Automatic mixed precision for optimizing gained time with constrained loss mean-squared-error based on model partition to sequential sub-graphs

Shmulik Markovich-Golan* **Daniel Ohayon*** **Itay Niv** **Yair Hanani**
Intel Corporation / Habana Labs
{shmulik.markovich-golan, daniel1.ohayon, itay.niv, yair.hanani}@intel.com

Abstract

Quantization is essential for Neural Network (NN) compression, reducing model size and computational demands by using lower bit-width data types, though aggressive reduction often hampers accuracy. Mixed Precision (MP) mitigates this tradeoff by varying the numerical precision across network layers. This study focuses on automatically selecting an optimal MP configuration within Post-Training Quantization (PTQ) for inference. The first key contribution is a novel sensitivity metric derived from a first-order Taylor series expansion of the loss function as a function of quantization errors in weights and activations. This metric, based on the Mean Square Error (MSE) of the loss, is efficiently calculated per layer using high-precision forward and backward passes over a small calibration dataset. The metric is additive across layers, with low calibration memory overhead as weight optimization is unnecessary. The second contribution is an accurate hardware-aware method for predicting MP time gain by modeling it as additive for sequential sub-graphs. An algorithm partitions the model graph into sequential subgraphs, measuring time gain for each configuration using a few samples. After calibrating per-layer sensitivity and time gain, an Integer Programming (IP) problem is formulated to maximize time gain while keeping loss MSE below a set threshold. Memory gain and theoretical time gain based on Multiply and Accumulate (MAC) operations are also considered. Rigorous experiments on the Intel Gaudi 2 accelerator validate the approach on several Large Language Models (LLMs).

1. Introduction

Quantization is a key technique for compressing neural networks by converting high-precision weights and activations to lower-precision formats, significantly reducing model size and computational load ([17], [6], [13], [7], [19][11]). This is crucial for efficient inference on accelerators and edge devices. The main quantization approaches are Quantization-Aware Training (QAT), Post-Training Quantization (PTQ), and Quantization-Aware Fine-Tuning (QFT).

QAT incorporates quantization during training, maintaining accuracy despite aggressive quantization (e.g., INT4) but requires substantial data and computational resources ([9], [10]). In contrast, PTQ quantizes pre-trained models efficiently but may reduce accuracy, especially in low-precision scenarios ([12], [2]). QFT combines PTQ with fine-tuning to balance accuracy and resource efficiency ([1]).

Mixed Precision (MP) has emerged as a key technique for optimizing Neural Network (NN) performance across hardware platforms ([16]). Automatic selection of MP configuration for inference involves search-based methods (e.g., Hessian AWare Quantization (HAWQ) and Orthogonal Mixed

* Equal contribution.

Precision Quantization (OMPQ)) and optimization-based approaches (e.g., Differentiable Neural Architecture Search (DNAS), Hardware-Aware Automated Quantization (HAQ), Reinforcement Learning Approach for Deep Quantization (ReLeQ), AUTOQ). Optimization-based methods tackle the non-differentiability of bit-widths through reinforcement learning and hardware feedback.

Pandey et al. [14] proposed an MP algorithm for post-training scenarios, minimizing data usage and considering hardware limitations. The algorithm first measures layer sensitivity, then reduces bit-width iteratively while maintaining performance. Wu et al. [20] introduced Structured Mixed-precision (StruM), tailored for compatible hardware, while [4] formulated loss minimization as a Multiple-Choice Knapsack Problem (MCKP), solved with a greedy search algorithm.

In this work, we address the challenge of selecting an MP configuration that optimizes gained time or memory compared to the high-precision model for PTQ, while maintaining a constraint on the Mean Square Error (MSE) of the loss. To solve this problem, we introduce a novel sensitivity metric derived from the MSE of the loss of the quantized model. This metric is formulated by approximating the loss error as a first-order Taylor series expansion of quantization errors from weights and activations. It is estimated using both forward and backward passes of the model at high precision with a calibration dataset. The metric predicts the MSE of the loss for arbitrary MP configurations by considering the loss error components from different quantized layers as statistically independent. The MSE of a given component from a layer is calculated as the product of its sensitivity and the MSE of the quantization error of an individual element. Although the method requires a backward pass, the additional memory requirement is minimal, mainly consisting of stored activations (since weighta optimizer is not required).

We also introduce a method for predicting the empirical time gained from a MP configuration, based on the additive execution time of sequentially computed sub-graphs. The model structure is analyzed to find sequential sub-graphs, each potentially comprising multiple layers or a single layer. Time gains for all MP configurations are measured for each group, enabling the prediction of gained time for any configuration. The total time gain is estimated as the sum of the gained times per group.

2. Proposed method

In Sec. 2.1 the problem is formulated and the solution is derived based on Integer Programming (IP) optimizing a generic objective function with constrained loss MSE per group (sequential sub-graph). A full derivation of the sensitivity and the loss MSE, are given Appendix E. Various performance metrics which can substitute the generic objective function are defined in Sec. 2.2. We also discuss the motivation and method for partitioning the model graph to sequential sub-graphs for accurately assessing the time gained by MP. The method is summarized in Sec. A.

2.1. Formulation

Let $\mathcal{M}$, $\mathbf{X}$ and $\mathbf{Y}$ respectively denote a NN, the input and output, such that:

$$\mathbf{Y} \triangleq \mathcal{M}\left(\mathbf{X}\right) \tag{1}$$

and let $g\left(\mathcal{M}\left(\mathbf{X}\right), \mathbf{Y}_{\text{true}}\right)$ denote the loss function where $\mathbf{Y}_{\text{true}}$ represents the *ground-truth target* corresponding to $\mathbf{Y}$. The NN is composed of L linear operations, including both standard linear layers and Batch General Matrix Multiplication (BGEMM) layers. Assume that the underlying hardware accelerator supports F distinct numerical formats. A per-layer MP configuration $\mathcal{I}^{\text{layer}}$

is defined by a set of $L \times F$ binary indicators (one per each combination of layer and numerical format):

$$\mathcal{I}^{\text{layer}} \triangleq \left\{ i_{\ell,f}^{\text{layer}} \in \{0,1\} \right\}_{\ell \in [0,L-1], f \in [0,F-1]} \tag{2}$$

where ℓ indexes the layers and f indexes the numerical formats. Each layer is assigned exactly one numerical format, enforcing the constraint $\sum_f i_{\ell,f}^{\text{layer}} = 1$. The numerical formats are assumed to be various floating-point representations, differentiated by their mantissa bit widths, denoted m_f.

Now, suppose the model is partitioned into J disjoint groups of layers, $\{V_j\}_{j=0}^{J-1}$, such that layers within a group exhibit dependent performance characteristics, while different groups are independent. Define each group as the set of layer indices comprising it, i.e., $V_j \triangleq \left\{\ell_{j,0}, \ldots, \ell_{j,L_j-1}\right\}$, where L_j is the number of layers in group j. Let $\mathbf{Q}_j \in \mathbb{Z}^{L_j \times F^{L_j}}$ be the matrix enumerating all per-layer possible quantization configurations for group j, where each column specifies a choice of numerical formats for the group layers, and each of its elements is in the range $[0, F-1]$.

We extend the standard per-layer binary indicator into a per-group binary indicator as follows. A per-group MP configuration $\mathcal{I}$, also denoted here as an MP configuration for brevity, is defined by a set of $J \times F^{L_j}$ binary indicators (one per each combination of group and any of its F^{L_j} possible quantization combinations):

$$\mathcal{I} \triangleq \{i_{j,p} \in \{0,1\}\}_{j \in [0,J-1], p \in \left[0, F^{L_j}-1\right]} \tag{3}$$

where j indexes the groups and p indexes its quantization configurations (indicating that the configuration in p-th column of $\mathbf{Q}_j$ is selected). Each group is assigned exactly one configuration, enforcing the constraint $\sum_p i_{j,p} = 1$. A special case arises when the entire model is sequential; this corresponds to $J = L$ single layer groups with $V_\ell = \{\ell\}$.

Let $\mathbf{c}_j \in \mathbb{R}^{F^{L_j}}$ be the vector of performance metric values associated with the configurations in $\mathbf{Q}_j$, and $\mathbf{d}_j \in \mathbb{R}^{F^{L_j}}$ the corresponding loss MSE values. Define the MSE of the loss function due to approximation under an MP configuration as:

$$\mathrm{E}\left[\tilde{g}^2\right] = \mathrm{E}\left[(\hat{g} - g)^2\right] \tag{4}$$

where $\mathrm{E}[\bullet]$ denotes the expectation operator and $\hat{g}$ is the perturbed loss under the MP configuration.

Let c be a performance metric to be maximized. In this study, we evaluate several metrics: *empirical time gain* denoted c^{ET}, *theoretical time gain* estimated from the number of Multiply and Accumulate (MAC) operations denoted c^{TT} and *memory gain* from reduced model size denoted c^{M}. Execution under an MP configuration $\mathcal{I}$ aims to improve performance, while potentially increasing the loss MSE.

Assuming a maximum allowable loss MSE of $\tau^2 \mathrm{E}\left[g^2\right]$, for a parameter $\tau < 1$ (which is the normalized-Root Mean Square Error (RMSE) threshold), our objective is to determine the optimal MP configuration by solving:

$$\begin{aligned} \{i_{j,p}\}_{j,p} = \text{argmax}_{\{\underline{i}_{j,p}\}_{j,p}} & \sum_{j,p} \underline{i}_{j,p} \mathbf{c}_{j,p} \\ \text{s.t.:} & \sum_{j,p} \underline{i}_{j,p} d_{j,p} \leq \tau^2 \mathrm{E}\left[g^2\right], \ \sum_p \underline{i}_{j,p} = 1 : \ \forall j, \ \underline{i}_{j,p} \in \{0,1\} : \ \forall j,p. \end{aligned} \tag{5}$$

Define the IP loss MSE and performance metrics as:

$$d \triangleq \sum_{j,p} \underline{i}_{j,p} d_{j,p} \qquad (6) \qquad\qquad c \triangleq \sum_{j,p} \underline{i}_{j,p} \mathbf{c}_{j,p} \qquad (7)$$

In Appendix E and Sec. 2.2, we derive explicit expressions for the latter, respectively. The loss MSE components $d_{j,p}$ are defined as:

$$d_{j,p} \triangleq \sum_{l=0}^{L_j-1} s_{\ell_{j,l}} \alpha_{Q_{j,lp}} \qquad (8) \qquad\qquad s_\ell^r \triangleq \|\mathbf{z}_\ell^r \odot \dot{\mathbf{z}}_\ell^r\|^2 \qquad (9)$$

with s_ℓ^r being the sensitivity of the loss to the quantization of tensor $\mathbf{z}_\ell^r$, comprising the activations and parameters of the ℓ-th layer, with $\dot{\mathbf{z}}_\ell^r$ being the gradient of the loss with respect to this tensor, $s_\ell \triangleq \frac{1}{R}\sum_r s_\ell^r$ is its average over the sample index r, and $\alpha_{Q_{j,lp}}$ the respective quantization noise variance according to configuration $Q_{j,lp}$.

2.2. Performance metric

The choice of the performance metric c significantly impacts the resulting MP configuration. We consider three metrics: empirical time gain, theoretical time gain, and memory gain.

2.2.1. EMPIRICAL TIME GAIN c^{ET}

Model partition to sequential sub-graphs: The partition process is briefly described. For more details please refer to Sec. B. Consider representing the computation of a model as a Directed Acyclic Graph (DAG). Note that two adjacent sub-graphs that are connected by a single edge are computed sequentially since the second sub-graph *depends* on the output of the first sub-graph. This sequential computation allows us to model their combined computation time as the sum of their individual times, which also applies to their gained time. Our partition procedure identifies *single-entry/single-exit sub-graphs* bounded by branching and merging nodes, splitting the computation graph into as many sequential sub-graphs as possible. These sub-graphs form an ordered sequence $\{V_j\}_{j=0}^{J-1}$ that executes strictly sequentially at run-time. Predicting the computation time of concurrent layers within sub-graphs presents significant challenges. Operations within a sub-graph may execute in parallel, while the compiler is free to fuse or reorder operations. Additionally, latency depends on complex interactions between layer dependencies, hardware resources, and scheduling rules. We propose to avoid this complication and measure the gained time of each sub-graph, represented as a group of layers comprising it, for all their possible quantization configurations.

Gained time based on empirical time measurements per-group vs. per-layer: Consider the gained time of the Attention sub-graph in LLAMA-3.1-8B, illustrated in Figure 5, which contains the quantizable layers: `q_proj`, `v_proj`, `k_proj`, `qk_matmul` and `av_matmul`. Figure 1 compares the measured empirical time gain $c_{j,p}^{\text{ET}}$ of the attention sub-graph against the theoretical time gain predicted as the corresponding sum of per-layer time gain measurements. The large discrepancies demonstrate that simple summation of per-layer measurements does not yield a good estimate for the time gain of a sub-graph which contains concurrent computations. It shows the gap that the proposed method addresses.

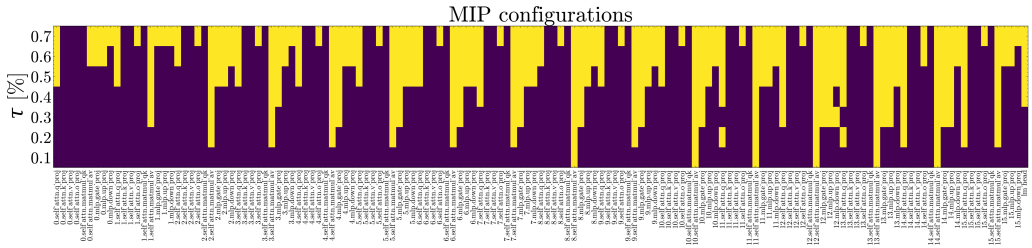


Figure 2: Layer-wise quantization patterns across MP configurations (rows) and model layers (columns) for IP-EmpiricalTime (IP-ET). Yellow: FP8, purple: BF16.

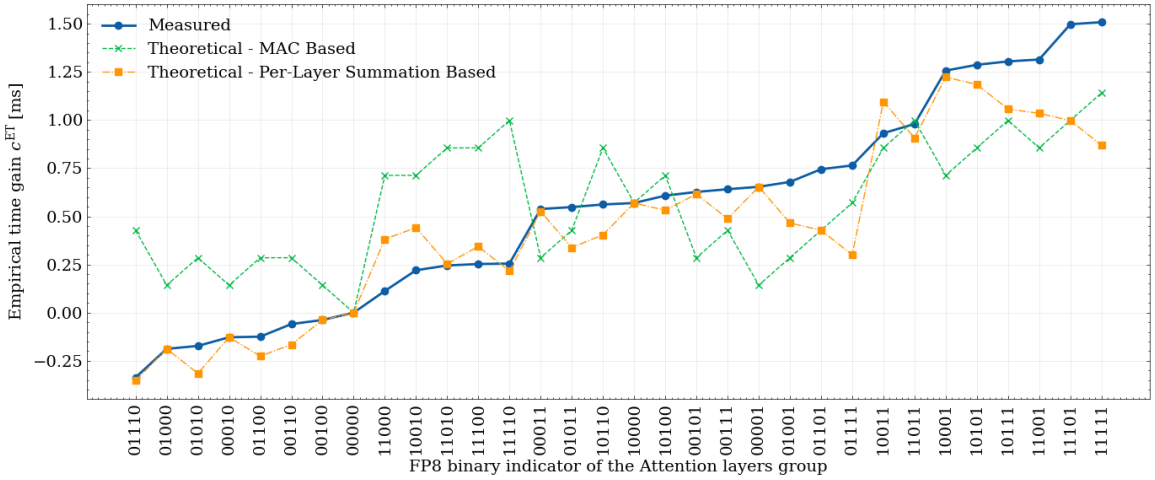


Figure 1: Measured empirical time gain $c^{\text{ET}}_{j,p}$ of the Attention sub-graph in LLAMA-3.1-8B (in blue) compared to its prediction based on the summation of per-layer time gain measurements (in orange) and for the theoretical time gain $c^{\text{TT}}_{j,p}$ (green) for any of its 2^5 MP configurations. The various configurations are ordered in *ascending order of empirical time gain*. Configurations are labeled as 5-bit binary words which represent the numerical format of each of the 5 linear operations (`q_proj`, `v_proj`, `k_proj`, `qk_matmul`, and `av_matmul`) with BF16 and FP8 denoted as 0 and 1, respectively.

Gained time measurements: The time gain of the p-th MP configuration of the j-th group is measured by subtracting the end-to-end Time To First Token (TTFT) of the model with the j-th group configured correspondingly and the other groups configured to BF16 from the end-to-end TTFT of the model in BF16.

3. Experimental results

We evaluate our proposed mixed-precision strategies (IP-ET, IP-TT, IP-M) against Random and Prefix baselines using 1B and 8B models on an Intel Gaudi 2 accelerator. We first validate the key additivity assumptions of our method in Sec. D.3 before analyzing the final trade-off curves for loss vs. time (Sec. 3.1) and accuracy vs. performance (Sec. 3.2). The complete experimental setup is detailed in Appendix D.1.

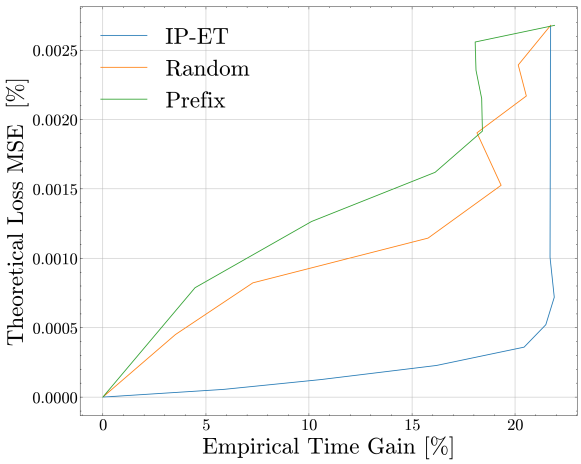


Figure 3: Theoretical loss MSE vs. empirical time gain on the Meta-Llama-3.2-1B-Instruct (1B) model across four tasks.

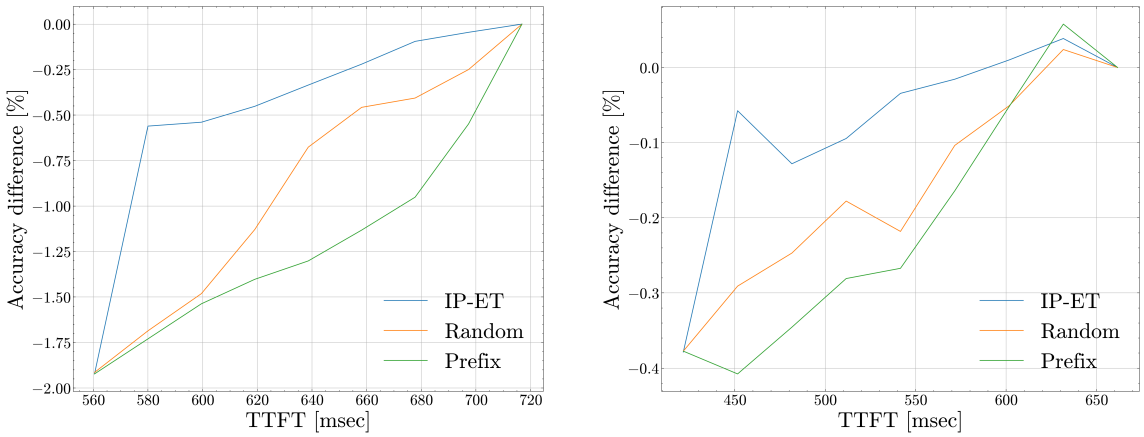
3.1. Loss MSE vs. empirical time gain curve

Figure 3 demonstrates that the IP-ET strategy is significantly and consistently better then the Random and Prefix strategies, yielding an appealing loss MSE vs. empirical time gain curve. Furthermore, it maintains markedly low loss MSE vs. empirical time gain.

3.2. Accuracy vs. performance curve

Figure 4(*a*)subfigure and Figure 4(*b*)subfigure illustrate the accuracy degradation vs. TTFT curves of different strategies for the 1B and 8B models, respectively. For both models, IP-ET consistently achieves better accuracy at comparable latency than the baselines. E.g., with Meta-Llama-3.1-8B-Instruct (8B), IP-ET achieves accuracy loss below 0.1% at 450ms TTFT, whereas other strategies require $\sim$600ms for similar accuracy—a 30% speedup.

Table 1 provides a comprehensive comparison across all strategies and models. The proposed IP-based methods consistently outperform the baselines. Despite its limited quantization scope (linear layers only), IP-Memory (IP-M) still surpasses the baselines in most cases, with one exception: for 8B on LAMBADA, the Prefix strategy achieves slightly higher accuracy. These results confirm that sensitivity-aware, hardware-informed quantization significantly improves inference efficiency while preserving model quality. The improvement of the proposed method for the 1B model is better then the 8B model since the gap between the FP8 and BF16 accuracies there is larger. See Sec. D for additional results on per-task time gains, MAC-based gains, and memory gains.



(*a*) 1B (*b*) 8B

Figure 4: Average accuracy difference [%] vs. TTFT across HellaSwag, LAMBADA, Winogrande, and PIQA. Comparing MP quantization strategies (IP-ET, Random and Prefix)

4. Conclusions

By utilizing a novel *loss MSE* and *empirical time gain per sequential sub-graphs* metrics, we introduce an automatic MP method based on IP for PTQ. The proposed loss MSE metric, which exhibits additive properties per layer, serves as a proxy for model accuracy. We efficiently approximate this metric using forward- and backward-passes over a small calibration dataset. Recognizing that the empirical time gain exhibits additivity solely for sequential sub-graphs—attributable to parallel capabilities and advanced compiler optimizations in the hardware accelerator— we formulate an algorithm for model partitioning. In this approach, each sub-graph is characterized as a group of constituent layers, and we define a performance objective function by summing the empirical time gain for each group. To achieve this, we measure the empirical time gains of each sub-graph over a limited set of samples. We validate both the approximation of the loss MSE and its additive nature across layers. Furthermore, we demonstrate that the empirical time gain is additive per group, resulting in a highly accurate estimate of the measured time gain. Finally, we evaluate the proposed method by comparing it against baseline strategies (Random and Prefix configurations), demonstrating that it consistently outperforms these approaches across various Large Language Models (LLMs).

References

[1] Saleh Ashkboos, Bram Verhoef, Torsten Hoefler, Evangelos Eleftheriou, and Martino Dazzi. Efqat: An efficient framework for quantization-aware training. arXiv preprint arXiv:2411.11038, 2024. URL https://arxiv.org/abs/2411.11038.

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1810.05723.

[3] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[4] Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization, 2021. URL https://arxiv.org/abs/2110.06554.

[5] EleutherAI. Language model evaluation harness. https://github.com/EleutherAI/lm-evaluation-harness, 2025.

[6] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. 2021. https://arxiv.org/abs/2103.13630.

[7] Yunhui Guo. A survey on methods and theories of quantized neural networks. 2018. https://arxiv.org/abs/1808.04752.

[8] Intel Corporation. Intel neural compressor. https://github.com/intel/neural-compressor, 2024.

[9] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018. URL https://arxiv.org/abs/1712.05877.

[10] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, 2018. URL https://arxiv.org/abs/1806.08342.

[11] Joonhyung Lee, Shmulik Markovich-Golan, Daniel Ohayon, Yair Hanani, Gunho Park, Byeongwook Kim, Asaf Karnieli, Uri Livne, Haihao Shen, Tai Huang, Se Jung Kwon, and Dongsoo Lee. Faster inference of llms using fp8 on the intel gaudi, 2025. URL https://arxiv.org/abs/2503.09975.

[12] Szymon Migacz. 8-bit inference with tensorrt. NVIDIA GPU Technology Conference (GTC), 2017. URL https://www.cse.iitd.ac.in/~rijurekha/course/tensorrt.pdf. Presentation.

[13] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. 2021. https://arxiv.org/abs/2106.08295.

[14] Nilesh Prasad Pandey, Markus Nagel, Mart van Baalen, Yin Huang, Chirag Patel, and Tijmen Blankevoort. A practical mixed precision algorithm for post-training quantization. *arXiv preprint arXiv:2302.05397*, 2023.

[15] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[16] Mariam Rakka, Mohammed E Fouda, Pramod Khargonekar, and Fadi Kurdahi. A review of state-of-the-art mixed-precision neural network frameworks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[17] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 2023. https://arxiv.org/abs/2205.07877.

[18] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

[19] Olivia Weng. Neural network quantization for efficient inference: A survey. 2021. https://arxiv.org/abs/2112.06126.

[20] Michael Wu, Arnab Raha, Deepak A. Mathaikutty, Martin Langhammer, and Engin Tunali. Strum: Structured mixed precision for efficient deep learning hardware codesign, 2025. URL https://arxiv.org/abs/2501.18953.

[21] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Appendix

Appendix A. Proposed method summary

Algorithm 1 summarizes our end-to-end approach for automatic MP configuration. The method integrates hardware-aware timing measurements with gradient-based sensitivity analysis to determine optimal precision assignments. After partitioning the model into sequential sub-graphs (line 1), we perform sensitivity calibration through forward and backward passes on the calibration dataset (line 2). We then measure empirical time gains for each sub-graph across different precision configurations (line 3), before formulating and solving the IP optimization that maximizes performance while respecting the loss MSE threshold (line 4). This algorithm forms the foundation for all three optimization strategies (IP-ET, IP-TheoreticalTime (IP-TT), and IP-M).

Algorithm 1 Proposed automatic MP algorithm summary

Input: A model $\mathcal{M}$, a calibration dataset $\mathcal{D}_{\text{calib}}$ and a relative RMSE threshold τ
Output: MP configuration $\mathcal{I}$ (see (3))

1: Analyze model and partition it to J sequential sub-graphs $\{V_j\}_j$ as described in B
2: Sensitivity calibration

- Wrap the model $\mathcal{M}$ to enable sensitivity measurement
- Run forward- and backward- passes over $\mathcal{D}_{\text{calib}}$, and obtain: sensitivity $\{s_\ell\}_\ell$ and mean-square loss $\mathrm{E}\left[g^2\right]$ (see (26))

3: Empirical time gain measurement

- Measure TTFT of j-th group and p-th MP configuration, for $j \in [0, J-1]$ and $p \in \left[0, F^{L_j} - 1\right]$
- Compute c^{ET} by subtracting the measurements from the TTFT of the model in BF16,

4: Obtain $\mathcal{I}$ by solving the IP optimization problem (see (5))
5: **return** $\mathcal{I}$

Appendix B. Model Partitioning into sequential groups of layers

Effective MP assignment requires identifying model sub-graphs which execution time is additive. Given a network's computation graph, that can be formulated as a DAG with a single sink vertex - our partitioning algorithm splits the model to sequential sub-graphs with a single entry and a single exit points. Figure 5 illustrates the resulting partitioning for a Llama-3 transformer layer, showing the Attention and MLP blocks split into single-entry/single-exit sub-graphs (V_1–V_4) that serve as the fundamental units for our MP optimization.

Appendix C. Alternative Optimization Metrics

C.0.1. THEORETICAL TIME GAIN c^{TT}

This performance metric is defined per-layer as the theoretical time gain based on the number of MAC operations multiplied by the gained time of a single MAC in the f-th numerical format (compared to BF16), denoted $\delta_{\text{T},f}$.

For a linear layer $\ell \in \{\mathcal{L}_{\text{lin}}, \mathcal{L}_{\text{BGEMM}}\}$ with N samples, input dimension C_ℓ, and output dimension K_ℓ the theoretical time gain is defined as:

$$c_{\ell,f} \triangleq \begin{cases} NC_\ell K_\ell \delta_{\text{T},f} & ;\ell \in \mathcal{L}_{\text{lin}} \\ NC_\ell^2 \delta_{\text{T},f} & ;\ell \in \mathcal{L}_{\text{BGEMM}} \end{cases} \tag{10}$$

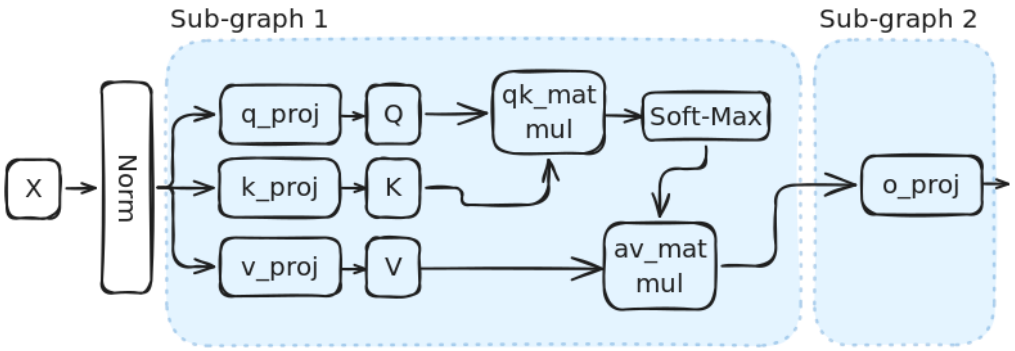
It is a simple performance metric that approximates the gained time without requiring any timing measurements.

We compare theoretical and empirical time gains, i.e., $c^{\text{TT}}_{j,p}$ and $c^{\text{ET}}_{j,p}$, for the Attention sub-graph in LLAMA-3.1-8B. By definition, since the theoretical time gain is based on number of MACs, it is additive across layers. Therefore, the theoretical time gain of the p-th configuration of the j-group

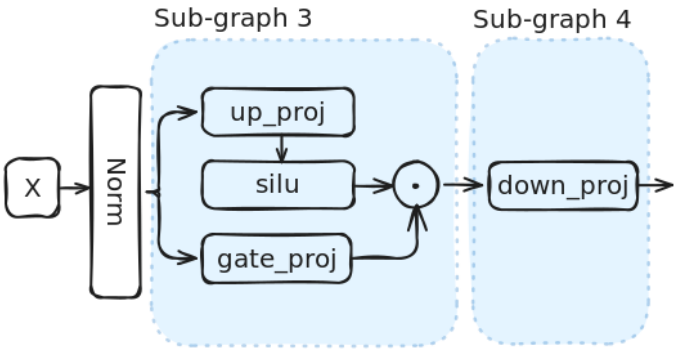
Algorithm 2 Partition model to sequential groups of layers

Input: A model $\mathcal{M}$
Output: Model partition $\{V_j\}_j$

Construct a DAG graph of the model computation $\{\text{Vertices}, \text{Edges}\}$
Add a start vertex start_vertex and denote the end vertex as end_vertex
Run Breadth-first search (BFS) and denote the longest path from start_vertex to vertex as $\text{path\_len}\,[\text{vertex}]$ for each $\text{vertex} \in \text{Vertices}$
$V = [], \text{vertex} = \text{start\_vertex}$
while $\text{vertex} \neq \text{end\_vertex}$ **do**
    Define set $V' = \{\}$
    $\text{cur\_len} = \text{path\_len}[\text{vertex}] + 1$
    Define the set $A = \text{next}[\text{vertex}]$
    while $|A| > 1$ **do**
        for $\text{vertex}' \in A$ **do**
            if $\text{path\_len}[\text{vertex}'] \leq \text{cur\_len}$ **then**
                $A.\text{pop}(\text{vertex}')$
                $V'.\text{push}(\text{vertex}')$
                $A.\text{push}(\text{next}[\text{vertex}'])$
            end if
        end for
        $\text{cur\_len} = \text{cur\_len} + 1$
    end while
    $\text{vertex} = A.\text{pop}()$
    $V'.\text{push}(\text{vertex})$
    Pop non-quantizable vertices/layers from V'
    if |V'|>0 **then**
        $V.\text{append}(V')$
    end if
end while
return V



(*a*) Attention block



(*b*) MLP block

Figure 5: Single-entry/single-exit sub-graphs (V_1–V_4) identified in one Llama-3 transformer layer. Dashed blue regions denote the latency-additive sub-graphs used in 2.2.1; residual adds are omitted for clarity. The final LM-head forms an additional single-layer sub-graph that is omitted from the illustration for brevity.

is $c^{\mathrm{TT}}_{j,p} = \sum_{l \in V_j} c^{\mathrm{TT}}_{\ell, Q_{j,p}}$. Figure 1 compares the aforementioned theoretical versus measured time gain. In order to simplify the comparison we fit the theoretical and empirical time gains, by constant scale and bias which we apply to the theoretical time gain. Even after optimal fitting, the theoretical proxy fails to capture the measured behavior, indicating that MAC counts do not reflect kernel fusion, memory traffic, or scheduler effects. Note that the IP is not affected by multiplying the performance metric by a scale factor and adding a bias to it.

C.0.2. MEMORY GAIN c^{M}

Memory savings arise exclusively from storing weights at lower precision. Intermediate tensors produced by BGEMM kernels can certainly be computed in FP8, but since they are not persistent they are stored in the stack memory. Quantizing them therefore improves latency but *does not* change the static model size. Under these observations, the additivity assumption across layers holds. Let $\delta_{M,f}$ be the byte reduction obtained when a single parameter element is stored in format f instead of BF16.

Since memory is additive across layers, we treat each primitive layer as its own group, i.e. $J = L$ and $V_j = \{\ell_j\}$. For a (trivial) group j and bit-width assignment $Q_{j,p}$ the memory gain is:

$$c_{\ell,f} \triangleq \begin{cases} C_\ell K_\ell \delta_{M,f} & \ell \in \mathcal{L}_{\text{lin}}, \\ 0 & \ell \in \mathcal{L}_{\text{BGEMM}}. \end{cases} \tag{11}$$

$$\mathbf{c}_{j,p} \triangleq \sum_{l=0}^{L_j-1} c_{\ell_{j,l},Q_{j,p}}. \tag{12}$$

These $\mathbf{c}_{j,p}$ values are used by the IP with the objective of maximizing memory gain c^{M}.

Appendix D. Additional experimental results

D.1. Experimental setup details

We evaluate MP quantization during the prefill stage of LLM inference using Intel's Gaudi 2 accelerator with $F = 2$ numerical formats (BF16 and FP8-E4M3: 4 exponent, 3 mantissa bits), the lm-evaluation-harness [5], and Neural Compressor [8]. Our evaluation spans four tasks (HellaSwag [21], LAMBADA [15], Winogrande [18], and PIQA [3]), averaging 5 iterations per configuration for time measurement, 20% of the samples in each dataset for calibration and sensitivity measurements, and the full datasets for final evaluation. Results are reported for 1B (with batch size 40) and 8B models (with batch size 10). Each evaluation is run over 10 different randomization seeds in which we perturb the scales before quantization in order to assess the accuracy statistics (mean and standard-deviation) and not just a single noisy realization of it.

Our proposed method combined with the different metrics yields the following strategies: IP-ET maximizes empirical time gain (Sec. 2.2.1), IP-TT maximizes theoretical time gain (Sec. C.0.1), and IP-M maximizes memory gain (Sec. C.0.2). Both IP-ET and IP-TT quantize linear and BGEMM layers , while IP-M quantizes only linear layers.

Each of the IP strategies is compared against two baseline strategies: *Random* which arbitrarily selects layers to quantize, resulting in scattered patterns and *Prefix* which quantizes layers in a sequential order. Both baseline strategies adhere to the loss MSE threshold. Figure 6 illustrates how each strategy selects layers for quantization given normalized-RMSE threshold τ. Our proposed IP-ET strategy produces *optimal* configurations which maximize the performance metric under the loss MSE constraint, leading to its superior accuracy-performance curve shown in subsequent results.

D.2. Layer-wise quantization patterns

Figure 6 illustrates the distinct layer-wise quantization patterns produced by our proposed IP-ET strategy in contrast to the Random and Prefix baseline methods.

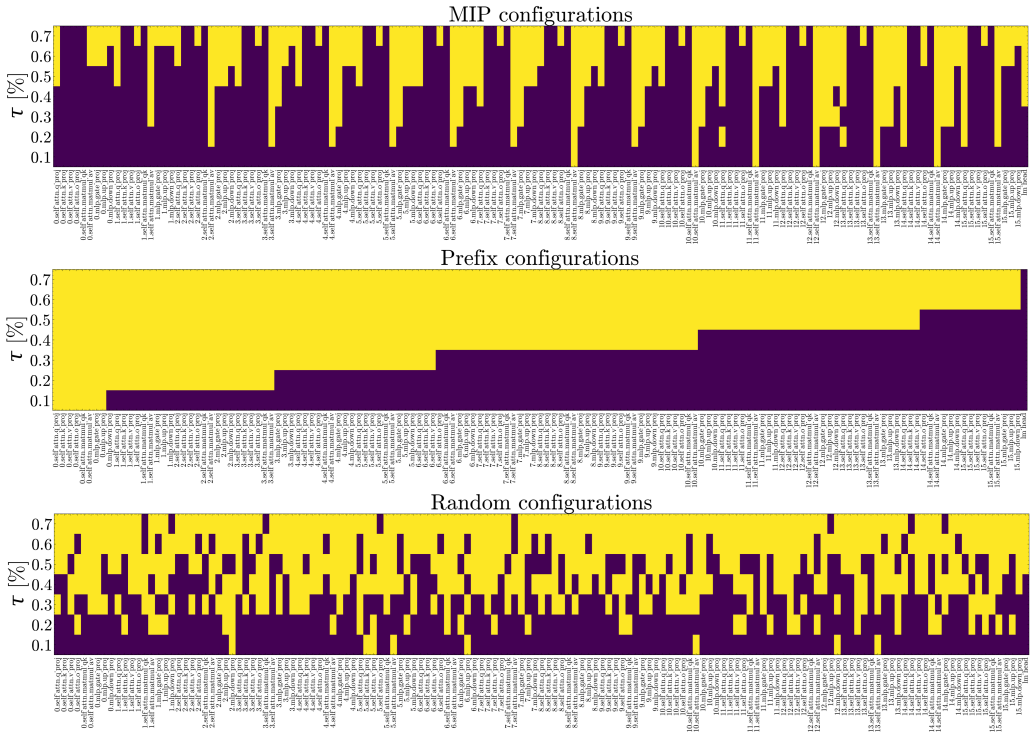


Figure 6: Layer-wise quantization patterns across MP configurations (rows) and model layers (columns) for IP-ET (top), Prefix (middle), and Random (bottom). Yellow: FP8, purple: BF16.

D.3. Time gain and loss MSE model validation

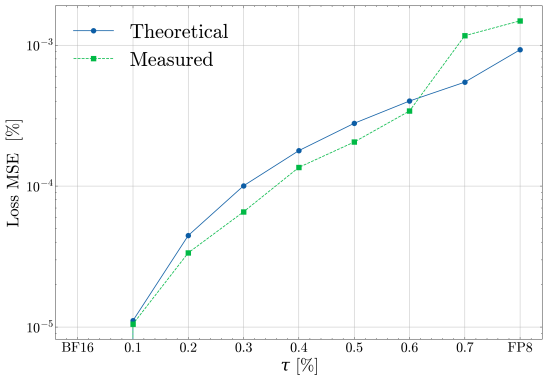
Considering the 1B model for MP configurations attained using the proposed method with $\tau \in \{0, 0.1\%, \ldots, 0.7\%\}$ in addition to the all-FP8 configuration, we depict the measured vs. theoretical empirical time gain (see (7)) and loss MSE, respectively, in Figure 7(*a*)subfigure and Figure 7(*b*)subfigure. Evidently our assumptions hold as the empirical time gain appears additive across groups and the theoretical loss MSE, assuming the per-layer model (26) and additivity (6), is a reliable estimate for the measured loss MSE.

D.4. Per task accuracy

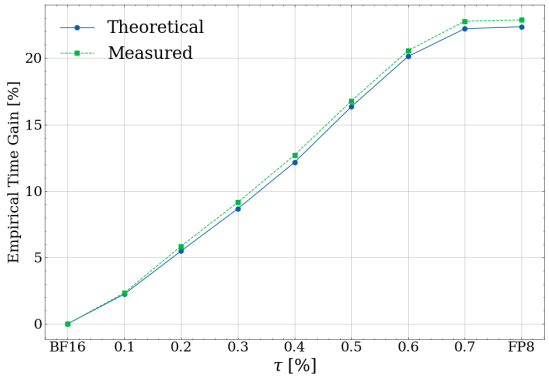
This subsection reports the per-task outcomes that complement the main text's task-average accuracy–TTFT curves. For each tolerance, we compare Random, Prefix, and our IP-ET (see 2.2.1), IP-TT (see C.0.1) and IP-M (see C.0.2) strategies against the full-BF16 baseline.

D.5. Per task: gained time based on measurements

Figure 8 reports for each individual task the accuracy difference (relatively to BF16) as a function of TTFT. The proposed IP-ET outperforms Random and Prefix strategies on most of the settings, particularly in the 1B model. For example, in HellaSwag using the 1B model (Figures 8(*a*)subfigure and 8(*b*)subfigure), IP-ET shows a significant advantage across all MP configurations.



(a) Loss MSE versus τ. Blue - theoretical loss MSE; Green - measured loss MSE using the chosen configurations by IP-ET



(b) Relative TTFT reduction versus τ. Blue - theoretical gain from the group aware IP-ET; Green - measured gain on Gaudi 2.

Figure 7: Empirical validation of the additivity assumption on different MP configurations

Model	Strategy	LAMBADA	LAMBADA	HellaSwag	Winogrande	PIQA	Tasks Avg.
		ppl diff ↓ [%]	*acc diff ↑ [%]*	*acc diff ↑ [%]*	*acc diff ↑ [%]*	*acc diff ↑ [%]*	*acc diff ↑ [%]*
IP-ET - Empirical Time Gain Optimization (both BGEMMs and linear layers)							
Llama-3.2-1B-Instruct	Random	4.938 ± 0.96	-2.107 ± 0.45	-1.077 ± 0.35	**0.077 ± 0.93**	-0.449 ± 0.34	-0.889 ± 0.52
	Prefix	5.986 ± 1.61	-2.206 ± 0.51	-1.586 ± 0.43	-0.271 ± 0.78	-0.615 ± 0.55	-1.170 ± 0.57
	IP-ET	**2.170 ± 0.32**	**-1.401 ± 0.26**	**-0.303 ± 0.14**	0.020 ± 0.59	**-0.169 ± 0.21**	**-0.463 ± 0.30**
Llama-3.1-8B-Instruct	Random	1.290 ± 0.15	-0.256 ± 0.25	-0.071 ± 0.08	0.085 ± 0.55	-0.399 ± 0.26	-0.160 ± 0.286
	Prefix	1.075 ± 0.15	-0.029 ± 0.24	-0.157 ± 0.12	-0.065 ± 0.66	-0.566 ± 0.30	-0.204 ± 0.33
	IP-ET	**0.922 ± 0.08**	**-0.229 ± 0.17**	**2.53e−4 ± 0.06**	**0.276 ± 0.41**	**-0.341 ± 0.17**	**-0.073 ± 0.20**
IP-TT - Theoretical Time Gain Optimization (both BGEMMs and linear layers)							
Llama-3.2-1B-Instruct	Random	4.938 ± 0.98	-2.107 ± 0.46	-1.077 ± 0.35	0.077 ± 0.85	-0.449 ± 0.33	-0.889 ± 0.49
	Prefix	5.986 ± 1.62	-2.206 ± 0.50	-1.586 ± 0.43	-0.271 ± 0.79	-0.615 ± 0.54	-1.170 ± 0.57
	IP-TT	**2.744 ± 0.43**	**-1.697 ± 0.41**	**-0.429 ± 0.14**	**0.096 ± 0.61**	**-0.102 ± 0.26**	**-0.533 ± 0.35**
Llama-3.1-8B-Instruct	Random	1.290 ± 0.15	-0.256 ± 0.25	-0.071 ± 0.08	0.085 ± 0.55	-0.399 ± 0.26	-0.160 ± 0.28
	Prefix	1.075 ± 0.15	-0.029 ± 0.24	-0.157 ± 0.12	-0.065 ± 0.67	-0.566 ± 0.31	-0.204 ± 0.33
	IP-TT	**1.002 ± 0.08**	**-0.178 ± 0.15**	**2.58e−4 ± 0.06**	**0.185 ± 0.43**	**-0.279 ± 0.19**	**-0.068 ± 0.21**
IP-M - Memory Gain Optimization (only linear layers)							
Llama-3.2-1B-Instruct	Random	4.151 ± 1.25	-1.886 ± 0.52	-0.980 ± 0.35	0.396 ± 0.87	-0.363 ± 0.30	-0.708 ± 0.51
	Prefix	4.483 ± 1.41	-1.693 ± 0.64	-1.361 ± 0.41	**0.435 ± 0.86**	-0.554 ± 0.44	-0.794 ± 0.59
	IP-M	**2.497 ± 0.34**	**-1.512 ± 0.33**	**-0.421 ± 0.15**	0.230 ± 0.67	**-0.075 ± 0.26**	**-0.445 ± 0.35**
Llama-3.1-8B-Instruct	Random	1.073 ± 0.10	-0.267 ± 0.21	-0.024 ± 0.08	0.180 ± 0.49	-0.321 ± 0.24	-0.108 ± 0.25
	Prefix	**0.567 ± 0.13**	**0.015 ± 0.18**	-0.092 ± 0.07	0.271 ± 0.47	-0.457 ± 0.22	-0.066 ± 0.23
	IP-M	0.981 ± 0.08	-0.160 ± 0.17	**0.012 ± 0.06**	**0.280 ± 0.37**	**-0.262 ± 0.16**	**-0.032 ± 0.19**

Table 1: Accuracy and perplexity difference across three optimization strategies, averaged over different quantization configurations from high-precision (BF16) to low-precision (FP8).

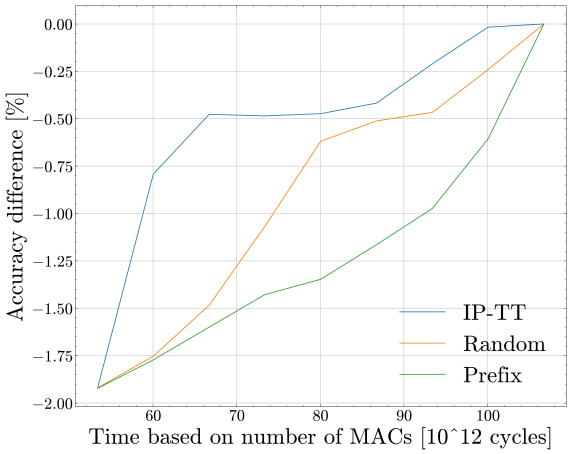
In LAMBADA using the 8B model (Figure 8(g)subfigure), Prefix yields higher accuracy, but IP-ET achieves lower perplexity (Figure 8(i)subfigure), highlighting that loss-based optimization (which is correlated to perplexity) doesn't necessarily translate to accuracy gains.

In Winogrande on the 1B model (Figure 8(e)subfigure) is particularly noisy, as reflected by large standard deviations in Table 1 which can explain the reason IP-ET shows no clear advantage.

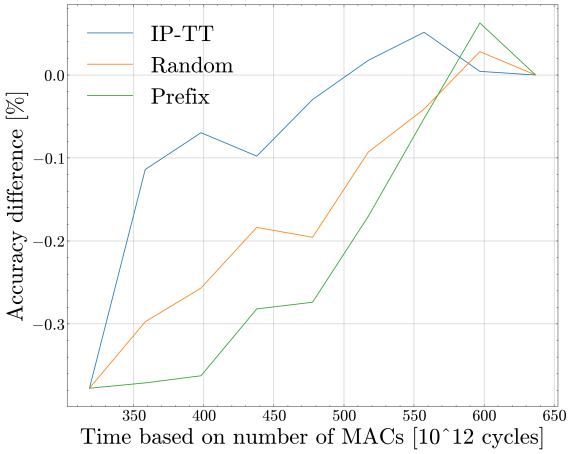
However, the rapid rise of the blue line indicates that IP-ET, achieves good accuracy by not quantizing only a few layers.

D.6. Gained time based on number of MACs

Figure 9 shows the tradeoff between accuracy and theoretical compute time, measured by MACs. The x-axis denotes the theoretical time gain based on the number of MAC operations as defined in Sec. C.0.1. While the y-axis reports accuracy difference relative to BF16, averaged across tasks. Our IP-TT (blue) consistently outperforms Random (orange) and Prefix (green) strategies, achieving a smaller accuracy degradation on both model sizes.

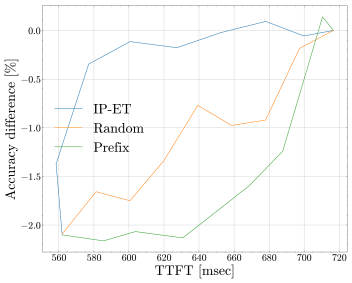


(*a*) 1B

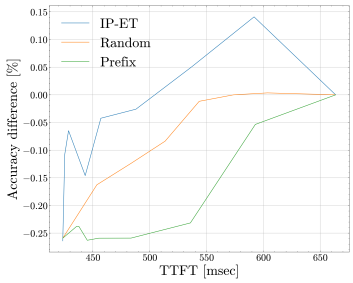


(*b*) 8B

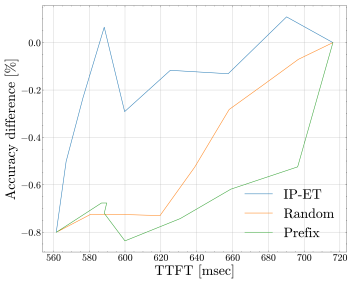
Figure 9: Average accuracy difference [%] vs. time based on number of MACs [cycles], across HellaSwag, LAMBADA, Winogrande, and PIQA. Comparing layer selection strategies for quantization (IP-TT, Random, Prefix).



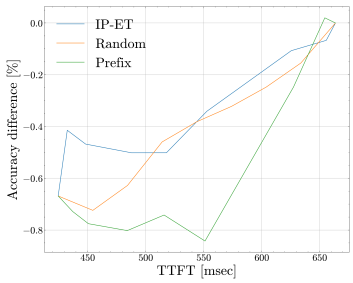
(*a*) 1B - HellaSwag



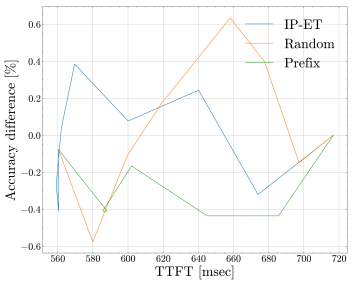
(*b*) 8B - HellaSwag



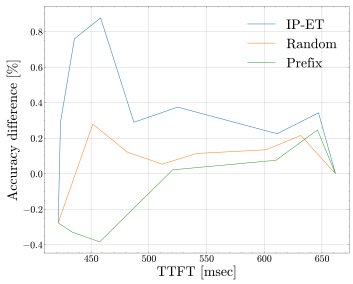
(*c*) 1B - PIQA



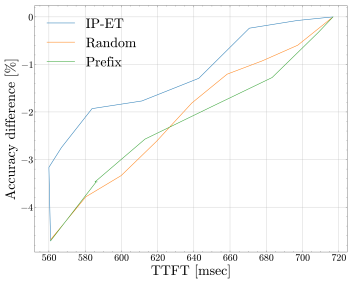
(*d*) 8B - PIQA



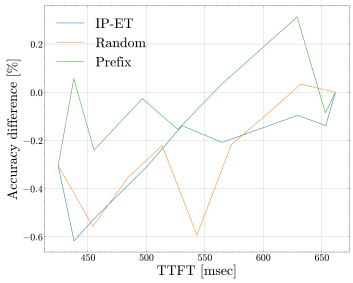
(*e*) 1B - Winogrande



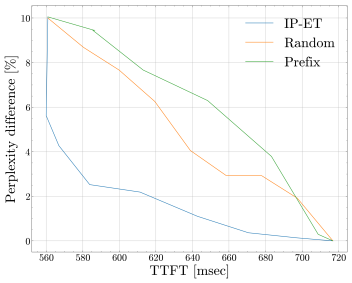
(*f*) 8B - Winogrande



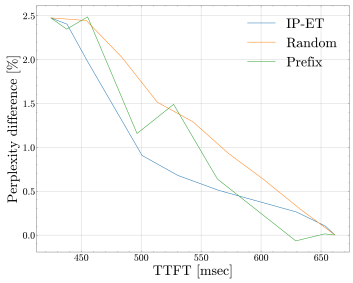
(*g*) 1B - LAMBADA accuracy



(*h*) 8B - LAMBADA accuracy



(*i*) 1B - LAMBADA perplexity



(*j*) 8B - LAMBADA perplexity

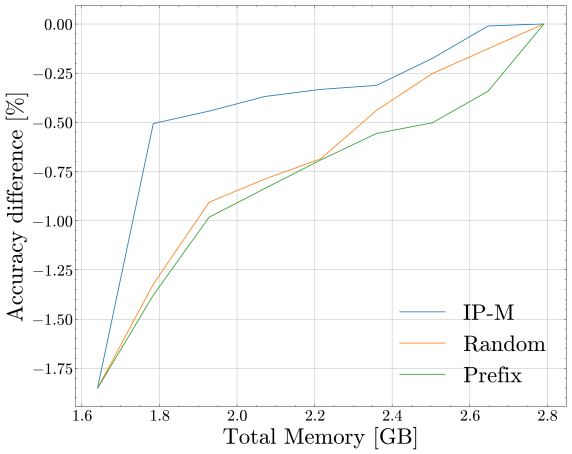
Figure 8: Per task accuracy/perplexity difference vs. TTFT. Comparing layer selection strategies for quantization (IP-ET, Random, Prefix

D.7. Gained memory

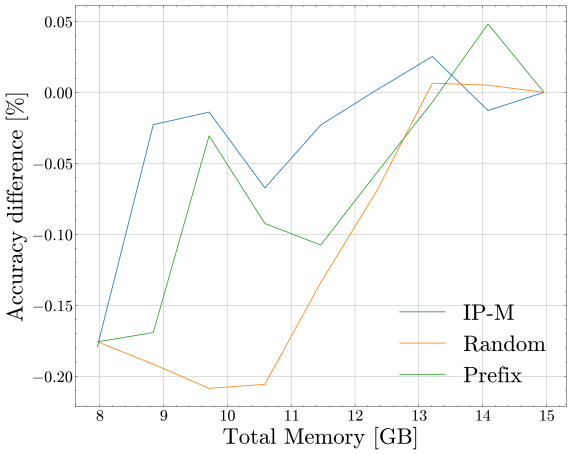
Figure 10 shows the tradeoff between accuracy and total model's memory. The x-axis values were calculated by subtracting BF16 model's total memory and the memory gain (defined in Sec. C.0.1) of each configuration. While the y-axis reports accuracy difference relative to BF16, averaged across tasks.

For the 1B model (Figure 10(*a*)subfigure), IP-TT (blue) consistently outperforms Random (orange) and Prefix (green) strategies, achieving lower accuracy loss for a given memory budget.

For the 8B model (Figure 10(*b*)subfigure), IP-TT also performs better than other strategies, though the margin is small; notably, the initial FP8 configuration results in less than 0.2% accuracy difference range, since only linear layers are quantized in these experiments. All 8B's MP configurations yield averaged accuracy difference close to zero.



(*a*) 1B



(*b*) 8B

Figure 10: Average accuracy difference [%] vs. total memory across HellaSwag, LAMBADA, Winogrande, and PIQA. Comparing layer selection strategies for quantization (IP-M, Random, Prefix).

Appendix E. Loss MSE metric

The model comprises a set of standard linear layers, denoted by $\mathcal{L}_{\text{lin}}$, and a set of BGEMM layers, denoted by $\mathcal{L}_{\text{BGEMM}}$. A linear layer $\ell \in \mathcal{L}_{\text{lin}}$ is defined by the operation:

$$\mathbf{Y}_\ell = \mathbf{X}_\ell \mathbf{W}_\ell^T + \mathbf{1}_{N\times 1}\mathbf{b}_\ell^T \tag{13}$$

where the dimensions of the matrices are as follows: $\mathbf{X}_\ell \in \mathbb{R}^{N\times C_\ell}$, $\mathbf{W}_\ell \in \mathbb{R}^{K_\ell \times C_\ell}$, $\mathbf{Y}_\ell \in \mathbb{R}^{N\times K_\ell}$, and $\mathbf{b}_\ell \in \mathbb{R}^{K_\ell \times 1}$. Here, N represents the number of input samples.

A BGEMM layer ℓ' is defined as:

$$\mathbf{Y}_{\ell'} = \mathbf{X}_{0,\ell'} \otimes \mathbf{X}_{1,\ell'} \tag{14}$$

where $\mathbf{X}_{0,\ell'}$ and $\mathbf{X}_{1,\ell'} \in \mathbb{R}^{N\times C_{\ell'}}$, and the output $\mathbf{Y}_{\ell'} \in \mathbb{R}^{N\times 1}$. The operator $\otimes$ is defined such that the n-th element of $\mathbf{Y}_{\ell'}$ is computed by $Y_{\ell',n,0} \triangleq \left(\mathbf{e}_n^T \mathbf{X}_{0,\ell'}\right)\left(\mathbf{e}_n^T \mathbf{X}_{1,\ell'}\right)^T$ with the selection vector $\mathbf{e}_n \in \mathbb{R}^{N\times 1}$ defined as $\mathbf{e}_n^T \triangleq [\mathbf{0}_{1\times n-1}, 1, \mathbf{0}_{1\times N-n}]$ and is used to extract the n-th row of a matrix.

Let $\mathbf{z}_\ell$ represent the extended input of layer ℓ, obtained by vectorizing the possibly quantized inputs. It is defined as:

$$\mathbf{z}_\ell \triangleq \begin{bmatrix} \left[\mathbf{x}_\ell^T, \mathbf{w}_\ell^T\right]^T & ;\ell \in \mathcal{L}_{\text{lin}} \\ \left[\mathbf{x}_{0,\ell}^T, \mathbf{x}_{1,\ell}^T\right]^T & ;\ell \in \mathcal{L}_{\text{BGEMM}} \end{bmatrix}. \tag{15}$$

Define the vectorized representations as:

$$\mathbf{x}_\ell \triangleq \text{vec}\left(\mathbf{X}_\ell\right) \tag{16a}$$
$$\mathbf{w}_\ell \triangleq \text{vec}\left(\mathbf{W}_\ell\right) \tag{16b}$$
$$\mathbf{x}_{0,\ell} \triangleq \text{vec}\left(\mathbf{X}_{0,\ell}\right) \tag{17a}$$
$$\mathbf{x}_{1,\ell} \triangleq \text{vec}\left(\mathbf{X}_{1,\ell}\right) \tag{17b}$$

with dimensions $\mathbf{x}_\ell \in \mathbb{R}^{NC_\ell\times 1}$, $\mathbf{w}_\ell \in \mathbb{R}^{C_\ell K_\ell \times 1}$ and $\mathbf{x}_{0,\ell}, \mathbf{x}_{1,\ell} \in \mathbb{R}^{NC_\ell\times 1}$.

We now respectively derive expressions for the noisy loss arising from model quantization and the quantized extended input:

$$\hat{g} \triangleq g + \tilde{g}, \tag{18}$$
$$\hat{\mathbf{z}}_\ell \triangleq \mathbf{z}_\ell + \tilde{\mathbf{z}}_\ell \tag{19}$$

where $\tilde{\mathbf{z}}_\ell$ is the quantization noise for layer $\ell \in \mathcal{L}_{\text{lin}} \bigcup \mathcal{L}_{\text{BGEMM}}$. And since f represents a floating-point format with m_f mantissa bits, the noise, modeled as a scaled Uniform random variable, and its respective variance are given by:

$$\tilde{z}_{\ell,k} \sim |z_{\ell,k}|\, 2^{-m_f} \text{U}[\pm 1/2] \tag{20}$$
$$\text{E}\left[\tilde{z}_{\ell,k}^2\right] = |z_{\ell,k}|^2\, \alpha_f \tag{21}$$

for $k \in [0, |\mathbf{z}_\ell|]$, where $\text{U}[\pm 1/2]$ is a Uniform random distribution over $[-0.5, 0.5]$, and $|\mathbf{z}_\ell|$ denotes the number of elements in $\mathbf{z}_\ell$ with $\alpha_f \triangleq \frac{2^{-2m_f}}{12}$ for $f \in [0, F-1]$.

Considering the r-th input sample and (18), the noisy loss is expressed as $\hat{g}^r \triangleq g^r + \tilde{g}^r$. Assuming that the quantization noise is small compared to the full-precision values, a first-order Taylor series approximation yields:

$$\hat{g}^r \approx g^r + \sum_{\ell \in \mathcal{L}_{\text{lin}} \bigcup \mathcal{L}_{\text{BGEMM}}} (\tilde{\mathbf{z}}_\ell^r)^T \, \dot{\mathbf{z}}_\ell^r \quad (22)$$

$$\dot{\mathbf{z}}_\ell^r \triangleq \left. \frac{\partial g}{\partial \mathbf{z}_\ell} \right|_{\mathbf{z}_\ell^r} \quad (23)$$

where $\dot{\mathbf{z}}_\ell^r$ is the gradient of the loss with respect to the extended input $\mathbf{z}_\ell$ of sample r.

The *sensitivity* of layer ℓ and its corresponding loss MSE for numerical format f and sample r are respectively defined as:

$$s_\ell^r \triangleq \|\mathbf{z}_\ell^r \odot \dot{\mathbf{z}}_\ell^r\|^2 \quad (24)$$

$$d_{\ell,f}^{\text{layer},r} \triangleq s_\ell^r \alpha_f. \quad (25)$$

The variance of the contributions to the loss MSE which correspond to the elements of the extended input are added in super-position, and with $\odot$ denoting the element-wise product. Averaging over R input samples yields the *average* sensitivity and corresponding loss MSE component:

$$s_\ell \triangleq \frac{1}{R} \sum_r s_\ell^r \quad (26)$$

$$d_{\ell,f}^{\text{layer}} \triangleq s_\ell \alpha_f. \quad (27)$$

For the p-th quantization configuration, and under the assumption that quantization noise is statistically independent across layers, the loss MSE component which corresponds to the j-th group is given by the sum of per-layer contributions:

$$d_{j,p} \triangleq \sum_{l=0}^{L_j - 1} s_{\ell_{j,l}} \alpha_{Q_{j,lp}}. \quad (28)$$