# Energy-Aware Vision Model Partitioning for Edge AI

Katare, Dewant; Zhou, Mengying; Chen, Yang; Janssen, Marijn; Ding, Aaron Yi

**DOI**
10.1145/3672608.3707792

**Publication date**
2025

**Document Version**
Final published version

**Published in**
40th Annual ACM Symposium on Applied Computing, SAC 2025

# Energy-Aware Vision Model Partitioning for Edge AI

Dewant Katare [§], Mengying Zhou [§,†], Yang Chen [†], Marijn Janssen [§], Aaron Yi Ding [§]
[§]Delft University of Technology, [†]Fudan University

## ABSTRACT

Deploying scalable Vision Transformer applications on mobile and edge devices is constrained by limited memory and computational resources. Existing model development and deployment strategies include distributed computing and inference methods such as federated learning, split computing, collaborative inference and edge-cloud offloading mechanisms. While these strategies have deployment advantages, they fail to optimize memory usage and processing efficiency, resulting in increased energy consumption. This paper optimizes energy consumption by introducing adaptive model partitioning mechanisms and dynamic scaling methods for ViTs such as EfficientViT and TinyViT, adjusting model complexity based on the available computational resources and operating conditions. We implement energy-efficient strategies that minimize inter-layer communication for distributed machine learning across edge devices, thereby reducing energy consumption from data flow and computation. Our evaluations on a series of benchmark models show improvements, including up to a 32.6% reduction in latency and 16.6% energy savings, while maintaining mean average precision sacrifices within 2.5 to 4.5% of baseline models. These results show that our proposal is a practical approach for improving edge AI sustainability and efficiency.

## CCS CONCEPTS

• **Computing Methodologies** → **Vision Transformers**; Energy-aware Computing; • **Computer Systems** → *Edge AI*; Edge Computing.

## KEYWORDS

Energy-Aware Computing, Model Partition, Edge Computing

**ACM Reference Format:**
Dewant Katare, Mengying Zhou, Yang Chen, Marijn Janssen, Aaron Yi Ding. 2025. Energy-Aware Vision Model Partitioning for Edge AI. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3672608.3707792

## 1 INTRODUCTION

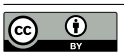
Advanced artificial intelligence (AI) models, particularly Vision Transformers (ViTs) [6], have shown improved performance for computer vision systems and applications [12, 19, 27]. However, deploying these models in real-world applications, especially in resource-constrained environments like mobile and edge devices, presents optimization challenges. This is critical for applications and use cases such as connected autonomous vehicles (CAVs), where balancing computational efficiency, energy consumption, and real-time processing is necessary. CAVs rely on sensing, computing, and communication within their ecosystems, necessitating low-latency processing [8] and the transmission of large volumes of data and model parameters [25] between the connected vehicles, heterogeneous edge devices, and the cloud. These requirements introduce memory bottleneck and computational challenges [9, 10, 20, 21, 24]. Model partitioning offers a promising solution for deploying efficient and scalable applications in distributed environments [7, 9, 24]. By decomposing a model into submodels, less complex submodels can be executed locally on edge devices, while high-computational-demand submodels are offloaded to powerful devices or cloud servers [9, 21]. Compared with the fully offloading strategies, model partitioning approach reduces data transmission by processing raw inputs locally [20] and decreases inference time by using more powerful servers than resource-constrained local devices [21]. These advantages make the execution of complex models more energy-efficient and faster, particularly for limited battery-powered devices.

Current research primarily focuses on reducing latency in offloading partitioned models and improving model accuracy [12, 15]. However, these studies often introduce a notable energy overhead, impacting the range of vehicles powered by limited-capacity batteries. Our motivation measurements reveal that three classic models can trade some precision for substantial energy savings, while maintaining acceptable accuracy for fault-tolerant applications. Inspired by this, we propose an energy-efficient model partitioning algorithm tailored to heterogeneous edge devices, which considers energy consumption, multi-metric evaluation, and adaptability to various vision models. This work advances the state-of-the-art by prioritizing energy efficiency in edge AI deployments within heterogeneous environments while maintaining accuracy and latency performance. The contributions of this paper are as follows:

**1) Energy-Efficient Model Partitioning:** Addressing the issue of energy efficiency in edge devices by developing algorithms that optimize model partitioning based on the computational capabilities of the heterogeneous devices, enhancing on-device processing and overall energy sustainability in edge deployments.

**2) Reduced Communication Overhead:** We minimize energy consumption from frequent device communications in distributed AI on edge networks to alleviate network congestion and improve system efficiency.

**3) Multi-Metric Evaluation:** We evaluate the performance of our partitioning method using energy consumption as the primary metric and secondarily considering accuracy and latency. This combined evaluation empirically assesses the proposed method in real-world edge scenarios. We believe these results can be used as a reference for making balanced trade-offs across multiple metrics.




*SAC '25, March 31-April 4, 2025, Catania, Italy*

ACM ISBN 979-8-4007-0629-5/25/03.
https://doi.org/10.1145/3672608.3707792

## 2 BACKGROUND: MODEL PARTITIONS

AI models can be partitioned by parallelizing matrix operations [18]. In Convolutional Neural Networks (CNNs), model partitioning can be implemented by independently computing tensors for a given input image while maintaining the model's input and output dimensions. The classic channel partitioning approach on CNNs for classification tasks [28] shows the principles of parallel execution of layers in a CNN model, e.g., differentiating sequential and parallel operations. In the convolutional layer, a feature map is generated from each filter, which can be separated with respective channels for partitioning. Also, output feature maps can be partitioned along the channel dimension so that each participating edge device computes a subset of the output feature maps [9]. Here, the corresponding set of filters is mapped to distributed devices, which at the same time also contain the dimensions of the model to prevent any adversarial effects by the model.

Similarly, fully connected layers can also be executed in parallel using spatial partitioning across height and width dimensions. In this case, output feature maps are considered. Similar to the convolutional layer, participating devices maintain dimensions (structure) and compute a subset of the output feature maps independently for the fully connected layer operations. In these operations, only a subset of input must be transferred, reducing overall communication costs. Based on the principles of channel partitioning, our proposed mechanism for CNN includes three dedicated approaches: Meta, Ensembled and Hybrid, respectively (also shown in Figure 1). These approaches are covered in detail in section 3.

### 2.1 ViTs for Mobile and Edge Devices

Recent studies have also explored various approaches to reduce the computational demands of the complex model like ViTs, without compromising their performance. Yu et al. [26] explored the "Width & Depth Pruning" method that addresses the width and depth dimensions of modules in ViTs. Its evaluation on ILSVRC-12 showed a decrease in Floating Point Of Operations (FLOPs) with maintaining baseline accuracy. Chen et al. [3] integrated the concept of sparsity in CNNs into ViTs to reduce training memory requirements and inference complexity. Their method dynamically extracted and trained sparse subnetworks throughout the training phase, while maintaining robust model performance. Zheng et al. [29] proposed SAViT, a structure-aware pruning method that leverages the interactions between different ViT components. This approach enhanced model efficiency and improved accuracy on benchmark datasets such as ImageNet and COCO by considering the structural interactions within the model. Liu et al. [13] utilised token pruning for dense prediction tasks such as object detection and instance segmentation. They proposed methods that preserve pruned tokens within feature maps for potential reactivation. This approach allowed for enhanced flexibility, reduced performance decline, and speed improvements during inference.

### 2.2 Collaborative Training and Inference

The deployment of ViTs in distributed heterogeneous edge environments requires collaborative training and inference strategies, which enable these models to benefit from the diverse computational resources available across different nodes. This approach involves partitioning the ViT model so that different blocks can be processed on various devices, optimizing latency and energy efficiency [15]. The inference speed can be improved by distributing or offloading the workload, and the training phase can be parallelized for computations and data handling [17]. This methodology ensures that edge devices with limited computational capabilities can still perform complex vision tasks by sharing the computational load across the network and extending the applicability of ViTs to resource-constrained environments in real-world applications.

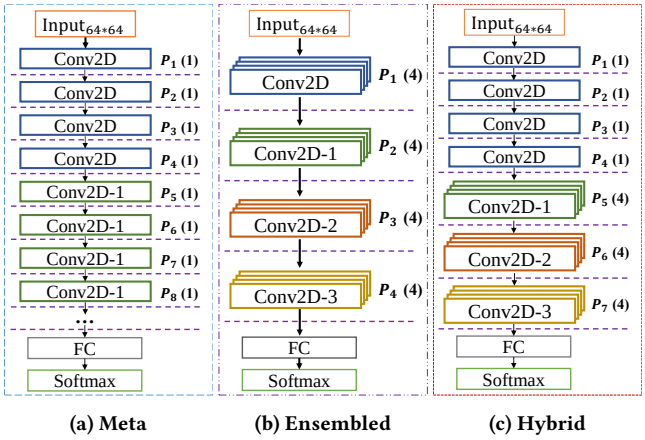



**Figure 1: Resource-based Model Partitioning Approaches**

### 2.3 Energy Efficient Mechanisms

Several approaches have been proposed to improve the energy efficiency of ViTs. Xu et al. [21] presented a decomposition-and-ensemble algorithm that breaks down large ViTs into smaller parts for collaborative inference, reducing latency and energy consumption with minimal performance loss. An approach to reduce the computational demand of ViTs is proposed by Nag et al. [14] by integrating ViTs with Weightless Neural Networks. They replaced computationally intensive layers with weightless alternatives to reduces hardware requirements and enhance energy efficiency and latency on edge devices. Li et al. [11] addressed inefficiencies in softmax computations within ViTs by approximating softmax functions with simpler or minimum computational elements, reducing resource consumption while maintaining high accuracy.

## 3 MOTIVATION AND SCOPE

The previous section provides background on the techniques for partitioning AI models, such as CNNs and ViTs, across different devices for distributed computing. This section will describe the motivation and fundamentals behind energy-saving partitioning strategies. Specifically, we explore how a balanced tradeoff can be established for energy and accuracy. These approaches are suitable for application optimization in error-tolerable scenarios and use cases. In this motivating experiment, we use classic CNN models, including ResNet, MobileNet, and SqueezeNet, to examine how partitioning impacts energy usage and model accuracy.

### 3.1 Model Partitioning for Classic Models

Based on channel partitioning principles, our partitioning mechanism includes three approaches: Meta, Ensembled, and Hybrid, as shown in Figure 1.

**Meta Partitioning:** In this approach, the channel partitioning technique is used to separate each layer independently, as shown in Figure 1(a). In this partitioning approach, a model can be partitioned by calculating the tensors of the current layer as a subgroup. Each output of the subgroup (independent layers) is combined to calculate the relative tensors before processing the next layer. As each layer is compiled independently, the output tensors from all layers should be merged to combine calculated weights and provide respective outputs. Within a dynamic wireless network, this approach results in excessive communication costs. However, in fully connected vehicular ecosystem scenario, the tradeoff in layer-wise parallel execution can be balanced with excessive communication costs from the individual vehicle and vision sensors generating and transferring large volumes of data.

**Ensembled Partitioning:** Meta approach requires frequent data transfer between layers, which increases communication costs. Therefore, to reduce the communication costs inherent in the Meta approach, the Ensembled approach combines layers with similar dimensions to counter this problem, which can further optimize end-to-end energy consumption. It has also been used as a hardware acceleration approach for machine learning applications to reduce memory footprints. We explore this approach by combining operations from multiple convolutional layers as a single block, as shown in Figure 1(b), instead of the single layers described in the Meta approach. Partitioning is implemented following the concept of the same dimension of input and output. The number of fused layers can vary depending on the availability and computation resources of participating devices. However, it is important to note increasing the number of blocks will further increase the communication cost. This approach is ideal for edge-cloud use cases and for AI models with repetitive layers, such as ResNet.

**Hybrid Partitioning:** Vehicle-edge ecosystem is formed of hybrid devices with different computing and processing abilities. The model partition approach should be dynamic and adjusted flexibly, as shown in Figure 1(c), according to each participating device's task requirements and current computational resource availability. In this approach, the coordinator device or host is aware of the computational resources currently available on the participating devices. The partitioning algorithm optimizes the partition depending on the vehicular task and resources required for layers. Computing resources (e.g., CPU, GPU), estimated memory, and latency can be three parameters we have considered in this approach as they can be directly associated with power consumption. It is important to note, an extra computing load is encountered on the coordinator device, which may increase power consumption on it. However, from the holistic perspective, we achieve a optimized end-to-end energy savings across all other devices.

### 3.2 Test and Analysis

We evaluate the above three partioning approaches on three classic models, namely MobileNet, SqueezeNet, and ResNet. We conduct experiments on an NVIDIA Jetson Nano and use the nuScenes [1] object classification dataset to assess model accuracy. To measure on-board power, we use Tegrastats and HV power monitor for energy measurements. The tested results are listed in Table 1.

**Table 1: Impact of Partitioning Approaches on Accuracy and Energy of Classic Models**

| Model | Approach | Accuracy (%) | Power (Wh) |
|---|---|---|---|
| MobileNet (3.7 MB) | Meta | 70.90 | 1.93 |
| | Ensembled | 67.40 | 1.69 |
| | Hybrid | 65.10 | 1.81 |
| SqueezeNet (11.0 MB) | Meta | 78.59 | 1.88 |
| | Ensembled | 79.38 | 2.58 |
| | Hybrid | 74.29 | 2.34 |
| ResNet (45.9 MB) | Meta | 88.43 | 5.71 |
| | Ensembled | 89.61 | 5.49 |
| | Hybrid | 84.31 | 5.22 |

We find that accuracy and energy consumption are influenced by the model type, partitioning method, and the corresponding transmitted partitions. Different partitioning methods result in varying energy consumptions, with reductions of up to 27%. However, this reduction in energy consumption also sacrifices a certain accuracy. For the SqueezeNet model, the meta-partitioning approach shows reduced energy consumption along with around 1% loss in accuracy, which is acceptable for some applications. We also observe that, in terms of accuracy, the accuracy of meta and ensembled methods is comparable, while the hybrid method achieves the lowest accuracy. This is because the hybrid approach creates large-sized ensembled partitions along with smaller meta partitions. In such cases, the computed results from meta partitions could be overshadowed by the results from ensembled partitions during merging, resulting in small numerical differences that could be concealed during backpropagation.

**Takeaway:** From the above preliminary measurements, it is empirical that energy consumption and accuracy are influenced by model architecture, partitioning method, and convergence mechanism. By adjusting these parameters within the system, we can meet user energy consumption requirements while maintaining an acceptable level of accuracy. This has inspired us to propose an energy-aware model partitioning system, with energy as the primary optimization criterion.

## 4 ENERGY-AWARE MODEL PARTITIONING

Inspired by the above insight, we propose the Layer-Adaptive Partitioning with Dynamic Task Redistribution (LAP-DTR) approach for partitioning large-scale ViTs, such as ViT [6], DeiT [16], and Swin [2], across edge devices. The approach considers energy consumption and resource allocation for used devices by ensuring efficient collaborative strategies without excessive communication or model performance degradation.

### 4.1 Modeling and Formulation

The LAP-DTR method partitions ViT tasks across heterogeneous edge devices (CPU, GPU), by dynamically adjusting allocation based on device memory, computational capabilities, and energy consumption (Algorithm 1). This algorithm describes the step-by-step process for the distribution mechanism and reallocation among

individual devices, ensuring that each device operates within its optimal parameters. Using Algorithm 1, the method enables real-time inference with minimal accuracy tradeoff, optimizing resource utilization and energy efficiency across the entire system. The following subsections cover detailed problem descriptions and definitions of the relevant formulas.

**Energy and Resource Allocation-Aware Partitioning:** To ensure efficient deployment, the LAP-DTR framework includes an energy and resource allocation model that guides how the large ViT is partitioned. For each edge device, the available computational power, memory, and current energy level are factored into the partitioning strategy.

Let $E_i$ be the remaining energy on device $i$, $C_i$ the computational capacity (in terms of GFLOPs), and $M_i$ is the available memory. The layer partitioning is solved as an optimization problem to minimize overall energy consumption and computational load while meeting latency constraints. The objective is to allocate layers such that:

$$\min \sum_{i=1}^{N} \frac{E_i}{C_i} \cdot L_i \quad \text{such that:} \quad \sum_{i=1}^{N} L_i = L_{total},$$

where $L_i$ is the computational load (number of layers) assigned to device $i$, and $L_{total}$ is the total number of layers in the ViT. Each layer $l$ of the transformer has a computational complexity $C_l$ and a memory footprint $M_l$. The partitioning seeks to ensure that for any device $i$:

$$C_l \leq C_i \quad \text{and} \quad M_l \leq M_i,$$

ensuring that the assigned layer fits within the device's computational and memory constraints. The framework dynamically updates the partitioning based on the current energy state of each device.

**Dynamic Task Redistribution:** In addition to energy-aware partitioning, the LAP-DTR framework continuously monitors the workload and battery levels of each edge device during runtime. If a device's energy $E_i$ falls below a threshold $E_{min}$ or if its current workload exceeds its computational limits, the task is dynamically redistributed to other devices.

This redistribution is achieved by solving the following minimization problem, which seeks to balance the computational load across devices while considering their energy constraints:

$$\min \sum_{i=1}^{N} \left( \frac{C_i}{L_i} + \lambda \cdot \frac{1}{E_i} \right),$$

where $\lambda$ is a weighting factor that adjusts the importance of energy savings relative to computational load balancing. The objective is to offload tasks to devices that have both available energy and computational capacity, avoiding bottlenecks.

**Attention-based Feature Summarization:** To further reduce communication overhead, an attention-based feature summarization mechanism is used. Instead of transmitting full feature maps, only the most important attention weights and features are shared between devices. Given an attention matrix $A$ generated by a self-attention layer, the feature summarization is performed by applying a compression function $f(\cdot)$ to extract key elements:

$$X_{\text{summarized}} = f(A) \cdot X,$$

**Algorithm 1** Selecting a Model Partition Strategy

**Require:** Device capability set $C$, required accuracy $Acc$, required energy $E$, required latency $L$, weights for accuracy $w_{Acc}$, energy $w_E$, and latency $w_L$

Initialize the list of candidate profiles $Candidates = \emptyset$
**for** each model partition profile **do**
  Get the accuracy $Acc_{\text{profile}}$ from the profile
  Get the energy consumption $E_{\text{profile}}$ from the profile
  Get the latency $L_{\text{profile}}$ from the profile
  **if** $Acc_{\text{profile}} \geq Acc$ **and** $E_{\text{profile}} \leq E$ **and** $L_{\text{profile}} \leq L$ **then**
    Add the profile to $Candidates$
  **end if**
**end for**
**if** $Candidates$ is empty **then**
  **return** $profile where Acc_{\text{profile}} is max$
**end if**
Initialize the best model partition $BestProfile$ = None
Initialize the best score $BestScore = -\infty$
**for** each candidate profile **do**
  Get the accuracy $Acc_{\text{candidate}}$ from the candidate profile
  Get the energy consumption $E_{\text{candidate}}$ from the candidate profile
  Get the latency $L_{\text{candidate}}$ from the candidate profile
  Calculate the accuracy score $Score_{\text{acc}} = w_{Acc} \cdot Acc_{\text{candidate}}$
  Calculate the energy score $Score_{\text{energy}} = w_E \cdot \frac{E_{\text{candidate}}}{E_{max}}$
  Calculate the latency score $Score_{\text{latency}} = w_L \cdot \frac{L_{\text{candidate}}}{L_{max}}$
  Calculate the overall score $Score = Score_{\text{acc}} + Score_{\text{energy}} + Score_{\text{latency}}$
  **if** $Score > BestScore$ **then**
    Update the best model partition $BestProfile$ = candidate
    Update the best score $BestScore = Score$
  **end if**
**end for**
**return** The best model partition $BestProfile$

where $X$ is the original feature map, and $X_{\text{summarized}}$ is the compressed version that is transmitted between devices. This helps reduce the communication load while preserving the essential information required for accurate inference.

**Knowledge Transfer Between Sub-networks:** Knowledge transfer across sub-networks is facilitated by passing intermediate representations from one device to another. Each sub-network outputs a feature map $X_i$ on device $i$, which is used as input to the next sub-network on device $j$. The transfer function $T(\cdot)$ ensures that the knowledge from $X_i$ is optimally adapted to the sub-network on device $j$:

$$X_j = T(X_i) \quad \text{where} \quad T(X_i) = W \cdot X_i,$$

where $W$ is a learned transformation matrix that adjusts the intermediate feature map to match the dimensions and structure expected by the sub-network on device $j$.

**Layer Elasticity:** To enhance resource efficiency, non-contributing layers of the transformer are rendered elastic. This enables them to dynamically adjust their complexity by reducing the number of attention heads $h$ or the hidden dimension size $d$ in response to the available device resources. The elastic adjustment is given as:

$$h = h_{max} \cdot \left(1 - \frac{E_i}{E_{max}}\right) \quad \text{and} \quad d = d_{max} \cdot \left(1 - \frac{E_i}{E_{max}}\right),$$

where $h_{max}$ and $d_{max}$ are the maximum values for the number of heads and hidden dimension, and $E_i$ is the current energy of device $i$. As the energy level decreases, the number of heads and dimension size are scaled down, reducing the computational load on the device.

## 4.2 Model Partitioning Profile: ViTs

ViTs are structured differently from traditional CNNs. They consist of layers of multi-head self-attention mechanisms and feed-forward networks, which are known to be computationally demanding. In particular, the attention mechanism requires each input token to interact with every other token, making the computation complexity grow quadratically with input size. For partitioning ViTs, the main objective is to distribute the attention heads and feed-forward layers across the available devices, balancing the workload and ensuring that the computational or memory demands are within device range. Each layer in a ViT can be characterized by the following factors:

- Attention Heads: The number of attention heads in a self-attention layer directly influences the computation cost. Devices with higher computational capacity are assigned layers with more heads.
- Embedding Dimension: The size of the embedding dimension affects the memory and processing power required. Larger embedding dimensions increase the complexity of matrix operations, so they are allocated to devices with sufficient resources.
- Feed-Forward Network Size: Each transformer layer includes a feed-forward network that processes the output of the attention heads. The size of this network is a key factor in determining the memory and computation needs.
- Token Count: The number of tokens passed through the layers influences the overall complexity, as the attention mechanism scales quadratically with the number of tokens.

The partitioning profile for ViTs is optimized to minimize energy consumption and maximize efficiency. Layers with more attention heads and larger embedding dimensions are assigned to devices with higher capabilities, while the layers with reduced complexity are allocated to less powerful devices. During runtime, the system dynamically adjusts the partitioning based on the energy availability of each device. If a device becomes low on energy, the framework can reduce the number of attention heads or compress the feed-forward network to allow the model to continue processing without interruption. The partitioning profile for ViTs considers both the computational and memory requirements, ensuring that the workload is appropriately distributed across the edge-device network.

# 5 ENERGY-AWARE EDGE FRAMEWORK

As the framework operates in a distributed manner, where edge devices collaboratively perform inference tasks. Intermediate results are cached locally to ensure robustness, and the system adapts to network conditions by redistributing tasks when necessary. Communication is minimized through the attention-based summarization technique, while fault tolerance is ensured by local result caching.

**Algorithm 2** Distributed Training for ViTs

**Require:** ViT model $M$, dataset $D$, edge devices $E = \{E_1, E_2, \ldots, E_N\}$ with properties energy $E_i$, computation capacity $C_i$, memory $M_i$, and communication bandwidth $B_{ij}$ between devices $E_i$ and $E_j$.
**Ensure:** Optimized and partitioned model $M$ distributed across $E$.
**Initialization:** Profile energy, compute resources, and memory for each device in $E$.
**for all** $l \in \text{layers}(M)$ **do**
  Compute $C_L$ and $M_L$ for layer $L$
  Assign layer $l$ to device $E_k$ where $k = \arg\min_i \left( \frac{E_i}{C_i} \cdot C_l, M_l \leq M_i \right)$
**end for**
**for all** $E_I \in E$ **do**
  Train sub-model $M_i$ using data $D_i$
  Transfer intermediate feature maps $X_i$ to $E_{i+1}$, where $X_{\text{sum}} = \text{Attention}(X_i)$
  **if** $E_I$ runs out of energy **then**
    Reassign layers of $M_i$ to minimize energy consumption:
    $L_j = \arg\min_j \left( \sum_{j=1}^{N} \left( \frac{C_j}{L_j} + \lambda \cdot \frac{1}{E_j} \right) \right)$
  **end if**
**end for**

## 5.1 Orchestration of Training and Inference

The LAP-DTR framework orchestrates both training and inference phases across multiple devices by dynamically partitioning the model and adapting to device constraints. The following steps show how the orchestration is managed using multiple mixed metrics and the specific steps involved in managing the orchestration.

**Evaluation Metrics:** The LAP-DTR framework is evaluated using the following metrics for training and inference:

- **Latency**: The total time taken for inference across all edge devices.
- **Energy Consumption**: The energy used by each device during inference, including both computation and communication.
- **Accuracy**: The model's performance in terms of classification accuracy.
- **Resource Utilization**: The proportion of available computational and memory resources used by each device.
- **Communication Overhead**: The data transmitted between edge devices during inference.

**Training Procedure:** The training process is managed as described in the Algorithm 2, which details the steps for distributing and managing model layers across multiple edge devices. This algorithm ensures that each device is assigned layers based on its available energy, computational capacity, and memory. By following Algorithm 2, the system achieves a balanced load distribution, optimizing energy consumption while maintaining efficient training performance across the entire network.

- The system first profiles each device's available energy, compute capacity, and memory.
- Layers are assigned based on the energy and computational capacities of devices, ensuring balanced load distribution.
- During training, intermediate feature maps are transferred between devices using attention-based summarization, minimizing the communication overhead.

**Algorithm 3** Distributed Inference for ViTs

**Require:** Trained models $\{M_1, M_2, \ldots, M_N\}$ assigned across devices $E$.
**Ensure:** Aggregated inference results $Y$.

```
for all inference tasks do
    for all E_i ∈ E do
        Monitor current energy E_i and workload
        Adjust layer settings for current energy state:
        h_i = h_max (1 − E_i/E_max),   d_i = d_max (1 − E_i/E_max)
        Execute inference on M_i
        Transfer summarized features X_sum to E_{i+1}
        if E_i fails or disconnects then
            Resume inference using cached results at E_{i+1}
        end if
    end for
    Aggregate outputs from all models: Y = G(M_1(X), M_2(X), ..., M_N(X))
end for
```

- If a device runs out of energy, its layers are redistributed to other devices in the network based on available resources.

**Inference Steps:** The inference steps are managed by Algorithm 3, which outlines the procedures for monitoring device states and adjusting layer complexities in real-time. This algorithm enables the dynamic redistribution of tasks and ensures that each device operates within its current energy constraints. By implementing Algorithm 3, the framework maintains robust inference performance and minimizes communication overhead, even in environments with heterogeneous and fluctuating device capabilities.

- For each inference task, the system monitors the workload using Flops, latency and energy levels of each device.
- As an optimized strategy during partition, the algorithm adjusts assigned layers' complexity based on its current energy level (e.g., reducing the number of attention heads or hidden dimensions) for devices.
- Intermediate feature maps are transferred between devices using summarized attention scores to minimize inter-layer data transmission.
- If a device fails or disconnects, the cached intermediate results are used to resume inference from the last state.
- Once all devices complete their assigned tasks, the final result is aggregated using a simple aggregation function $G$.

The orchestration handles device heterogeneity, ensuring transitions between energy states and task redistribution. The system is highly adaptable and fault-tolerant, providing robust performance even under varying network and hardware conditions.

## 5.2 Complexity Analysis

The computational complexity of this algorithm consists of : the partitioning and the distributed training/inference phases.

**Partitioning Phase:** For the partitioning of $L$ layers among $N$ edge devices, each layer's computational complexity $C_l$ and memory $M_l$ must be calculated. This involves $O(L)$ complexity for profiling the model. Assigning layers to the edge devices involves solving a resource allocation problem that depends on the device's energy $E_i$, computational capacity $C_i$, and memory $M_i$. This allocation is done in $O(L \cdot N)$ time, as each layer is assigned based on minimizing the energy-to-computation ratio.

**Training and Inference Phase:** During training, each device trains its assigned sub-model. The overall time complexity is measured by communication overhead and the computational complexity of each sub-model. The communication complexity between devices is reduced by summarizing attention features, resulting in $O(N)$ communication rounds, where $N$ is the number of edge devices. Since the layers are partitioned adaptively, the computational load on each device is balanced, ensuring no single device becomes a bottleneck. The overall training time can be approximated as:

$$T_{\text{train}} = \sum_{i=1}^{N} T(M_i) + O(N),$$

where $T(M_i)$ is the training time of sub-model $M_i$ on device $E_i$.

Inference follows a similar complexity profile, where each device processes its assigned layers, adjusting complexity based on energy availability. The dynamic adjustment of layer complexity results in a time complexity for inference as:

$$T_{\text{infer}} = O\left(\sum_{i=1}^{N} C(M_i) \cdot \frac{E_i}{E_{\max}}\right) + O(N),$$

where $C(M_i)$ is the computation cost of sub-model $M_i$ on device $E_i$ and $O(N)$ accounts for communication overhead.

# 6 MULTI-METRIC EVALUATION

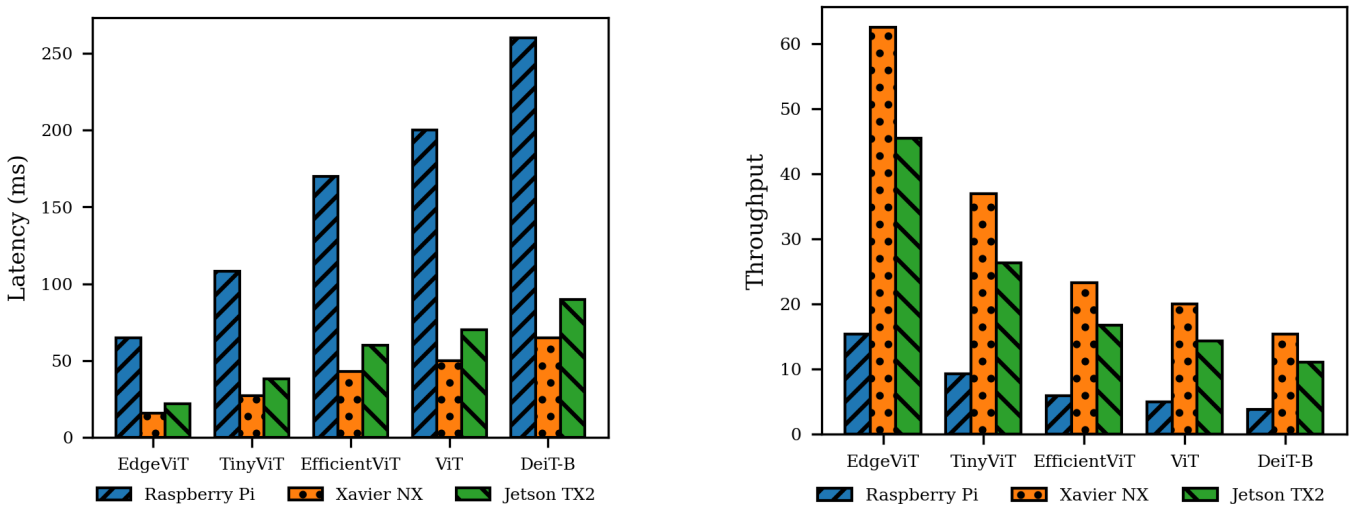
To validate our energy-aware partitioning approach, we conduct tests using following hardware setup and ViT models.

**Hardware Setup:** We utilize a heterogeneous set of edge devices, including Nvidia Xavier NX, Jetson TX2, and Raspberry Pi 4, featuring both CPU and GPU resources. These devices provide varying levels of computational power and memory resources, enabling us to test the adaptability of our partitioning strategies in real-world edge AI scenarios. Each device's energy consumption and latency are recorded during training and inference.

**ViT Models:** We evaluate the following ViT models:

- ViT [6]: The original ViT model that applies transformer architectures to image recognition tasks, leveraging self-attention mechanisms for high performance in various computer vision applications.
- EdgeVit [4]: A ViT variant for edge devices, enhancing computational efficiency and reducing memory usage to enable real-time processing in resource-constrained environments.
- TinyViT [19]: A compact transformer designed for resource-constrained environments, offering reduced size and computational requirements without significant loss in accuracy.
- DeiT-B [16]: A data-efficient image transformer that enhances training efficiency and performance by utilizing knowledge distillation techniques, making it suitable for scenarios with limited labeled data.
- EfficientViT [12]: Lightweight version of ViT optimized for inference on edge devices, balancing performance and energy consumption for deployment in low-resource settings.

**Tested Datasets:** These above five ViT models are tested on the OPV2V dataset [24], which is the first large-scale open dataset focused on vehicle-to-vehicle (V2V) communication for perception




(a) Latency (b) Throughput

**Figure 2: Latency and Throughput Comparison for Distributed Models**

**Table 2: Accuracy and Energy Comparison Between Centralized and LAP-DTR on GPU**

| Models | Methods | mAP (%) | Latency (ms) | Energy (mJ) |
|---|---|---|---|---|
| EdgeViT | Centralized | 56.7 | 4.18 | 362.5 |
| | LAP-DTR | 54.2 | 3.95 | 301.1 |
| TinyViT | Centralized | 55.3 | 3.82 | 418.3 |
| | LAP-DTR | 52.7 | 3.15 | 376.5 |
| EfficientViT | Centralized | 54.8 | 3.75 | 450.7 |
| | LAP-DTR | 53.1 | 3.22 | 392.4 |
| ViT | Centralized | 48.3 | 4.60 | 481.2 |
| | LAP-DTR | 43.8 | 3.11 | 413.4 |
| DeiT-B | Centralized | 45.9 | 4.93 | 421.5 |
| | LAP-DTR | 41.4 | 3.64 | 351.4 |

tasks [23, 24]. This dataset, collected using the OpenCDA framework and CARLA simulator [5, 22], contains: 73 diverse driving scenes across 9 cities and 6 road types. 12K frames of LiDAR point clouds and RGB camera images. 230K annotated 3D bounding boxes. Benchmarks with 4 LiDAR detectors and 4 different fusion strategies (16 models in total). This dataset provides a comprehensive evaluation platform for multi-vehicle sensing and cooperation, which is particularly relevant for autonomous driving scenarios.

**Metrics:** We measure three key metrics: Energy Consumption, the total energy usage by each device during inference, measured using external power monitoring tools; Accuracy, the model's performance on 3D object detection tasks using the annotated bounding boxes; latency, the time taken for model inference across the distributed edge devices.

**Trade-off Between Accuracy and Energy:** Table 2 compares the performance of the centralized method and the LAP-DTR method across five ViT models. Overall, the LAP-DTR method reduces energy consumption and latency. For example, EdgeViT's energy consumption decreased by approximately 17%, and latency decreased by about 5%. Similarly, TinyViT and EfficientViT saw energy reductions of around 10% and 13%, respectively, along with significant decreases in latency. Although the accuracy of ViT and DeiT-B decreased from 48.3% to 43.8% and from 45.9% to 41.4%, respectively, the reductions in energy consumption and latency remain within acceptable ranges. This indicates that the LAP-DTR method can achieve energy savings while maintaining model performance, making it suitable for applications that require high energy efficiency and can tolerate minor losses in accuracy.

**Latency and Throughput:** Figures 2 compare the latency and throughput of models across three devices: Raspberry Pi, Xavier NX, and Jetson TX2. Latency is measured in milliseconds and shows the time each device takes to process an input for each model. Throughput is measured as images processed per second. Raspberry Pi shows higher latency across all models, for example around 68 ms for EdgeViT and 259 for DeiT-B due to limited processing power, while Xavier NX shows the highest throughput because of processing power, showing its efficiency in handling intensive computations.

**Energy Consumption:** We monitor energy consumption using the NVIDIA Management Library (NVML) and Tegrastats for NVIDIA devices. The library allows tracking and optimization of power consumption during computational tasks. Figure 3 shows energy usage across memory, CPU, and GPU on a device. Because of the large output feature map and complex computation in handling token representations the GPUs consume most energy, while memory and CPUs use less energy which is associated with input feature map, optimizers and activations. Additionally, the energy consumption of memory and CPUs varies depending on the ViT model architecture. For all ViT models except DeiT-B, CPU energy consumption exceeds that of memory. Also as shown, DeiT-B model consumes more energy in memory than on CPUs, reflecting their optimizations for memory efficiency.

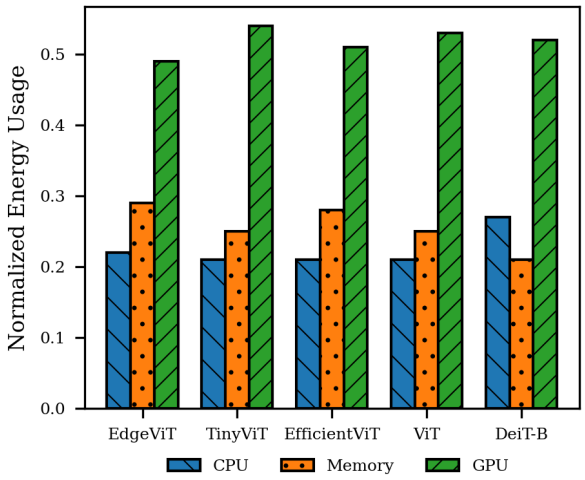



**Figure 3: Energy Comparison for Models on Xavier NX**

## 7 CONCLUSION AND FUTURE WORK

This paper introduces the LAP-DTR framework which dynamically partitions Vision Transformer (ViT) models across heterogeneous edge devices to enhance energy efficiency and optimize resource use. The focus is to adapt model complexity during training and inference based on the device's resources, memory and processing capacity. The strategy also considers device heterogeneity to ensure robust performance under varying operational conditions, making it a practical solution for error-tolerable distributed ViTs on edge networks. Our test on various ViT models shows that LAP-DTR can reduce energy consumption by up to 17% and decrease latency with a balanced tradeoff in accuracy. The analysis shows the practicality of LAP-DTR in real-world edge AI scenarios, especially for applications where energy resources are limited and minor accuracy trade-offs are acceptable. While LAP-DTR shows potential results, few areas require further exploration: **1). Extending to Other Architectures.** Using LAP-DTR to multi-modality and transformer models, such as vision language models, to further test potentials. **2). Model Approximation Strategies.** Combine model partitioning with bit-wise approximation techniques to achieve device-specific energy savings with balanced trade-offs for applications. **3). User-Centric Optimization.** Incorporate user preferences and quality-of-service requirements to balance energy consumption, latency, and accuracy. Addressing these areas will improve the LAP-DTR framework, making it more adaptable and effective for data and compute-intensive applications.

## ACKNOWLEDGMENTS

This work was supported by the funding from European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska Curie grant agreement No. 956090 (APROPOS: Approximate Computing for Power and Energy Optimisation).

## REFERENCES

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuScenes: A multimodal dataset for autonomous driving. In *Proc. of CVPR*.

[2] Lu Chen, Yong Bai, Qiang Cheng, and Mei Wu. 2022. Swin Transformer with Local Aggregation. In *Proc. of ISPDS*.

[3] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. 2021. Chasing Sparsity in Vision Transformers: An End-to-End Exploration. In *Proc. of NeurIPS*.

[4] Zekai Chen, Fangtian Zhong, Qi Luo, Xiao Zhang, and Yanwei Zheng. 2022. EdgeViT: Efficient Visual Modeling for Edge Computing. In *Proc. of WASA*.

[5] Jean-Emmanuel Deschaud. 2021. KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator. *arXiv preprint arXiv:2109.00892* (2021).

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of ICLR*.

[7] Joren Dumoulin, Pouya Houshmand, Vikram Jain, and Marian Verhelst. 2024. Enabling Efficient Hardware Acceleration of Hybrid Vision Transformer (ViT) Networks at the Edge. In *Proc. of ISCAS*. 1–5.

[8] Chongtao Guo, Xijun Wang, Le Liang, and Geoffrey Ye Li. 2022. Age of Information, Latency, and Reliability in Intelligent Vehicular Networks. *IEEE Network* 37, 6 (2022), 109–116.

[9] Yu Huang, Heli Zhang, Xun Shao, Xi Li, and Hong Ji. 2023. RoofSplit: An edge computing framework with heterogeneous nodes collaboration considering optimal CNN model splitting. *Future Generation Computer Systems* 140 (2023), 79–90.

[10] Dewant Katare and Aaron Yi Ding. 2023. Energy-efficient Edge Approximation for Connected Vehicular Services. In *Proc. of CISS*.

[11] Suhang Li, Bo Yin, and Hao Zhang. 2023. An Energy-Efficient Architecture of Approximate Softmax Functions for Transformer in Edge Computing. In *Proc. of EEET*.

[12] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. 2023. EfficientViT: Memory Efficient Vision Transformer With Cascaded Group Attention. In *Proc. of CVPR*.

[13] Yifei Liu, Mathias Gehrig, Nico Messikommer, Marco Cannici, and Davide Scaramuzza. 2024. Revisiting Token Pruning for Object Detection and Instance Segmentation. In *Proc. of WACV*.

[14] Shashank Nag, Logan Liberty, Aishwarya Sivakumar, Neeraja J Yadwadkar, and Lizy Kurian John. 2024. Lightweight Vision Transformers for Low Energy Edge Inference. In *Proc. of ISCA workshop*.

[15] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, et al. 2023. Efficiently Scaling Transformer Inference. In *Proc. of MLSys*.

[16] Hugo Touvron, Matthieu Cord, and Hervé Jégou. 2022. DeiT III: Revenge of the ViT. In *Proc. of ECCV*.

[17] Haonan Wang, Connor Imes, Souvik Kundu, Peter A Beerel, Stephen P Crago, and John Paul Walters. 2023. QuantPipe: Applying Adaptive Post-Training Quantization for Distributed Transformer Pipelines in Dynamic Edge Environments. In *Proc. of ICASSP*.

[18] Huaming Wu, William J Knottenbelt, and Katinka Wolter. 2019. An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems* 30, 7 (2019), 1464–1480.

[19] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. 2022. TinyViT: Fast Pretraining Distillation for Small Vision Transformers. In *Proc. of EECV*.

[20] Mengyao Wu, F Richard Yu, and Peter Xiaoping Liu. 2022. Intelligence Networking for Autonomous Driving in Beyond 5G Networks With Multi-Access Edge Computing. *IEEE Transactions on Vehicular Technology* 71, 6 (2022), 5853–5866.

[21] Guanyu Xu, Zhiwei Hao, Yong Luo, Han Hu, Jianping An, and Shiwen Mao. 2024. DeViT: Decomposing Vision Transformers for Collaborative Inference in Edge Devices. *IEEE Transactions on Mobile Computing* 23, 5 (2024), 5917–5932.

[22] Runsheng Xu, Yi Guo, Xu Han, Xin Xia, Hao Xiang, and Jiaqi Ma. 2021. OpenCDA: An Open Cooperative Driving Automation Framework Integrated with Co-Simulation. In *Proc. of ITSC*.

[23] Runsheng Xu, Zhengzhong Tu, Hao Xiang, Wei Shao, Bolei Zhou, and Jiaqi Ma. 2023. CoBEVT: Cooperative Bird's Eye View Semantic Segmentation with Sparse Transformers. In *Proc. of CoRL*.

[24] Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. 2022. OPV2V: An Open Benchmark Dataset and Fusion Pipeline for Perception with Vehicle-to-Vehicle Communication. In *Proc. of ICRA*.

[25] Xiuwen Yin, Jianqi Liu, Xiaochun Cheng, and Xiaoming Xiong. 2021. Large-Size Data Distribution in IoV Based on 5G/6G Compatible Heterogeneous Network. *IEEE Transactions on Intelligent Transportation Systems* 23, 7 (2021), 9840–9852.

[26] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. 2022. Width & Depth Pruning for Vision Transformers. In *Proc. of AAAI*.

[27] Li Yuan, Yunpeng Chen, Tao Wang, et al. 2021. Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In *Proc. of ICCV*.

[28] Wanpeng Zhang, Nuo Wang, Liying Li, and Tongquan Wei. 2022. Joint compressing and partitioning of CNNs for fast edge-cloud collaborative intelligence for IoT. *Journal of Systems Architecture* 125 (2022), 102461.

[29] Chuanyang Zheng, Kai Zhang, Zhi Yang, et al. 2022. SAViT: Structure-Aware Vision Transformer Pruning via Collaborative Optimization. In *Proc. of NeurIPS*.