partition point j, by removing all the redundant filters in all the previous layers.

$$\min_{(\gamma_{1..j-1})} [t_{1..j-1}(\gamma_{1..j-1})] \tag{9}$$

The constraint on mAP in Eq. 9, is given by

$$mAP(M_p'') \geq mAP(M) - th \tag{10}$$

where M_p'' is M_p with pruning in all the layers from layer 1 to the partitioned layer. In the process outlined within this study, the last stage prior to deploying the model on the edge device is compression. The data present at the partitioning point is compressed using ZIP compression, and the data is subsequently transmitted through a designated communication channel. Consequently, the initial operation performed by the edge node involves decompression. A comprehensive depiction of the entire workflow undertaken in this research can be found in Fig. 2.

To summarise, for every partition point considered in the CNN model, initially an interface I is created, where quantization and de-quantization are perfomed on both sides of the

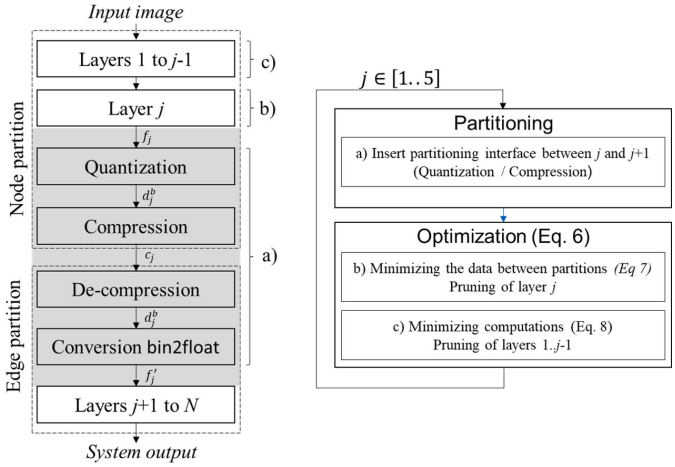

Fig. 2. Flow chart of the work process done in this work.

network, in the IoT node and in the edge node, respectively. This is followed by a two-step pruning on the IoT node, The first step involves pruning the partition layer j, followed by pruning its preceding layers to eliminate redundant filters. The key constraint in this optimization procedure is to retain the accuracy drop within the predefined threshold th.

IV. Results

A. Experimental setup

The scope of the presented method is to reduce the overall energy consumption at the IoT node as a combination of node offloading, quantization, and pruning. For the scope of this analysis, TY3 is implemented in both the IoT and the edge node to evaluate different node offloading configurations. The energy and latency metrics for processing the TY3 model in the IoT node were obtained from the work of Ivanov [22]. The implementation of TY3 and calculation of energy and latency metrics including the performance metrics were done on a workstation implementation with a 12^{th} Gen Intel(R) Core(TM) i9-12900 with a 2.40 GHz CPU, 32GB of RAM and with NVIDIA GeForce RTX 3090 GPU. The energy consumption and latency metrics of several communication technologies used in this analysis were derived from the estimation model of Krug et al. [23].

Fig. 1 shows the partitioning points j, where $j \in [1:5]$ in TY3 architecture as defined in [20]. The rationale for partitioning the model at these points is that data flow at these layers is sequential, allowing for more streamlined data processing. Apart from these five partitioning configurations, two more cases are included, one when all the processing is done in the edge device ($j = 0$) and the other when all the processing is done in the IoT node ($j = 6$). The partitioning part remains the same as in our previous work since we focus on the combination with pruning in this work. The TY3 model was trained using the MIUN-Feet dataset [9] as a baseline. The training process resulted in a 99.11% mAP at an intersection over union (IOU) of 0.5 and a confidence level of 0.45. To optimize for IoT deployment, we partitioned the CNN model, introducing quantization and pruning at the partition layer. Finally, we fine-tuned the model by retraining it to compensate for the loss it encountered during quantization and pruning.

B. Step 1 - Pruning only in layer j

In the present study, feature maps are strategically pruned at designated partitioning points. This pruning inevitably results in information loss, consequently affecting the mAP of the model, thus retraining is employed to address this issue. Upon retraining, a substantial improvement in overall accuracy is observed. In particular, at j=1, 2 and 3, the model maintained an accuracy exceeding 90% despite high pruning levels of 95%, 98%, and 99%, respectively, with only a single feature map remaining. Conversely, at partitioning points 4 and 5, the accuracy declined below 90% under the same number of remaining feature maps. To ensure the model's mAP remains above 95%, a systematic examination of different pruning

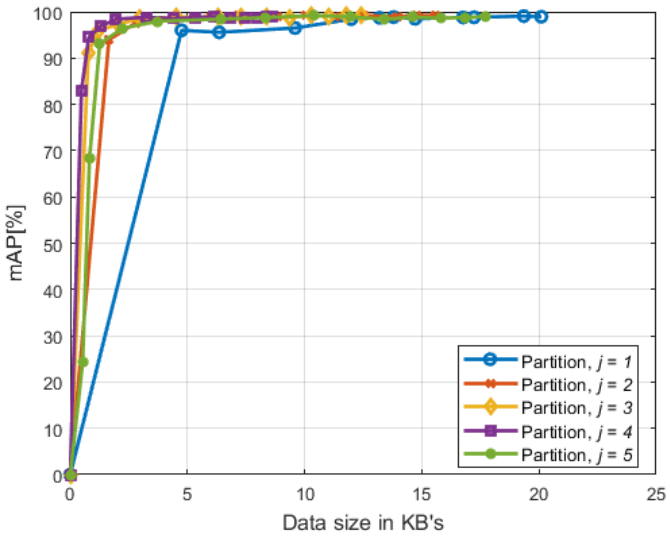

Fig. 3. mAP for models partitioned in j [1...5] vs. the size c_j of the data transferred from the partitioning point.

percentages was conducted at each j. After implementing the pruning process, the data at j undergoes compression and packing prior to being offloaded to the edge device. Fig. 3 illustrates the amount of remaining data at each partitioning point after the processes of pruning, compression, and packing have been applied. For all partitioning points, it can be observed that when achieving an mAP of over 95% across all partitions, the data required to be offloaded is less than 5 kByte.

C. Step 2 - Pruning in all the layers $[1..j]$

The filters at the partitioning point are fed with inputs from the previous layers $j-1$, and when filters at j are removed, the inputs to these removed filters are also eliminated. This reduction in input can lead to the presence of redundant filters in the previous layers, which in turn presents an opportunity for pruning in all these previous layers to reduce the overall computation. The overall energy consumption of the neural network can be significantly reduced by decreasing the computation required in the previous layers, specifically from $t_1 - t_{j-1}$.

Pruning all the layers from $j = 1$ to $j = 5$ reduces t_i which can be calculated by the number of multiplication accumulators that the model has to compute, as the layers are pruned, a subsequent reduction in the number of required multiplications is observed, contributing to a decrease in the overall energy requirement. Fig. 4 illustrates the the effect of reduction of multiplication accumulators on the mAP of the CNN model.

Our findings indicate that even when the remaining t_i is around 10% of the original model, the mAP exceeds the 95% threshold for all partitioning points. This reduction in computational tasks is a critical factor in making neural networks more practical for real-world IoT applications, especially for

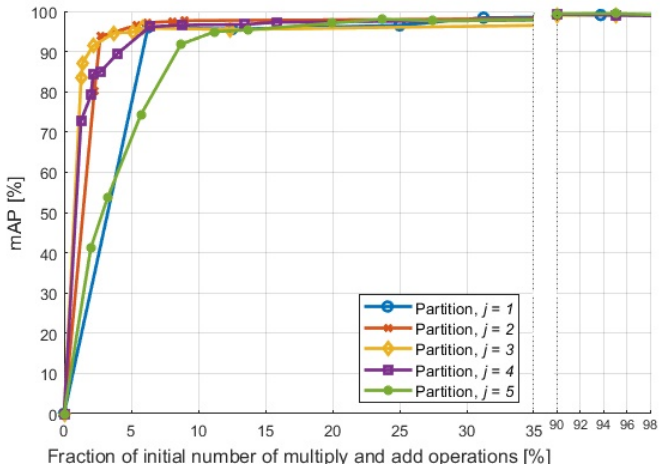

Fig. 4. mAP for the partitioned model M'' vs. the remaining multiplication and add operations after the pruning of layers 1 to $j-1$.

devices with limited computational resources such as mobile devices and embedded systems.

D. Energy and Latency

The energy requirement of the node after partitioning is the summation of processing energy and the communication energy required to transfer the data from the IoT node to the edge node for a given communication technology. In this case study, four common IoT communication models i.e BLE-4, BLE-5, LTE-C4, and Wi-Fi, were utilized and the amount of energy required for data transmission and induced latency were analyzed. Fig. 5 gives us information about the latency and energy requirements at each partitioning point.

The findings presented in Fig. 5 demonstrate that partitioning a CNN model results in an optimal solution in terms of both latency and energy consumption. This is particularly true when compared to the scenarios where all processing occurs solely on the edge device or the IoT node without partitioning.

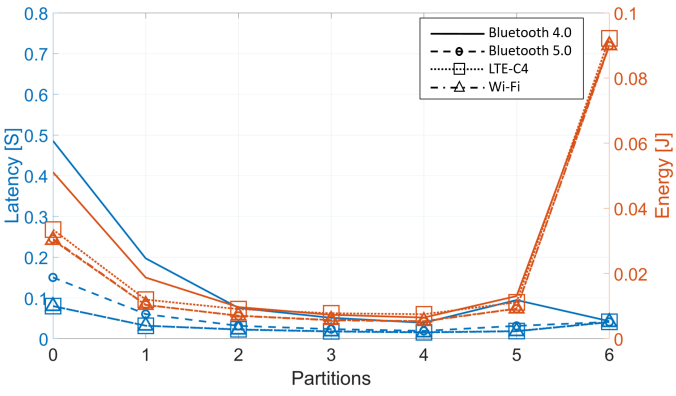

Fig. 5. Energy and latency after partitioning and pruning the CNN model for four different communication technologies.

Partitioning points ranging from $j = 1$ to $j = 5$ offer lower latency and energy consumption compared to $j = 0$ and $j = 6$

V. Discussion

Table I summarises the results obtained from the proposed method, drawing attention to the enhanced performance in energy consumption and latency. Each cell of this table is derived as a comparison in performance between the optimal partitioning point and the baseline model implemented either fully in-edge ($j = 0$) or fully in-node ($j = 6$). For the case without pruning, the optimal partition point is a combination of partition $j = 1$ and Wi-Fi communication technology. Instead, in the case of pruning, partitioning point $j = 4$ with BLE-5 is the most energy-efficient point while from the latency point of view same partitioning point $j = 4$ but with Wi-Fi is the best solution. By leveraging optimal partitioning points and advanced communication technologies, the proposed method achieves an astounding 17 times in energy savings as compared to processing all the models in an IoT node while significantly reducing the latency as well. These results emphasize the efficacy of partitioning as a viable approach for improving the performance and energy efficiency of CNN models.

The scope of the proposed method is to enable the energy-efficient design of CNN-based IoT nodes. However, considering that this IoT-node will be part of a larger system that transfers data through a communication channel, then it becomes necessary to include the latency component in our trade-off analysis. Fig. 5 presents the results from both these perspectives, visualizing the conflicting nature between latency and energy consumption for a smart IoT node. This also indicates that the optimum is on neither extremity of the plot, but somewhere in the middle, relying on node offloading.

Upon closer examination of Fig. 5, it becomes evident that both energy consumption and latency exhibit significant variations, which are dependent on the chosen partitioning point and the utilized communication model. For instance, when BLE-4 is used for data transmission, the energy consumption is significantly higher than that of other communication models at all partitioning points. This highlights the importance of selecting the most suitable communication technology for IoT nodes, as it can significantly impact overall system performance and efficiency.

Across the communication technologies examined, the majority of energy consumption results from the computational workload of the CNN and data compression, with the excep-

TABLE I
Advantages of the Partitioning Design Solution.

	Without Pruning		With Pruning	
	All In-Edge	**All In-Node**	**All In-Edge**	**All In-Node**
Energy saving	x1.26	x3.8	x5.74	x17
System speed-up	x2.05	x1.05	x5.24	x2.65

tion of the cases relying on BLE-4 technology. The negligible impact of communication energy on the overall energy consumption can be attributed to the significant reduction in the data size after pruning and compression, thereby requiring less data to be transmitted from the IoT device to the edge node. Consequently, the required communication bandwidth and energy are minimized, facilitating an energy-efficient IoT node.

The latency analysis offers a comprehensive view of the CNN models' performance at different partitioning points across various communication technologies. The results in Fig. 5 show that the communication overhead is a significant contributor to the overall latency of the model, especially for BLE-4 and BLE-5 technologies. These findings suggest that optimizing the communication between the IoT and the edge is essential to minimize the overall latency for the execution of the CNN model. When examining the partitioning points for BLE-4, it is observed that the lowest latency is achieved when all processing occurs in the node, i.e., $j = 6$. However, at this partitioning point, the energy consumption is also the highest, making it an unfeasible option for practical deployment. Instead, for the remainder of the communication technologies analyzed, partitioning $j = 4$, emerges as the most favorable choice, with the lowest latency and energy requirements. Upon evaluating the impact of pruning on the partitioned CNN model, it becomes apparent that the amount of data reduction achieved is substantial. Consequently, in terms of energy consumption and latency, the processing and communication within the CNN model are not major factors, whereas the compression method employed becomes the primary energy consumer.

It is evident that prior to pruning, the effect of compression on energy consumption and latency is negligible. However, after pruning and subsequent reduction of the model's energy and latency, compression emerges as a critical factor. Thus, the energy efficiency of the compression method becomes a critical consideration in determining the optimal partitioning point, which may vary across different communication technologies. Future research could explore various compression methods to further enhance the energy efficiency of the model and facilitate its implementation on smaller IoT devices.

VI. Conclusions

This paper presents a study that demonstrates the effectiveness of off-loading IoT nodes by partitioning CNN models. The results show that this approach can lead to up to 17 times more efficient implementation compared to running the CNN models entirely on the node or on an edge server. The efficiency improvement is enabled by decreasing communicated data size and computational load through pruning in combination with quantization and compression.

To further validate these results, the plan is to extend our investigation to other CNN models and datasets. We also aim to improve the partitioning method by using more efficient compression techniques and automating the design transformation process. Overall, our study highlights the potential of node-offloading, including pruning, as a means of achieving efficient CNN-based imaging in IoT contexts.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 12 2015.

[3] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[4] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1501–1510.

[5] M.Courbariaux, Y.Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

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

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[8] C. V. Giménez, S. Krug, F. Z. Qureshi, and M. O'Nils, "Evaluation of 2d-/3d-feet-detection methods for semi-autonomous powered wheelchair navigation," *Journal of Imaging*, vol. 7, no. 12, p. 255, 2021.

[9] C. Vilar, "MIUN dataset för semi-autonom manövrering av eldrivna rullstolar," 2021. [Online]. Available: https://doi.org/10.5878/k44d-3y06

[10] Y.Choi, M.El-Khamy, and J.Lee, "Towards the limit of network quantization," *arXiv preprint arXiv:1612.01543*, 2016.

[11] H.Jegou, M.Douze, and C.Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[12] Y. Wei, X. Pan, H. Qin, W. Ouyang, and J. Yan, "Quantization mimic: Towards very tiny cnn for object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 9 2018.

[13] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "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.

[14] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2498–2507.

[15] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[17] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "Modnn: Local distributed mobile computing system for deep neural network," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1396–1401.

[18] I. Shallari, S. Krug, and M. O'Nils, "Communication and computation inter-effects in people counting using intelligence partitioning," *Journal of Real-Time Image Processing*, vol. 17, no. 6, pp. 1869–1882, 2020.

[19] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[20] I. Sanchez Leal, E. Saqib, I. Shallari, A. Jantsch, S. Krug, and M. O'Nils, "Waist tightening of cnns: A case study on tiny yolov3 for distributed iot implementations," in *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, 2023, pp. 241–246.

[21] F. Martins Campos de Oliveira and E. Borin, "Partitioning convolutional neural networks to maximize the inference rate on constrained iot devices," *Future Internet*, vol. 11, no. 10, p. 209, 2019.

[22] M. Ivanov, "Embedded machine learning demonstrator," Bachelor Thesis, TU Wien, 2021. [Online]. Available: https://publik.tuwien.ac.at/files/publik_296007.pdf

[23] S. Krug and M. O'Nils, "Modeling and comparison of delay and energy cost of iot data transfers," *IEEE Access*, vol. 7, pp. 58 654–58 675, 2019.