Bespoke LUT: Non-Linear Approximation for Integer-only Transformer Inference on NPUs

Seonyoung Kim, Jooeun Kim, Hayoung Yun, Meejeong Park,
Sangjeong Lee, Hanjoo Cho, *Member, IEEE,* and Heonjae Ha

***Abstract*—Efficient deployment of Transformer models on neural processing units (NPUs) is fundamentally limited by the underutilization of hardware resources caused by the interleaved structure of linear and non-linear layers. Linear operations are mapped to matrix arithmetic units (MAUs), which are highly optimized for fast and energy-efficient matrix multiplications. Non-linear operations are mapped to auxiliary units such as digital signal processors (DSPs). DSPs often present a performance bottleneck, as operators are decomposed into multiple instructions. To address this, we propose Bespoke LUT, a hardware-efficient integer-only approximation solution for non-linear operations on NPUs. Bespoke LUT employs several key innovations. First, it customizes lookup tables (LUTs) for each layer to account for variations in input ranges across layers without requiring training. Moreover, a Dual-ranged LUT is used to improve precision for small inputs in large language models (LLMs), ensuring more accurate results. Finally, Bespoke LUT implements an efficient 8-bit indexing scheme, eliminating the need for an additional comparator module and simplifying hardware integration. Experiments on Transformer models demonstrate up to $3.3\times$ speedup with less than 6% accuracy degradation, highlighting its effectiveness for efficient NPU deployment.**

***Index Terms*—Neural Processing Units, Look-up table, Non-linear function, Integer-only inference, Post-training quantization, Transformer**

I. Introduction

With the increasing demand for on-device artificial intelligence (AI) applications, efficient deployment of AI models on NPUs has become essential. Achieving this efficiency requires careful alignment between model characteristics and hardware architecture [1]. NPUs specialize in maximizing throughput and energy efficiency through highly parallelized compute units: MAUs, optimized for high-throughput linear operations such as matrix multiplications, and auxiliary compute units such as DSPs equipped to handle more complex non-linear functions [2], [3]. As AI models are typically composed of alternating linear and non-linear operations, efficient deployment on NPUs requires careful scheduling of heterogeneous computing resources to maximize hardware utilization [4], [5].

Recently, rapid advancement of Transformer-based AI models has driven breakthroughs across various domains, including image processing [6], [7] and natural language understanding [8], [9]. While so, they increasingly utilize complex and novel non-linear functions such as Softmax, GeLU, and Layer Normalization (LayerNorm).

Executing these non-linear operations often induces high latency because such operations must be decomposed into multiple DSP instructions, significantly increasing latency. Additionally, scheduling between MAUs and the other compute units introduces poor hardware utilization [10] and further overhead due to frequent data movement [11]. Consequently, non-linear operations often emerge as a major computation bottleneck in Transformer architectures [12], [13]. Profiling in Qualcomm's Snapdragon 8 NPU [14], [15] and our in-house NPU deployed in commercial TVs[1] shows that non-linear operations can account for up to 75% of total inference latency (Figure 1), highlighting them as a major performance bottleneck. This motivates the need for accelerating non-linear operations directly on MAUs to reduce latency and improve hardware utilization.

To maximize hardware efficiency, many edge NPUs prioritize INT8 or INT16 arithmetic, with limited support for floating-point operations [16]–[21]. Integer-arithmetic offers benefits such as reduced hardware complexity, lower power consumption, and faster computation compared to floating-point operations, making it particularly attractive for resource-constrained on-device AI applications [22], [23]. However, certain non-linear operations in Transformer architectures exhibit high sensitivity to integer quantization, which results in significant accuracy degradation when executed in integers [24]. Therefore, to maintain model quality, non-linear functions are typically computed in floating-point, while linear operations continue to benefit from integer acceleration [25], [26]. Interleaving linear and non-linear operations between integer and floating-point formats incurs frequent conversions, leading to additional latency, quantization errors, and inefficiencies in data flow. Therefore, enabling accurate integer execution of non-linear operations is beneficial for maximizing efficiency in Transformer inference on NPUs.

To this end, many approximation approaches were proposed to fully utilize MAUs by using integer operations only [27], [28]. Recently, LUTs have been employed to approximate non-linear operations on NPUs, replacing expensive transcendental functions such as exponentials and square roots with simple table lookups [29]–[31]. LUT-based methods offer constant-time execution and are well-suited for integer-based implementation. This dual advantage of low latency and design

S. Kim and J. Kim contributed equally to this work. This work was performed while J. Kim and H. Cho were with Samsung Research. (Corresponding author: Heonjae Ha.) S. Kim, H. Yun, M. Park, S. Lee, and H. Ha are with Samsung Research, Seoul, South Korea (e-mail: {sy007.kim, hayoung.yun, mia.park, sangjeong.lee, heonjae.ha}@samsung.com). J. Kim is now with Naver Corporation, Seongnam, South Korea (e-mail: jooeun.kim@navercorp.com). H. Cho is now with 42dot, Seoul, South Korea (e-mail: hanjoo.cho@42dot.ai).

[1]4 TOPS MAU and 32 GOPS DSP

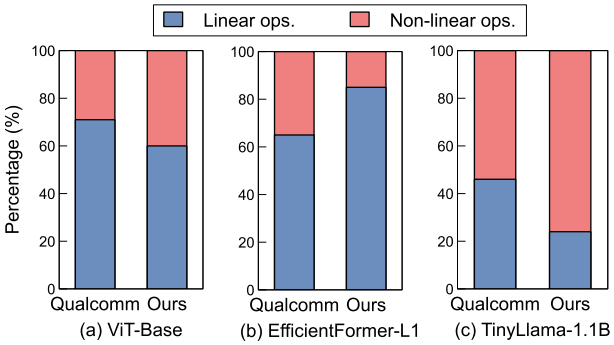

Fig. 1: Latency breakdown on our NPU and Qualcomm.

flexibility makes LUT-based methods particularly attractive for accelerating non-linear operations in modern Transformer models.

Despite these advantages, LUTs' fixed breakpoint allocation imposes approximation error [30]. While prior LUT approaches attempt to address these challenges, they continue to exhibit structural limitations. NN-LUT [30] trains one-hidden-layer ReLU network to generate breakpoints and piece-wise approximation LUT parameters for its non-uniform intervals based on the output values of the target function as ground truth. This training requires extensive parameter search and additional comparator modules for LUT indexing. Auto-LUT [31] automates the training process but still inherits the same comparator-based design, which relies on additional comparator logic and complicates deployment. Such architectures complicate the design and limit the scalability on edge NPUs.

To enable efficient execution of non-linear functions on NPUs, we introduce *Bespoke LUT*, a hardware-friendly lookup table architecture and configuration algorithms designed for simplicity, low precision, and deployability. Our design addresses three key aspects that enhance the practicality of LUT-based approximations in real-world systems.

First, Bespoke LUT employs a lightweight uniform indexing scheme, illustrated in Figure 2, that partitions inputs into an index and an interpolation weight. More details on this simple architecture are elaborated in Section V-A. This simplifies the hardware design by removing the associated comparison logic [5], making it a versatile solution for broad function coverage. In contrast to prior methods [29]–[31], which rely on non-uniform indexing or training-based configuration, Bespoke LUT achieves competitive approximation accuracy with minimal hardware complexity, avoiding reliance on floating-point units. Moreover, we observe that a properly configured uniform-ranged LUT is sufficient to minimize approximation error, while preserving hardware efficiency [32].

Second, Bespoke LUT addresses the substantial variation in input distributions observed across layers [33]. Figure 3 illustrates the case of RSQRT operations in DETR-R50, where the input distributions vary significantly between layers. Such variation makes it difficult for a single global LUT to maintain high approximation accuracy across the network [30]. To address this, Bespoke LUT proposes a calibration data-driven per-layer configuration algorithm during post-training quantization (PTQ) [22], which notably reduces quantization error in non-linear operations. Section V-B provides a comprehensive description on this algorithm, which enables a complete and low precision integer-only inference with affordable accuracy loss while requiring no retraining and introducing negligible runtime and storage overhead.

Third, Bespoke LUT mitigates numerical instability arising from asymptotic growth of non-linear operations. In particular, functions such as Reciprocal and RSQRT exhibit rapid output divergence for inputs near zero. To address this, Bespoke LUT introduces a small auxiliary high-resolution LUT specifically designed for small input ranges, and applies it to sensitive intervals using a selective algorithm as detailed in Section V-C. Unlike previous LUT designs that allocate disproportionately large value ranges to asymptotic regions that result in significant approximation errors, Bespoke LUT preserves precision while maintaining a minimal memory footprint.

In summary, our main contributions are as follows:

- We discover that a lightweight, uniform LUT-based approach for non-linear approximation is sufficient to achieve high generalizability and hardware-friendly deployment.
- We introduce a calibration data-driven per-layer LUT configuration algorithm that accurately captures layer-wise input variation, enabling integer-only training-free deployment without runtime cost or accuracy loss.
- We propose a Dual-ranged LUT that selectively focuses resolution on error-sensitive regions, improving approximation quality near asymptotic boundaries with minimal overhead.
- We demonstrate that Bespoke LUT is deployable on real-world NPUs and achieves up to 3.3× inference speedup on Transformer models in both vision and language domains.

II. Background

A. Uniform Post-Training Quantization and Calibration

Uniform asymmetric quantization maps a real-valued vector, x, to the integer grid of which range is $[q_{\min}, q_{\max}]$. The range is $[0, 2^b - 1]$ for b-bit unsigned integer and $[-2^{b-1}, 2^{b-1} - 1]$ for signed integer. The quantized value, x_{int}, is described using two quantization parameters, the scaling factor, s, and the zero point, z, as follows:

$$x_{\text{int}} = \text{clamp}(\left\lfloor \frac{x}{s} \right\rceil + z, q_{\min}, q_{\max}), \tag{1}$$

with clamp defined as

$$\text{clamp}(x, a, b) = \begin{cases} a & \text{if } x < a \\ x & \text{if } a \leq x \leq b \\ b & \text{if } x > b \end{cases}, \tag{2}$$

where $\lfloor . \rceil$ denotes the nearest round. The quantization parameters are defined as follows:

$$s = \frac{x_{\max} - x_{\min}}{2^b} \quad \text{and} \quad z = \left\lfloor \frac{x_{\min}}{s} \right\rceil, \tag{3}$$

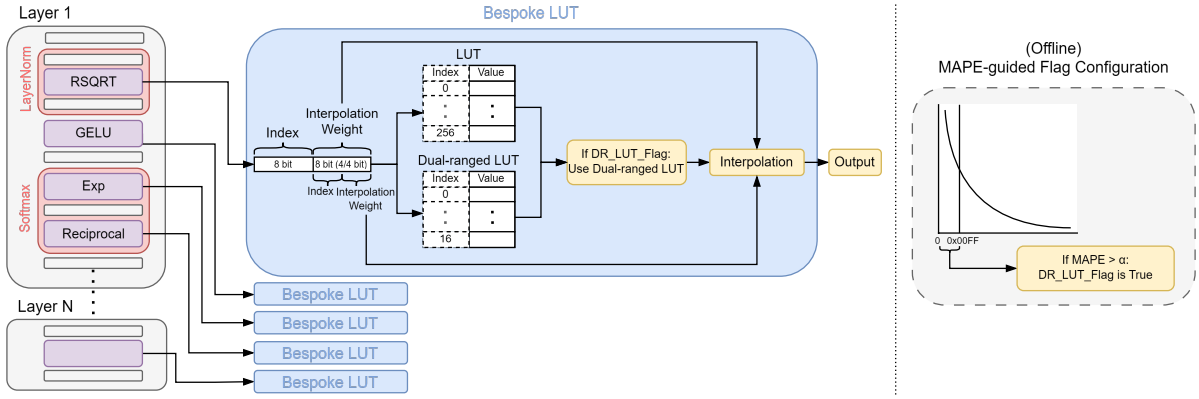

Fig. 2: Overview of the proposed Bespoke LUT. The dotted index entries indicate virtual positions that do not occupy physical hardware space, enabling efficient range coverage with minimal area overhead.

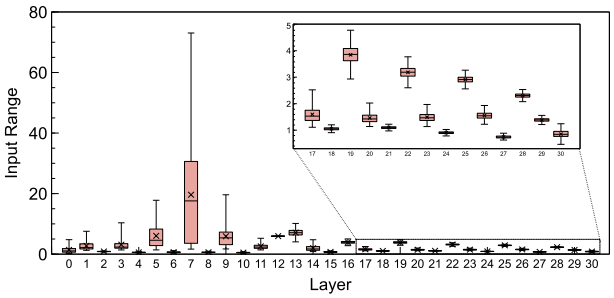

Fig. 3: Input distribution of RSQRT operations across layers in DETR-R50.

where $x_{\max}$ and $x_{\min}$ mean the maximum and minimum of x, respectively. While the range of weights is constant, the range of activations varies according to inputs. Hence, in PTQ, sample inputs (called calibration data) are fed into the target network to measure the activation range of each layer. The activation range, $x_{\max}$ to $x_{\min}$, is statistically estimated using methods such as absolute min/max tracking or exponential moving average (EMA) [34]. This process is called *calibration*.

We adopt uniform quantization to maximize hardware efficiency and reduce memory footprint, both critical for deployment in resource-constrained environments [35], [36]. Additionally, our experiments assume a PTQ setting, where the simplicity and consistent behavior of uniform quantization are particularly advantageous.

B. Non-linear components in Transformer models

A typical Transformer block includes non-linear operations such as GELU, Softmax, and LayerNorm.

$$\text{GELU}(x) := \frac{x}{2}[1 + \text{erf}(\frac{x}{\sqrt{2}})]; \ \text{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x \exp^{-t^2} dt; \quad (4)$$

$$\text{Softmax}(x_i) := \exp(x_i - \max(x)) \cdot \frac{1}{\sum(\exp(x_i + \max(x)))}; \quad (5)$$

$$\text{LayerNorm}(x_i) := (x_i - \bar{x}) \cdot \frac{1}{\sqrt{\text{Var}(x)}}, \quad \text{where} \quad \bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \quad \text{and} \quad \text{Var}(x) := \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}. \quad (6)$$

When approximating these non-linear operations using integer-arithmetics in MAU, the operations can be broken into a set of linear and non-linear components, as shown in Figure 4. Further details on the mechanism of I-BERT [28] and I-ViT [27] in Figure are provided in Section III-B.

III. Related Works

A. LUT-based Approaches

Modern deep learning accelerators have incorporated LUTs to approximate non-linear functions to reduce computational complexity [37]–[39]. This approach discretizes the function's input domain and stores representative output values for efficient retrieval. A critical design choice is whether to partition the input space into uniform or non-uniform intervals. Uniform intervals simplify address computation but may lead to high approximation errors in regions where the function changes rapidly. Non-uniform intervals can reduce such errors by exploiting finer granularity where non-linearity is extreme, at the cost of increased indexing complexity. Thus, LUT design involves a trade-off between approximation accuracy and hardware efficiency [32], [40].

NN-LUT [30] approximates non-linear functions by training a one-hidden-layer perceptron with ReLU activations, representing the function as a set of piecewise linear segments. These segments are encoded in a non-uniform LUT containing their coefficients. NN-LUT utilizes a "universal LUT", which we define as a single lookup table shared across all layers for identical non-linear operations. While this approach simplifies deployment, it may cause accuracy degradation due to variations in dynamic ranges across different layers. Auto-LUT [31] extends this framework by automatically determining the optimal number of segments and bit-widths for each function to balance accuracy and hardware cost.

In parallel, FIGLUT [41] has been proposed to compress model weights by learning LUT-based centroids. However, it mainly targets weights and retains floating-point activations, limiting full integer acceleration. To address activations, Range-Invariant Approximation for BERT Fine-Tuning [42] introduced a range-scaling method that achieves single-entry LUT approximation with significant area savings, though

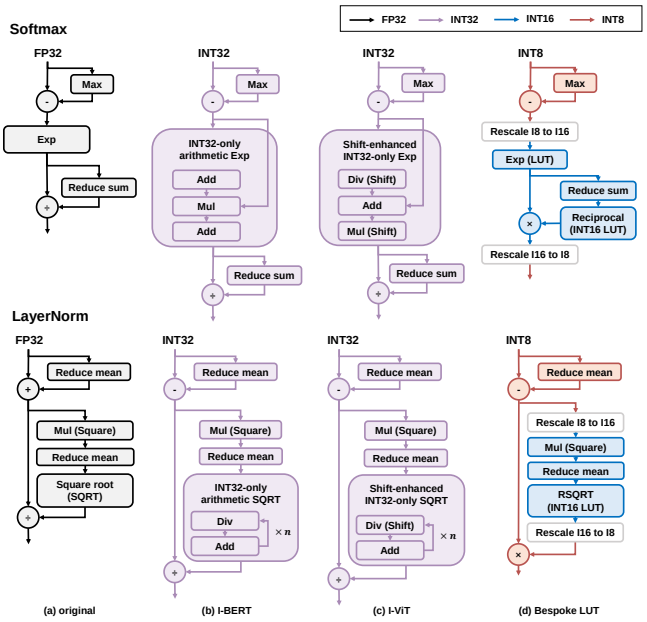

Fig. 4: Decomposition of non-linear operations.

its design is specialized for BERT architectures. Separately, CompressedLUT [43] focuses on lossless compression of generic LUTs to optimize memory, rather than direct function approximation.

Building upon these insights, we aim to accelerate non-linear operations using only 8-bit and 16-bit representations with hardware-friendly LUT designs, proposing a complete training-free approximation scheme that achieves high accuracy and full integer-only execution without fine-tuning.

B. Polynomial Approximation-based Approaches

Prior works have explored replacing non-linear functions with polynomials to enable integer-only inference using only the MAU. While our method focuses on LUT-based approximation, we include MAU-based solutions as baselines in the evaluation. These executions avoid communication between MAU and DSP.

I-BERT [28] facilitates the entire inference of Transformer models through integer-only arithmetic by substituting non-linear components, such as GELU, Softmax, and LayerNorm, with polynomial approximations. For instance, it adapts the SQRT in LayerNorm to a well-known integer calculation [44]. I-ViT [27] improves the efficiency of integer-only inference by adopting hardware-friendly bit-shift operations in an integer iterative manner, using dyadic arithmetic pipeline. Such approximation methods enable efficient, integer-only inference without modifying the model architecture, improving computational efficiency, memory usage, and compatibility.

However, the drop-in replacement of the non-linear components with polynomial approximations introduces significant approximation errors, making it hard to leverage pre-trained models without adjustments. Both I-BERT and I-ViT conducted quantization-aware fine-tuning to recover the accuracy degradation. Furthermore, they demand the manual design of polynomial approximations for each non-linear function,

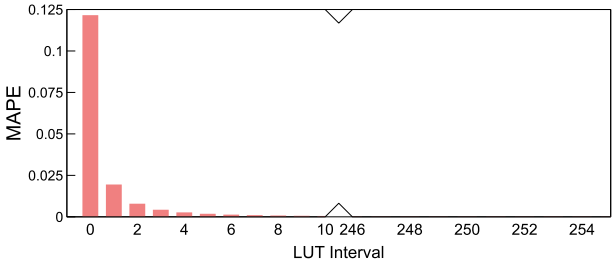

Fig. 5: MAPE measured across all LUT intervals.

involving extensive tuning to find optimal forms and parameters. This manual overhead restricts scalability, particularly as newer activations like SiLU, RMSNorm, GeGLU, and SwiGLU emerge in recent models such as LLaMA [45], PaLM [46], and T5 [47].

IV. Observation

Transformer models running on integer-only NPUs suffer from significant approximation errors, particularly in non-linear operations. Through profiling across vision and language models, we identified two key bottlenecks in this process: layer-wise input variation and asymptotic divergence of non-linear functions. These insights form the basis of our Bespoke LUT design, enabling accurate and hardware-friendly approximation under tight NPU constraints.

A. Layer-wise Input Range Distribution

Transformer models apply the same non-linear operations repeatedly across multiple layers, yet each layer exhibits its own input value distributions due to differences in computation history. To understand how this affects approximation accuracy in integer-only NPUs, we analyzed the input value distributions of RSQRT across layers in DETR-R50 [7].

As shown in Figure 3, the input ranges of RSQRT vary significantly across layers. Some layers operate within narrow and dense ranges, while others span broader intervals. However, current LUT-based methods typically construct a single LUT per non-linear operation, shared across all layers, assuming a fixed global input range. This approach forces the LUT to uniformly cover the union of all per-layer ranges, leading to wide discretization intervals and reduced precision—especially for layers with narrow inputs.

This suggests that the mismatch between LUT coverage and actual input distributions can become a source of quantization error, even if the LUT itself is well-configured. It also highlights the potential benefit of aligning LUT granularity with layer-specific distributions. We explore this design direction in Section V-B.

B. Approximation Bottleneck in Asymptotic Regions of Non-Linear Functions

In analyzing Transformer-based LLMs such as TinyLlama [8], we observed that the approximation error of LUT-based integer computation is not uniformly distributed across the input space. Instead, a disproportionate concentration of error

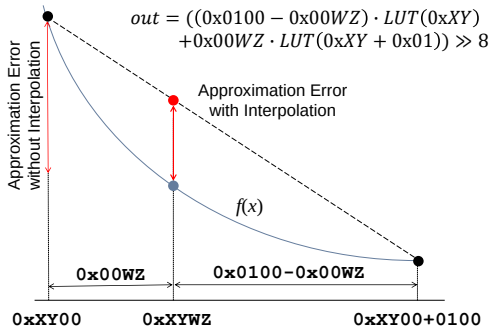

Fig. 6: Interpolation of a 16-bit input.

occurs in the asymptotic regions of non-linear functions such as Reciprocal and RSQRT, particularly near input values close to zero.

These functions exhibit rapid growth as the input approaches zero, causing LUT entries corresponding to small inputs to cover extremely wide output ranges. This results in substantial quantization errors, as the uniform discretization of input intervals is fundamentally mismatched to the steep non-linearity in this region.

Figure 5 illustrates this phenomenon: in a uniformly indexed LUT, the first few entries—corresponding to small positive inputs—incur the highest approximation errors. We used the mean absolute percentage error (MAPE) as an empirical measure of LUT error across input intervals. These errors are not evenly distributed across the input domain, but are sharply concentrated in the earliest LUT intervals, leading to both functional inaccuracies and reduced model-level performance.

This profiling reveals a key bottleneck in deploying integer-only LUT approximations for non-linear functions. It motivates the need for a finer-grained approximation strategy that can allocate higher resolution to these error-sensitive regions, without sacrificing hardware simplicity. We introduce our Dual-ranged LUT design to address this bottleneck in Section V-C.

V. Proposed Method: Bespoke LUT

A. 8-bit Index-based LUT

To approximate non-linear functions efficiently on resource constrained edge NPUs, Bespoke LUT adopts a uniform indexing scheme with 8-bit precision and linear interpolation. This design achieves a balance between lookup granularity, memory footprint, and hardware simplicity, while maintaining high approximation accuracy.

While LUTs with different bit-widths—such as 6 to 13 bits—can be constructed to trade off precision and memory usage, we adopt an 8-bit indexing strategy as a practical sweet spot. It provides sufficient resolution for most activation distributions observed in Transformer models, enables fast $O(1)$ access, and avoids excessive memory overhead associated with wider LUTs. We empirically validate this design choice in Section VI-E1.

In this structure, the upper 8 bits of a 16-bit quantized input serve as the LUT index, and the lower 8 bits are used as interpolation weights [48]. This approach yields constant-time access and requires no comparator modules or control branching, making it well suited for integration into integer-only MAU pipelines. Figure 6 illustrates this process: for a 16-bit integer input x, two neighboring entries l_{low} and l_{high} are selected as follows:

$$l_{\text{low}} = \text{LUT}(x \gg 8), \tag{7a}$$

$$l_{\text{high}} = \text{LUT}\Big((x \gg 8) + 1\Big). \tag{7b}$$

The final output y is computed via a weighted sum:

$$y = \frac{(\texttt{0x0100} - w) \cdot l_{\text{low}} + w \cdot l_{\text{high}}}{\texttt{0x0100}}, \tag{8}$$
$$\text{where } w = x \ \&\ \texttt{0x00FF}.$$

This simple indexing and interpolation mechanism allows a fast, accurate function approximation with a minimal 257-entry table and no floating-point operations or comparators. The compact structure also supports parallel LUT access and predictable latency, facilitating scalable deployment across various model sizes and hardware. Moreover, this function-independent architecture enables it to flexibly support a wide range of non-linear functions. Its modularity allows seamless scaling to different model sizes and operations without requiring function-specific customization or retraining.

B. Per-Layer LUT Configuration via Calibration

Transformer models contain multiple non-linear operations that are repeated across layers, but each layer exhibits distinct input value distributions due to the combination of learned parameter shifts and the evolving nature of semantic representations [49], [50]. As discussed in Section IV-A, these input ranges can differ significantly across layers, and using a single LUT to approximate the same operation across all layers often leads to approximation errors, particularly in layers with narrow or skewed input distributions.

To address this, Bespoke LUT constructs a separate lookup table for each instance of a non-linear operation, calibrated to the dynamic range observed in that specific layer. This approach allows each LUT to focus its approximation capacity on the actual range used by the layer, instead of stretching to cover a union of all possible input values across the model.

The per-layer LUTs are generated using statistics obtained during the calibration phase of PTQ. In this process, representative input data are passed through the model, and each layer's minimum and maximum input values are recorded using EMA [34]. These statistics are essential for computing the quantization parameters, and thus are necessarily collected during the calibration process. In other words, no additional computation overhead is incurred to generate the LUTs. Using these calibrated ranges, each LUT is configured to match the effective input domain of its layer.

Because Bespoke LUT uses uniform 8-bit indexing, this per-layer configuration does not introduce control complexity or comparator overhead. Each LUT is simply generated with a corresponding input range, but the indexing logic remains identical. As a result, the hardware simplicity of a global LUT

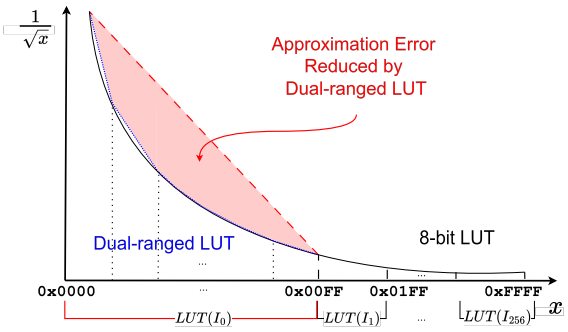

Fig. 7: Dual-ranged LUT to mitigate approximation errors in the first LUT interval with asymptotic growth.

is preserved, improving approximation precision across the model at the same time.

This strategy also scales efficiently to large models with many non-linear layers, since the memory overhead per LUT is minimal (257 entries), and all required configuration data are extracted from existing PTQ workflows. Section VI-E2 demonstrates that this per-layer strategy achieves consistent accuracy improvements over global LUT baselines, especially in models with high layerwise variability like DETR [7] and TinyLlama [8].

C. Dual-ranged LUT for Handling Asymptotic Approximation Errors

While our per-layer 8-bit LUTs are generally sufficient to approximate non-linear functions in most Transformer models, we observed a critical bottleneck in layers involving asymptotic behavior such as RSQRT and Reciprocal. These functions exhibit steep gradients near zero, where small input values lead to large output variations. As illustrated in Figure 7, the first interval of the LUT often fails to preserve sufficient precision, particularly with language models of compact size such as TinyLlama [8].

To identify layers that suffer from such issues, we use the MAPE as a decision metric. By leveraging calibration-time statistics, it captures realistic, layer-specific input behavior without requiring any additional training.

We first obtain the layer's quantization parameters — the minimum and maximum input values $(x_{\min}, x_{\max})$, the scale s, and zero-point z — during PTQ. From these values, we enumerate all integer inputs within the valid range and compute the dequantized values using:

$$x_{\text{int}} \in \left[\left\lfloor \frac{x_{\min}}{s} \right\rceil + z,\ \left\lfloor \frac{x_{\max}}{s} \right\rceil + z\right] \quad \text{and} \quad x = s \cdot (x_{\text{int}} - z). \tag{9}$$

We then pass these values through the original non-linear function to obtain the ground truth outputs y_i. We compare them with the LUT-approximated outputs $\hat{y}_i$ to compute the average MAPE for each LUT interval. To evaluate the approximation error of each LUT interval, we group input samples by the LUT entry they map to— specifically, inputs with the same upper 8 bits (i.e., index j) are assigned to the same interval. The MAPE for the j-th LUT entry is defined as:

$$\text{MAPE}_j = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{10}$$

where $\mathcal{I}_j$ denotes the set of inputs that map to the j-th LUT entry. This entire process is performed offline during calibration and introduces no inference time overhead.

If the first LUT interval of a given layer exhibits $MAPE_0 > 0.1$, we attach a small auxiliary LUT, *Dual-ranged LUT*, to that layer. This 4-bit LUT refines approximation within a 16-bit integer test data $x \in [\texttt{0x0000}, \texttt{0x00FF}]$ by subdividing it into 16 regions. The upper 4 bits of x (i.e., x & `0x00F0`) act as the index, and the lower 4 bits (i.e., x & `0x000F`) serve as interpolation weights:

$$d_{\text{low}} = \text{DR\_LUT}(x \gg 4), \tag{11a}$$

$$d_{\text{high}} = \text{DR\_LUT}\Big((x \gg 4) + 1\Big). \tag{11b}$$

The output $\hat{y}$ is computed similarly using 4-bit interpolation:

$$\hat{y} = \frac{(\texttt{0x10} - w') \cdot d_{\text{low}} + w' \cdot d_{\text{high}}}{\texttt{0x10}}, \quad \text{where } w' = x \,\&\, \texttt{0x000F}. \tag{12}$$

The Dual-ranged LUT (DR_LUT) contains only 17 entries (16 + 1) and requires just 34 bytes of memory. Despite this small overhead, it significantly improves approximation in error-sensitive regions. In our experiments on TinyLlama [8], SmolLM [9], and OPT [51], only the former two models triggered this refinement, demonstrating the adaptive and efficient nature of our approach.

Notably, within regions where the Dual-ranged LUT is applied, it achieves an effective resolution equivalent to that of a 12-bit LUT. However, instead of requiring a full 12-bit LUT with 4097 entries, our method achieves similar precision by adding only a small 4-bit Dual-ranged LUT on top of the base 8-bit LUT, resulting in a total of just 374 entries (257 + 17). This leads to a highly memory-efficient structure without sacrificing accuracy. Further details related to this design choice can be found in the Section VI-E1.

VI. Experiments

We design our experimental study to answer a core question: can Bespoke LUT deliver practical model- and hardware-level benefits while maintaining accuracy and reducing system-level cost? To answer this, we evaluate our method from three complementary perspectives:

1. Model-Level Effectiveness. We evaluate accuracy on both vision and language Transformer models under PTQ, comparing Bespoke LUT with representative approximators (I-BERT [28], I-ViT [27], NN-LUT [30]) and FP32 baselines.

2. System-Level Efficiency. We assess latency and memory overhead under a commercial edge NPU, focusing on non-linear operation cost and the benefits of using Bespoke LUT without DSP intervention.

3. Design Justification. We conduct ablations on LUT design parameters (e.g., per-layer LUTs, indexing schemes) and compare microarchitectural traits (e.g., indexing complexity,

integer-only execution) with prior work to validate hardware feasibility.

Together, these experiments demonstrate that Bespoke LUT offers not only high accuracy but also practical deployability under resource-constrained settings.

A. Experiment Settings

1) Hardware and Simulator Configuration: We used an in-house event-driven simulator targeting our commercial NPU for latency evaluation. This simulator runs the control processor (CP) and DSP modules at the cycle level, while profiling the MAU modules using an instruction-level simulation based on a performance model derived from actual RTL results. The NPU includes a MAC array (MAU) to accelerate linear transformation using integer-arithmetic inference and a programmable DSP to handle more complex operations. MAU and DSP each deliver 4 TOPS and 32 GOPS of computing power, with 3 GB/s of external DRAM memory bandwidth allocated. Although this might be considered a relatively low-performance NPU, our aim is to demonstrate the capability of running Transformer models effectively in highly constrained edge-device environments [16], [17] , such as TVs or robotic vacuum cleaners, not only in mobile phones.

2) Implementation Details: For accuracy evaluation, we tested all models in PyTorch. To focus on PTQ deployment scenarios, we evaluated all baselines without any fine-tuning, reflecting realistic hardware deployment where retraining is infeasible. I-BERT [28] and I-ViT [27] were evaluated using official implementations provided by the authors. NN-LUT [30] was re-implemented based on the original paper. For reproduction, we trained LUTs for four non-linear operations (Exp and Reciprocal for Softmax, RSQRT for LayerNorm, and GELU), using 128 hidden units (resulting in 256 parameters: 128 weights + 128 biases). We adopted a conservative configuration (learning rate = $1e-5$, $300K$ steps) to ensure stable convergence. This setup isolates the approximation quality of each method in a fair PTQ environment without introducing confounding effects from fine-tuning or domain-specific heuristics.

As I-BERT, I-ViT and NN-LUT were designed for specific non-linear functions, GELU, Softmax and LayerNorm, we adapted existing implementation of LayerNorm with additional modifications for RMSNorm. In contrast, Bespoke LUT's function-agnostic attribute enables direct support for all non-linear operations included in the evaluated models. Notably, Bespoke LUT was effortlessly applied to new non-linear functions such as SiLU and RMSNorm introduced in TinyLlama [8] and SmolLM [9], without requiring any function-specific adaptation or retraining.

3) Datasets and Evaluation Metrics: We evaluate our method on both vision and language Transformer models, using widely recognized datasets. ViT [6], DeiT [52], and EfficientFormer [53] are tested on ImageNet-1k [54], while DETR [7] is evaluated on MS COCO [55]. For language models—TinyLlama [8], SmolLM [9], and OPT [51]—we use WikiText-2 [56]. We used top-1 accuracy as the evaluation metric for image classification. For object detection, we measured the mean average precision (mAP) [57] of bounding boxes, with mAP@50:95 used to represent accuracy. The metric, mAP@50:95, averages mAP values across a range of intersection-over-union (IoU) thresholds from 0.50 to 0.95, in increments of 0.05. For language models, we used perplexity (ppl) [58] as the evaluation metric, where lower perplexity values indicate better predictive performance.

B. Mixed Precision Pattern

Before conducting the experiments, we employed a commonly used technique called mixed precision quantization (MPQ) to efficiently apply quantization [59], [60]. Traditional approaches to mixed precision determine the bit-width for different layers by measuring their quantization sensitivity, but this process becomes increasingly costly as model size grows. Our experiments revealed that assigning higher precision to certain operation patterns can substantially enhance accuracy with minimal latency impact, eliminating the need for extensive parameter tuning. We applied this mixed precision strategy to a typical Transformer block.

One particular challenge with 8-bit quantization is *element-wise addition*, which is highly susceptible to accuracy degradation. When there is a substantial difference in scale between inputs, the larger input can dominate the output [22], [61], [62]. However, the computational overhead of performing element-wise addition in 16-bit is negligible. Thus, all residual paths in the Transformer block were computed in 16-bit to preserve accuracy.

Another component prone to accuracy loss in 8-bit is the *Square* operation within LayerNorm, where each input is multiplied by itself [22], [63]. Representing the square of an 8-bit value using only 8 bits results in severe information loss, because squared values greater than 255 are clipped. Among the 256 possible input values, only 17 unique outputs remain (the squares of inputs 0-15 and the clipped value 255), leading to approximately 93.4% loss of representable information. This loss is especially pronounced when the scale is small, leading to a significant accuracy degradation [62]. To mitigate this, we compute the variance in LayerNorm using 16-bit precision.

C. Accuracy Evaluation

In this section, we compare the performance of our proposed method against several benchmark approaches on various Transformer models. Unless otherwise noted, all experiments used MPQ (INT8 and INT16) for the linear layers, as described in Section VI-B. For non-linear layers, I-BERT [28], I-ViT [27] and NN-LUT [30] used 32-bit integers, whereas Bespoke LUT used 16-bit integers for LUT output values and 8-bit integers for other computations.

1) Vision models: In Table I, we compare the accuracy of Bespoke LUT with I-BERT [28], I-ViT [27], and NN-LUT [30] across vision Transformer models such as ViT [6], DeiT [52], EfficientFormer [53], and DETR [7]. Despite using low-precision arithmetic (16-bit LUT output), Bespoke LUT achieves accuracy on par with full-precision (FP32) baselines and consistently outperforms all prior approximators.

TABLE I: Accuracy comparison for non-linear approximation on vision tasks. NN-LUT marked with an asterisk (*) indicates the use of LUTs exclusively for the Exponential and Reciprocal functions in Softmax.

	Image Classification				Object Detection
	ViT-Base	DeiT-Tiny	Eformer-L1	Eformer-L3	DETR-R50
FP32	0.8510	0.7216	0.8050	0.8255	0.4201
MPQ	0.8425	0.7069	0.7899	0.8204	0.3796
I-BERT (INT32)	0.7964	0.6590	0.7593	0.8146	0.0191
I-ViT (INT32)	0.7119	0.5081	0.7540	0.8074	0.1065
NN-LUT* (INT32)	0.7919	0.4978	0.7855	0.8171	0.2446
NN-LUT (INT32)	0.0012	0.0009	0.0013	0.0018	0.0760
Bespoke LUT (INT8+16)	**0.8441**	**0.7069**	**0.7907**	**0.8195**	**0.3782**

TABLE II: Perplexity comparison for non-linear approximation on language models.

	TinyLlama	SmolLM	OPT
FP32	6.9624	8.5236	12.5163
MPQ	7.1249	9.6684	13.0443
I-BERT (INT32)	**7.1611**	32.406	13.6231
I-ViT (INT32)	35.6617	87.8625	14.5739
NN-LUT (INT32)	-	-	-
Bespoke LUT (INT16)	7.4225	**10.2741**	**12.9053**
w/o Dual-ranged LUT	68.4218	27692.1640	12.8935

TABLE III: Hardware latency of the benchmark methods for each operation.

Latency (ns)	GELU	Softmax	LayerNorm
DSP	211	251	127
I-BERT	149	362	151
I-ViT	569	451	795
Bespoke LUT	**67**	**111**	**55**

Although I-BERT and I-ViT employed 32-bit precision for some of their approximators, they experienced notable accuracy drops in the absence of fine-tuning. This observation suggests that polynomial approximations of non-linear operations may be inadequate in capturing the complexities inherent in non-linearity across a variety of models under a resource-constraint environment.

For NN-LUT, results with an asterisk denote that LUTs were applied only to the Softmax function. When NN-LUT was applied to other non-linear operations, such as GELU and RSQRT in LayerNorm, we observed extreme accuracy deterioration. Consequently, we conducted a comprehensive analysis of input distributions across all layers and operations for each model. While the input ranges for Exponential and Reciprocal in Softmax were aligned within the bounds assumed by the original authors, the actual input ranges of GELU and RSQRT proved much wider. This discrepancy indicates that the input distributions assumed in NN-LUT are insufficient for a broader variety of models. NN-LUT originally focused on BERT-based models [64], [65], yet our experiments across a more diverse set of tasks and models revealed considerably greater variation in input ranges.

We observed consistent trends with DETR-R50 model, which utilizes ReLU activation instead of GELU, focusing on Softmax and LayerNorm. Both I-BERT and I-ViT, when not fine-tuned, experienced significant accuracy degradation. Moreover, applying NN-LUT to both Softmax and LayerNorm resulted in a marked decline in accuracy compared to the results obtained when NN-LUT was applied only to Softmax. A noteworthy observation is that although the input range of RSQRT in DETR-R50 is aligned with the defined range, applying NN-LUT to DETR-R50 still incurs critical accuracy loss because of the extreme variation in input ranges of RSQRT layers in DETR model as shown in Figure 3.

Bespoke LUT consistently achieves high accuracy across all models and tasks, matching MPQ baselines that rely on full-precision DSP execution. Despite using only 8-bit indexed LUTs and 16-bit outputs, it effectively approximates non-linear operations. Its per-layer design ensures robustness to model-specific variations, and the entire process operates under PTQ without requiring any fine-tuning.

2) Language Models: In Table II, we extend the evaluation to language models, comparing Bespoke LUT against I-BERT [28], I-ViT [27], and NN-LUT [30] on models such as TinyLlama (1.1B) [8], SmolLM (1.7B) [9], and OPT (1.3B) [51]. Compared to vision models, LLMs pose significantly greater challenges for quantization. Their activations exhibit much wider and more uneven distributions [26], [66], particularly in non-linear layers such as LayerNorm and Softmax. These characteristics exacerbate approximation errors under low-precision settings, making LLMs far more sensitive to quantization artifacts than vision models.

Despite these challenges, Bespoke LUT achieves accuracy comparable to full-precision (FP32) and consistently outperforms all prior approximators under pure PTQ without any fine-tuning. This robustness stems from two design principles: per-layer LUT construction, which adapts to diverse input distributions, and the selective use of Dual-ranged LUTs to refine approximations in high-error regions. In both TinyLlama and SmolLM, we observed significant approximation error in the first LUT interval due to asymptotic divergence. Applying a 4-bit auxiliary LUT in that region yielded notable accuracy gains with minimal memory overhead. In contrast, this refinement was not needed in OPT, where input distributions were more uniform.

I-BERT performed well on TinyLlama and OPT but poorly on SmolLM, whereas I-ViT performed well only on OPT. While I-ViT's original paper recommended a scaling factor of 23 for exponential approximation, we found that a much smaller value of 12 worked better for OPT, highlighting its sensitivity to model-specific tuning. Analyzing the Bespoke LUT results, we found that disabling the Dual-ranged LUT caused notable performance drops on TinyLlama and especially SmolLM, but not on OPT. I-ViT underperformed on both TinyLlama and SmolLM, while I-BERT showed

TABLE IV: Latency comparison of processing non-linear operations within Transformer blocks using DSP and LUT.

	BW (GB/s)	ViT-Base DSP	ViT-Base MAU (LUT)	Eformer-L1 DSP	Eformer-L1 MAU (LUT)	TinyLlama DSP	TinyLlama MAU (LUT)
Latency (ms)	3	16.73	12.87	2.69	2.00	1506.05	448.93
	20	8.47	3.87	1.25	0.58	1233.57	142.56
Non-linear op (ms)	3	5.90	0.19	0.37	0.08	1113.26	27.20
Communication (ms)	3	1.74	0.94	0.15	0.10	35.50	19.88
Non-linear proportion (%)	3	34.67	0.35	26.72	0.28	73.92	0.38

degradation mainly on SmolLM, which appeared particularly sensitive to extreme input values. This suggests that I-BERT and I-ViT are susceptible to extreme inputs. NN-LUT showed the weakest performance when applied beyond the Softmax layer. We extended its application to GELU and RSQRT to enable a fair and comprehensive comparison. This broader application resulted in sharp accuracy degradation, especially in TinyLlama and SmolLM, where RSQRT input distributions vary drastically across layers.

These results confirm that Bespoke LUT is not only accurate under PTQ without any retraining, but also robust across a variety of LLM architectures. Its ability to adapt to distributional shifts—especially in quantization-sensitive regions—makes it a practical and generalizable solution for on-device LLM deployment.

D. Hardware Efficiency Evaluation

1) Latency evaluation of Non-linear Modules: We assessed the latency of key non-linear operations—GELU, Softmax with Exponential and Reciprocal, and LayerNorm with RSQRT—across our proposed method and benchmark approaches, as detailed in Table III. I-BERT [28] and I-ViT [27] approximate entire modules rather than individual operations. Thus, we measured latency at the module level, such as Softmax and LayerNorm, to provide a comprehensive comparison. This evaluation was conducted using the simulator emulating our commercial NPU, which supports only 8-bit and 16-bit computations for on-device applications. As such, although I-BERT and I-ViT can utilize both 8-bit and 32-bit precision, all latency measurements were taken assuming 8-bit precision. Still, Bespoke LUT exhibited the fastest latency due to its simple architecture.

Regarding NN-LUT [30], its architecture is structurally similar to Bespoke LUT, NN-LUT requires iterative comparator modules based on if-else conditions. Due to the lack of optimization for these comparator modules in our simulator, we were unable to obtain reliable latency measurements for NN-LUT and have therefore excluded it from the evaluation. However, since Bespoke LUT allows immediate value retrieval without the need for additional comparator overhead, we estimate that Bespoke LUT would exhibit lower latency than NN-LUT. Overall, the results show that Bespoke LUT achieves $2-14\times$ faster latency compared to the other benchmarks, including the 8-bit DSP, demonstrating its high efficiency in handling non-linear operations.

2) Latency evaluation of Transformer blocks: In Table IV, we evaluated the overall latency, the latency and the communication overhead of non-linear operations, and the proportion of latency attributed to non-linear operations in a single Transformer block of our baseline models. The evaluation was conducted using both DSP and MAU (LUT). We conducted the experiments with the bandwidth of 3 GB/s as the average bandwidth on our target NPU. Further, we evaluated the latency using the bandwidth of 20 GB/s to experimentally minimize the influence of data transfer and compare the pure execution time in DSP and MAU.

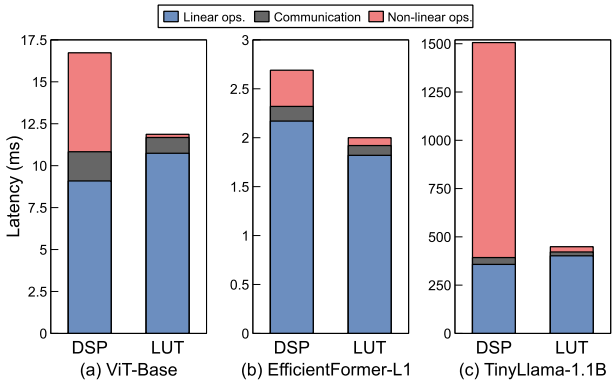

Fig. 8: Latency comparison of Transformer blocks using DSP and Bespoke LUT.

TABLE V: Number of model parameters and memory usage of Bespoke LUT. The values in parentheses represent the size of the Dual-ranged LUT.

	Model Size	LUT Size (KB)	Proportion (%)
ViT-Base	87M	31.354	0.036
DeiT-Tiny	6M	31.354	0.523
Eformer-L1	14M	10.28	0.073
Eformer-L3	39M	22.102	0.057
DETR-R50	41M	34.438	0.084
TinyLlama	1.1B	44.19 (2.19)	0.004
SmolLM	1.7B	48.18 (2.39)	0.003
OPT	1.3B	48.18 (2.39)	0.004

By minimizing the computational burden of non-linear operations, the Bespoke LUT improves the execution time by up to $3.3\times$ faster as shown in Figure 8. This is achieved by lowering the communication overhead between the MAU and DSP, thereby enhancing overall performance efficiency.

As the bandwidth increases, the advantage of Bespoke LUT becomes more pronounced, as the utilization of LUT brings $2.1-8.6\times$ gain in latency. This suggests that LUT-based approach provides substantial gains in overall latency, particularly when minimizing the impact of communication overhead. By reducing the relative contribution of overhead incurred by DMA access within the system, Bespoke LUT enables more efficient computation, thereby boosting the model performance as the operational bandwidth expands.

TABLE VI: Comparison of the approximation methods for non-linear operations on NPUs. Cycles for NN-LUT marked in parentheses are estimated on our NPU.

	Bespoke LUT	NN-LUT	I-BERT	I-ViT
Indexing Scheme	8-bit uniform + optional 4-bit	Learned (non-uniform)	Polynomial (Taylor)	Shift-based approx.
Precision	INT16	INT32 (FP32)	INT32	INT32
# Entries	257 + 17	1-hidden layer (128)	-	-
Comparator Needed	No	Yes	No	No
Cycles	2	2 (3)	3-5	3-5

3) Memory evaluation: We conducted an analysis of the number of model parameters, the memory size of per-layer LUT, and its proportion to the model size as detailed in Table V. For vision Transformers, the LUT memory usage remained low at around 30KB, demonstrating its excellent compatibility. Even for LLMs exceeding 1B parameters, the LUT memory usage was limited to 50KB, including the Dual-ranged LUT.

While the size of Bespoke LUT increases with the number of model parameters, the most significant factor affecting its overall size is the number of layers. Nonetheless, the size of Bespoke LUT occupies no more than 0.5% of the model size, and the loading process these LUTs imposes a negligible burden on the system. This minimal memory footprint demonstrates that our method is a versatile solution that can be applied to a wide range of models, from smaller vision Transformers to LLMs, and across various hardware environments.

4) Microarchitectural Comparison: To assess hardware efficiency, we compare the microarchitectural characteristics of Bespoke LUT with prior approximation methods in Table VI, focusing on indexing logic, output precision, number of entries, and execution cycles. Our Bespoke LUT uses an 8-bit uniform indexing scheme with optional 4-bit uniform indexing (Dual-ranged LUT), allowing deterministic lookups without requiring training or complex access logic. It stores INT16 outputs and avoids comparators altogether, enabling fast and predictable execution. While the number of entries is larger than that of prior work, this is an intentional choice to ensure high accuracy without increasing logic complexity.

We estimate cycle counts under a 1 GHz operating frequency, which reflects typical hardware conditions in resource-constrained NPUs [17]–[21], [67]. Cycle counts in the table represent the minimal execution latency under integer-based execution units, serving as a common baseline for comparison. For Bespoke LUT, the architecture uses a uniform indexing scheme and a single INT16 interpolation per access. This simple and deterministic execution path enables completion in 2 cycles. For NN-LUT [30], the paper reports a 2-cycle latency, but under typical hardware assumptions with INT32 support, the operation would require at least 3 cycles. We report this in the table with values in parentheses to distinguish estimated values from ones reported in the paper. For I-BERT [28] and I-ViT [27], latency varies depending on the target non-linear function. Both use polynomial approximations evaluated through a sequence of INT32 operations. Considering both the data precision and the iterative nature of their evaluation, we estimate 3–5 cycles per operation depending on complexity.

E. Ablation Study

TABLE VII: Model performance across varying LUT bit-widths.

LUT bit	# entry	DeiT	ViT	TinyLlama	SmolLM
4	16	69.836	63.258	21566710	151108.8
5	32	**70.598**	82.664	24542.02	226507.9
6	64	70.596	**84.248**	21609.91	48965.43
7	128	70.564	84.388	14029.01	42430.7
8	256	70.548	84.372	5673.575	27893.1
9	512	70.568	84.368	959.9969	1200.457
10	1024	70.606	84.274	87.3619	74.0579
11	2048	70.564	84.366	14.9897	16.6283
12	4096	70.57	84.412	**7.9888**	**10.2663**
13	8192	70.62	84.414	7.4676	9.7067

1) Design Choice for LUT Architecture: To determine the appropriate LUT entry bit-width for Bespoke LUT, we conducted an ablation study evaluating the performance impact of varying the bit-width from 4 to 13 bits in Table VII. This study was performed across four representative models—DeiT [52], ViT [6], TinyLlama [8], and SmolLM [9]—without employing the Dual-ranged LUT structure. Instead, all non-linear operations within the Transformer blocks were approximated uniformly using N-bit LUT per layer.

Our findings reveal that the required bit-width for maintaining acceptable model performance varies depending on model size and architecture. For vision models such as DeiT and ViT, LUTs with 8-bit entries (257 entries) were sufficient to achieve accuracy comparable to their FP32 baselines. In contrast, the larger and more complex TinyLlama and SmolLM required higher precision, reaching FP32-level performance only when using a 12-bit LUT (4097 entries).

Based on these observations, we adopted a hybrid 8-bit + 4-bit LUT architecture, leveraging powers-of-two for hardware efficiency. The 8-bit LUT serves as the default for all layers, while the additional 4-bit LUT is selectively enabled to enhance precision in layers where approximation error is critical, as identified in prior analysis. This design provides a balanced trade-off between performance quality and resource usage.

2) Universal LUT vs Per-Layer LUT: As observed in Figure 3, even for the same non-linear operation, the input distribution can vary significantly depending on the layer's position within the model. Even if a universal LUT is created to cover all these distributions, it might be inadequate for layers that need to handle smaller ranges. Table VIII demonstrates that using a single LUT across the entire model results in a drop in accuracy compared to using a per-layer approach. Particularly in TinyLlama and SmolLM, the significant variation in input ranges across layers, along with the sensitivity of non-linear functions to small input values, made the universal LUT approach especially unsuitable for this model. Moreover, Bespoke LUT significantly outperforms across various models,

TABLE VIII: Accuracy comparison between universal LUT and per-layer LUT.

	Universal	Per-Layer
ViT-Base	0.7932	**0.8441**
DeiT-Tiny	0.7065	**0.7069**
Eformer-L1	0.3207	**0.7907**
Eformer-L3	0.6487	**0.8195**
DETR-R50	0.2370	**0.3782**
TinyLlama	47556.62	**7.4225**
SmolLM	6408402.0	**10.2741**
OPT	13.2678	**12.9053**

TABLE IX: Model accuracy and the number of Dual-LUTs under varying MAPE thresholds.

	TinyLlama			SmolLM		
Threshold	# dr-lut	Memory (KB)	ppl	# dr-lut	Memory (KB)	ppl
0.00	69	2.19	7.90	72	2.39	10.27
0.05	39	1.29	7.96	27	0.90	10.27
0.10	34	1.13	7.99	25	0.83	10.30
0.15	31	1.03	8.05	21	0.70	10.32
0.20	29	0.96	8.22	18	0.60	10.81
0.25	27	0.90	8.43	18	0.60	10.81
0.30	26	0.86	8.54	16	0.53	11.53
0.35	24	0.80	9.12	16	0.53	11.53

implying that utilizing a "Bespoke" LUT tailored to each layer is crucial. Since it can be derived from the min/max parameters obtained during the calibration, which is a prerequisite for PTQ, this approach requires little to no additional effort.

3) Threshold Selection for Dual-ranged LUT Application: As previously discussed, the Dual-ranged LUT is selectively applied to layers in which the approximation error—measured by the MAPE in the first LUT interval—exceeds a predefined threshold. Since this threshold acts as a tunable hyperparameter, we conducted a series of experiments to explore its effect on model performance and resource usage.

We varied the MAPE threshold across values $\{0.00, 0.05, 0.10, 0.15, \quad 0.20, 0.25, 0.30, 0.35\}$, measuring the resulting accuracy and memory footprint for each configuration in Table IX. As expected, lower thresholds result in more layers receiving the additional 4-bit LUT, thereby improving approximation precision and model accuracy. However, this comes with the cost of increased memory usage due to the larger number of dual-range LUT entries required.

This trade-off between performance and memory efficiency in TinyLlama is illustrated in Figure 9. As the threshold value decreases, memory consumption continues to increase, while improvements in perplexity gradually diminish, indicating a point of diminishing returns.

VII. Discussion

While large language models are often considered memory-bound, this characterization mainly applies to the token generation phase [68], [69]. During the prefill stage—which includes long context processing and early-stage attention—non-linear operations such as Softmax and normalization impose considerable computational overhead, and are increasingly accounting for a larger fraction of the inference cost, especially in emerging deployment paradigms like agent-based AI [70], [71] and multi-modal reasoning [72], [73]. Optimizing non-linear computation in this stage is therefore critical for enabling faster and more efficient multi-stage inference [74], [75]. Bespoke LUT naturally aligns with this trend by offering a lightweight, training-free mechanism to accelerate non-linear operations without requiring architectural changes or retraining.

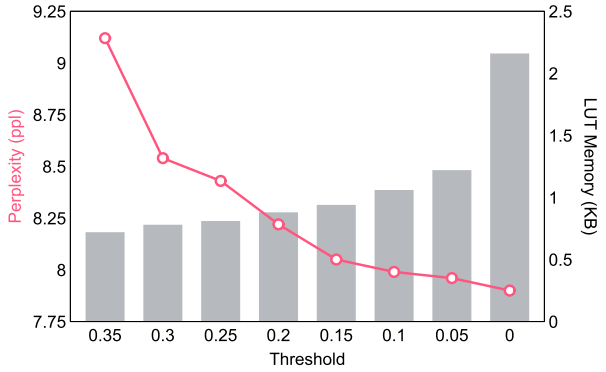

Fig. 9: Trade-off between perplexity and memory usage as MAPE threshold for Dual-ranged LUT decreases.

VIII. Conclusion

In this paper, we present Bespoke LUT, whose name reflects our design philosophy of customizing LUT behavior per layer to align with both model characteristics and hardware constraints. By leveraging calibration data for per-layer LUT configuration, we achieve accurate integer-only inference with no need for retraining or runtime cost. Moreover, our Dual-ranged LUT mitigates the divergent behavior of non-linear functions in asymptotic regions using a selective application algorithm to minimize overhead while preserving accuracy. Taken together, our methods reveal that a simple and flexible LUT design can meet the dual goals of accuracy and hardware efficiency, making our approach practical and efficient for real-world deployment. Our experimental results demonstrate that Bespoke LUT consistently outperforms existing benchmarks across both vision and language tasks, significantly improving latency and achieving up to $3.3\times$ speedup.

The simplicity and scalability of Bespoke LUT make it highly suitable for future applications, especially as Transformer models grow in size and as the demand for their efficient deployment on edge devices increases. Looking ahead, we plan to extend the application of Bespoke LUT beyond Transformer architectures to other domains requiring efficient non-linear function acceleration. Its simplicity, effectiveness, and minimal hardware overhead suggest that Bespoke LUT has strong potential as a practical solution for a wide range of models and domains, delivering both accuracy and hardware efficiency without requiring extensive parameter tuning or retraining.

References

[1] A. Raha, S. K. Kim, D. A. Mathaikutty, G. Venkataramanan, D. Mohapatra, R. Sung, C. Brick, and G. N. Chinya, "Design considerations for edge neural network accelerators: An industry perspective," in *2021 34th*

International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID), 2021, pp. 328–333.

[2] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080246

[3] H. Liao, J. Tu, J. Xia, and X. Zhou, "Davinci: A scalable architecture for neural network computing," in *2019 IEEE Hot Chips 31 Symposium (HCS)*, 2019, pp. 1–44.

[4] D. Larkin, A. Kinane, V. Muresan, and N. O'Connor, "An efficient hardware architecture for a neural network activation function generator," in *Advances in Neural Networks - ISNN 2006*, J. Wang, Z. Yi, J. M. Zurada, B.-L. Lu, and H. Yin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1319–1327.

[5] L. Li, S. Zhang, and J. Wu, "An efficient hardware architecture for activation function in deep learning processor," 06 2018, pp. 911–918.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[7] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[8] P. Zhang, G. Zeng, T. Wang, and W. Lu, "Tinyllama: An open-source small language model," *arXiv preprint arXiv:2401.02385*, 2024.

[9] L. B. Allal, A. Lozhkov, E. Bakouch, L. von Werra, and T. Wolf, "Smollm - blazingly fast and remarkably powerful," 2024.

[10] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[11] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH computer architecture news*, vol. 44, no. 3, pp. 367–379, 2016.

[12] NXP Semiconductors, "Improving dsp performance using efficient spe coding," NXP Semiconductors, Tech. Rep., 2008. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN3733.pdf

[13] ARM Ltd., "The dsp capabilities of arm cortex-m4 and cortex-m7 processors," ARM Ltd., Tech. Rep., 2016. [Online]. Available: https://community.arm.com/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-21-42/7563.ARM-white-paper-_2D00_-DSP-capabilities-of-Cortex_2D00_M4-and-Cortex_2D00_M7.pdf

[14] Qualcomm Technologies, Inc., "Qualcomm snapdragon 8 gen 1 mobile platform," https://www.qualcomm.com/products/snapdragon-8-gen-1-mobile-platform, 2021, accessed: 2025-04-11.

[15] ——, "Qualcomm ai hub," https://app.aihub.qualcomm.com, 2024, accessed: 2025-04-11.

[16] A. D. Documentation, "Arm ethos-u85 npu technical overview," https://developer.arm.com/Processors/Ethos-U85, 2024, accessed: 2024-06-05.

[17] V. P. Release, "Verisilicon vip9000 and zsp are adopted by icatch next generation ai-powered automotive image processing soc." https://www.verisilicon.com/en/PressRelease/VIP9000andZSPAdoptedbyiCatch, 2024, accessed: 2024-05-25.

[18] C. L. Team, "Ceva-neupro: Npu ip for embedded ai," https://dev2.ceva-ip.com/product/ceva-neupro-nano/, 2023, accessed: 2025-04-10.

[19] M. Team, "Mediatek dimensity 9000," https://www.mediatek.com.es/products/smartphones-2/mediatek-dimensity-9000, 2021, accessed: 2025-04-10.

[20] C. L. Team, "Ceva-neupro: Npu ip for embedded ai," https://www.synaptics.com/assets/product-brief/vs680-uhd-smart-display-soc, 2022, accessed: 2025-04-10.

[21] H. Team, "Hailo-8 ai accelerator," https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator/, 2019, accessed: 2025-04-10.

[22] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[23] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

[24] Y. Bondarenko, M. Nagel, and T. Blankevoort, "Understanding and overcoming the challenges of efficient transformer quantization," *arXiv preprint arXiv:2109.12948*, 2021.

[25] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "SmoothQuant: Accurate and efficient post-training quantization for large language models," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[26] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[27] Z. Li and Q. Gu, "I-vit: Integer-only quantization for efficient vision transformer inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 065–17 075.

[28] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-bert: Integer-only bert quantization," *International Conference on Machine Learning (Accepted)*, 2021.

[29] Y. Xie, A. N. Joseph Raj, Z. Hu, S. Huang, Z. Fan, and M. Joler, "A twofold lookup table architecture for efficient approximation of activation functions," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 12, pp. 2540–2550, 2020.

[30] J. Yu, J. Park, S. Park, M. Kim, S. Lee, D. H. Lee, and J. Choi, "Nn-lut: neural approximation of non-linear operations for efficient transformer inference," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 577–582. [Online]. Available: https://doi.org/10.1145/3489517.3530505

[31] H. Lu, Q. Mei, and K. Wang, "Auto-lut: Auto approximation of non-linear operations for neural networks on fpga," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[32] E. Reggiani, R. Andri, and L. Cavigelli, "Flex-sfu: Accelerating dnn activation functions by non-uniform piecewise approximation," 07 2023, pp. 1–6.

[33] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "Fq-vit: Post-training quantization for fully quantized vision transformer," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022, pp. 1173–1179.

[34] D. Haynes, S. Corns, and G. K. Venayagamoorthy, "An exponential moving average algorithm," in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

[35] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," arXiv preprint arXiv:1412.6115, 2014.

[36] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations (ICLR)*, 2016.

[37] J.-S. Park, J.-W. Jang, H. Lee, D. Lee, S. Lee, H. Jung, S. Lee, S. Kwon, K. Jeong, J.-H. Song, S. Lim, and I. Kang, "9.5 a 6k-mac feature-map-sparsity-aware neural processing unit in 5nm flagship mobile soc," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 152–154.

[38] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, H. Abdel-Aziz *et al.*, "Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 15–28.

[39] N. Corporation, "Nvdla primer," http://nvdla.org/primer.html, 2021, accessed: 2024-03-07.

[40] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable lookup table for neural network quantization," in *CVPR*, 2022, pp. 12 423–12 433.

[41] G. Park, H. Kwon, J. Kim, J. Bae, B. Park, D. Lee, and Y. Lee, "Figlut: An energy-efficient accelerator design for fp-int gemm using look-up tables," *arXiv preprint arXiv:2503.06862*, 2025.

[42] J. Kim, J. Lee, J. Choi, J. Han, and S. Lee, "Range-invariant approximation of non-linear operations for efficient bert fine-tuning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[43] A. Khataei and K. Bazargan, "Compressedlut: An open source tool for lossless compression of lookup tables for function evaluation and beyond," in *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 2–11. [Online]. Available: https://doi.org/10.1145/3626202.3637575

[44] R. E. Crandall and C. Pomerance, *Prime numbers: a computational perspective*. Springer, 2005, vol. 2.

[45] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[46] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[47] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[48] E. J. Kirkland and E. J. Kirkland, "Bilinear interpolation," *Advanced computing in electron microscopy*, pp. 261–263, 2010.

[49] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, "On layer normalization in the transformer architecture," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[50] A. Langedijk, H. Mohebbi, G. Sarti, W. Zuidema, and J. Jumelet, "DecoderLens: Layerwise interpretation of encoder-decoder transformers," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 4764–4780. [Online]. Available: https://aclanthology.org/2024.findings-naacl.296/

[51] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin *et al.*, "Opt: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[52] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, vol. 139, July 2021, pp. 10 347–10 357.

[53] Y. Li, G. Yuan, Y. Wen, J. Hu, G. Evangelidis, S. Tulyakov, Y. Wang, and J. Ren, "Efficientformer: Vision transformers at mobilenet speed," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 934–12 949, 2022.

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

[55] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[56] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016. [Online]. Available: https://arxiv.org/abs/1609.07843

[57] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[58] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker, "Perplexity—a measure of the difficulty of speech recognition tasks," *The Journal of the Acoustical Society of America*, vol. 62, no. S1, pp. S63–S63, 1977.

[59] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[60] M. Rakka, M. E. Fouda, P. Khargonekar, and F. Kurdahi, "Mixed-precision neural networks: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2208.06064

[61] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018. [Online]. Available: https://arxiv.org/abs/1806.08342

[62] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, "Ptq4vit: Post-training quantization fo vision transformers with twin uniform quantization," in *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XII*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 191–207. [Online]. Available: https://doi.org/10.1007/978-3-031-19775-8_12

[63] Y. Ding, H. Qin, Q. Yan, Z. Chai, J. Liu, X. Wei, and X. Liu, "Towards accurate post-training quantization for vision transformer," in *Proceedings of the 30th ACM International Conference on Multimedia*, ser. MM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 5380–5388. [Online]. Available: https://doi.org/10.1145/3503161.3547826

[64] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[65] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.

[66] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.

[67] A. D. Documentation, "Arm ethos," chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://armkeil.blob.core.windows.net/developer/Files/pdf/arm-ethos-u-processor-series-brief-v2.pdf, 2021, accessed: 2025-04-10.

[68] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[69] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro, A. Phanishayee, and M. Zaharia, "Efficient large-scale language model training on gpu clusters using megatron-lm," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3458817.3476209

[70] Z. Durante, B. Sarkar, R. Gong, R. Taori, Y. Noda, P. Tang, E. Adeli, S. K. Lakshmikanth, K. Schulman, A. Milstein, D. Terzopoulos, A. Famoti, N. Kuno, A. Llorens, H. Vo, K. Ikeuchi, L. Fei-Fei, J. Gao, N. Wake, and Q. Huang, "An interactive agent foundation model," 2024. [Online]. Available: https://arxiv.org/abs/2402.05929

[71] R. Gong, Q. Huang, X. Ma, Y. Noda, Z. Durante, Z. Zheng, D. Terzopoulos, L. Fei-Fei, J. Gao, and H. Vo, "MindAgent: Emergent gaming interaction," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 3154–3183. [Online]. Available: https://aclanthology.org/2024.findings-naacl.200/

[72] A. Gritsevskiy, A. Panickssery, A. Kirtland, D. Kauffman, H. Gundlach, I. Gritsevskaya, J. Cavanagh, J. Chiang, L. L. Roux, and M. Hung, "Rebus: A robust evaluation benchmark of understanding symbols," 2024.

[73] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, "Multimodal chain-of-thought reasoning in language models," 2024. [Online]. Available: https://arxiv.org/abs/2302.00923

[74] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[75] T. Dao, "FlashAttention-2: Faster attention with better parallelism and work partitioning," in *International Conference on Learning Representations (ICLR)*, 2024.

Seonyoung Kim received the B.S. degree in Computer Engineering from Hongik University in 2019, and the M.S. degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2022. Since 2022, she has been working as an AI Researcher with Samsung Research. Her research interests include efficient machine learning/artificial intelligence, efficient training/inference, optimization, and hardware-aware AI design.

Jooeun Kim received the B.S. degree in Applied Statistics and Computer Science from Yonsei University in 2021, and the M.S. degree in Data Science from Seoul National University in 2023. From 2023 to 2025, she was a Software Engineer with Samsung Research. Since 2025, she has been working as a Machine Learning Engineer with Naver Corporation. Her research interests include machine learning, software development, optimization, and computer vision models.

Hayoung Yun received the B.S. degree in Electrical and Electronic Engineering from Korea University in 2018. Since 2018, she has been working as an AI Researcher with Samsung Research. Her research interests include computer vision software, on-device AI application analysis, and optimization.

Meejeong Park received the B.S. degree in Computer System Engineering from Duksung Women's University in 2014. Since 2014, she has been working as an AI Researcher with Samsung Research. Her research interests include machine learning software, on-device AI application development, and optimization.

Sangjeong Lee received the B.S. degree in Physics from Korea Advanced Institute of Science and Technology in 1999, the M.S. degree in Electrical Engineering from KAIST in 2001, and the Ph.D. degree in Computer Science from KAIST in 2012. Since 2014, he has been working as an AI Researcher with Samsung Research. His research interests include DNN model trends, quantization, and optimization.

Hanjoo Cho (S'13-M'17) received the B.S. degree in electrical and electronic engineering from Kyungpook National University, Daegu, South Korea, in 2012, and the Ph.D. degree in electrical and computer engineering from Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2017.

From 2017 to 2024, he was a Senior Research Engineer with Samsung Research, Seoul, where he worked on neural network optimization for the NPU. He is currently a Senior Engineer in the Model Optimization Team with 42dot. His research interests include neural network quantization, model optimization, and neural processing units.

Heonjae Ha received the B.Eng. degree in Electronics Engineering from Korea University in 2006, the M.S. degree in Electrical Engineering from Stanford University in 2009, and the Ph.D. degree in Electrical Engineering from Stanford University in 2018.

From 2009 to 2013, he was a DRAM Design Engineer with SyHynix. He then served as a DRAM Design Validation Engineer with Apple from 2018 to 2022, and as a Hardware System Engineer with Meta from 2022 to 2024. Since May 2024, he has been the Head of the SoC Architecture Team with Samsung Research. His current research is focused on the strategic planning and execution of Custom IP development for mass-produced chips for battery-operated devices and large-scale data centers.