Mercury: Towards Optimal Accuracy-Latency Trade-off for Collaborative Transformer Inference

Yumeng Liang[†], Jianhui Chang[†], Sijia Li[‡†], Mingyuan Zang[†], Jie Wu[†*]
[†]China Telecom Cloud Computing Research Institute, China
[‡]University of Science and Technology Beijing, China
[*]Temple University, USA
{liangym9, changjh1}@chinatelecom.cn, lisijia@xs.ustb.edu.cn, zangmy1@chinatelecom.cn, jiewu@temple.edu.

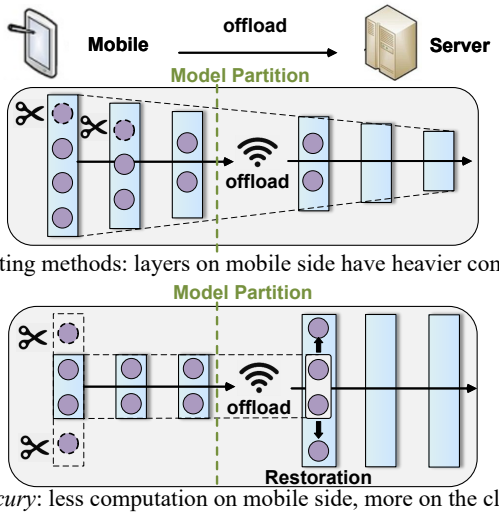
***Abstract*—Vision Transformers (ViT) achieve remarkable performance across a wide range of visual tasks but incur high inference latency on resource-constrained mobile devices. Mobile-cloud collaborative inference, *i.e.*, partially offloading the workload to the cloud, is a promising solution for acceleration by leveraging both mobile and cloud resources. Existing approaches reduce token numbers in a multi-round manner to shrink the data to be offloaded to the cloud, so as to reduce the communication latency. However, they cause imbalanced workloads, leading to heavy latency on the resource-constrained mobile devices. This paper presents Mercury, a collaborative ViT inference framework that accounts for mobile-cloud resource disparity. Unlike multi-round strategies, it performs single-round token pruning at the initial layer on mobile devices to aggressively reduce computation and communication overhead, and then restores tokens in the cloud to maintain accuracy. A similarity-based regionally decentralized token pruning method is designed to preserve semantic features, while an efficient meta-parameter-based token reconstruction method is used to improve accuracy with minimal overhead. Experiments on real-world testbeds under 4G/5G traces show that Mercury achieves up to a 2.27× speedup and halves the mobile-side energy consumption, while preserving comparable accuracy to state-of-the-art methods.**

***Index Terms*—mobile-cloud collaboration, collaborative transformer inference, vision transformer, IoT, mobile computing**

I. Introduction

The widespread deployment of cameras has enabled lots of emerging applications on mobile devices, such as object/person re-identification [1], [2] and anomaly detection [3], [4]. These tasks heavily rely on deep learning methods for real-time processing. Recently, Vision Transformer (ViT) has gained increasing attention due to its superior accuracy compared to convolutional neural networks (CNNs) across various vision tasks [5]. However, despite the performance benefits, ViT comes with substantial computational costs, making it unsuitable for deployment on mobile devices where resources are limited. To address this, various model compression methods have been proposed to reduce memory and computation by modifying model structures [6]–[8]. While effective in reducing overhead, these approaches often degrade accuracy, failing to achieve an optimal accuracy-latency trade-off.

An alternative solution is *mobile-cloud collaborative inference*, *i.e.*, partially offloading the workload to the cloud by transmitting intermediate features over networks to leverages both mobile and cloud capabilities for acceleration. Existing collaborative approaches [10]–[13] mainly focus on CNNs, whose intermediate features shrink progressively during workloads, enabling reduced communication overhead. In contrast, ViT maintains a fixed token size and large computational footprint throughout the inference, making traditional collaborative strategies less effective in reducing the overall latency.



(a) Existing methods: layers on mobile side have heavier computation

(b) *Mercury*: less computation on mobile side, more on the cloud side

Fig. 1: Comparison of collaborative frameworks. Unlike the existing method [9], Mercury fully considers the mobile-cloud differences in computational resources: aggressively reducing tokens at the initial layer and restoring them on the cloud.

Several recent works [9], [14] aim to reduce ViT's collaborative inference latency by *gradually reducing token counts layer-by-layer*, thereby shrinking the offloading data by splitting at later layers. However, this multi-round reduction overlooks the mobile-cloud resource disparity and *leads to imbalanced workloads*: the computation-intensive early layers have to run on resource-constrained mobile devices, resulting in significant computational latency. In addition, it ignores the possibility of regenerating the pruned tokens through restoration, which can be done in the cloud to boost accuracy.

In this paper, we propose **Mercury**[1], a resource-aware

This work was supported by Young Elite Scientists Sponsorship Program of the Beijing High Innovation Plan. Corresponding author: Jie Wu.

[1]Mercury, the Roman god with winged sandals, symbolizes swift message delivery between cloud and mobile devices.

collaborative inference framework for ViT. As shown in Fig. 1, Mercury differs from existing multi-round token reduction methods by adopting *single-round pruning at the initial layer* to minimize mobile-side computation and communication latency, and then *restores tokens in the cloud* to preserve accuracy. Note that the early token pruning significantly reduces both mobile computation and intermediate features to be offloaded, enabling flexible layer partitioning unconstrained by communication volume. Meanwhile, token reconstruction incurs negligible overhead due to abundant cloud resources.

To deliver efficient mobile-cloud collaboration, *i.e.*, significantly reducing the overall latency while preserving high inference accuracy, Mercury addresses two main challenges:

1) How to significantly reduce the token numbers at the initial layer while preserving semantic information? Our investigation experiments reveal that, although reducing tokens at the initial transformer layer delivers substantial inference acceleration, it drastically lowers accuracy compared to pruning in deeper layers. This is mainly because early-stage features are more sparse and less aggregated, which makes large-scale pruning in these layers prone to losing important features. Considering this, Mercury adopts *a similarity-based regionally decentralized token pruning algorithm*, which preserves tokens with higher semantic similarity within each subregion, ensuring semantic integrity under high pruning rates.

2) How to efficiently reconstruct tokens to enhance inference accuracy? A straightforward approach would introduce a dedicated neural network to regenerate the pruned tokens, but this adds heavy task-specific computational and storage overhead, which is impractical for supporting diverse tasks in the cloud. Instead, Mercury leverages the inherent attention mechanism of transformers: the restored tokens are treated as *meta-parameters*, which are trained offline and stored in the cloud. They are concatenated with the retained tokens as input to the cloud-side model, thereby restoring the essential semantic features for enhanced inference accuracy. This method avoids incurring substantial additional computational cost and adds minimum parameters, only 0.13% of the original model size, achieving efficient accuracy enhancement.

We evaluate Mercury using a real-world setup: a Raspberry Pi 4B as the mobile device, a cloud server, and real 4G/5G bandwidth traces to simulate bandwidth fluctuations during transmission. Compared with two state-of-the-art methods [9], [14], Mercury achieves up to **2.27× speedup** under the same accuracy threshold and reduces the energy consumption of mobile devices to **50%** of the competing method. Moreover, *each of our two core designs independently contributes to performance improvements*: under the same cloud-side token reconstruction setting, our token pruning strategy alone achieves a 1.73× speedup, while our token reconstruction alone can further boost accuracy by up to 15.62%.

Our contributions can be summarized as follows:

- We propose a novel collaborative Transformer inference paradigm that performs early token reduction on mobile devices and token reconstruction in the cloud to achieve both speed acceleration and accuracy preservation.
- We introduce Mercury, an efficient collaborative framework with a similarity-based regionally decentralized pruning strategy and meta-parameter-driven token reconstruction for optimal accuracy-latency trade-offs.
- Extensive experiments under realistic settings demonstrate that Mercury consistently outperforms existing methods in terms of the accuracy-latency trade-offs, whether using single-round reduction alone, token reconstruction alone, or both combined.

II. Background and motivation

A. Vision Transformer and Token Reduction

ViTs [15] are a series of groundbreaking deep learning models that adapt the Transformer architecture—originally designed for natural language processing—to the domain of computer vision. They begin by dividing an input image into a series of non-overlapping patches. For instance, an image with dimensions of H (height), W (width) can be divided into $N = HW/P^2$ patches. Each patch has the shape $P \times P$, and is linearly embedded into a vector of fixed dimension d_f, referred to as *tokens*, which are the input to the Transformer. Taking the tokenized features X as input, the computation process of the Transformer with L layers is formulated as:

$$f(X) = \Gamma \circ \mathcal{F}^L \circ \mathcal{F}^{L-1} \circ \cdots \circ \mathcal{F}^1 \circ X, \tag{1}$$

where $\mathcal{F}^l(\cdot)$ denotes the transformer block at layer l, and $\Gamma(\cdot)$ is the post-processing of downstream tasks. Each block uses a multi-head attention (MHA) layer, followed by a multi-layer perceptron, to transform the output tokens of the previous block:

$$t^l = \mathcal{F}^l(t^{l-1}) = \Theta^l(O_f^l), \tag{2}$$

where t^l and t^{l-1} are the output of layer l and $l-1$, and $\Theta^l(\cdot)$ is the multi-layer perceptron of layer l. O_f^l is the global attention among all patches extracted by the MHA in layer l, which could be formulated as:

$$O_f^l = Softmax(\frac{QK^T}{\sqrt{d_f}})V, \quad \text{where } V, Q, K = \Lambda(t^{l-1}). \tag{3}$$

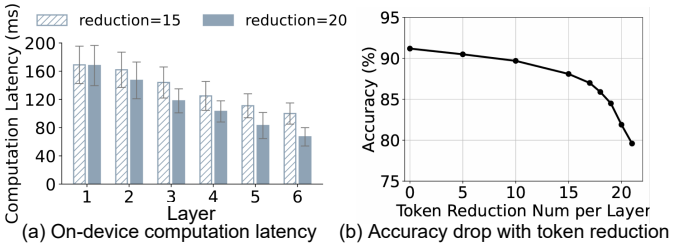
$\Lambda(\cdot)$ is the linear projection function and generates features of the same shape as the input.

In traditional transformers, t^l typically has the same shape as t^{l-1}. Hence, the computational complexity of Eq. 3 remains consistent across all layers, which is $O(N^2)$. For high-resolution images, the number of tokens N is typically substantial, leading to greater computation than that of CNNs. To address this, many existing works propose to reduce the number of tokens at runtime for faster inference speed [16]–[18]. They either prune the less informative tokens or combine similar tokens in a layer-wise manner so that the token length progressively decreases:

$$|t^l| = |t^{l-1}| - r^l, \quad \text{where} \quad r^l \geq 1. \tag{4}$$

Specifically, the token length at layer l is constrained by:

$$|t^l| < |t^j|, \quad \forall j \in \{1, 2, \ldots, l\}. \tag{5}$$



(a) On-device computation latency (b) Accuracy drop with token reduction

Fig. 2: The per-layer computation latency of ViT-Base on Raspberry Pi 4B and the accuracy on CIFAR-100 when progressively reducing a fixed number of tokens at each layer following prior method [14]. In mobile-cloud collaboration, shallow layers are deployed on the mobile device, yet consume longer computation latency than deep layers.

This ensures that tokens in later layers are significantly fewer than those in early layers, reducing the quadratic complexity $O(|t^l|^2)$ of self-attention computation in deeper layers.

B. Mobile-Cloud Collaborative Transformer Inference

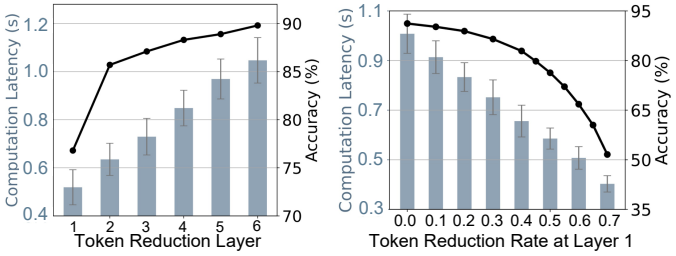
This progressive token reduction described in Eq. 5 leads to *a gradual decrease in intermediate feature sizes across layers*, just like that of CNNs, and opens up opportunities for mobile-cloud collaborative Transformer inference [9], [14]. Specifically, more layers can be processed on the mobile device, which reduces communication overhead due to reduced feature size, especially under limited network bandwidth conditions.

However, such methods *fail to sufficiently decrease the computation overhead of shallow layers that are deployed on the mobile device and lead to significant computational latency*. Specifically, when layers 1 to l are executed on the mobile device and layers $l+1$ to L are offloaded to the cloud, there exists an imbalance between computation latency on the mobile device L_m and cloud L_c, which can be formulated as:

$$L_m = \frac{\sum_{i=1}^{l} O(|t^i|^2)}{P_m} \gg \frac{\sum_{j=l+1}^{L} O(|t^j|^2)}{P_c} = L_c, \quad (6)$$

where $\forall i \in [1, l]$, $j \in [l+1, L]$, $|t^i| > |t^j|$. P_m and P_c denote the processing power of the mobile device and cloud server, and satisfy the constraint $P_m \ll P_c$. This imbalance in token length and processing power leads to a significant computational bottleneck on the mobile device. In the following, we conduct experiments to investigate this issue in real-world IoT settings.

Investigation Experiments. We use a Raspberry Pi 4B as the mobile device, which collaborates with the cloud server via 2.4 GHz WiFi. The cloud server is equipped with an Nvidia 4090 GPU, enabling high-speed inference. On this testbed, we evaluate the performance of ViT-Base model [19], which is a 12-layer ViT model. Following the prior method [14], we apply the token reduction method [18], *i.e.*, progressively reducing a fixed number of tokens. The computation latency of each layer with a batch size of 16 is shown in Fig. 2 (a). Meanwhile, we plot the inference accuracy of CIFAR-100 dataset in Fig. 2 (b) with various per-layer token reduction settings. From the results, following limitations are observed:



(a) Impact of token reduction point (b) Impact of token reduction rate

Fig. 3: Computation latency of the shallow 6 layers on Raspberry Pi (blue bar) and accuracy (black line) when applying token reduction at a single specific layer. Reducing more tokens in the shallower layer significantly reduces computation latency, yet sacrificing accuracy drastically.

1) In mobile-cloud collaboration, shallow layers are deployed on the mobile device, yet consume longer computation latency than deeper layers. For instance, on the Raspberry Pi, the execution latency of the first three layers is 433 ms, which is 1.71× that of Layer 4 to 6 (253 ms), when the token reduction number is set to 20. For the model split at Layer 3, on-device inference latency becomes the dominant component of the overall computation latency, whereas cloud inference latency remains below 1 ms. The primary reason is that existing pruning strategies overlook the mobile-cloud resource disparity, leading to inadequate pruning of the shallow layers on the mobile device.

2) While reducing a large number of tokens can accelerate inference, it often results in significant accuracy degradation. When the token reduction number per layer increases from 15 to 20, the total inference latency of the 6 shallow layers decreases by 15.4%. However, the inference accuracy drops sharply by 6.2% (from 88.1% to 81.9%), which severely impacts accuracy-critical applications such as action recognition and medical image analysis.

Motivation. The analysis above reveals a key performance bottleneck in mobile-cloud collaborative ViT inference: shallow layers executed on resource-constrained mobile devices suffer from high computation latency. Yet existing token pruning methods typically apply gradual token reduction across layers, and can not sufficiently reduce token numbers initially while preserving high accuracy. Inspired by this, we wonder *whether we can aggressively reduce the number of tokens in the initial layer to lower the on-device computing delay?*

C. Challenges

We conduct experiments where token reduction is applied to *only a single selected layer*, instead of progressively reducing tokens across all layers. Following the existing method [17], we calculate the significance score of each token based on the attention values calculated in the transformer layer and preserve the top-K important tokens with the highest scores. We then reduce token numbers at different layers with various token reduction rates. The latency for computing the first 6 layers on Raspberry Pi and corresponding accuracies are shown in Fig. 3. The following observations are obtained:

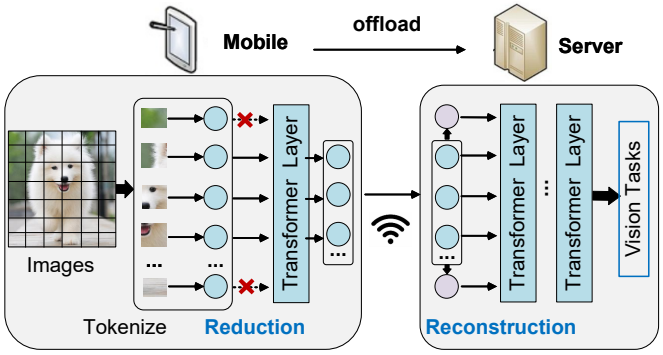


Fig. 4: The illustration of Mercury. A remarkable number of tokens are reduced at the initial transformer layer on the mobile device to significantly reduce computation and communication latency, and are reconstructed on the cloud side to achieve higher inference accuracy.

1) Applying token reduction in the shallower layer remarkably reduces computation latency, but at the cost of substantial accuracy degradation. For instance, pruning 50% of tokens at Layer 1 instead of Layer 6 can reduce the total computational latency by half (from 1.05 s to 0.52 s). However, this comes with a drastic drop in accuracy of 13% (from 89.8% to 76.8%). The reason behind this is that the shallower layers generate fewer redundant intermediate features [20], which are more severely affected by token reduction. Therefore, maintaining accuracy is much more challenging when aggressively reducing tokens in the earlier layers.

2) As the token reduction rate increases, computation latency decreases approximately linearly, while accuracy drops non-linearly at an accelerating rate. For example, increasing the token reduction rate from 0 to 0.3 reduces latency by 0.25 s, with only an accuracy drop of 4.7%. While increasing the rate from 0.4 to 0.7 also yields a 0.25 s latency reduction, it causes a dramatic 31.3% drop in accuracy. This indicates that as the reduction becomes more aggressive, it becomes increasingly difficult to preserve inference accuracy.

These findings highlight that while aggressively reducing tokens in the initial layers can significantly lower latency, it becomes increasingly challenging to maintain high accuracy.

III. System Design

Motivated by the above observations, we propose Mercury to fully minimize the computational latency on mobile side, and leverage the powerful computation resources on the cloud side to enhance the inference accuracy.

A. Problem Formulation and System Overview

The optimization objective of Mercury is formulated as:

$$\begin{aligned} \min_{\tau, M} \quad & L_o(\tau, M) = L_m(\tau) + L_t(\tau) + L_c(\tau, M), \\ \text{s.t.} \quad & \Delta A(\tau, M) \leq \epsilon. \end{aligned} \tag{7}$$

The overall latency $L_o(\tau, M)$ with token reduction strategy τ and reconstructed tokens M are composed of three parts: latencies of mobile inference $L_m(\tau)$, feature transmission $L_t(\tau)$, and cloud inference $L_c(\tau, M)$. While $\Delta A(\tau, M)$ denotes the accuracy drop resulting from token reduction and reconstruction with an upper bound of ϵ.

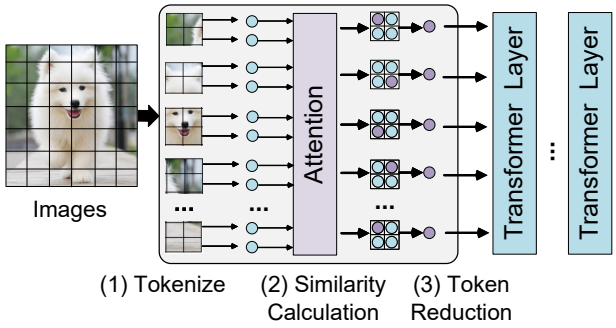


Fig. 5: The illustration of our token reduction method. After global similarity among tokens is calculated, the token in each subregion (*e.g.*, 2×2 grid) with the highest score (purple circle) is preserved, while others can be pruned.

Using the Lagrange multiplier λ, the overall optimization objective of the system could be defined as:

$$\min \left(L_m(\tau) + L_t(\tau) + L_c(\tau, M) + \lambda \Delta A(\tau, M) \right). \tag{8}$$

In practical systems, the optimization variables τ (token pruning strategy) and M (reconstructed tokens) affect both latency and accuracy in complex, non-linear, and system-dependent ways, which are difficult to model analytically in a tractable form. To cope with this challenge, Mercury adopts a two-stage framework (as shown in Fig. 4) that decouples the optimization of τ and M. In the first stage, τ is optimized to significantly reduce the number of tokens on the mobile side while retaining task-relevant information to mitigate accuracy degradation. In the second stage, the pruned tokens are reconstructed on the cloud side via M to further compensate for accuracy loss, with minimal additional cost.

Problem Definition. Given a tokenized image feature set $X = \{x_1, x_2, \ldots, x_N\}$, the first stage, *token reduction*, aims to decrease the number of tokens N while preserving the essential information required for downstream tasks. Formally, the objective is to find an optimal strategy τ that minimizes:

$$\arg\min_{\tau} \| f(X) - f(\tau(X)) \|, \tag{9}$$

where $\tau(X) \subseteq X$ is a selected subset after token pruning. The Transformer f is composed of a mobile-side component f_m and a cloud-side component f_c. The token features transmitted to the cloud, denoted by $X' = f_m(X - X_r)$, have length $N - K$, where K is the number of pruned tokens.

In the second stage, *token reconstruction* aims to generate tokens $M = \{m_1, m_2, \ldots, m_K\}$ that match the shape of the removed tokens X_r and satisfy the following optimization:

$$\arg\min_{M} \| f(X) - f_c(X' + M) \|, \tag{10}$$

where X' and M are concatenated and passed into the cloud-side Transformer f_c.

Algorithm 1: Regionally Decentralized Token Pruning

Input: Tokenized features $X = \{x_1, x_2, \ldots, x_N\}$,
Global attention values $O_f = \{o_1, o_2, \ldots, o_N\}$.
Output: Tokens to prune $X_r = \{x_1, x_2, \ldots, x_K\}$.

for $i = 1$ **to** N **do**
  for $j = 1$ **to** N **do**
    Compute the cosine similarity via Eq. 11;
  Compute semantic score for each token via Eq. 12;
for $u = 1$ **to** $\sqrt{N}$ **do**
  for $v = 1$ **to** $\sqrt{N}$ **do**
    Perform 2D gridding via Eq. 13;
for $p = 0$ **to** $\frac{\sqrt{N}}{l} - 1$ **do**
  for $q = 0$ **to** $\frac{\sqrt{N}}{l} - 1$ **do**
    Partition into subregions via Eq. 14;
    Find token with maximum score via Eq. 15;
    Calculate the global index as Eq. 16;
Determine the preserved tokens X^* as Eq. 17;
Determine the pruned tokens X_r as Eq. 18;
return X_r

B. Token Reduction at the Initial Transformer Layer

Motivation. Existing methods [9], [14] adopt the similarity-based pairing strategy to reduce tokens [18], [21], which randomly divides all tokens into two groups, then selects the *most similar tokens* from each group for merging. This random bipartition method only supports small-scale pruning (less than 50%), and cannot be applied to our scenario of large-scale (*e.g.*, 70%) pruning in the initial layers. Alternatively, the *significance score* based token pruning methods [17], [22] can support any pruning ratio, but cause severe accuracy degradation when applied to initial layers as shown in recent literature [23]. Therefore, in Mercury, a novel token reduction method is required. Inspired by the fact that features extracted by initial layers exhibit low aggregation and are scattered across various subregions of the image, we propose to *preserve tokens of higher similarity scores within each subregion*, as illustrated by purple circles in Fig. 5. It performs token pruning in a regionally decentralized manner while maintaining the scattered semantic features for higher inference accuracy.

Algorithm Design. Inspired by [31], we identify the important tokens to be preserved in each region and prune the other tokens as shown in Fig. 5. The selection of important tokens is mainly based on the semantic similarity scores computed from the attention of the initial transformer layer. After tokens with the highest similarity are selected to be preserved, the other tokens are sorted by their similarity to the preserved ones: pruning tokens with higher similarity incurs less semantic information loss. Given that semantic features in the early layer are less aggregated, *i.e.*, scattered across various subregions of the image, we propose the *Regionally Decentralized Token Pruning Algorithm*, as described in Algorithm 1.

Input and similarity score calculation. Besides the tokenized image features $X = \{x_1, x_2, \ldots, x_N\}$, we reuse the global attention features $O_f = \{o_1, o_2, \ldots, o_N\}$ that are already calculated by the initial transformer layer, which is described in Eq. 3. It is then used to calculate the semantic similarity among all tokens in the following:

$$\text{Sim}_{i,j} = \frac{o_i \cdot o_j}{\|o_i\| \|o_j\|}, \tag{11}$$

where $Sim_{i,j}$ denotes the cosine similarity among the i_{th} and j_{th} token features.

For each token, semantic similarity score s_i is calculated:

$$s_i = \sum\nolimits_{j=1}^{N} \text{Sim}_{i,j}. \tag{12}$$

A higher score indicates more semantic similarity to others, and thus is more representative of the global context.

Subregion partition and preserved tokens selection. Afterwards, as shown in Fig. 5, during the token selection, we fully consider the unaggregated features scattered across various regions of the image, and divide the similarity score matrix into grids to preserve scattered tokens across different regions. Using $k = \sqrt{N}$ to denote the width and height of the 2D-grid transformed from the score set s, each element in the grid at position $(u,\ v)$ is defined as:

$$G(u, v) = s_{(u-1)k+v}. \tag{13}$$

They are further partitioned into subregions with the scale size of l. A subregion in row p and column q maintains the following elements:

$$\text{Subregion}_{p,q} = \{G(pl + u, ql + v) \mid u, v \in \{1, 2, \ldots, l\}\}, \tag{14}$$

where $p, q \in \left\{0, 1, \ldots, \frac{k}{l} - 1\right\}$. For each subregion, we only preserve the one with the maximum score. The index of preserved tokens in $\text{Subregion}_{p,q}$ is given by:

$$u^*_{p,q}, v^*_{p,q} = \underset{u,v \in \{1,2,\ldots,l\}}{\arg\max}\ \text{Subregion}_{p,q}(u, v), \tag{15}$$

where $u^*_{p,q}, v^*_{p,q}$ are the row and column offsets within the $\text{Subregion}_{p,q}$. They are then converted to the global index in the original token set X:

$$\text{index}_{p,q} = (p \cdot l + u^*_{p,q}) \cdot k + (q \cdot l + v^*_{p,q}). \tag{16}$$

Leveraging the index, the preserved tokens X^* are selected:

$$X^* = \{x_{\text{index}_{p,q}} \mid \forall p, q\}. \tag{17}$$

In this way, the most informative tokens are preserved within each subregion. In our experiments, l is set to 2, which selects 25% tokens to preserve.

Pruned Token Selection. Through the above approach, up to 75% of the tokens can be pruned. To enable a more flexible token reduction rate, *e.g.*, any proportion from 0% to 75%, we then sort the prunable set of tokens $X - X^*$ according to their similarity to X^*, and prune the top-K ones, denoted as X_r:

$$X_r = \underset{x_i \in X - X^*}{\arg\text{top-K}} \left(\max_{x_j \in X^*} \text{Sim}(x_i, x_j) \right). \tag{18}$$

After selecting the pruned tokens X_r, the other tokens $X - X_r$ are further processed by remaining transformer layers on the mobile device and then transmitted to the cloud for inference.

As described in Sec. IV-C, by effectively preserving critical semantic features that are sparsely distributed at the initial layer, our method achieves *an accuracy improvement of 10.4% to 23.1%* compared to the existing pruning method [17] under a token reduction rate of 50% to 75%.

C. Token Reconstruction on the Cloud Server

Motivation. Although our token pruning method achieves significant improvements, a high reduction rate still causes a sharp drop in accuracy (verified in Sec. IV-C). The underlying reason lies not only in the loss of semantic information, but also in the drastic reduction of computational load. For instance, when 75% of tokens (N) are reduced, transformer's computational complexity of $O(N^2)$ is reduced by 93.75% (to 1/16th of the original), which limits the model's ability to extract information. To verify this, we *zero-pad the received tokens* to the original length N on the cloud, and observe that the inference accuracy with a 75% pruning rate at the initial layer *improves from 67.8% to 75.1%* on CIFAR-100. *This verifies the importance of recovering the number of tokens for inference accuracy.* Note that, given the substantial disparity in computational resources between mobile devices and the cloud, the computational overhead incurred by the increased token count on the cloud side results in only negligible latency. Hence, token reconstruction on the cloud server is quite favorable for the accuracy-latency trade-off.

Reconstruction Method. The intuitive and straightforward approach is to introduce numerous additional parameters to learn the mapping, such as introducing additional neural networks with heavy parameters like RCNet [24]. However, such methods ignore the inherent architecture and capacity of the existing Transformer model f_c. Inspired by [25], we adopt a more efficient strategy: introduce M as trainable meta-parameters in the cloud-side Transformer f_c, and learn to mitigate the adverse effects of token reduction. This enables implicit learning of feature completion through the Transformer's native attention operations across all layers.

The additional tokens M are randomly initialized as trainable parameters: $M \in \mathbb{R}^{K \times d_f}$, where K is the number of reduced tokens (same as the length of X_r). Then M are concatenated with the cloud-received token features X' as the input of cloud-side Transformer component f_c. During the training process, we employ a standard cross-entropy loss for the classification task using the training set data:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N_s} \sum\nolimits_{i=1}^{N_s} y_i \log(p_i), \tag{19}$$

where y_i denotes the ground-truth label, p_i represents the predicted class probability, and N_s is the number of data samples. This classification loss serves as the sole supervision signal for end-to-end optimization. Note that, during training, we freeze the parameters of each transformer layer in f_c, and only fine-tune the small number of parameters in the classification head (Γ in Eq.1), as well as the added parameters of M. As for inference, the well-trained M are directly concatenated with X' as prefixes to the input of f_c.

Analysis of Overhead. Our reconstruction method incurs minimal computational latency and storage overhead on the cloud side. *As for computational latency*, the supplementary features M are stored as hyperparameters on the cloud server and are directly used during inference, introducing no generation-related computational overhead. Since the expanded token sequence ($M + X'$) maintains the same length as the original input tokens X, the cloud-side Transformer f_c incurs the same computational complexity as the unpruned case. Given the powerful computational resources in the cloud server, this results in negligible overhead of inference latency. *As for storage overhead*, the only additional parameters come from M, which is minimal in size. For instance, reconstructing 75% tokens for a ViT-Base model *only introduces 0.11M extra parameters, which is just 0.13% of the original model.*

IV. Evaluation

In this section, we introduce the implementation details of Mercury and compare it with state-of-the-art methods in terms of latency, accuracy, and energy consumption.

A. Implementation

Hardware. We utilize a Raspberry Pi 4B with 8GB RAM and Cortex-A72 (ARM v8) SoC as the mobile device, where the shallow transformer layers are deployed to perform mobile-side computation. The quad-core 1.5GHz processor is a typical resource-constrained environment for on-device computation. The calculated intermediate features are then transmitted to the cloud server via wireless networks. On the cloud server, an NVIDIA 4090 GPU and Intel(R) Xeon W7-3455 @3.3 GHz processors are deployed to provide typical cloud-side computing resources, which are capable of completing the ViT inference in less than 1 ms. On both devices, we install Ubuntu operating system as well as Python 3.12.7 and PyTorch 2.2.1 to support the transformer model construction.

Wireless Network Traces. To evaluate Mercury's performance under real-world scenarios, we utilize the dataset of 4G and 5G wireless networks uplink traces to simulate the network conditions [26] during feature transmission. The dataset contains the bandwidth variations under three different scenarios: static, walking, and driving. After cleaning out the abnormal data, the dataset contains the record of 126,216 seconds 5G uplink bandwidth variation as well as 74,119 seconds of 4G performance. The average bandwidth of 5G and 4G dataset is 48.2 Mbps and 37.3 Mbps, respectively.

Baselines. Two state-of-the-art transformer inference methods are implemented as the comparison baselines: 1) OTAS [14], an elastic transformer serving system that flexibly adjusts the number of tokens (either reducing [18] or increasing [25]) to enhance throughput or improve accuracy. We adopt its elastic strategy to modify the model, and partition the model into mobile and cloud parts for collaborative deployment. 2) Janus [9], a collaborative transformer inference framework

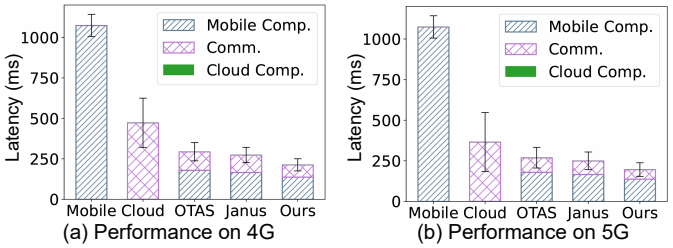


Fig. 6: Comparison of mobile-only, cloud-only, OTAS [14], Janus [9], and our method in terms of computation and communication latency.

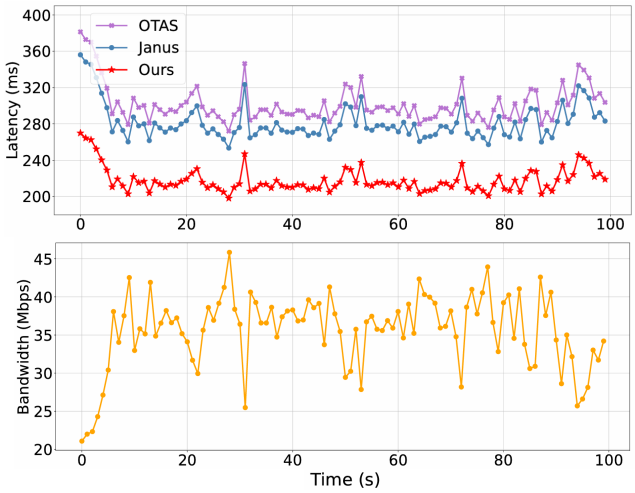


Fig. 7: Overall latency comparison under real 4G traces (yellow lines). Our method reduces the latency by 26% on average compared to the state-of-the-art methods [9], [14].

that adopts an exponential token reduction strategy to reduce more tokens on resource-constrained mobile devices, thereby reducing the overall latency. We implement their methods based on the timm library [27] and the code of TOME [18].

Tasks and Evaluation Metrics. We employ the image classification dataset of CIFAR-100 [28] and ImageNet-1K [29] as the task to test Mercury's performance in terms of accuracy and latency. We employ the well-trained DeiT model with distillation token [19] as backbone, and train our additional feature reconstruction parameters on the cloud using the training dataset. The *classification accuracy* on the test set is then utilized as the evaluation metric. As for the latency, we separately measure the *computation latency* on the mobile and cloud, then use the bandwidth trace in the wireless network dataset to calculate the average *communication latency*.

B. System-level Evaluation

Effectiveness of Mobile-Cloud Collaborative Inference. We first compare the collaborative methods with cloud-only and mobile-only inference performance. The ViT-Base model is split after Layer 1, and we apply token reduction methods to make the recognition accuracy on ImageNet at a similar level (2.8% accuracy loss from the original model) for comparability. *1) Mobile-Only.* When the whole model is conducted on the mobile device, the latency only contains the on-device

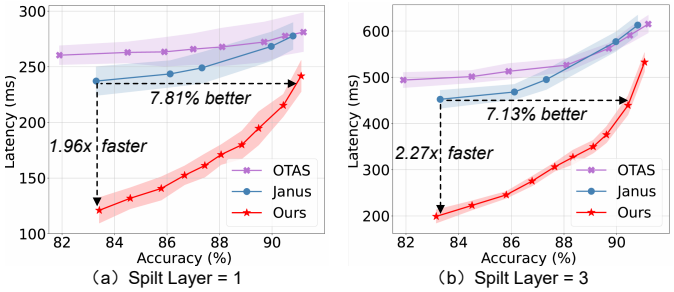


Fig. 8: Comparison of accuracy-latency trade-off under 5G networks on CIFAR-100 dataset with different splitting layers.

computing latency. As shown in Fig. 6, due to the poor computing resources of the mobile device, the computation latency of mobile-only inference is as high as 1073 ms. *2) Cloud-only.* When the whole transformer inference is completed on the cloud, the high-resolution images with an average size of 2.2 MB require uploading. The transmission latency of 472 ms in 4G, 365 ms in 5G dominates the whole latency. Meanwhile, the upload of original images causes a risk of privacy leakage, which is not applicable in scenarios like face recognition where strict privacy requirements exist. *3) Collaborative methods.* Compared to mobile-only and cloud-only methods, the collaborative methods significantly decrease the computation latency by offloading partial layers to the cloud and minimize the communication latency by transmitting the compact and non-privacy intermediate features (less than 0.6 MB) extracted by the first transformer layer. Therefore, the collaborative methods achieve better performance under both 4G and 5G networks. Specifically, our Mercury achieves speedups of *5.06*× and *2.23*× compared to mobile-only and cloud-only methods under 4G networks.

Comparison with the State-Of-The-Art Methods. *1) Latency comparison.* We first set the splitting position after the first layer, and adjust the token reduction rate to keep the accuracies of the three methods at a similar level (80.5% on ImageNet, with a 2.8% accuracy drop compared to the original model), then compare their overall latency. As shown in Fig. 6, the average overall inference latency of Mercury is 212 ms under 4G and 195 ms under 5G, representing approximately a *26% reduction* compared to the comparative methods (OTAS [14] and Janus [9]). A more fine-grained comparison of latency under different network bandwidths is shown in Fig. 7. Compared with the comparative methods, our method achieves *accelerations of 1.32*× *and 1.23*×, respectively. This is because our method enables substantial token reduction in the initial layer, effectively lowering both computation and communication latency.

2) Accuracy-latency trade-off. All three collaborative works exhibit *elastic service characteristics*: they can achieve a trade-off between accuracy and latency by adjusting the token reduction rate, and can dynamically select the partitioning layer to alleviate cloud burden or accelerate the inference process. The accuracy-latency trade-off performance under 5G networks on CIFAR-100 dataset is shown in Fig. 8. When the token reduction rate is substantial (accuracy is lower

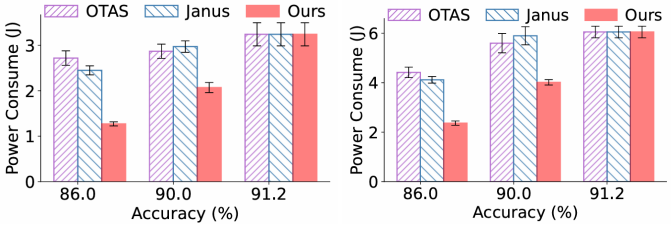


Fig. 9: Comparison of the power consumption on the Raspberry Pi with different accuracy thresholds and splitting layers.

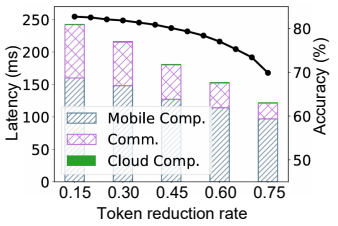


Fig. 10: Token reduction after Layer 1 on ImageNet.

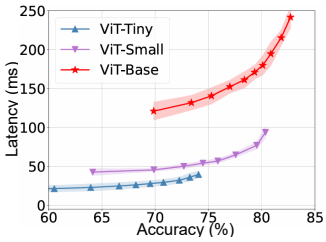


Fig. 11: Effectiveness on different ViT models.

TABLE I: Comparison of overhead when split after Layer 3.

Method	Accuracy On CIFAR-100	Latency		Parameters on cloud	FLOPs on cloud
		Overall	Cloud-side		
Vanilla	91.2%	636 ms	0.86 ms	63.79 M	26.3 G
Janus [9]	89.9%	577 ms	0.85 ms	63.79 M	20.0 G
Ours	**89.9%**	**407 ms**	**0.86 ms**	**63.84 M**	**26.3 G**

than 86%), Janus [9] outperforms OTAS [14] serving as a stronger baseline, due to its nonlinear pruning strategy, which prioritizes token reduction in the shallow layers. Compared with Janus [9], our method further *accelerates the inference speed by 1.17× up to 2.27×* with the same accuracy threshold. When adopting the same latency threshold, our method *can enhance the accuracy by 2.96% up to 7.81%*. These substantial improvements are attributed to our method's ability to prune a large number of tokens at the initial layer while ensuring accuracy through the innovative design of regionally decentralized pruning and meta-parameter-based token reconstruction. These results verify that Mercury achieves optimal accuracy-latency trade-off leveraging the novel design.

3) Power consumption on the mobile device. Since IoT devices typically rely on battery power, the power consumption on mobile devices directly affects the device's battery life. Given the maximum power consumption of 6W for Raspberry Pi 4B, the energy consumption of Janus [9], OTAS [14], and our Mercury is shown in Fig. 9. Without token reduction (with 91.2% accuracy), the three methods exhibit similar energy consumption: 3.24 J and 6.05 J when executing 3 layers and 6 layers, respectively. With an accuracy loss of 1.2% on CIFAR-100, the comparative method reduces energy consumption by 7.4%, while our method achieves *a significantly greater reduction of over 33.6%*, which is *4.54× the energy savings of comparative methods*. When the accuracy threshold is 86%, the power consumption of our method is *only half that of the comparative methods*, and *40% of the vanilla model*. This demonstrates that our method can substantially extend the battery life of IoT devices during collaborative inference tasks.

4) Overhead. Our method's primary overheads stem from the increased parameters and the additional computation incurred during token number reconstruction. The comparison with the vanilla model (without token reduction) and Janus [9] is shown in Table I, where Transformer is split after Layer 3 and the accuracy threshold on CIFAR-100 is set equivalently. The cloud-side parameter increment is approximately *0.05 M, constituting merely 0.08% of the original cloud-side model*. This introduces an exceptionally minimal cost. In terms of computational load, the cost of our method is *identical to that of the vanilla model of 26.3 GFLOPs*. This equivalence arises because our token reconstruction method treats the additional tokens as hyperparameters, which are learned during the training stage and introduce no extra computations during the inference. Consequently, the inference latency matches the vanilla approach, completing within 1 ms on the cloud server. Overall, the introduced storage and computation overhead is negligible compared to the 1.56× overall acceleration.

C. Micro-Benchmark Experiments

Impact of Token Reduction Rate. We plot the performance on ImageNet in Fig. 10 with various token reduction rates applied after Layer 1. The accuracy loss remains minimal (within 3%) when the token reduction rate is below 40%, and accelerates after exceeding 60%. For tasks with low latency requirements, the reduction rate can be increased to 60%, achieving a *1.72× speedup* at the cost of a 5.7% drop in accuracy. In extreme cases, a 75% reduction rate can be used to achieve coarse-grained but fast identification, resulting in a *2.00× speedup* while maintaining a *top-5 accuracy of 88.9%*.

Generalization on Different ViT Models. We separately apply our methods to various ViT models, *i.e.*, ViT-Base, ViT-Small, and ViT-Tiny, and plot the performance on ImageNet in Fig. 11. Across all three models, our method achieves a favorable trade-off between accuracy and latency: compared to the original models, *a 5% reduction in accuracy leads to speedups of 1.50×, 1.65×, and 1.43×, respectively.* These results demonstrate that our method can be universally applied to ViT architectures of varying parameter scales, providing elastic performance tuning with optimal accuracy-latency trade-off.

Effectiveness of Our Design. We evaluate four designs on the ViT-Base model with CIFAR-100: 1) multi-round token reductions without restoration (OTAS [14] and Janus [9]); 2) multi-round reductions with restoration (OTAS+R and Janus+R); 3) single-round reduction without restoration (our pruning strategy and the existing method [17]); 4) single-round reduction with restoration (ours). We first compare methods 1, 2, and 4 using the splitting point after Layer 3 as shown in Fig. 12. Adding our reconstruction module to OTAS and Janus improves accuracy by up to 7.9% and 4.2%. Compared to the improved performance, our single-round reduction with reconstruction *achieves a 1.73× speedup* under the same accuracy threshold, demonstrating the efficiency of

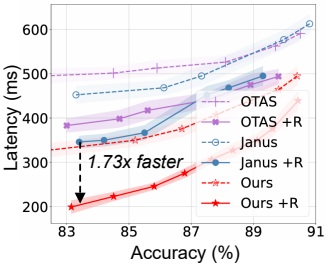


Fig. 12: Comparison of different token pruning strategies under the same cloud-side token reconstruction (+R) setup.

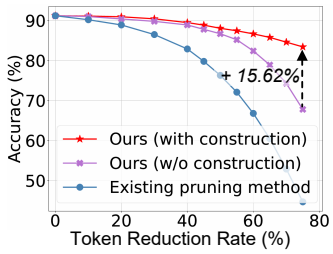


Fig. 13: Performance with and without token reconstruction, compared with the existing token pruning method [17].

our early-layer pruning strategy. We then compare methods 3 and 4 splitting after the initial layer, as shown in Fig. 13. Unlike the existing method [17], which prunes tokens with lower significance scores, our regionally decentralized pruning strategy better retains feature diversity in the early layers, leading to a *10.4% accuracy improvement* when 50% of tokens are pruned. Based on this, integrating our token reconstruction module provides an additional accuracy improvement: *15.62% higher* under a 75% token pruning rate. These results highlight strengths of our token pruning and reconstruction designs.

Choice of Token Reconstruction Method. We compare our meta-parameter-based token reconstruction method with two alternatives: duplicating tokens for each region and generating tokens using a deconvolutional layer. Table II presents the comparison in terms of accuracy gains and parameters/computation overhead on CIFAR-100, where 75% of tokens are reduced after Layer 3. The *duplicate method* only replicates the remaining tokens in each region, thus introducing almost no additional parameters or computations. But it achieves low accuracy due to its inability to enhance the remaining token features. The *generation method* achieves a competitive accuracy of 82.55%, leveraging the abundant parameters in deconvolutional layers for upsampling. Nevertheless, its excessive additional parameters, memory footprint, and computational overhead make it inefficient for large-scale deployments. In contrast, our meta-parameter-based token reconstruction method introduces *86× fewer additional parameters*, incurs *no extra computation beyond vanilla ViT inference*, and achieves the *highest accuracy of 83.15%*.

V. Related Work

Inference on Mobile Devices. Given the limited computing resources of mobile devices, many methods employ model compression techniques [17], [18], [30], [31] to improve the inference speed. However, these methods typically require sacrificing a relatively high degree of accuracy to achieve substantial acceleration. An emerging approach [32]–[34] leverages multiple mobile devices for collaboration to achieve fast and accurate inference. Nevertheless, resource constraints in mobile networks severely limit the performance of such collaboration, which requires frequent communication [7], [35]. In contrast, the mobile-cloud collaborative approach, by harnessing the powerful resources of the cloud, can achieve more significant acceleration and enables full reuse of existing models without distillation and other adjustments.

TABLE II: Comparison of token reconstruction methods to recover 75% tokens on CIFAR-100 when split after Layer 3.

Method	Accuracy	Parameters	FLOPs
Duplicate	57.71%	63.79 (+0.00) M	26.3 (+0.0) G
Generation	82.55%	73.23 (+9.44) M	28.1 (+1.8) G
Ours	**83.15%**	**63.90 (+0.11) M**	**26.3 (+0.0) G**

Mobile-Cloud Collaborative Inference. Partially offloading the inference task to the cloud combines the advantages of both mobile-side and cloud-side inference. It not only avoids privacy concerns that may arise from fully offloading raw data to the cloud, but also fully leverages the powerful resources of the cloud for acceleration. Most existing methods are designed for CNN inference: they either focus on selecting the optimal split layer [10], [36]–[40] based on the computational and communication overhead, or design redundancy and progressive feature transmission algorithm [11]–[13], [24], [41] to counter the potential packet loss and jittery bandwidth during wireless transmission. Few existing works have focused on the mobile-cloud collaborative inference of transformers, which exhibits unique characteristics: intermediate features generated by each layer during computation maintain consistent dimensions. This means that, unlike CNN-based methods where feature counts decrease as the split layer depth increases, splitting at later layers of Transformer can not reduce the amount of transmitted data. Janus [9] first introduces the layer-by-layer token reduction method [18] into collaborative ViT inference, enabling transformers to exhibit decreasing feature counts across layers. In this paper, we point out that such a layer-wise feature reduction framework fails to fully account for the computational discrepancies between mobile devices and the cloud, resulting in excessively high latency on the mobile side. To address this issue, we propose an innovative architecture in Mercury that drastically reduces tokens on the mobile device at the initial layer, and reconstructs them in the cloud, achieving an optimal accuracy-latency trade-off.

VI. Conclusion

In this work, we present Mercury, a novel mobile-cloud collaborative framework for ViT inference. By strategically pruning tokens early on mobile devices and reconstructing them in the cloud, Mercury fully considers the mobile-cloud resource disparity and achieves optimal efficiency. The similarity-based regional pruning preserves critical semantic information, while the meta-parameter-based reconstruction introduces minimal computation and parameter overhead. Experimental evaluations across real network conditions validate that Mercury outperforms existing methods in latency reduction and energy efficiency, while preserving comparable accuracy. Our framework offers a promising solution for deploying the high-performance ViT in real-time IoT scenarios, where an optimal accuracy-latency trade-off is essential.

REFERENCES

[1] C. Wang, Y. Yang, M. Qi, H. Zhang, and H. Ma, "Towards efficient object re-identification with a novel cloud-edge collaborative framework," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 7, 2025, pp. 7600–7608.

[2] H. Fu, K. Cui, C. Wang, M. Qi, and H. Ma, "Mutual distillation learning for person re-identification," *IEEE Transactions on Multimedia*, 2024.

[3] K. Liu, M. Zhu, H. Fu, H. Ma, and T.-S. Chua, "Enhancing anomaly detection in surveillance videos with transfer learning from action recognition," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 4664–4668.

[4] L. Perini, V. Vercruyssen, and J. Davis, "Transferring the contamination factor between anomaly detection domains by shape similarity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4128–4136.

[5] T. Yao, Y. Li, Y. Pan, and T. Mei, "Hiri-vit: Scaling vision transformer with high resolution inputs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[6] S. Mehta and M. Rastegari, "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer," *arXiv preprint arXiv:2110.02178*, 2021.

[7] G. Xu, Z. Hao, Y. Luo, H. Hu, J. An, and S. Mao, "Devit: Decomposing vision transformers for collaborative inference in edge devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5917–5932, 2023.

[8] L. Zhu, X. Wang, Z. Ke, W. Zhang, and R. W. Lau, "Biformer: Vision transformer with bi-level routing attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 10 323–10 333.

[9] L. Jiang, S. D. Fu, Y. Zhu, and B. Li, "Janus: Collaborative vision transformer under dynamic network environment," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025.

[10] Y. Duan and J. Wu, "Optimizing job offloading schedule for collaborative dnn inference," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 3436–3451, 2023.

[11] J. Huang, H. Guan, and D. Ganesan, "Re-thinking computation offload for efficient inference on iot devices with duty-cycled radios," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–15.

[12] S. Wang and X. Zhang, "Neuromessenger: Towards error tolerant distributed machine learning over edge networks," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 2058–2067.

[13] Y. Cheng, Z. Zhang, and S. Wang, "Rcif: Towards robust distributed dnn collaborative inference under highly lossy iot networks," *IEEE Internet of Things Journal*, 2024.

[14] J. Chen, W. Xu, Z. Hong, S. Guo, H. Wang, J. Zhang, and D. Zeng, "Otas: An elastic transformer serving system via token adaptation," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1021–1030.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[16] H. Yin, A. Vahdat, J. M. Alvarez, A. Mallya, J. Kautz, and P. Molchanov, "A-vit: Adaptive tokens for efficient vision transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 809–10 818.

[17] M. Fayyaz, S. A. Koohpayegani, F. R. Jafari, S. Sengupta, H. R. V. Joze, E. Sommerlade, H. Pirsiavash, and J. Gall, "Adaptive token sampling for efficient vision transformers," in *European Conference on Computer Vision*. Springer, 2022, pp. 396–414.

[18] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," in *The Eleventh International Conference on Learning Representations*, 2023.

[19] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[20] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan, "Efficientvit: Memory efficient vision transformer with cascaded group attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 14 420–14 430.

[21] Y. Duan, X. Li, and J. Wu, "Topology design and graph embedding for decentralized federated learning," *Intelligent and Converged Networks*, vol. 5, no. 2, pp. 100–115, 2024.

[22] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang *et al.*, "Spvit: Enabling faster vision transformers via latency-aware soft token pruning," in *European conference on computer vision*. Springer, 2022, pp. 620–640.

[23] X. Wu, F. Zeng, X. Wang, and X. Chen, "Ppt: Token pruning and pooling for efficient vision transformers," *arXiv preprint arXiv:2310.01812*, 2023.

[24] Y. Liang, J. Chang, M. Zang, and J. Wu, "Rcnet: Resilient collaborative dnn inference for wireless networks with high packet loss," *IEEE Transactions on Network Science and Engineering*, 2025.

[25] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, and S.-N. Lim, "Visual prompt tuning," in *European conference on computer vision*. Springer, 2022, pp. 709–727.

[26] M. Ghoshal, Z. J. Kong, Q. Xu, Z. Lu, S. Aggarwal, I. Khan, Y. Li, Y. C. Hu, and D. Koutsonikolas, "An in-depth study of uplink performance of 5g mmwave networks," in *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, 2022, pp. 29–35.

[27] R. Wightman, "Pytorch image models," https://github.com/huggingface/pytorch-image-models.

[28] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[30] L. Han, Z. Xiao, and Z. Li, "Dtmm: Deploying tinyml models on extremely weak iot devices with pruning," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1999–2008.

[31] E. Zhang, J. Tang, X. Ning, and L. Zhang, "Training-free and hardware-friendly acceleration for diffusion models via similarity-based token pruning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 9, 2025, pp. 9878–9886.

[32] C. Hu and B. Li, "When the edge meets transformers: Distributed inference with transformer models," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2024, pp. 82–92.

[33] S. Ye, J. Du, L. Zeng, W. Ou, X. Chu, Y. Lu, and X. Chen, "Galaxy: A resource-efficient collaborative edge ai system for in-situ transformer inference," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1001–1010.

[34] Y. Chen, Z. Niu, M. Roveri, and G. Casale, "Ceed: Collaborative early exit neural network inference at the edge," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[35] J. Du, Y. Wei, S. Ye, J. Jiang, X. Chen, D. Huang, and Y. Lu, "Co-designing transformer architectures for distributed inference with low communication," *IEEE Transactions on Parallel and Distributed Systems*, 2024.

[36] 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 SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[37] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[38] Y. Duan and J. Wu, "Joint optimization of dnn partition and scheduling for mobile cloud computing," in *Proceedings of the 50th International Conference on Parallel Processing*, 2021, pp. 1–10.

[39] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "Deepslicing: Collaborative and adaptive cnn inference with low latency," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2175–2187, 2021.

[40] J. Wu, L. Wang, Q. Jin, and F. Liu, "Graft: Efficient inference serving for hybrid deep learning with slo guarantees via dnn re-alignment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 2, pp. 280–296, 2024.

[41] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, and H. Kwon, "Clio: Enabling automatic compilation of deep learning pipelines across iot and cloud," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–12.