

RESEARCH ARTICLE

TCL: Time-Dependent Clustering Loss for Optimizing Post-Training Feature Map Quantization for Partitioned DNNs

OSCAR ARTUR BERND BERG^{1,2}, EIRAJ SAQIB¹, AXEL JANTSCH^{1,2}, (Fellow, IEEE), IRIDA SHALLARI¹, (Member, IEEE), SILVIA KRUG³, ISAAC SÁNCHEZ LEAL¹, AND MATTIAS O'NILS¹, (Member, IEEE)

¹Department of Computer and Electrical Engineering, Mid Sweden University, 85230 Sundsvall, Sweden

²Institut für Computertechnik, TU Wien, 1040 Wien, Austria

³Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH (IMMS GmbH), 98693 Ilmenau, Germany

Corresponding author: Mattias O'Nils (mattias.onils@miun.se)

This work was supported by The Swedish Knowledge Foundation.

ABSTRACT This paper introduces an enhanced approach for deploying deep learning models on resource-constrained IoT devices by combining model partitioning, autoencoder-based compression, quantization with Time Dependent Clustering Loss (TCL) regularization, and lossless compression, to reduce communication overhead, minimizing latency while maintaining accuracy. The autoencoder compresses feature maps at the partitioning point before quantization, effectively reducing data size and preserving accuracy. TCL regularization clusters activations at the partitioning point to align with quantization levels, minimizing quantization error and ensuring accuracy even with extreme low-bitwidth quantization. Our method is evaluated on classification models (ResNet-50, EfficientNetV2-S) and an object detection model (YOLOv10n) using the TinyImageNet-200 and Pascal VOC datasets. Deployed on Raspberry Pi 4 B and GPU, each model is tested across various partitioning points, quantization bit-widths (1-bit, 2-bit, and 3-bit), communication datarate (1MB/s to 10MB/s), and LZMA lossless compression. For a partitioned ResNet-50 after the convolutional stem block, the speed-up against a server solution is 2.33× and 1.85× compared to the all-in-node solution, with only a minimal accuracy drop of less than one percentage points. The proposed framework offers a scalable solution for deploying high-performance AI models on IoT devices, extending the feasibility of real-time inference in resource-constrained environments.

INDEX TERMS CNN, partitioning, quantization, IoT.

I. INTRODUCTION

The rapid growth of Internet of Things (IoT) devices has led to a new era of interconnected systems, supporting applications from smart homes to industrial automation. However, deploying deep learning models on these resource-constrained devices presents challenges due to their limited processing capacity [1], [2]. Deep learning models are typically too resource intensive for direct deployment on IoT devices due to high computational demands [3]. To address this, techniques for model compression (e.g., quantization,

pruning, knowledge distillation) are used to reduce model size and processing needs, although they can compromise accuracy [4]. Another approach is offloading computation to cloud servers, though this adds communication delays, which is problematic for real-time applications.

In this paper, we propose a methodology to address the challenges of deploying deep learning on IoT devices by integrating model partitioning, autoencoder deployment, specialized regularization function called Time-dependent Clustering Loss (TCL) and activation quantization. Our approach divides the model into two parts: the initial layers are processed on the IoT device, while the remaining layers are handled on a server. This reduces the data transmission

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

by leveraging the local computational power. In this work, our goal is to minimize the latency on the IoT-node by balancing communicational load at the partitioning point (PP) and model's accuracy. To reduce the communication load, we apply n-bit quantization (1-3 bits) to activations at the PP. Rather than standard Post-Training quantization (PTQ) [5], [6], TCL guides activations to cluster around centroids initialized by K-means clustering of feature maps before re-training with TCL, minimizing quantization error. This technique prepares activations for efficient quantization, aligning them to discrete values with minimal loss, and integrates seamlessly with standard task-specific training for applications like image classification or object detection. To further reduce communication overhead, we apply lossless compression via the Lempel-Ziv-Markov chain Algorithm (LZMA) to the quantized activations before transmission. This step minimizes data size without losing information, thereby lowering communication latency while maintaining accuracy.

Our contributions can be summarized as follows:

- We introduce **Time Dependent Clustering Loss (TCL)**, a specialized loss function that encourages activation clustering toward centroids during training. Afterward, we quantize these clustered activations to discrete levels based on the selected bit-width.
- **Comprehensive Deployment Scenarios:** We analyze the deployment of classification models (ResNet-50, EfficientNetV2-S) and an object detection model (YOLOv10n) on TinyImageNet-200 and PASCAL VOC datasets. Using different communication datarates, we explore the trade-offs in latency, and accuracy across different quantization levels and partitioning points.

The paper is organized as follows: Section II reviews related work on deploying deep learning models on resource-constrained devices. Section III details our methodology, including model partitioning, quantization, custom loss function design, and compression techniques. Section IV presents the experimental setup and results, while Section V discusses these findings. Finally, Section VII concludes the paper and outlines future research directions.

II. RELATED WORK

Deploying deep neural networks (DNNs) on resource-constrained devices such as IoT nodes has motivated extensive research on reducing computation, storage, and communication demands. We review the literature along four key dimensions: (i) model compression and hardware acceleration, (ii) model partitioning and collaborative inference, (iii) quantization techniques, and (iv) autoencoder-based intermediate data reduction, and discuss how our approach builds on or contrasts with these studies.

A. MODEL COMPRESSION AND HARDWARE ACCELERATION

A substantial body of work has focused on compressing DNNs to reduce computational and storage costs. Surveys

by Deng et al. [1] and Li et al. [3] provide comprehensive overviews of techniques including pruning, quantization, and knowledge distillation. Mishra et al. [4] highlight the inherent challenges in compressing DNNs, while Hoyer et al. [2] explore hardware accelerators that facilitate real-time inference.

Pruning methods have also been widely adopted to eliminate redundant parameters and reduce intermediate feature map sizes. For example, Ruan et al. [7] propose a dynamic and progressive filter pruning strategy for compressing convolutional networks, and Saqib et al. [8] demonstrate that pruning can significantly reduce communication overhead in distributed settings.

Our work integrates these compression principles by incorporating a quantization scheme guided by our novel Time-dependent Clustering Loss (TCL) into a partitioned architecture, and thus leverages existing insights on reducing model complexity and extends them by tightly coupling quantization with intermediate data compression.

B. MODEL PARTITIONING AND COLLABORATIVE INFERENCE

Distributed inference techniques have been broadly classified into vertical and horizontal partitioning. Vertical partitioning splits the network along its depth allocating early layers to the IoT device and later layers to a server to reduce on-device computation and communication demands. In contrast, horizontal partitioning distributes computations within the same layer without modifying the overall architecture. Detailed comparisons of these paradigms can be found in [9] and [10]. Early approaches like Neurosurgeon [11] and DeepSplit [12] focused on finding optimal partition points by balancing computational and communication costs, while more recent techniques dynamically select partition points as seen in the work of Zhang et al. [13] and the edge-cloud framework proposed by Tian et al. [14]. Additionally, Sanchez Leal et al. [6] offer strategies tailored to distributed IoT systems.

Our work adopts a vertical partitioning strategy but distinguishes itself by integrating an autoencoder for intermediate data compression and applying TCL to guide quantization at the partition point. This integration not only optimizes the transmission of intermediate data but also enhances the overall efficiency of the collaborative inference process.

C. QUANTIZATION TECHNIQUES

Quantization methods are pivotal for reducing model size and communication overhead by converting full-precision representations to lower-bitwidth formats. Early methods such as post-training quantization [5] and quantization-aware training (QAT) [15], [16] have effectively reduced precision while retaining accuracy. More advanced approaches, including BNN+ [17] and adaptive layerwise quantization [18], aim to further reduce quantization error by refining the distribution of activations. Additionally, distillation-based

strategies [19] and architectures like BottleNet [20] integrate dedicated modules for data reduction.

Our approach builds on these foundational techniques by introducing TCL, a regularization method that explicitly encourages activations to cluster around discrete centroids. By doing so, our method minimizes quantization error even under extreme low-bitwidth conditions, thereby preserving accuracy while substantially reducing the communication load in partitioned DNN deployments.

D. AUTOENCODER-BASED INTERMEDIATE DATA REDUCTION

Recent studies have demonstrated that autoencoder architectures can effectively compress intermediate feature maps, reducing communication overhead in distributed deep learning systems. Li et al. [21] show that deep autoencoders can reduce the dimensionality of representations with minimal loss in accuracy. Concurrently, Yao et al. [22] propose deep compressive offloading strategies that leverage edge computation to mitigate network latency, while surveys by Wang et al. [9] and Duan et al. [10] provide broader perspectives on end-edge-cloud collaborative computing architectures.

While the literature has extensively addressed individual aspects such as compression [1], [3], [4], partitioning [6], [11], [12], [13], [14], quantization [5], [15], [16], [17], [18], [19], and data reduction through both pruning and autoencoder-based methods [7], [8], [21], [22], our work integrates these components. By leveraging TCL-based quantization together with an autoencoder for intermediate data compression, we achieve a balanced trade-off between inference accuracy, communication latency, and model size. This integrated strategy—supported by both foundational and recent works—advances scalable DNN deployment on resource-constrained devices.

III. METHODOLOGY

This section outlines our methodology for deploying large deep learning models on resource-constrained Internet of Things (IoT) devices. It includes model partitioning, quantization at Partitioning Points (PPs), and an integrated autoencoder-based compression technique to reduce communication overhead. We evaluate our method on classification and object detection tasks using ResNet-50, EfficientNetV2-S, and YOLOv10n models on the TinyImageNet-200 and PASCAL VOC datasets.

A. MODEL PARTITIONING STRATEGY

We consider potential PPs at sequential locations within the model’s computational graph. Furthermore, we restrict them to those that exhibit distinct shapes compared to other layers. To adapt complex neural networks for resource constrained IoT devices, we use a partitioning strategy that divides the model into two segments: One executed on the IoT device, denoted as $M_{IoT}^{1..i}$, and the other processed on an unconstrained server, denoted as $M_{server}^{i+1..N}$, where i represents

the PP of N possible points. The PPs are chosen at locations where the data flow is sequential, avoiding layers with parallel data paths as controlling and synchronizing parallel outputs is complex and resource intensive.

The partitioned model, incorporating the IoT- and server model, is defined as:

$$M' = M_{IoT}^{1..i} \cup I \cup M_{server}^{i+1..N} \quad (1)$$

The interface I handles the transmission of activations from the IoT device to the server. By carefully selecting the PP i , we aim to balance the computational load between the device and the server while minimizing the communication overhead.

B. COMPRESSION PIPELINE

We introduce an autoencoder-based compression mechanism at the partitioning point, followed by TCL quantization and subsequent Post training quantization (PTQ). The autoencoder is trained jointly with the rest of the model. This not only reduces the communication load but also maintains the relevance and effectiveness of TCL in optimizing post-training feature map quantization. During inference, we deploy LZMA as a lossless compression to further reduce the datasize at this point, reducing the communicational overhead between IoT and server device.

The autoencoder architecture is designed to compress feature maps at the partitioning point efficiently. It consists of the following layers:

- Encoder:
 - Conv2D Layer 1: Kernel size of 1×1 reduces the number of channels by a factor of 4.
 - Conv2D Layer 2: Reduces the spatial dimensions (height and width) by half, achieving an overall reduction in feature map size by a factor of 16.
 - Activation and Normalization: SiLU activation and Batch Normalization are applied after each layer to enhance non-linearity and maintain stable learning.
- Decoder:
 - Transposed Convolutional Layers: The decoder is symmetric to the encoder, using transposed convolutional layers to reconstruct the compressed feature maps to their original dimensions.
 - Activation and Normalization: Similar to the encoder, Batch Normalization and SiLU activation are used.

C. TIME DEPENDENT CLUSTERING LOSS (TCL)

To prepare activations at the PP for efficient quantization, we use the Time Dependent Clustering Loss (TCL), denoted as L_{TC} . This loss function encourages activations to cluster around predefined centroids, adapting their distribution for better quantization.

We build on “BNN+: Improved Binary Network Training” [17], originally designed for 1-bit quantization, to support various bit-widths. TCL aligns activations with quantization levels during training, mitigating the

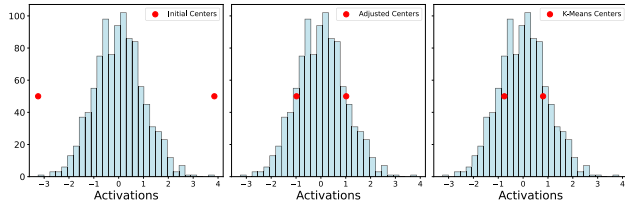


FIGURE 1. K-Means clustering process on the feature map distribution before training.

performance degradation typically seen with quantization and enhancing model accuracy post-quantization.

Before training the model with TCL-regularization, we initialize the cluster centroids by applying K-Means clustering to the activations at the PP_i of the pre-trained model, as illustrated in Figure 1. Initially, the model is trained with the autoencoder inserted and K-Means clustering is applied afterwards at the feature maps at the partitioning point within the autoencoder. The number of clusters K is set to 2^n , where n represents the desired bit-width for quantization. This clustering captures the inherent structure of the activations, providing meaningful reference points for the TCL function.

For 1-bit quantization, the TCL-regularization function is defined as:

$$L_{TC}(x, T) = \mathbb{E} \left[|\alpha - |x - \beta||^T \right] \quad (2)$$

$$\alpha = \frac{\mu_2 - \mu_1}{2} \quad (3)$$

$$\beta = \frac{\mu_1 + \mu_2}{2} \quad (4)$$

where x denotes the activation values, T is number of epochs, α and β are non-trainable parameters, which are initialized based on the cluster center values (i.e., μ_1 and μ_2), which remain static during the TCL-guided training phase and τ is a temperature parameter decreasing over epochs T . As illustrated in Figure 2, the training process iteratively encourages the activation values to move toward the centroids. During the early stages of training, a higher temperature τ results in a smoother loss landscape, allowing the activations to make substantial adjustments and shift towards the centroids. This movement enables the layers preceding the quantized layer to adapt their activations to align with the quantization levels. As training progresses, τ is gradually decreased, sharpening the loss landscape and leading to finer tuning of activations around the centroids. This process reduces the variance of activations around the centroids, preparing them for effective quantization.

For 2-bit quantization, the activations are clustered into four centroids μ_1, μ_2, μ_3 , and μ_4 . The TCL-function is extended to accommodate the increased number of centroids:

$$C_{b_0} = \mathbb{E} \left[|\alpha_0 - |x - \beta_0||^T \right] \quad (5)$$

$$C_{b_1} = \mathbb{E} \left[|\alpha_1 - |x - \beta_1||^T \right] \quad (6)$$

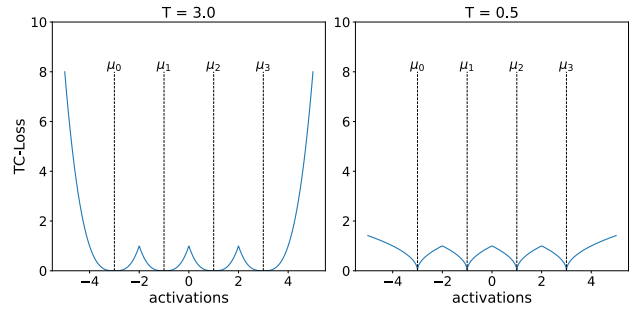


FIGURE 2. Clustering of activations into four centroids during training with TCL for $T = 3.0$ (left) and $T = 0.1$ (right).

$$D = \begin{cases} -C_{b_1}, & \text{if } x \leq I \\ -C_{b_0}, & \text{if } x > I \end{cases} \quad (7)$$

Here, I is the intersection of C_{b_0} and C_{b_1} , i.e.,

$$C_{b_1} = C_{b_0} \quad (8)$$

Using Equation 5, 6, and 7 we formulate the equation of TCL for 2-bit quantization:

$$L_{TC}^{2\text{bit}}(x, T) = C_{b_0} + C_{b_1} + D \quad (9)$$

As shown in Figure 2, increasing the number of centroids allows the model to represent the activations with finer granularity at the PP. This finer granularity refers to the increased resolution in representing activation values due to the higher number of quantization levels. This process can be generalized for any bit-width n . The TCL function is adapted by combining multiple 1-bit TCL functions with respect to each interval between centroids corresponding to 2-bit quantization (Eq. 5 to 9). By structuring the TCL regularization in this way, we can systematically guide the activations toward the quantization levels appropriate for n -bit quantization.

The temperature parameter τ plays a critical role in the training process by controlling the sharpness of the TCL function. By controlling τ , the training process first allows for significant shifts of activations toward the cluster centers and then refines these activations to closely match the quantization levels. This approach minimizes quantization errors at the PP, enabling the model to maintain high accuracy despite the reduced precision of activations due to quantization.

The total loss function used during retraining combines the task-specific loss L_{TS} (e.g., cross-entropy loss for classification tasks) with the TCL regularization term L_{TC} :

$$L_{total} = \lambda L_{TS} + (1 - \lambda) L_{TC} \quad (10)$$

where λ is a weighting factor between 0 and 1 that balances the contributions of the two losses.

D. QUANTIZATION OF ACTIVATIONS AT PP

After training the model with TCL, we apply quantization to the activations at the selected PP_i . The quantization process is

applied only at this point and involves mapping the clustered activations to their nearest centroids, effectively reducing the bit-width to the specified level (1, 2, or 3 bits). In addition, the server part of the model is retrained to adjust to the quantized activations, while the IoT part of the model up to the partitioning point remains frozen. The quantization levels can be efficiently stored in a Look-Up Table (LUT). At inference time, the server maps the transmitted bits via this LUT to its floating-point counterpart values. This post-training quantization converts the clustered activations into discrete values, preparing them for efficient transmission to the server. By quantizing the output activations of the IoT model segment $M_{IoT}^{1..i}$ before transmitting them, we reduce both the computational load on the IoT device and the data transfer size to the server. The alignment of activations with the quantization levels, achieved through TCL during training, reduces the variance of activations around the centroids, minimizing quantization error and preserving model accuracy.

E. TRAINING PROCEDURE

After inserting the autoencoder at the partitioning point, we train this model including the autoencoder without regularization for a defined number of epochs. Next, we initialize TCL cluster centroid values based on the K-Means clustering method and we re-train this model with TCL regularization subsequently. Finally, we incorporate a retraining phase following the post-training quantization of activations at the partitioning point. Specifically, the activations at the interface between the IoT-side and server-side components of the model are quantized, after which only the server-side portion is retrained to mitigate any performance degradation resulting from quantization. During this process, the weights of the IoT-side component are kept frozen up to the partitioning point.

During the TCL re-training phase, the model is trained for a set number of epochs, as specified in the training schedule. This approach ensures a consistent training duration and allows for controlled experimentation and comparison across different configurations.

We decrease the parameter τ by the following equation:

$$\tau = \frac{\tau_{end} - \tau_{start}}{num_{epochs}} \cdot e + \tau_{start} \quad (11)$$

where τ represents the temperature at epoch $0 \leq e \leq num_{epochs}$ and τ_{start} and τ_{end} are the start and end temperature values, respectively. Algorithm 1 highlights the K-Means clustering of the activations at the partitioning point i , and subsequent training with TCL-regularization.

IV. RESULTS

In this section, we present the results of deploying deep learning models—ResNet-50 (RN), EfficientNetV2-S (EN), and YOLOv10n (YOLO) on a resource-constrained IoT setup comprising a Raspberry Pi 4 (RP), with further processing offloaded to a server. We analyze our proposed method

Algorithm 1 Partitioned Model Training With TCL

Require: Pre-trained model M with already inserted Autoencoder at Partitioning point i , dataset DS , bitwidth bw , regularization factor λ TCLoss function $TCLoss$, Task-Loss Function $TaskLossFct$

Ensure: Trained & Partitioned model M_p

- 1: Extract feature maps at point i
- 2: Apply K-Means clustering on the distribution of feature maps for 2^{bw} cluster centroids
- 3: Initialize cluster centroids for TCL
- 4: **for** count, batch in enumerate(DS) **do**
- 5: $y, fm = M(batch)$ # fm : feature maps at point i
- 6: $loss = TaskLossFct(y)$
- 7: $loss_{TCL} = TCLoss(fm, i)$
- 8: $loss_{total} = \lambda \cdot loss + (1 - \lambda) \cdot loss_{TCL}$
- 9: $loss_{total}.backward()$
- 10: **end for**
- 11: **return** trained model M_p

against the partitioning method presented in [6], as it exhibits the greatest similarity to our approach. Furthermore, we investigate the impact of the proposed TCL regularization function on overall accuracy compared to the standard BNN+ quantization method described in [17].

Each model was evaluated across various PPs (from initial to later PPs) quantization levels (1-bit, 2-bit, and 3-bit), and communication datarates, ranging from 1 Mb/s to 10 Mb/s.

A. HARDWARE AND COMMUNICATION SETUP

The IoT segment of our partitioned models is implemented on Raspberry Pi 4 Model B, implementing a quad-core Cortex-A72 processor and 4 GB of RAM. We model the server with a NVIDIA GeForce® RTX 4090 Laptop version.

To transmit the quantized activations from the IoT devices to the server, we explore multiple communication datarate, ranging from 1MB/s to 10MB/s targeting low datarate environments. This variety allows us to evaluate the impact of different communication methods on accuracy and latency.

1) LOSSLESS COMPRESSION

To further reduce the communication overhead associated with transmitting quantized activations from the IoT device to the server, we apply lossless compression algorithms. Specifically, we utilize the LZMA compression to compress the quantized data without introducing additional information loss.

B. EVALUATION METRICS

To assess the performance and efficiency of the proposed methodology, we employ a set of evaluation metrics that quantify model accuracy, communication load, and latency. These metrics provide a rigorous and quantitative analysis of the tradeoffs involved in deploying deep learning models on resource-constrained IoT devices.

1) MODEL ACCURACY

For classification tasks, we measure the model's performance using Top-1 Accuracy. For object detection tasks, we use the mean Average Precision (mAP) metric, which calculates the average precision across all classes, providing a comprehensive measure of the model's detection performance. We calculate the task performance (mAP) by averaging over different intersection-over-union (IoU) ratios from 0.5 to 0.95.

2) COMMUNICATION LOAD

The Communication Load at a PP_i is defined as the total size of the quantized and compressed activations transmitted from the IoT device to the server.

3) LATENCY

The total Latency L_{total} is the sum of the computation latency L_{comp} from the input to the PP_i of the network, quantization and compression latency L_{QC} , and communication latency L_{comm} . The derivation of L_{comm} follows the modeling approach outlined in [23]:

$$L_{total}(P_i) = \sum_{i=0}^L L_{comm}(P_i) + L_{QC}(P_i) + L_{comp}(P_i) \quad (12)$$

By defining these evaluation metrics, we establish a comprehensive framework to quantitatively assess the performance, efficiency, and trade-offs of our proposed methodology. It allows us to make informed decisions to optimize for accuracy, and latency, which are critical factors in real world IoT applications.

C. COMPREHENSIVE EXPERIMENTAL OUTCOMES

We evaluate our method on three deep neural networks—ResNet50, EfficientNetV2-S, and YOLOv10n comparing our approach against baseline methods (PTQ, BNN+, and BottleNet) in terms of top-1 accuracy (classification models) and mean average precision (mAP) with a Intersection-over-Union (IoU) at 50% (YOLOv10n) and latency on the IoT-node at various partitioning points (PP1 to PP4). The accuracy-latency relationship is visualized in figures 3, 4 and 5.

For ResNet50, our approach achieves an accuracy of 77.12% at PP1 with 1 bit quantization and transmitted data size of 2.4kB. In comparison, the PTQ baseline reports 74.81%, BNN+ achieves 75.78%, with similar size of 2.4kB and BottleNet reduces the data to 0.5kB but reaches only 71.20% at PP1. As the partition point moves deeper into the network, our accuracy improves modestly 77.28% at PP2 (with 9.1kB) and 77.40% at PP3 (with 4.7kB). Figure 3 illustrates the latency for ResNet50, at 1-bit quantization which ranges between 37.5ms and 25ms for PP1 ranging from 1Mb/s to 10 Mb/s, indicating minimal overhead from our autoencoder-based compression and TCL-guided quantization.

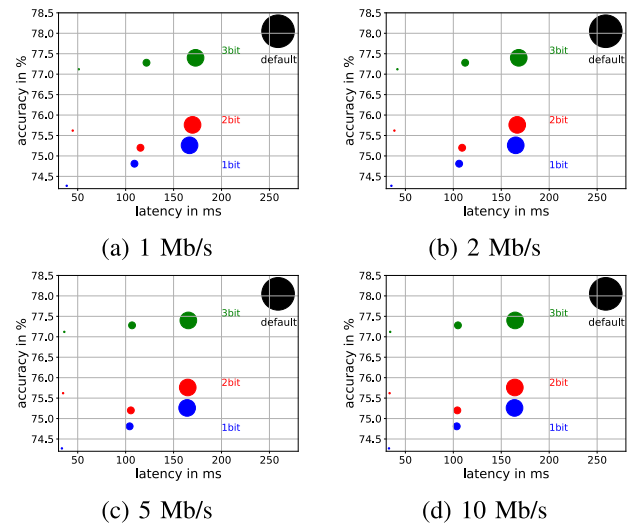


FIGURE 3. Accuracy vs. latency for ResNet50 at varying quantization levels. Blob size indicates the partitioning layer—the smaller the blob, the earlier the PP, meaning a smaller model on the IoT device. Default refers to the full precision unpartitioned model on the IoT device. Subfigures (a)–(d) represent communication rates of 1, 2, 5, and 10 Mb/s, respectively.

Similarly for 1-bit quantization of EfficientNetV2-S, our method obtains an accuracy of 83.95% at PP1 with a data size of 3.5kB. Accuracy increases gradually to 84.06% at PP2, 84.17% at PP3, and reaches 84.32% at PP4. Meanwhile, the data size decreases from 3.5kB at PP1 and PP2 to 1.8kB at PP3 and 0.6kB at PP4. In contrast, the PTQ and BNN+ baselines report lower accuracies (78.90% and 81.27% at PP1, respectively) with the same data size as of our model. While as BottleNet reduces the datasize to 0.7kB at the same PP albeit with a lower accuracy of 77.71%. Figure 4 shows that the latency for EfficientNetV2-S ranges from 61.2ms to 48.6ms, demonstrating that our approach maintains competitive real-time performance while reducing the communication payload.

For the YOLOv10n model, our method achieves an accuracy of 82.15% at PP1 with a data size of 18.4kB at 1-bit quantization. As the partition point shifts deeper (PP2 and PP3), accuracy increases slightly to 82.40% and 82.73%, respectively, and reaches 82.88% at PP4 while the data size drops to 4.6kB. In comparison, the PTQ and BNN+ baselines yield accuracies of 77.90% and 80.59% at PP1, and datasize of 18.4kB respectively. In addition the BottleNet paper reduces the data to 3.78kb but with a huge accuracy degradation to 78.71% Figure 5 shows that the latency for YOLOv10n lies within a narrow band (150ms–102ms) for PP1 at 1-bit, confirming that our integrated method does not impose significant delay.

D. OVERALL PERFORMANCE

Table 1 summarizes the key performance metrics (accuracy and data size) across different partition points and quantization schemes. The results across all three models

TABLE 1. Partitioning points (PP1 to PP4) for different models, with and without TCL for 3 bit Post-Training Quantization, including accuracy and data size details.

Model	Algorithm	PP1		PP2		PP3		PP4	
		Accuracy	Data size	Accuracy	Data size	Accuracy	Data size	Accuracy	Data size
ResNet50	PTQ [6]	74.81	2.4kB	74.94	9.1kB	75.18	4.7kB	x	x
	BNN+ [17]	75.78	2.4kB	75.95	9.1kB	76.28	4.7kB	x	x
	BottleNet [20]	71.20	0.5kB	71.91	2kB	71.57	1kB	x	x
	Ours	77.12	2.4kB	77.28	9.1kB	77.40	4.7kB	x	x
EfficientNetv2-S	PTQ [6]	78.90	3.5kB	80.41	3.5kB	80.96	1.8kB	81.84	0.6kB
	BNN+ [17]	81.27	3.5kB	81.82	3.5kB	82.10	1.8kB	82.92	0.6kB
	BottleNet [20]	77.71	0.7kB	77.90	0.8kB	77.51	0.3kB	78.09	0.1kB
	Ours	83.95	3.5kB	84.06	3.5kB	84.17	1.8kB	84.32	0.6kB
YOLOv10n	PTQ [6]	77.90	18.4kB	78.29	9.2kB	77.99	9.2kB	79.59	4.6kB
	BNN+ [17]	80.59	18.4kB	81.03	9.2kB	80.95	9.2kB	81.06	4.6kB
	BottleNet [20]	78.71	3.78kB	78.13	1.90kB	79.09	1.88kB	79.02	0.95kB
	Ours	82.15	18.4kB	82.40	9.2kB	82.73	9.2kB	82.88	4.6kB

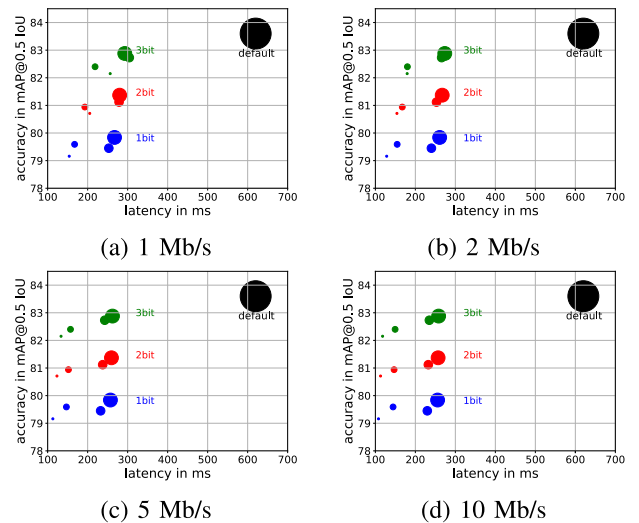
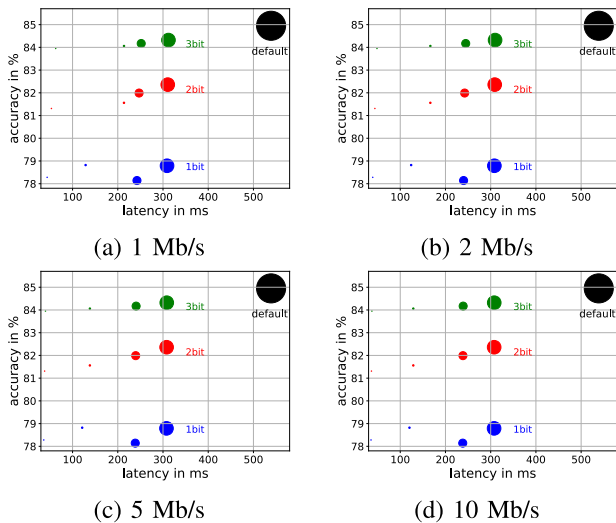


FIGURE 4. Accuracy vs. latency for EfficientNetV2-S. Color and blob and default represent the same as in Figure 3. Subfigures (a)–(d) show performance at different data rates.

FIGURE 5. Accuracy (mAP) vs. latency for YOLOv10n. Color and blob and default represent the same as in Figure 3. Subfigures (a)–(d) show performance at different data rates.

indicate that our integrated framework—combining vertical partitioning with an autoencoder-based compression module and TCL-guided quantization consistently outperforms the baseline methods in accuracy while substantially reducing the intermediate data size. Moreover, the latency overhead introduced by our approach remains minimal (typically within a few milliseconds), confirming its suitability for real-time inference in resource-constrained edge environments.

V. DISCUSSION

The deployment of deep learning models on resource-constrained IoT devices presents significant challenges due to limited computational resources and the stringent

requirements for real-time processing. Our study addresses these challenges by demonstrating the efficacy of AI model partitioning, in conjunction with activation regularization (TCL) and quantization, as a means to facilitate the deployment of complex models on IoT devices. This approach effectively balances accuracy, latency, and model size, thereby enabling practical implementations in resource-constrained environments.

One notable trend observed in our experiments is that later partitioning points generally result in higher accuracy. This phenomenon can be attributed to the autoencoder structure, where a greater number of trainable parameters, due to an increased number of channels, enhances the model's

TABLE 2. Model speed-up comparison at PP1 with 3-bit quantization for 1MB/s communicational data rate.

Model	Speed-Up Server	Speed-Up Node
RN	1.24	3.83
EN	1.04	4.76
YOLO	2.33	1.85

representational capacity. Furthermore, our results indicate a consistent increase in accuracy with higher bitwidths across all models and partitioning points. Specifically, a 3-bit quantization level is sufficient to maintain baseline accuracy, with an accuracy drop of less than 2 percentage points compared to the full-precision model.

Furthermore, we observe that the model size on the IoT node increases for later partitioning points, as a greater portion of the network remains on the edge device. Additionally, the latency of each IoT submodel decreases as the communication data rate increases.

Notably, for partitioned YOLOv10n models, the latency behavior at the first partitioning point exhibits an interesting trend. At a communication data rate of 1 MB/s, the first partitioning point results in a higher overall IoT latency than the second partitioning point. However, as the data rate increases to 2, 5, and 10 MB/s, this trend reverses, and later partitioning points begin to exhibit higher latencies on the IoT device. This highlights the crucial role of communication bandwidth in determining overall inference latency.

These findings emphasize that the total latency of the system is influenced by multiple factors, including the data size at the partitioning point, the computational latency of both submodels, and the available communication bandwidth. A careful selection of the partitioning point, tailored to the specific hardware and network conditions, is therefore essential to achieving an optimal balance between inference speed and model performance.

Comparing our partitioning framework with the baseline configurations—where inference is performed entirely on the IoT device (all-in-node) or offloaded completely to the server (all-in-server)—our approach achieves a significant speed-up over both. In particular, for the earliest partitioning point with 1MB/s communication datarate, we observe a substantial reduction in latency, as highlighted in Table 2.

To mitigate bandwidth constraints when transmitting input images directly to the server, we employ JPEG. This preserves baseline accuracy while reducing communication overhead.

Lastly, the relative speed-up between the Raspberry Pi 4 (RP4) and the server is highly dependent on both the communication data rate and the computational speed of the IoT device. These factors introduce variability in the observed speed-up.

VI. CONCLUSION

This work presents a methodology for deploying large deep learning models on resource-constrained IoT devices

by leveraging model partitioning, quantization with Time-Dependent Clustering Loss (TCL) regularization, and loss-less compression techniques. By implementing this approach on a Raspberry Pi 4 and evaluating it across low communication data rates (ranging from 1 MB/s to 10 MB/s), we have developed a flexible framework that adapts to the diverse constraints and requirements of real-world IoT environments.

For the partitioned YOLOv10n model at the first partitioning point, our approach achieved a speed-up of $\times 2.33$ compared to an all-in-server solution and $\times 1.85$ compared to an all-in-node solution. These results highlight the effectiveness of model partitioning, particularly for earlier partitioning points, in balancing the trade-off between inference latency on the IoT device and model accuracy.

Future work should focus on further optimizing the IoT-side model by employing additional compression techniques tailored to resource-constrained devices. Such improvements could enable even greater speed-ups and facilitate the deployment of increasingly complex deep learning models on low-power IoT hardware.

ACKNOWLEDGMENT

(OSCAR ARTUR BERND BERG AND EIRAJ SAQIB contributed equally to this work.)

REFERENCES

- [1] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.
- [2] I. Hoyer, O. Berg, L. Krupp, A. Utz, C. Wiede, and K. Seidl, "Hardware accelerators for a convolutional neural network in condition monitoring of CNC machines," in *Proc. IEEE SENSORS*, Oct. 2023, pp. 1–4.
- [3] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, Mar. 2023. [Online]. Available: <https://www.mdpi.com/2073-431X/12/3/60>
- [4] R. Mishra, H. Prabhat Gupta, and T. Dutta, "A survey on deep neural network compression: Challenges, overview, and solutions," 2020, *arXiv:2010.03954*.
- [5] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," 2015, *arXiv:1511.06488*.
- [6] I. S. 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 *Proc. Cyber-Phys. Syst. Internet Things Week*, May 2023, pp. 241–246.
- [7] X. Ruan, Y. Liu, B. Li, C. Yuan, and W. Hu, "DPFPS: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 3, pp. 2495–2503.
- [8] E. Saqib, I. S. Leal, I. Shallari, A. Jantsch, S. Krug, and M. O'Nils, "Optimizing the IoT performance: A case study on pruning a distributed CNN," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Jul. 2023, pp. 1–6.
- [9] Y. Wang, C. Yang, S. Lan, L. Zhu, and Y. Zhang, "End-edge-cloud collaborative computing for deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 4, pp. 2647–2683, 4th Quart., 2024.
- [10] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 591–624, 1st Quart., Nov. 2022.
- [11] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, Apr. 2017.
- [12] R. Mehta and R. Shorey, "DeepSplit: Dynamic splitting of collaborative edge-cloud convolutional neural networks," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 720–725.

[13] J. Zhang, M. Shu-Fang, Z. Yan, and J. Huang, "Joint DNN partitioning and task offloading in mobile edge computing via deep reinforcement learning," *J. Cloud Comput.*, vol. 12, no. 1, p. 116, Aug. 2023.

[14] Y. Tian, Z. Zhang, Y. Yang, Z. Chen, Z. Yang, R. Jin, T. Q. S. Quek, and K.-K. Wong, "An edge-cloud collaboration framework for generative AI service provision with synergetic big cloud model and small edge models," 2024, *arXiv:2401.01666*.

[15] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," 2020, *arXiv:2006.10518*.

[16] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*.

[17] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia, "BNN+: Improved binary network training," in *Proc. ICLR*, 2018, pp. 1–10.

[18] X. Zhu, W. Zhou, and H. Li, "Adaptive layerwise quantization for deep neural network compression," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[19] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*.

[20] A. E. Eshratifar, A. Esmaili, and M. Pedram, "BottleNet: A deep learning architecture for intelligent mobile cloud computing services," 2019, *arXiv:1902.01000*.

[21] N. Li, A. Iosifidis, and Q. Zhang, "Attention-based feature compression for CNN inference offloading in edge computing," in *Proc. IEEE Int. Conf. Commun.*, Jan. 2022, pp. 967–972.

[22] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proc. 18th Conf. Embedded Networked Sensor Syst.*, Nov. 2020, pp. 476–488.

[23] S. Krug and M. O’Nils, "Modeling and comparison of delay and energy cost of IoT data transfers," *IEEE Access*, vol. 7, pp. 58654–58675, 2019.



IRIDA SHALLARI (Member, IEEE) received the joint M.Sc. degree in embedded systems from the University of Southampton, UK and NTNU, Norway, in 2016, and the Ph.D. degree in embedded vision systems from Mid Sweden University, Sweden. She is currently an Assistant Professor with Mid Sweden University. From 2023 to 2024, she was a Visiting Researcher with Ontario Tech University and the University of Salerno investigating on anomaly detection and AI-based measurement systems. She has co-authored around 30 publications, and she has garnered competitive research funding through both national and international grants. Her research focuses on the design and optimization of AI-based IoT nodes intended for intelligent monitoring, resource-constrained computations, and robust fault detection, often in challenging domains (e.g., railroad condition monitoring and snow depth measurement systems).



SILVIA KRUG received the B.Sc. degree in computer science from the University of Applied Sciences, Darmstadt, Germany, and the M.Sc. degree in computer engineering and the Dr.Ing. degree in communication networks from Technische Universität Ilmenau, Germany, in 2013 and 2017, respectively. She currently holds a postdoctoral position with the Embedded Systems Design Research Group, Mid Sweden University, Sweden. Her research interests include communication aspects for wireless sensor systems with a focus on energy efficiency and design considerations for the distributed IoT systems.



OSCAR ARTUR BERND BERG received the B.Sc. degree from the University of Wuppertal, in 2020, and the Master of Science degree in electro- and information-technology from the University of Stuttgart, Stuttgart, in 2023. He is currently pursuing the dual Ph.D. degree from TU Wien and Mid Sweden University. His research interests include optimization of computer vision models for efficient IoT deployment and embedded systems design.



EIRAJ SAQIB received the B.Sc. degree from University of Kashmir, Srinagar, in 2018, the Master of Science degree in electronic system design engineering, Universiti Sains Malaysia, Malaysia, 2021. He is currently pursuing the Ph.D. degree with Mid Sweden University, Sweden. His research interests include optimization of CNN model for the IoT implementation and embedded systems design.



AXEL JANTSCH (Fellow, IEEE) received the Dipl.Ing. and Ph.D. degrees in computer science from Vienna University of Technology (TU Wien), Vienna, Austria, in 1987 and 1992, respectively. From 1997 to 2014, he was an Associate Professor and since 2002, he has been a Full Professor of Electronic Systems Design with the KTH and the Royal Institute of Technology, Stockholm, Sweden. Since 2014, he has been a Professor of Systems on Chips with TU Wien.



ISAAC SÁNCHEZ LEAL received the B.Hons. degree in aeronautics and mechanical engineering from Glyndwr University, U.K., in 2015, the B.Hons. degree in electronics and industrial automation engineering from Universidad de Alcalá, Spain, in 2016, and the M.Sc. degree in electronics design from Mid Sweden University, Sweden, in 2018. He is currently pursuing the Ph.D. degree with the Research Group in Embedded IoT Systems Designs, Department of Computer and Electrical Engineering, Mid Sweden University. His research interests include deployment methods for computer vision systems based on neural networks



MATTIAS O’NILS (Member, IEEE) received the B.S. degree in electrical engineering from Mid Sweden University, Sundsvall, Sweden, in 1993, and the Licentiate and Ph.D. degrees in electronic systems design from the Royal Institute of Technology, Stockholm, Sweden, in 1996 and 1999, respectively. He is currently a Professor with the Department of Computer and Electrical Engineering and leads the AI based IoT and Measurement System Research Group, Mid Sweden University. His current research interests include implementation of AI based systems and design of AI based measurement systems.

...