LIQOS-ViT: Learned Importance guided Quantization with Outlier awareness for Split inference ViTs

Amira Dhaouadi Cédric Adjih Nadjib Achir
Inria, France
{amira.dhaouadi, cedric.adjih, nadjib.achir}@inria.fr

***Abstract*—Vision Transformers (ViTs) are among the most powerful deep learning models, but their large computational and memory requirements make deployment on resource constrained IoT and edge devices difficult. Split computing addresses this limitation by dividing the inference process between the edge device and a remote server. However, transmitting intermediate features between the two sides can introduce a substantial communication overhead. To address this challenge, we propose an adaptive mixed-precision quantization framework for compressing ViT intermediate activations after model splitting. The proposed method first determines the importance of individual activation values and then applies grouping and normalization to reduce the impact of outliers. Based on the estimated importance scores, different bit-widths are assigned to activation values under a global compression constraint, allowing more important information to be represented with higher precision. The resulting quantized activations reduce transmission cost while preserving model accuracy, making the approach suitable for adaptive and efficient ViT inference in edge–server environments.**

***Index Terms*—Vision Transformer, split computing, adaptive quantization, mixed precision, gradient, bit allocation, outliers**

I. Introduction

Vision Transformers (ViTs) have become a dominant architecture in computer vision due to their ability to capture long-range dependencies and their competitive performance across diverse tasks [1]–[3]. However, deploying ViTs on edge and IoT devices remains challenging due to their high computational cost and large memory footprint. In particular, the computational complexity of self-attention scales rapidly with the number of input tokens, leading to a significant increase in both processing cost and memory usage. Furthermore, the large parameter count of ViT models further limits their deployment on resource-constrained hardware platforms [3]. These challenges are particularly pronounced in real-time edge settings, where full on-device inference is often infeasible.

The alternative is to offload inference to a remote server. However, doing so requires transmitting the input data, which may induce substantial communication overhead, increase end-to-end latency, and raise privacy concerns [4]. Split computing offers a more favorable trade-off between local execution and cloud or edge offloading [5]. Basically, the model is partitioned into two components. The first part of the model is executed at the edge-side and processes the input up to a designated split point, and transmits the resulting intermediate activations to the server-side, which executes the second part of the model. Compared with raw data offloading, this approach reduces the device workload and avoids direct transmission of the raw input, potentially offering additional privacy benefits [6].

Despite these benefits, the practical efficiency of split computing depends critically on the size of the transmitted intermediate representation. For ViTs, the activations at the split point can still be high-dimensional, making communication a major bottleneck in bandwidth and latency. A possible solution is to compress the transmitted activations. Among candidate approaches, quantization [7] is particularly attractive because it reduces communication costs by lowering the number of bits required to represent activations.

In this work, we focus on post-training activation quantization applied only to the transmitted activation in split ViT inference. Existing post-training quantization methods for ViTs mostly rely on fixed-precision quantization and often use a shared quantization policy at the tensor, channel, or token level. Some recent methods introduce improved scaling strategies or outlier-aware preprocessing, they generally remain limited to coarse-grained precision assignment [8]. This design may be suboptimal for split ViT inference, as activations at the split point do not contribute equally to the final prediction. Moreover, activation outliers can disproportionately enlarge the quantization range, thereby reducing the effective precision available to most activation values. As a result, fixed-precision quantization may introduce unnecessary information loss and lead to a suboptimal accuracy compression trade-off. To address these limitations, we propose an adaptive quantization method for split Vision Transformer inference that compresses intermediate activations while preserving predictive performance. Our method first estimates an activation importance map that captures the contribution of each activation to the model output. To reduce sensitivity to outliers, we then preprocess the activations by partitioning the values into groups, separating outliers from the main distribution. Each group can then be normalized independently. Based on these normalized representations, bit widths are allocated in an importance guided manner by solving an optimization problem that minimizes the quantization error under a global bit-budget constraint. This design is intended to improve communication efficiency, reduce transmission latency, and better preserve the

accuracy of the original ViT model.

The rest of this paper is structured as follows. In Section II, we review the state of the art on split computing for Vision Transformers and post-training quantization of ViT activations, followed by the motivation behind our work. Section III presents the pipeline of the proposed approach. The experimental results are then reported in Section IV. Finally, Section V concludes the paper.

II. Related Work and Motivation

A. Vision Transformer and Split Computing

A Vision Transformer (ViT) splits an input image into a sequence of patches. These patches are then processed by N Transformer blocks, each containing self-attention and feed-forward layers. After the final block, the resulting representation is passed to an Multi-Layer Perceptron (MLP) head to produce the prediction [9].

ViT inference is computationally expensive. Thus, split computing enables only part of the model to be executed on the device, while the remaining computation is offloaded to a server [5], [10]–[12]. However, transmitting large intermediate representations to the server can create a communication bottleneck in split computing. Previous works have addressed this issue using compression methods, such as autoencoder-based and sparsity-based techniques [5], [13]. These methods only target Convolutional Neural Network (CNNs), whose spatial feature maps differ significantly from ViT token embeddings, making their direct application to ViTs nontrivial. Intermediate data compression for transformer-based split inference remains largely unexplored, with recent approaches relying on sparsity or quantization [14], [15]. We therefore focus next on ViT quantization literature. Compared with sparsity-based methods, quantization provides compact low-bit representations, predictable communication overhead, and efficient hardware implementation [16].

B. Post-Training Quantization for ViT Activations

Post-training quantization (PTQ) improves model efficiency without retraining by mapping floating-point tensors to low-bit representations [17]. Unlike standard PTQ, which quantizes both weights and activations, a split-computing quantization framework must keep the weights fixed and quantize only the intermediate activations at the split point to reduce communication overhead. Quantization methods that are most applicable to our context can be generally categorized into two groups. The first category mainly reduces quantization error by handling outliers. For example, ADFQ-ViT [18] keeps values above a threshold in full precision and quantizes the remaining activations, while TSPTQ-ViT [19] uses separate scales for outlier and non-outlier channels. DopQ-ViT [20] and ORQ-ViTs [8] further improve robustness through outlier-aware scaling, decomposition, and clamping. However, aggressive outlier suppression or clamping can discard task-relevant high-magnitude responses, and quantizing outliers separately can be preferable. The second category allocates bit widths based on the required precision. Among these methods, PMQ [21] focuses on activation quantization and uses sensitivity-guided Pareto optimization to assign bit-widths. Other methods allocate more bits to important values from a rate-distortion perspective. Classical optimal bit allocation approaches minimize distortion under a rate constraint. The generalized BFOS algorithm assigns bits according to the rate-distortion slope of each component [22]. Gao et al. [23] apply rate-distortion theory to neural network compression by bounding the bit budget needed to achieve a target distortion. Similarly, RDO-Q [24] determines the number of bits per channel using a Lagrangian rate-distortion optimization framework. However, these methods target weight compression, whereas in split computing, the compressed object is an input-dependent intermediate activation tensor. This makes their direct application to activation compression non-applicable. Other methods, such as HAWQ-V2 [25], consider both weight and activation quantization, using a second-order Taylor approximation with the Hessian trace to estimate layer-wise sensitivity and determine the quantization configuration under a resource budget. In contrast, PTQ4ViT [26] uses Hessian information to guide quantization scale selection in ViTs, rather than performing explicit bit allocation under a rate-distortion objective. Although HAWQ-V2 could be adapted to ViTs, its sensitivity metric is based on a second-order approximation. This assumption is well-motivated for trained weights, where the loss is often treated as locally stationary, and the first-order term is neglected. Intermediate activations, however, are sample-dependent variables rather than optimized parameters, and their gradients with respect to the loss are generally nonzero. Therefore, the first-order Taylor term can be important in estimating the variation in loss caused by activation perturbations. This provides a direct and interpretable link between activation distortion and importance maps, thereby guiding mixed-precision quantization.

C. Importance Maps

Recently, importance maps have been used to guide quantization by allocating more bits to important components. For example, PMQ [21] performs patch-wise mixed-precision quantization for ViTs by estimating patch-level sensitivity and assigning different bit-widths across patches. However, this strategy is less suitable for fine-grained schemes that quantify individual activation values.

Finer-grained importance can be estimated using gradient-based importance map methods such as Grad-CAM and LRP [27], [28]. However, they are not always directly applicable to mixed-precision compression since they prioritize interpretability over quantization sensitivity. Perturbation-based methods estimate sensitivity by measuring the effect of noise or quantization on the final objective. DiffQ [29] injects pseudo-quantization noise into weights to jointly optimize weights and bit-widths, while NIPQ [30] extends this to weights and activations by learning bit-widths and truncation ranges through a noise proxy. UNIQ [31] and Noise Temper-

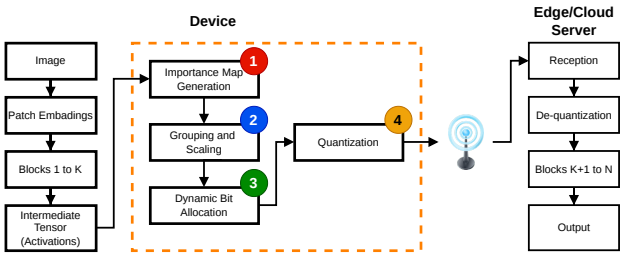


Fig. 1: Split Computing with the LIQOS approach for ViT

ing [32] similarly use controlled noise to emulate quantization errors and improve robustness to low-precision settings.

D. Motivation

In conclusion, existing compression methods are not well-suited to ViT split inference: CNN-based approaches do not transfer to token embeddings, and most mixed-precision schemes operate at the weight, layer, channel, or patch level rather than on individual activation values. ViT activation quantization methods, proposed outside split computing, cover the whole inference, whereas we quantize only at the split point and can thus reach much higher compression (below 1 bit per activation on average). Moreover, intermediate activations are input-dependent, so their sensitivity must be judged by their effect on final accuracy. This motivates our adaptive split computing framework, which assigns more bits to sensitive activations, fewer to less important ones, while explicitly handling outliers to enable higher compression, reduce communication overhead, and maintain accuracy as close as possible to the original.

III. Our Approach

Our splitting strategy places the embedding layer and a subset of transformer blocks on the edge device, while executing the remaining transformer blocks and the MLP head on the server. The objective is to compress the intermediate activations produced by the last transformer block running on the device before transmitting them to the server. For this purpose, LIQOS-ViT applies mixed-precision quantization by assigning each activation coefficient a number of bits adapted to its importance to the model. As shown in Figure 1, in the first step, we pre-compute an offline importance map that measures how strongly each activation location influences the model output. Next, to reduce the effect of outliers, we preprocess the activations by separating extreme values from the main distribution and normalizing each group independently. To keep the distortion model consistent after normalization, the importance map is rescaled accordingly. After this preprocessing, we perform importance-guided bit allocation by solving an optimization problem that minimizes the importance-weighted quantization error under a global bit budget constraint. Finally, we quantize the normalized activations using the resulting per-coefficient bit allocation, then reconstruct them on the server side via dequantization followed by denormalization.

In the rest of this section, we focus on an a high-level description of LIQOS principles and algorithms, see Appendix A for protocol and implementation considerations.

A. Learning Importance Maps

A central component of our approach is the design of an importance map that guides mixed-precision quantization. Formally, let $\mathbf{A} = H(\mathbf{X})$ denote the intermediate activation tensor produced by the device-side *head* H for a given input $\mathbf{X}$, with coefficients A_{ij} indexed by $(i,j) \in \Omega$, where $\Omega = \{(i,j) \mid i = 1,\dots,L,\ j = 1,\dots,D\}$ is the set of all activation indices (L rows of D elements each). The activation is quantized prior to transmission via the algorithm described later, yielding the quantized representation $\widehat{\mathbf{A}}$. The core idea is to allocate more bits to the most informative coefficients of $\mathbf{A}$ at the split point, thereby reducing quantization distortion where it matters most. Formally, for each coefficient $(i,j) \in \Omega$, we seek a bit allocation n_{ij} that minimizes the impact of quantization on the output of the *tail* model. Within LIQOS, we propose two methods for computing the importance map: a gradient-based approach (Section III-A2) and an optimization-based approach (Section III-A5).

1) General Distortion Model and Importance Score: We begin with a distortion model that characterizes the effect of quantization-induced perturbations on the tail output.

If we denote the quantization noise by $\boldsymbol{\eta} = \widehat{\mathbf{A}} - \mathbf{A}$, then, instead of observing the output $T(\mathbf{A})$ of the tail model, we obtain $T(\mathbf{A}+\boldsymbol{\eta})$. We assume a scalar distortion metric $d(\cdot,\cdot)$ applied to the output of the tail model. The distortion induced by quantization is therefore $d(T(\mathbf{A}+\boldsymbol{\eta}), T(\mathbf{A}))$.

We assume that the distortion induced by perturbing a coefficient A_{ij} depends linearly on the magnitude of the perturbation η_{ij}. Since we are interested in the distortion caused by each perturbation in isolation, we further assume that the total distortion is separable and can be expressed as the sum of the individual contributions of the coefficients. This leads to the following separable error model:

$$d(T(\mathbf{A}+\boldsymbol{\eta}), T(\mathbf{A})) = \sum_{(i,j)\in\Omega} S_{ij}|\eta_{ij}| + \nu(\mathbf{A},\boldsymbol{\eta}), \quad (1)$$

The term $\nu(\mathbf{A},\boldsymbol{\eta})$ captures the mismatch between the true distortion and this separable approximation, and is treated as noise and ignored. Once identified, $\mathbf{S}$ serves as the importance map. Each value S_{ij} is the importance score determining the number of bits allocated to coefficient (i,j).

2) Gradient-Based Importance Maps: It is possible to use gradient-based sensitivity, in a similar spirit to gradient saliency methods such as [33] and Grad-CAM [27]. There, the gradient with respect to the input image is computed and used to identify which parts of the image have the most impact on the output. In the context of model optimization, this has been used, for instance, for model pruning in [34], for identifying

less important activations and pruning (i.e., setting to 0) the weights of the model that lead to these activations. [1]

More precisely, considering a first-order Taylor approximation for a scalar tail model T (e.g., predicted-class logit), and treating the tensor $\mathbf{A}$ as a vector by stacking all its entries, we have:

$$T(\mathbf{A}+\boldsymbol{\eta}) - T(\mathbf{A}) = \nabla T(\mathbf{A})^\top \boldsymbol{\eta} + O(\|\boldsymbol{\eta}\|^2).$$

Thus, when using an ℓ_1 distortion, an approximate upper bound is

$$|T(\mathbf{A}+\boldsymbol{\eta}) - T(\mathbf{A})| \lesssim \sum_{(i,j)\in\Omega} |\nabla_{ij} T(\mathbf{A})|\,|\eta_{ij}|. \tag{2}$$

For the separable model (1), a natural choice is $S_{ij} = |\nabla_{ij} T(\mathbf{A})|$, as suggested by the upper bound (2).

3) Learning a Gradient-Based Importance Map: An important practical issue is that the gradient of the activation is only available after tail model computation, which is precisely what we wish to avoid. Therefore, we adopt a statistical approach, and *learn* the importance score as the expected gradient magnitude over a training dataset $\mathcal{T}$: $S_{ij} \triangleq \mathbb{E}_{\mathbf{A}\sim\mathcal{T}} |\nabla_{ij} T(\mathbf{A})|$.

4) Optimization-Based Importance Map: We propose an alternative approach, since (2) is loose. Here, our goal is to determine an importance score S_{ij} for each activation coefficient in $\mathbf{A}$ while making as few assumptions as possible. The approach is to estimate the sensitivity of the tail model by adding artificial noise to the activations and observing the resulting distortion at its output. By minimizing the expected distortion, the importance scores S_{ij} can be recovered.

We keep the separable linear model (1) in expectation over $\mathbf{A}$; however, the coefficients $\mathbf{S}$ are no longer assumed to be given by the gradient. Specifically, they satisfy

$$\mathbb{E}_{\mathbf{A}}[d(T(\mathbf{A}+\boldsymbol{\eta}), T(\mathbf{A}))] \approx \sum_{(i,j)\in\Omega} S_{ij}|\eta_{ij}|. \tag{3}$$

Formally, let

$$J(\mathbf{w}) \triangleq \mathbb{E}_{\mathbf{A},\,\mathbf{z}}\Big(d(T(\mathbf{A}+\mathbf{z}\odot f(\mathbf{w})), T(\mathbf{A}))\Big), \tag{4}$$

and consider the following general stochastic optimization problem:

$$\min_{\mathbf{w}\in\mathcal{W}} J(\mathbf{w}) \tag{5}$$

where the feasible set is:

$$\mathcal{W} = \{\mathbf{w} \mid \textstyle\sum_{(i,j)\in\Omega} \phi(w_{ij}) = K, \;\; \text{and } w_{ij} \geq w_{\min}\} \tag{6}$$

with $w_{\min}$ a given minimum value, $\phi : \mathbb{R} \to \mathbb{R}$ a given strictly increasing function, and K a chosen constant; $\mathbf{z}$ is a vector of i.i.d. random variables modeling pseudo-quantization noise for the activation coefficients, of average amplitude $\mu \triangleq \mathbb{E}[|z_{ij}|]$; and $f : \mathbb{R} \to \mathbb{R}^+$ is an arbitrary function applied component-wise. The choice of $w_{\min}, \phi, K, f$ and the distribution of $\mathbf{z}$ are left flexible to obtain desirable properties of the solution and recover the importance scores S_{ij}. In particular, $w_{\min}$ is chosen small enough and K large enough so that the optimum lies in the interior of $\mathcal{W}$, i.e. $w^*_{ij} > w_{\min}$ for all $(i,j) \in \Omega$.

Intuitively, this optimization estimates the sensitivity of the tail model to perturbations of each activation coefficient. We add random noise, scaled by $f(w_{ij})$, to each coefficient A_{ij}, and minimize the overall distortion at the output of the tail model. To reach this minimum, the optimization process naturally forces the noise scale $f(w_{ij})$ to be smaller for highly sensitive coefficients, effectively shielding them from heavy noise. At the optimum, the noise scales are automatically adjusted so that each coefficient contributes a roughly equal share of the total distortion. This is, of course, an approximation and depends on the exact constraints enforced by the set $\mathcal{W}$.

The following Theorem 1 shows that the parameters can be adjusted to recover the importance scores S_{ij} of (3) up to a constant factor, which is sufficient for our purposes.

Theorem 1. *For $d(x,y) = |x-y|$, $\phi(x) = \log_2(x)$, and $f(x) = 1/x$, the optimization problem* (5) *gives optimal parameters $\mathbf{w}^*$ satisfying $w^*_{ij} \propto S_{ij}$, i.e. $\mathbf{w}^*$ recovers the importance map $\mathbf{S}$ of* (3) *up to a constant factor, for the constraint set* (6)*.*

Proof. See Appendix B-A. □

5) Learning an Importance Map with Stochastic Gradient Descent: To solve the stochastic optimization problem (5), one can use stochastic gradient descent in a way that parallels the training of the model itself. This approach is often used for quantization-aware training (QAT). An example is DiffQ [29], which adds pseudo-quantization noise during training. The difference is that instead of fine-tuning the weights of the model, we treat them as frozen and learn the parameters $\mathbf{w}$.

More precisely, processing is performed over batches of samples. For each input sample $\mathbf{X}$, the activation $\mathbf{A} = H(\mathbf{X})$ is computed. Random noise $\mathbf{z}$ is then generated, and both $T(\mathbf{A})$ and $T(\mathbf{A} + \mathbf{z} \odot f(\mathbf{w}))$ are evaluated. The distortion between these two outputs defines the loss function. Stochastic gradient estimates with respect to $\mathbf{w}$ are then computed by backpropagation. This relies on the differentiability of all operations involved. Since $\mathbf{z}$ is sampled independently of $\mathbf{w}$, the noise does not prevent differentiation: this is the standard reparameterization trick [35], [36]. Indeed, the gradient with respect to $\mathbf{w}$ propagates through $\mathbf{z} \odot f(\mathbf{w})$ as $\mathbf{z} \odot f'(\mathbf{w})$. This approach has the main advantage that it makes only minimal assumptions on the tail model T, since T is evaluated under a similar perturbation model to the one induced by quantization.

One remaining issue is that, in the absence of constraints, $f(w_{ij}) \to 0$ for all $(i,j) \in \Omega$ would trivially minimize (5) since the noise would vanish; the feasible set $\mathcal{W}$ (6) prevents this. At each step, $\mathbf{w}$ is updated by stochastic gradient descent and then projected back onto $\mathcal{W}$; see Appendix B-B.

B. Group-wise Separation and Scaling

After determining the importance map, the next step is to preprocess the activations to reduce the effects of extreme values. Indeed, outliers can strongly influence the dynamic range,

[1] It is seldom used for weight quantization because the gradient of a well-trained model with respect to the weight is expected to be close to 0, and the Hessian can be used as in [25]. This is not the case here.

thereby degrading both compression efficiency and quantization accuracy. Many existing methods in the literature [8] handle outliers at the token or channel level by excluding them from the quantization range. In practice, these values are clamped to the quantization limits. This approach reduces the dynamic range and improves quantization precision for most values. However, it is not always the best way to treat outliers, since part of their information may be lost. Instead of simply clamping them, it can be more effective to keep them and quantize them separately from the remaining values.

In our approach, the activations in each row are divided into two groups using percentile-based thresholds. For each row i, let $\Omega_i = \{(i,j) \mid j = 1, \ldots, D\}$ denote the set of activation locations in that row, and let p denote a specified trimming ratio. The lower and upper thresholds are given by $v_\ell^{(i)} = Q_p\big(\{A_{ij}\}_{j=1}^{D}\big)$ and $v_u^{(i)} = Q_{1-p}\big(\{A_{ij}\}_{j=1}^{D}\big)$, respectively. $Q_p(\{A_{ij}\}_{j=1}^{D})$ is the p-th quantile of the activation values in row i. The two groups are then denoted by

$$g_{\text{out}}^{(i)} = \{(i,j) \in \Omega_i \mid A_{ij} < v_\ell^{(i)} \text{ or } A_{ij} > v_u^{(i)}\},$$
$$g_{\text{in}}^{(i)} = \{(i,j) \in \Omega_i \mid v_\ell^{(i)} \le A_{ij} \le v_u^{(i)}\}.$$

$g_{\text{out}}^{(i)}$ contains the outlier values from the lower and upper tails in row i, and $g_{\text{in}}^{(i)}$ contains the remaining values. Our objective is to separate activations with different dynamic ranges so that each group can be processed independently. Finally, for each row i and each group $h \in \{\text{in}, \text{out}\}$, we define the corresponding minimum, maximum, and dynamic range as

$$m_h^{(i)} = \min_{(i,j)\in g_h^{(i)}} A_{ij}, \quad M_h^{(i)} = \max_{(i,j)\in g_h^{(i)}} A_{ij},$$
$$\Delta_h^{(i)} = M_h^{(i)} - m_h^{(i)}.$$

According to the previous definitions, we can then normalize the activation coefficient as follows:

$$A_{ij}^{\text{norm}} = \begin{cases} \frac{A_{ij} - m_{g(i,j)}^{(i)}}{\Delta_{g(i,j)}^{(i)}}, & \Delta_{g(i,j)}^{(i)} > 0, \\ 0, & \Delta_{g(i,j)}^{(i)} = 0. \end{cases} \tag{7}$$

where $g(i,j) \in \{\text{in}, \text{out}\}$ denotes the group to which coefficient A_{ij} belongs. Note that, when $\Delta_{g(i,j)}^{(i)} = 0$, all normalized values in that group are set to zero to avoid division by zero. Because normalization is performed independently within each group, a quantization error in the normalized domain automatically translates into an error in the original domain. In order to capture this effect, we need to rescale the importance score S_{ij} (as defined in Section III-A) by the dynamic range of the corresponding group as follows:

$$\widetilde{S}_{ij} \triangleq S_{ij}\, \Delta_{g(i,j)}^{(i)}. \tag{8}$$

This group-wise preprocessing adds an extra overhead that must be taken into consideration in the total bit budget. The grouping is done for each row, containing D coefficients, of which $K_i = |g_{\text{out}}^{(i)}|$ are designated as outliers. In this case, the group-assignment map is determined by the selection of these K_i outlier positions from the available D positions. As a result, the number of possible masks for row i is given by $\binom{D}{K_i}$. Therefore, the ideal compressed cost needed to encode the row-wise assignment map is $\left\lceil \log_2 \binom{D}{K_i} \right\rceil$.

Finally, the decoder also needs to receive the normalization bounds of both groups in that row. If each transmitted scalar bound is represented with b_{meta} bits, the total overhead is

$$\text{overhead} = \sum_{i=1}^{L} \left(\left\lceil \log_2 \binom{D}{K_i} \right\rceil + 4\, b_{\text{meta}} \right).$$

Accordingly, the effective payload budget available for activation quantization is $C_b = C - \text{overhead}$.

C. Bit Allocation

To achieve an efficient compression ratio while preserving model performance, we propose a bit allocation algorithm. Basically, for each activation coefficient A_{ij}, we assign a quantization bit-width n_{ij} according to its importance. The main idea is that more important coefficients should be represented with higher precision.

1) Quantization Error Estimate: As defined in Section III-A, in this work, we approximate the effect of activation quantization on the model output by a weighted sum of coefficient-wise reconstruction errors:

$$d(T(\mathbf{A}), T(\widehat{\mathbf{A}})) \approx \sum_{(i,j)\in\Omega} S_{ij}\, |A_{ij} - \widehat{A}_{ij}|,$$

where, $T(\cdot)$ denotes the tail of the model and $\widehat{\mathbf{A}}$ is the reconstructed activation tensor after quantization and dequantization. Using the group-wise normalization defined in equation 7 and the scaling defined in equation 8, the distortion can be expressed as follows: $d(T(\mathbf{A}), T(\widehat{\mathbf{A}})) \approx \sum_{(i,j)\in\Omega} \widetilde{S}_{ij} \left| A_{ij}^{\text{norm}} - \widehat{A}_{ij}^{\text{norm}} \right|$.
Taking expectation, we obtain

$$\mathbb{E}\Big[d(T(\mathbf{A}), T(\widehat{\mathbf{A}})) \Big] \approx \sum_{(i,j)\in\Omega} \widetilde{S}_{ij}\, e_{ij}(n_{ij}),$$

where

$$e_{ij}(n_{ij}) \triangleq \mathbb{E}\Big[\left| A_{ij}^{\text{norm}} - \widehat{A}_{ij}^{\text{norm}} \right| \Big]$$

denotes the expected quantization error in the normalized domain when coefficient (i,j) is represented with n_{ij} bits. If we use a uniform scalar quantizer on the normalized interval $[0,1]$, the quantization step size for $n_{ij} \ge 1$, is

$$\delta(n_{ij}) = \frac{1}{2^{n_{ij}} - 1}. \tag{9}$$

According to the standard assumption that the quantization error is uniformly distributed over $\left[-\frac{\delta(n_{ij})}{2}, \frac{\delta(n_{ij})}{2} \right]$, the expected absolute error becomes:

$$e_{ij}(n_{ij}) = \frac{\delta(n_{ij})}{4} = \frac{1}{4(2^{n_{ij}} - 1)}, \qquad n_{ij} \ge 1.$$

When $n_{ij} = 0$, no information is transmitted for coefficient (i,j). If the decoder assigns a fixed reconstruction value in $[0,1]$, then the corresponding normalized error is bounded by

$e_{ij}(0) \leq \frac{1}{2}$. Finally, the contribution of coefficient (i,j) to the total distortion is modeled as:

$$E_{ij}(n_{ij}) = \begin{cases} \frac{1}{2}\widetilde{S}_{ij}, & n_{ij} = 0, \\ \frac{\widetilde{S}_{ij}}{4(2^{n_{ij}}-1)}, & n_{ij} \geq 1. \end{cases}$$

2) Optimization Problem Formulation: The final bit-allocation problem is formulated as minimizing the sum of these coefficient-wise propagated errors across all L tokens and D channels (i.e., all $(i,j) \in \Omega$).

$$\begin{aligned} \min_{\{n_{ij}\}} \quad & E_{\text{total}}(\mathbf{n}) = \sum_{(i,j)\in\Omega} E_{ij}(n_{ij}) \\ \text{s.t.} \quad & \sum_{(i,j)\in\Omega} n_{ij} \leq C_b = C - \text{overhead}, \\ & n_{ij} \in \{0, 1, \ldots, n_{\max}\}, \forall (i,j) \in \Omega. \end{aligned} \tag{10}$$

where, C_b denotes the available budget, C total budget, and overhead the side information required for reconstruction.

3) Problem Solving using a Lagrangian-Guided Greedy Search: To solve the optimization problem, we adopt a two-step allocation procedure. First, we derive a continuous estimate of the optimal bit assignment through a Lagrangian relaxation. Then, we map this relaxed solution onto the discrete feasible set and apply a greedy correction to exactly satisfy the bit budget. As the objective function is separable across coefficients, we can introduce a Lagrangian multiplier, $\lambda \geq 0$, related to the budget constraint as follows:

$$\mathcal{L}(\mathbf{n}, \lambda) = \sum_{i,j} E_{ij}(n_{ij}) + \lambda \left(\sum_{i,j} n_{ij} - C_b \right). \tag{11}$$

For a fixed λ, each n_{ij} can be optimized independently by considering the scalar objective

$$E_{ij}(n) + \lambda n.$$

In this case, we can deduce the optimal nonzero continuous number of bits, denoted by $\bar{n}_{ij}(\lambda)$, according to the following minimizer:

$$\bar{n}_{ij}(\lambda) = \arg\min_{n>0} \left\{ E_{ij}(n) + \lambda n \right\}. \tag{12}$$

The stationary condition of this relaxed problem is equal to:

$$\lambda = \frac{\widetilde{S}_{ij} \ln 2}{4} \frac{2^n}{(2^n - 1)^2},$$

Solving this stationarity condition yields the closed-form expression:

$$\bar{n}_{ij}(\lambda) = \log_2 \left(\frac{8\lambda + \widetilde{S}_{ij} \ln 2 + \sqrt{(\widetilde{S}_{ij} \ln 2)^2 + 16\lambda \widetilde{S}_{ij} \ln 2}}{8\lambda} \right).$$

Once the relaxed bit allocation has been obtained, the resulting values must be mapped onto a discrete set. To this end, we define the set of admissible bit-widths $\mathcal{B} = \{0, n_{\min}, n_{\min}+1, \ldots, n_{\max}\}$. We then propose a mapping function that takes the largest admissible value in $\mathcal{B}$ that does not exceed it, and is defined as follows:

$$\widetilde{n}_{ij} = \lfloor \bar{n}_{ij}(\lambda) \rfloor_{\mathcal{B}}, \tag{13}$$

Note that in LIQOS-ViT we adopt a strategy that discourages isolated 1-bit assignments during activation quantization. Specifically, we set $n_{\min}$ to 2 $bits$. Indeed, 1-bit quantization provides only two reconstruction levels, which is generally too coarse to represent the generally concentrated distribution of activation values and may lead to large reconstruction errors. This modification improves the overall rate-distortion trade-off, particularly in the low-budget regime where a naive allocation strategy tends to produce too many 1-bit assignments.

The resulting tensor $\widetilde{\mathbf{n}}$ provides an initial feasible discrete allocation; however, it may not match the exact amount of bits in the budget. Therefore, we need to correct this budget mismatch by applying a water-filling refinement. Basically, if the current allocation underuses the total available budget (i.e., $\sum_{i,j} \widetilde{n}_{ij} < C_b$), in this case we increase the bit-width of the coefficient with the largest marginal distortion reduction. However, in case the current allocation is exceeding the available budget (i.e., $\sum_{i,j} \widetilde{n}_{ij} > C_b$), we reduce in this case the bit-width of the coefficient with the smallest marginal distortion increase. This process is repeated until the exact bit budget is reached (i.e., $\sum_{i,j} \widetilde{n}_{ij} = C_b$).

D. Quantization & Dequantization

The final step quantizes the normalized activations according to the bit allocation determined by the proposed bit-allocation algorithm. For entries with $n_{ij} > 0$, we use a scalar uniform quantizer on the normalized interval $[0,1]$. The corresponding quantization step size is defined as in equation 9. The associated integer code is given by

$$A_{q,ij} = \text{Clamp}\left(\text{Round}\left(\frac{A_{ij}^{\text{norm}}}{\delta_{ij}} \right), 0, 2^{n_{ij}} - 1 \right),$$

and the reconstructed normalized coefficient is $\widehat{A}_{ij}^{\text{norm}} = A_{q,ij}\, \delta_{ij}$.

For entries with $n_{ij} = 0$, no quantized payload is transmitted. In this case, the decoder assigns a fixed reconstruction value $r_0 \in [0,1]$ in the normalized domain. In our implementation, we use $\widehat{A}_{ij}^{\text{norm}} = \frac{1}{2}$, which is consistent with the zero-bit error model used in the bit-allocation formulation.

After dequantization, the server applies the inverse row-wise group normalization to recover the activation in the original domain:

$$\widehat{A}_{ij} = m_{g(i,j)}^{(i)} + \widehat{A}_{ij}^{\text{norm}} \left(M_{g(i,j)}^{(i)} - m_{g(i,j)}^{(i)} \right),$$

where $g(i,j) \in \{\text{in}, \text{out}\}$ denotes the group assigned to coefficient (i,j) in row i, and $m_{g(i,j)}^{(i)}$ and $M_{g(i,j)}^{(i)}$ are the corresponding normalization bounds.

On the server side, reconstruction therefore consists of two successive steps: dequantization in the normalized domain, followed by inverse normalization in the original activation domain. Finally, the importance map does not need to be

transmitted, since it is learned offline and shared beforehand between the edge device and the server.

IV. Results

We implement LIQOS using the PyTorch framework. For both training the importance map and evaluating the proposed approach, we use the ImageNet-1K dataset for image classification. Several pretrained ViT variants, including ViT-B/32 ViT-B/16 and Dino-v2, are employed, with models obtained from the torchvision library. For calibration, we select five images from each of the 1,000 ImageNet-1K classes to train the importance map. Similarly, during evaluation, five images per class are sampled from the ImageNet-1K validation set and used as test set. The split point is set at Block 6, This parameter is a configurable and for each split point, we have an importance map. The patch embedding layer and the first six Transformer blocks run on the device, while the remaining blocks and the classification head run on the server. Although all experiments are reported for this split configuration, the split point is a configurable parameter, and the proposed method can be applied to other partitioning choices as well. In our experiments, we evaluate the compression factor, which is calculated as the compressed size divided by the original size. Our method involves two main design choices. The selection of the importance map for bit allocation and the percentile threshold for outlier-aware grouping. In this case, we first validate the reliability of the candidate importance maps, then study the effect of the percentile parameter, and finally compare the complete method against several baselines.

A. Learning and Validation of the Importance Maps

In our experiments, we computed two families of importance maps that guide the quantization process, as described in Section III-A: gradient-based importance maps (Section III-A3), denoted *Grad maps*, and importance maps obtained through SGD training (Section III-A5), denoted *SGD maps*. For the SGD maps, three hyperparameters must be specified. We conducted a grid search over a range of parameter values and selected the best-performing ones: for the ViT-B/32 model, we use $w_{\min} = 1$, $K = 4.5$, and $\mu = E(|\text{noise}|) = 1.73$.

Figure 2 provides an experimental validation of the idea behind importance maps: we artificially add uniform noise, with a given amplitude μ, to the intermediate activations and evaluate its final impact on model accuracy. The noise is scaled according to the importance map, as implied by (5) and described in Section III-A5. The Grad map already yields better performance, even though this was not its explicit design objective, while the SGD map performs best in this evaluation, for which it was specifically designed. Further figures and details on the learned maps can be found in Appendix C-A.

B. Selection of the Outlier Grouping Threshold

The second key hyperparameter is the percentile threshold α used to split activations into two groups for outlier-aware pre-

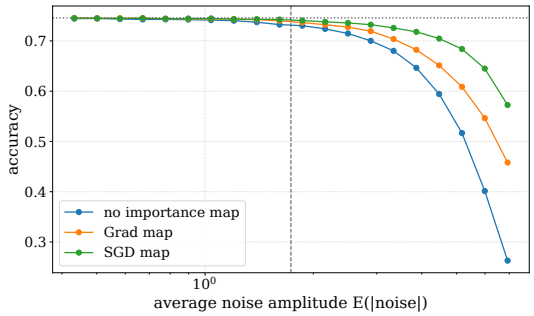
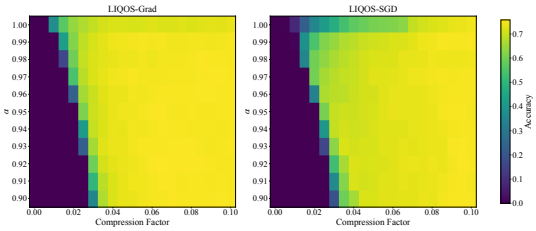
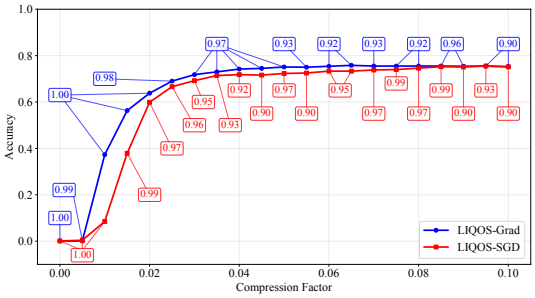


Fig. 2: Resilience of the model to pseudo-quantization noise using different importance maps.



(a) Accuracy heatmaps over compression ratio and α on ViT-B/32.



(b) Accuracy/compression Pareto Fronts on ViT-B/32

Fig. 3: Outlier Grouping Threshold.

processing. In LIQOS-ViT, we perform token-wise grouping rather than channel-wise grouping or across the full tensor.

We evaluated several values of α across different compression budgets on the training set, and we constructed the corresponding Pareto front. Each point is characterized by two objectives: the compression ratio and the resulting accuracy. A point belongs to the Pareto front if no other configuration simultaneously achieves a better compression ratio and better accuracy. This analysis enables identification, for each target compression regime, of the percentile value that offers the best trade-off.

Figure 3a shows that the effect of α is most visible in the transition region between low and moderate compression factor. Indeed, when the compression factor is too small, both LIQOS-Grad and LIQOS-SGD experience a significant drop in accuracy, indicating that the available bit budget is insufficient. Alternatively, when we increase the compression factor, the accuracy recovers rapidly, but the position of this recovery depends on the value of α. Finally, we also observe

that this dependence is more pronounced for LIQOS-SGD, whereas LIQOS-Grad exhibits a faster transition toward the high-accuracy regime.

In figure 3b, we plot the best accuracy per compression factor labeled by the associated value of α. In other words, each numeric annotation shows which percentile threshold produces the corresponding accuracy/compression operating point for the method, with blue labels for LIQOS-Grad and red labels for LIQOS-SGD. We may also observe that for low compression factors, constrained points are dominated by high α values, whereas for larger compression factors, the trend shifts, with a more diverse set of α values contributing to the accuracy/compression operating points. In this range, both LIQOS-Grad and LIQOS-SGD demonstrate improved performance.

C. Comparison with Baselines

In Figure 5, we compare the performance of the proposed LIQOS variants in ViT-B/32 with state-of-the-art quantization methods in terms of accuracy versus compression factor. The first baseline is uniform quantization, which applies no outlier handling or adaptive bit-allocation strategy; all values are quantized with the same number of bits. The second baseline is ORQ [8], a strong, recent reference method closely related to our work. In general, our approach achieves competitive results using either a mean gradient-based or a learned importance map. LIQOS preserves nearly the full accuracy of the original model even at a very small compression factor of 0.04, which corresponds to 96% compression. In contrast, at the same compression factor, both ORQ and uniform quantization show significant accuracy degradation, with accuracy values close to zero. For a compression factor close to 0.02, corresponding to about 98% compression the accuracy drop remains limited to around 10%. This result is notable because ImageNet is a large-scale, diverse dataset, making high accuracy under such aggressive compression particularly challenging. As well, all these results are also achieved with ViT-B/16 and Dino-v2, (See Appendix C-B). LIQOS, particularly the gradient-based variant, maintains the original model performance starting at a compression factor of 0.06. In contrast, uniform quantization still achieves weak accuracy at this compression level. Although ORQ achieves more competitive performance than uniform quantization, its accuracy remains lower than that of our approach. Thus, adaptive importance-aware quantization is considerably more effective than uniform allocation, especially under tight communication-budget constraints. We further evaluate the method under a lower communication budget to assess its robustness in more aggressive compression settings. When comparing the learned importance map with gradient-based allocation, the results show different behaviors depending on the compression budget. When the overhead is not considered, LIQOS can use the entire bit budget for activation quantization. In this higher-budget regime, starting from a compression factor of 0.04, the gradient-based strategy outperforms the learned importance map, as shown in 5. In contrast, at lower budgets, starting from a compression factor of 0.004, the learned importance map outperforms gradient-based allocation as shown in figure 4. This variant achieves an accuracy of around 60%, which is highly competitive under such an aggressive compression setting. This behavior can be explained by the fact that the learned map is optimized for low-budget scenarios, making it more effective at allocating bits when the compressed size ratio is very small.

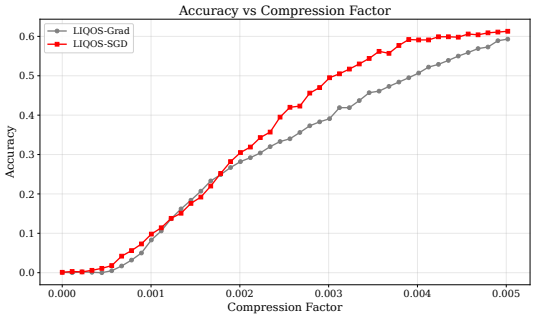


Fig. 4: LIQOS at Small Compression Budgets: Gradient-Based vs Learned Importance Maps

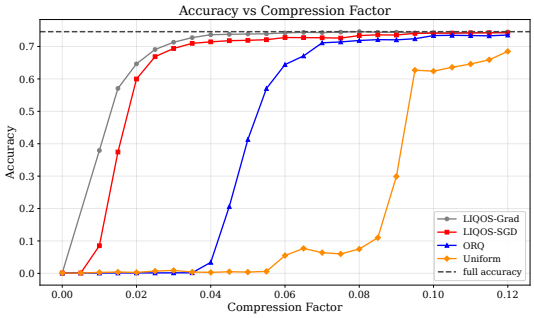


Fig. 5: Accuracy Across Compression Factors for LIQOS, ORQ, and Uniform Quantization for ViT-B/32

V. Conclusion

In conclusion, this work introduces a budget-aware mixed-precision quantization framework for compressing intermediate activations in split ViTs. By leveraging feature importance, estimated either from gradients or from a learned importance map, for adaptive bit allocation and incorporating preprocessing steps to reduce rate distortion, the proposed method achieves efficient compression with minimal impact on model accuracy. These characteristics make it a promising approach for reducing transmission overhead in edge-server inference, thereby improving the feasibility of deploying ViT models in resource-constrained environments. As future work, we plan to develop a grouping-based preprocessing method that further reduces overhead by eliminating the need to transmit group masks and range information.

VI. Acknowledgments

This work was funded by the French government under the France 2030 ANR program "PEPR Networks of the Future" (ref. ANR-22-PEFT-0007)

References

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[3] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Computing Surveys*, 2022.

[4] G. Xu *et al.*, "Devit: Decomposing vision transformers for collaborative inference in edge devices," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 5918–5933, 2024.

[5] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–30, 2022.

[6] S. Zhang, N. Yi, and Y. Ma, "A survey of computation offloading with task types," *IEEE transactions on intelligent transportation systems*, 2024.

[7] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers, 1992.

[8] X. He, Y. Lu, H. Liu, C. Gong, and W. He, "Orq-vit: Outlier resilient post training quantization for vision transformers via outlier decomposition," *Journal of Systems Architecture*, p. 103 530, 2025.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[10] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, 2022.

[11] H. Liu, M. E. Fouda, A. M. Eltawil, and S. A. Fahmy, "Split DNN Inference for Exploiting Near-Edge Accelerators," in *IEEE International Conference on Edge Computing and Communications (EDGE)*, IEEE, 2024, pp. 84–91.

[12] A. Tassi, O. Y. Kolawole, J. P. Roig, and D. Warren, "Optimized Split Computing Framework for Edge and Core Devices," *IEEE Transactions on Vehicular Technology*, vol. 75, no. 3, pp. 5233–5238, 2026.

[13] Y. Matsubara, D. Callegaro, S. Singh, M. Levorato, and F. Restuccia, "Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2022, pp. 337–346.

[14] A. Dhaouadi, N. Achir, and C. Adjih, "Enhancing split vit inference through sparsity-driven compression," in *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, IEEE, 2025, pp. 1–7.

[15] T. T. Nguyen, Y. H. Pua, T. V. Ngo, *et al.*, "Adaptive swin transformer partitioning over AI-RAN networks," in *2026 IEEE Vehicular Technology Conference (VTC2026-Spring) Workshops*, Nice, France, Jun. 2026.

[16] D. Du, G. Gong, and X. Chu, "Model quantization and hardware acceleration for vision transformers: A comprehensive survey," *arXiv preprint arXiv:2405.00314*, 2024.

[17] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*, Chapman and Hall/CRC, 2022, pp. 291–326.

[18] Y. Jiang, N. Sun, X. Xie, F. Yang, and T. Li, "ADFQ-ViT: Activation-distribution-friendly post-training quantization for vision transformers," *Neural Networks*, vol. 186, p. 107 289, 2025.

[19] Y.-S. Tai, M.-G. Lin, and A.-Y. A. Wu, "TSPTQ-ViT: Two-scaled post-training quantization for vision transformer," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[20] L. Yang, H. Gong, and Q. Gu, "Dopq-vit: Towards distribution-friendly and outlier-aware post-training quantization for vision transformers," *arXiv preprint arXiv:2408.03291*, 2024.

[21] J. Xiao, Z. Li, L. Yang, and Q. Gu, "Patch-wise mixed-precision quantization of vision transformer," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023, pp. 1–7.

[22] E. A. Riskin, "Optimal bit allocation via the generalized bfos algorithm," *IEEE Transactions on Information Theory*, vol. 37, no. 2, pp. 400–402, 1991.

[23] W. Gao, Y.-H. Liu, C. Wang, and S. Oh, "Rate distortion for model compression: From theory to practice," in *International Conference on Machine Learning*, PMLR, 2019, pp. 2102–2111.

[24] Z. Wang, J. Lin, X. Geng, M. M. Sabry Aly, and V. Chandrasekhar, "Rdo-q: Extremely fine-grained channel-wise quantization via rate-distortion optimization," in *Computer Vision – ECCV 2022*, Springer, 2022, pp. 157–172.

[25] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq-v2: Hessian aware trace-weighted quantization of neural networks," in *Advances in Neural Information Processing Systems*, 2020.

[26] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization," in *Computer Vision – ECCV 2022*, 2022.

[27] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in

Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626.

[28] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, e0130140, 2015.

[29] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," *Transactions on Machine Learning Research*, 2022.

[30] J. Shin, J. So, S. Park, S. Kang, S. Yoo, and E. Park, "Nipq: Noise proxy-based integrated pseudo-quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[31] C. Baskin, N. Liss, E. Schwartz, *et al.*, "Uniq: Uniform noise injection for non-uniform quantization of neural networks," *ACM Transactions on Computer Systems*, 2021.

[32] Z. Wang, J. B. Li, S. Qu, F. Metze, and E. Strubell, "Error-aware quantization through noise tempering," *arXiv preprint arXiv:2212.05603*, 2022.

[33] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR Workshop*, 2014.

[34] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *International Conference on Learning Representations*, 2017.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[36] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1278–1286.

[37] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the ℓ_1-ball for learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, ACM, 2008, pp. 272–279.

[38] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *Mathematical Programming*, vol. 158, no. 1, pp. 575–585, 2016.

[39] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, "DINOv2: Learning robust visual features without supervision," *Transactions on Machine Learning Research*, 2024.

Supplementary Material

Appendix A
Protocol and Implementation Considerations

In Section III, we described the high-level operation and core algorithms of LIQOS, Here, we provide additional discussion about the protocol and implementation of LIQOS.

LIQOS consists of two components that operate on different timescales. The first component (learning the importance map and computing the Pareto front) is performed offline, once, on a training set, and is therefore not time-critical. The resulting static parameters (importance maps, Pareto front, and a few additional parameters) are pre-shared between the edge device and the server before deployment. The second component, outlier grouping, bit allocation, and quantization, is executed at inference time for each input and is therefore time-critical.

The two components correspond to two distinct sources of variation: one static, namely the higher importance of certain coefficients (such as the class token or rows corresponding to the center of the image), and one dynamic, namely the outlier status of individual coefficients. We observed that the outlier status is highly input-dependent and did not seem to be reliably predictable. The dynamic aspect therefore appears to be an unavoidable part of the inference pipeline.

LIQOS is designed around this fact, making maximum use of the static component while efficiently accommodating the dynamic one. Crucially, the bit allocation $\mathbf{n}$ is not transmitted: doing so would introduce a prohibitive overhead, especially under the small bit budgets we target (e.g., fewer than 1 bit per activation). Instead, the server recomputes $\mathbf{n}$ locally from the pre-shared static parameters.

From a protocol perspective, the edge device transmits the following to the server:

- The quantized activations, which form the main payload.
- The side information associated with the dynamic component:
  - The group masks are binary masks that are efficiently compressible.
  - The range information, i.e., the minimum and maximum values per group, transmitted as floating-point scalars; in our experiments, we use 32-bit floats
- Additional short protocol metadata, such as the model identity, the split point, the bit budget, α, etc.

For the offline component, a significant part of the effort is spent on precomputing the importance map:

- For Grad maps, this is relatively fast, requiring approximately one epoch of forward–backward passes on the calibration set. We observed that even small subsets of ImageNet, with as few as 10 classes, yield results comparable to those obtained with the full training set ultimately used.
- For SGD maps, this step is more time-consuming, typically requiring 100 to 1000 epochs on the training set. The grid search over hyperparameters (learning rate, $w_{\min}$, K, etc.) adds further cost, although we did not observe substantial sensitivity to their exact values. Determining the scale values, which are used to construct the importance map, introduces only a moderate additional computational cost compared with a standard ViT forward pass. During training, the ViT backbone is frozen by default, so the optimization updates only the learnable scale parameters rather than all the model weights. The

main computational cost comes from forwarding the images through the ViT and backpropagating through the layers after the split point to compute gradients with respect to the scale values. In addition, two ViT forward passes are required. The first is used to obtain the predicted label for the original input, while the second is used to obtain the prediction after applying the perturbation. Overall, learning the importance map is computationally less expensive than training the entire ViT, since only the scale parameters are optimized while the backbone parameters remain fixed.

For the online component, we have designed GPU-efficient algorithms for both group creation, i.e., one-step outlier identification, and bit allocation, i.e., one-step selection of the top marginal gains $E_{ij}(n_{ij})$. Both steps are implemented primarily on the GPU, using GPU `top-k` operations as the core primitive.

APPENDIX B
LEARNING AN IMPORTANCE MAP WITH STOCHASTIC GRADIENT DESCENT: DETAILS AND PROOFS

A. Proofs of Equivalence of Distortion Model (3) and Optimization Problem (5): Theorems 2 and 1

Theorem 1 shows that, for the choices $\phi(x) = \log_2(x)$ and $f(x) = \frac{1}{x}$, the learning approach recovers the importance parameters S_{ij} up to a constant factor. We first establish a more general result (Theorem 2) that allows for arbitrary design choices of ϕ, f, and k.

Theorem 2. *Let $J(\mathbf{w})$ be as defined in (4), with $\mathbf{w} \in \mathcal{W}$ (as defined in (6)) and $f : \mathbb{R} \to \mathbb{R}^+$ applied component-wise.*

The optimization problem is:

$$\min_{\mathbf{w}\in\mathcal{W}} J(\mathbf{w}) \tag{14}$$

For $d(x, y) = |x - y|^k$ with $k \geq 1$, and under the separable distortion model $\mathbb{E}[d(T(\mathbf{A}+\boldsymbol{\eta}), T(\mathbf{A}))] = \sum_{(i,j)\in\Omega} S_{ij}|\eta_{ij}|^k$, satisfy:

$$S_{ij} \propto \frac{\phi'(w_{ij})}{(f(w_{ij}))^{k-1} f'(w_{ij})} \tag{15}$$

Proof. Let $k \geq 1$ be the exponent of the distortion $d(x, y) = |x-y|^k$. Under the separable distortion model, we can rewrite:

$$\begin{aligned} J(\mathbf{w}) &= \mathbb{E}_{A,z}\left[d(T(A + z \odot f(\mathbf{w})), T(A))\right] \\ &= \mathbb{E}_z\left[\sum_{(i,j)\in\Omega} S_{ij}|z_{ij} f(w_{ij})|^k\right]. \end{aligned}$$

Denoting $\mu \triangleq \mathbb{E}_z[|z_{ij}|^k]$ (independent of (i, j) since the noise is i.i.d.) and using $f(w_{ij}) > 0$:

$$J(\mathbf{w}) = \mu \sum_{(i,j)\in\Omega} S_{ij} \left(f(w_{ij})\right)^k.$$

Since $w_{\min}$ and K are chosen so that the optimum lies in the interior of $\mathcal{W}$ (i.e. $w^*_{ij} > w_{\min}$ for all $(i, j) \in \Omega$), the inequality constraints $w_{ij} \geq w_{\min}$ are inactive at the optimum and the KKT conditions reduce to those of the equality-constrained problem. We therefore apply the Lagrange multiplier method to the equality constraint alone:

$$\mathcal{L}(\mathbf{w}, \lambda) = J(\mathbf{w}) + \lambda\Big(\sum_{(i,j)\in\Omega} \phi(w_{ij}) - K\Big). \tag{16}$$

Setting $\partial\mathcal{L}/\partial w_{ij} = 0$:

$$k\mu S_{ij} \left(f(w_{ij})\right)^{k-1} f'(w_{ij}) + \lambda\phi'(w_{ij}) = 0.$$

Solving for S_{ij} and eliminating the common factor $k\mu/\lambda$ (constant across (i, j)) yields (15). □

Theorem 1. *For $d(x, y) = |x - y|$, $\phi(x) = \log_2(x)$, and $f(x) = 1/x$, the optimization problem (5) gives optimal parameters $\mathbf{w}^*$ satisfying $w^*_{ij} \propto S_{ij}$, i.e. $\mathbf{w}^*$ recovers the importance map $\mathbf{S}$ of (3) up to a constant factor, for the constraint set (6).*

Proof. This is an immediate corollary of Theorem 2. Taking $k = 1$, $f(x) = \frac{1}{x}$ and $\phi(x) = \log_2(x)$, so that $f'(x) = -\frac{1}{x^2}$ and $\phi'(x) = \frac{1}{x \ln 2}$. Since $(f(w_{ij}))^{k-1} = 1$ for $k = 1$, equation (15) gives:

$$S_{ij} \propto \frac{\phi'(w_{ij})}{f'(w_{ij})} = \frac{\frac{1}{w_{ij}\ln 2}}{-\frac{1}{w_{ij}^2}} = -\frac{w_{ij}}{\ln 2} \propto w_{ij}.$$

Hence $w^*_{ij} \propto S_{ij}$, as claimed. □

We note that an alternative choice, yielding the same conclusion as Theorem 1, would be: $f(x) = \log_2(x)$ and $\phi(x) = x$. Indeed, $\phi'(w_{ij})/f'(w_{ij}) = w_{ij} \ln 2 \propto w_{ij}$, so we also have: $S_{ij} \propto w_{ij}$. This is equivalent to a change of variable. The practical advantage of this parameterization is that ϕ being linear makes the constraint set $\mathcal{W} = \{\mathbf{w} \mid \sum_{(i,j)\in\Omega} w_{ij} = K,\ \forall(i, j) \in \Omega : w_{ij} \geq 1\}$ a standard truncated simplex, so projection applies directly.

Notice that the proofs hold for any i.i.d. noise distribution, and the distribution only changes the constant factor and the required choice of $w_{\min}$.

B. Projected Gradient Descent in the Reparameterized Space

To clarify the optimization procedure, we describe how the constraints defined by $\mathcal{W}$ (6) are efficiently enforced during optimization.

Let us define the change of variable $t_{ij} = \phi(w_{ij})$ and the associated feasible set that becomes:

$$\mathcal{T} = \phi(\mathcal{W}) = \Big\{\mathbf{t} \;\Big|\; \sum_{(i,j)\in\Omega} t_{ij} = K, \quad \forall(i, j) \in \Omega :\ t_{ij} \geq \phi(w_{\min})\Big\}.$$

The constraints of $\mathcal{W}$ are linear in the variables t_{ij}. It is a *shifted simplex*, hence feasibility can be enforced by classical algorithms. If our optimization problem were defined directly in the t_{ij} variables, then we would use classical Projected Stochastic Gradient Descent (PSGD), by enforcing the constraint $\mathbf{t} \in \mathcal{T}$ at each step. Critical to the efficiency of the method using training, the projection step needs to be efficient: it is the case with methods, for instance, found in [37], [38].

Since our optimization problem is defined in the w_{ij} variables, we can still use, as a heuristic, projected SGD by performing the projection step in the t_{ij} space, and after projection, the corresponding variables are recovered as $w_{ij}^{\text{proj}} = \phi^{-1}(t_{ij}^{\text{proj}})$. This change of variables transforms the complex constrained problem into a standard PSGD setup, as it ensures that $\mathbf{w}^{\text{proj}} \in \mathcal{W}$ after each step, since the constraints on $\mathbf{t}$ and $\mathbf{w}$ are equivalent under ϕ.

APPENDIX C
ADDITIONAL EXPERIMENTAL RESULTS

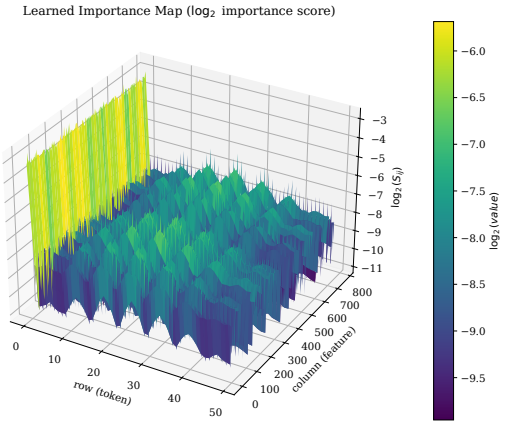
A. Additional Results on Learned Importance Maps

In this section, we provide additional information on the results regarding the importance maps. The importance maps are obtained using the two methods (gradient-based and SGD-based) described in Section III-A5.

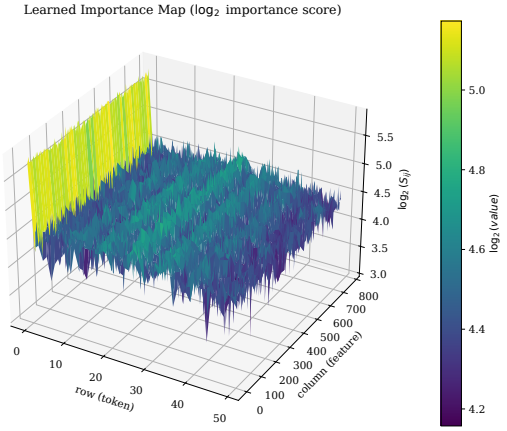
Both the gradient-based and SGD-learned importance maps are computed over a training set (denoted as the Grad map and SGD map, respectively). The SGD map is obtained via stochastic gradient descent (mirroring standard model training, but only updating importance coefficients), with additional hyperparameters for ViT-B/32: $w_{\min} = 1$, $K = 4.5$, and $\mu = \mathbb{E}(|\text{noise}|) = 1.73$, as previously indicated.

In Figure 6, we visualize both the gradient-based and SGD-learned importance maps for ViT-B/32, illustrating how the proposed method assigns different sensitivity levels across activation coefficients. The $\log_2$ of the importance score is shown: notably, the difference in $\log_2$ values between coefficients at different positions approximately corresponds to the difference in their allocated bit widths (as the bit budget increases), as deduced from (13) and related equations for the large bit-budgets.

Furthermore, from Figures 6a and 6b, we observe that the class token receives a significantly higher importance score than the remaining activations. This is clearly expected, as the class token aggregates information from all image patches through the self-attention mechanism and serves as the sole input to the final linear classifier. Consequently, small perturbations in the class token can lead to substantial drops in accuracy.



(a) Grad map for ViT-B/32



(b) SGD map for ViT-B/32

Fig. 6: Importance Maps for ViT-B/32

B. Additional Results for LIQOS with ViT-B/16 and DINO-v2

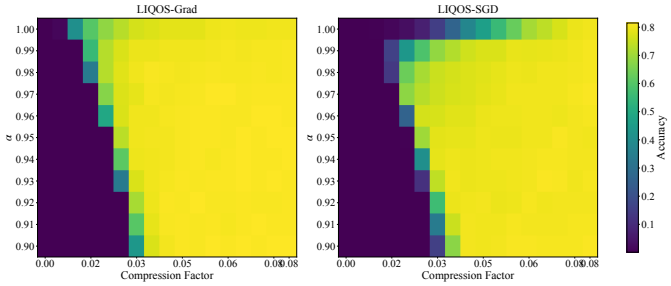
Figure 7 presents the same outlier-threshold analysis for ViT-B/16. Overall, the same observation for ViT-B/32 also holds for ViT-B/16. The percentile threshold α primarily affects the transition region between low and moderate compression factors. At very low compression factors, both LIQOS-Grad and LIQOS-SGD exhibit significant degradation in accuracy, whereas increasing the compression factor rapidly improves accuracy. Finally, as with ViT-B/32, the Pareto front analysis shown in 7b also confirms that high α values dominate at low compression factors, whereas for larger compression factors, a more diverse set of α values contributes to the accuracy/compression operating points.

Finally, Figure 8 plots the accuracy versus compression factor for ViT-B/16 for both LIQOS-Grad, LIQOS-SGD, ORQ, and uniform quantization approaches. The obtained results confirm the observations obtained from ViT-B/32. Both LIQOS-Grad and LIQOS-SGD outperform the baseline approaches, especially for low compression factors. This clearly shows that the proposed approach remains effective across different ViT architectures.

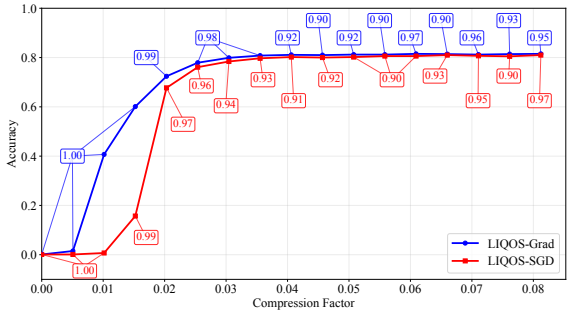
Similar results were obtained with LIQOS on another Vision Transformer model, DINOv2 [39], specifically the DINOv2-S/14 model with registers. As in the previous experiments, our approach consistently achieves better results than ORQ across the different importance maps, as shown in Figure 9. We still achieve a high compression factor (i.e., 0.05), but the "knee" of the curve is less pronounced. We conjecture that this may be because DINOv2 is designed to learn generic visual representations rather than features optimized for a specific image-classification objective. Consequently, information may be more evenly distributed across tokens, making perturbations to any token more likely to affect downstream performance.

C. Additional Results for LIQOS for Different Split Points

We also conducted experiments using split points at different places in the model. Namely, after the first block for each split point, we computed the gradient map and applied LiQoS-



(a) accuracy heatmaps over compression ratio and α on ViT-B/16.



(b) accuracy/compression Pareto Fronts on ViT-B/16

Fig. 7: Outlier Grouping Threshold – ViT-B/16.

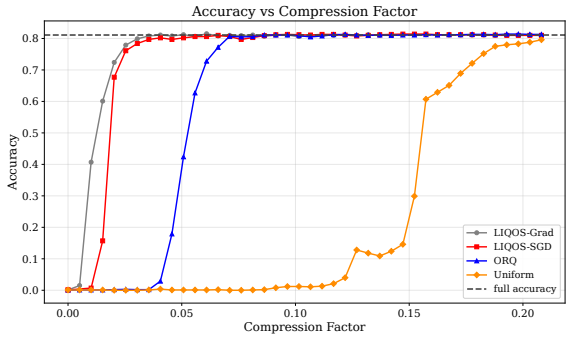


Fig. 8: Accuracy Across Compression Factors for LIQOS, ORQ, and Uniform Sampling for ViT-B/16

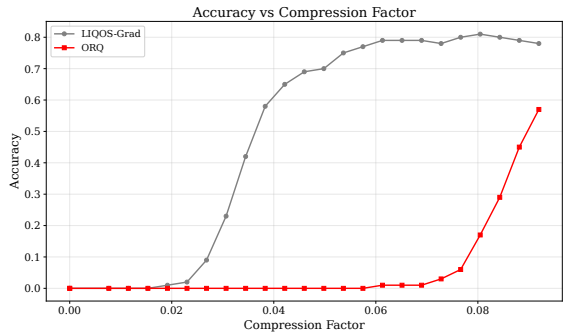


Fig. 9: Accuracy Across Compression Factors for LIQOS and ORQ Sampling for DINOv2

Grad. In Figure 10, we observed that split points located in the middle of the model provide similar performance (i.e., split

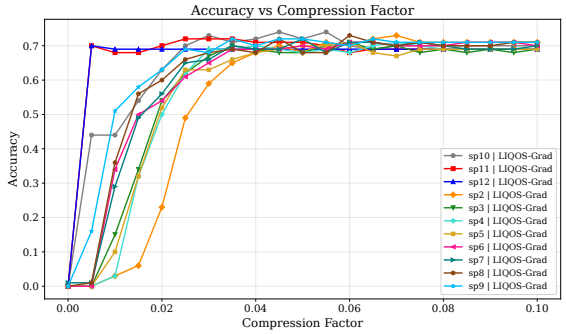


Fig. 10: Accuracy Across Compression Factors for LIQOS-Grad in several split point for ViT B/32

points 3 to 8). They also provide an interesting compression factor around 0.03. In contrast, extreme split points, such as 11 and 12, achieve better compression, reaching up to 99%. Or split point 2 seems to be too early (before significant processing occurred), hence provides worse performance. At critical accuracy levels (40%-60% accuracy), there are still potential gains to be obtained from accuracy-aware split-point selection. This was not the main focus of this work and is an interesting direction for future work.

D. Reference Dataset and Base Model Performance

We document the base model performance in this section.

Our reference dataset is a subset of ImageNet-1k, selecting 5 images per class (for each of the 1,000 classes) from both the training set and the validation set. It was chosen over datasets sometimes used in the split computing literature (e.g., CIFAR-10 or CIFAR-100), as it is more meaningful to apply split computing to higher-resolution images (natively averaging above 224×224 pixels, resized to that resolution for model input) than to 32×32 images.

We evaluate three off-the-shelf pretrained vision backbones: torchvision ViT-B/32 (`vit_b_32`, pretrained `IMAGENET1K_V1`), torchvision ViT-B/16 (`vit_b_16`, pretrained `IMAGENET1K_V1`), and the official DINOv2 ViT-S/14 register variant with linear classifier (`dinov2_vits14_reg_lc`), using the released DINOv2 [39] pretrained backbone and official linear-head checkpoint. No additional backbone pretraining or fine-tuning was performed for these baseline accuracy measurements.

For reference, Table I summarizes the performance of the tested models on our reference dataset, and the size of their latent space. As known, ViT-B/16 and ViT-B/32 overfit the ImageNet-1K training set (on which they were trained). In other sections, we therefore report performance on the validation subset.

TABLE I: Compact summary of the three models used.

Model	L×D	Top-1 Acc. (train / val)
ViT-B/32	50×768	89.48% / 74.58%
ViT-B/16	197×768	94.00% / 80.94%
DINOv2-S/14 (4 regs.)	261×384	84.86% / 80.48%