Dewant Katare[1], Salar Shakibhamedan[1], Nima Amirafshar[1], Nima Taherinejad[1], Axel Jantsch[1], Marijn Janssen[1], and Aaron Yi Ding[1]

[1]Affiliation not available

December 05, 2024

Approximation Strategies for Vision Models on Edge Devices: An Accuracy-Efficiency Trade-off

Dewant Katare , *Graduate Student Member, IEEE,* Salar Shakibhamedan , *Graduate Student Member, IEEE,* Nima Amirafshar , Nima Taherinejad , *Member, IEEE,* Axel Jantsch , *Senior Member, IEEE,* Marijn Janssen , and Aaron Yi Ding , *Member, IEEE*

***Abstract*—Autonomous applications having AI models and algorithms as backbone require high-performance computational and memory resources for efficient deployment and data processing. This complexity further increases while dealing with larger data sizes and computationally complex algorithms. Stochastic computing, precision-scaling, and model compression techniques such as quantization and pruning have addressed these issues. Although these approaches benefit model training and inference, performance and efficiency tradeoffs remain challenging. Our proposed approach includes three approximation schemes to address model performance and efficiency tradeoffs. The first scheme uses the concept of approximate multipliers. The second scheme approximates convolution operations using minimal multiplicative operations, and the third scheme uses variational inference using quantization-aware training and post-training quantization mechanisms. We evaluate the proposed schemes using performance metrics such as multiply-accumulate operations, floating-point operations, accuracy, latency, and on-device power consumption from CNNs, DNNs, and vision transformers. Tests show that our methods achieve up to 43% model compression while maintaining the accuracy within 3 to 4% of the baseline, with approximately 38% reduction in energy consumption compared to the baseline model. The proposed strategies provide mechanisms for optimized edge computing deployments by achieving a balanced tradeoff between model performance and energy efficiency. Code can be accessed at Approximate-Models.**

***Index Terms*—Approximation, Energy Efficiency, Edge AI, Machine Intelligence, Multipliers, Variational Inference**

I. Introduction

AI model development has advanced from traditional regression techniques to more complex statistical models and frameworks, such as graph neural networks, autoencoders, and generative adversarial networks (GANs) [1], [33]. This development trend also extends to computing practices such as the shift toward processing data locally on edge devices, complementing the existing high-performance servers and cloud-based deployments [4], [7], [15], [26]. Advancements in vision processing units (VPU), tensor processing units (TPU), and graphics processing units (GPUs) have enabled real-time applications and collaborative inference [42]. These developments provide methods for on-device learning, inference, and tiny machine learning within cyber-physical systems, such as autonomous vehicles, allowing them to perform complex tasks like image segmentation, object detection, precise localization, and high-definition mapping [19], [25], [48], [18], [53].

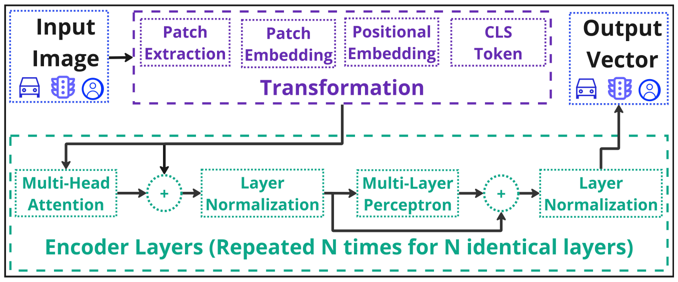


Fig. 1: Vision Transformer (ViT) block representation

The availability of automotive and vision datasets such as ImageNet, CIFAR-100, KITTI, Argo, nuScenes, and Waymo has complemented these advancements [19]. State-of-the-art neural network models such as ResNet [13], SSD [27], SqueezeNet [14], CenterNet [9], CenterPoint [55], and Yolo [36] have shown high accuracy [19], [58]. Recent research trends are shifting towards using vision transformers (ViT) such as EfficientVit [2], DeVit [52], TinyViT [49], and EdgeVit [3] to improve model performance by replacing traditional neural network operations with self-attention or multi-headed attention mechanisms, allowing for more precise image and object classification or detection (an overview of the ViT model is given in Figure 1) [8], [12], [56]. However, the high computational complexity, memory and energy demands of ViTs during training and inference present deployment challenges for compute-constrained edge devices [5], [44], [11], [37], [34]. Processing an n x n (size) image on classic models like ResNet-50 and Vgg-16 requires around 5.4 and 15.5 billion FLOPs, with a $O(n^2)$ complexity. With more complex modules, the ViTs have a computational complexity of $O(dn^2)$, where 'n' is the input sequence length or the number of patches. An overview of the modules and layers in these transformers are shown in Figure 1.

Similarly, the memory footprint varies for these models, ResNet-50 have approximately 25 million parameters, whereas larger ViT models have 100 million, requiring gigabytes of memory for processing (A trend of memory requirements for popular ViTs models against floating point operations per sec-

D. Katare, M. Janssen and A. Y. Ding are with the Department of Engineering, Systems and Services, Delft University of Technology (e-mail: d.katare@tudelft.nl; m.f.w.h.a.janssen@tudelft.nl; aaron.ding@tudelft.nl).

S. Shakibhamedan and A. Jantsch are with the Institute of Computer Technology, TU Wien, Vienna, Austria (e-mail: salar.shakibhamedan@tuwien.ac.at; axel.jantsch@tuwien.ac.at).

N. Amirafshar, N TaheriNejad are with the Institute for Computer Engineering, Heidelberg University, 69120 Heidelberg, Germany (e-mail: nima.amirafshar@uni-heidelberg.de; nima.taherinejad@ziti.uni-heidelberg.de). Corresponding author's e-mail: d.katare@tudelft.nl

ond (FLOPs) and parameters (millions) can be seen in Figure 2). The operations, including multiply-accumulate (MACs), range from 4-5 GMACs in efficient CNNs to over 15 GMACs in more complex ViTs [2], [12], [56], [49], [52]. These metrics show the challenges of deploying such models using Edge AI, which is often considered a sustainable alternative to centralized computing in the cloud or data centres which has additional power and energy demands [15], [42], [22], [17]. Also, recent analysis shows that vehicle automation's impact using AI and respective data processing emissions, including data transmission and wireless network energy use, are much higher than vehicle emissions [46], thus demanding energy-efficient computing [29]. For the mentioned challenges, this paper explores model approximation techniques for deploying models efficiently on embedded or Edge devices, focusing on the trade-offs between energy consumption and model performance metrics. The contributions are as follows:

1) We propose a multi-bit approximate multiplier mechanism for dynamic precision scaling and quantization-aware training of ViTs and neural network models.
2) A probabilistic approximation of convolution operation with reduced multiplicative complexity using signum function and bit-shifting, with a balanced trade-off between computational efficiency and model accuracy.
3) A variational inference (VI) mechanism for ViTs combined with post-training quantization and an energy-efficient training algorithm, including adaptive hyperparameter adjustment and joint-loss optimization.

The remaining paper is structured as follows: Section II covers the background and related work, focusing on model compression and acceleration. Section III covers the methodology, which includes the proposed approximation techniques. Section IV covers training processes and experimental results while evaluating the methods' accuracy and energy efficiency for training and inference. Sections V and VI discuss the results, summary, and future research directions for energy-efficient Edge AI.

II. Model Compression and Acceleration

Model optimization and acceleration techniques include pruning, normalization, distillation, quantization, and approximate computing [57], [23], [44], [50]. These methods offer model compression and reduction strategies for resource-constrained devices, but a knowledge gap exists in multi-parametric optimization of accuracy, energy efficiency, and computational complexity (memory and processing power). This gap is critical for applications which depend on either onboard heavy volume data processing or offloading data and computation to the high-performance computing units [19], [22], [42], [24]. This section covers optimization techniques for model deployment on edge devices, focusing on reducing memory and computational demands.

A. Multipliers

Approximate multipliers have been applied in various IoT and AI applications, such as object detection, speech recognition, and image classification, utilizing precision levels from

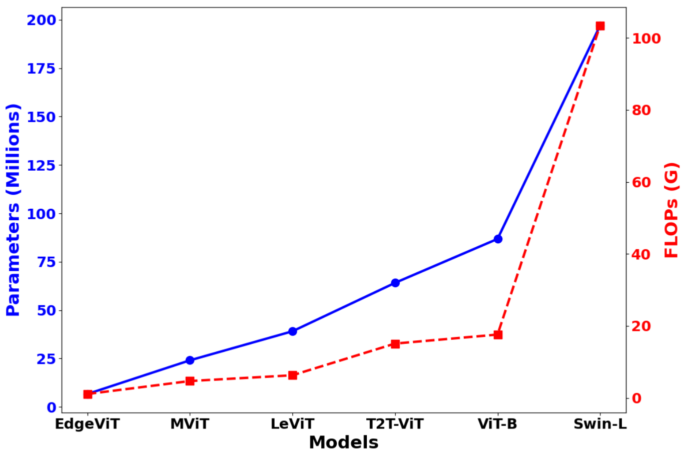


Fig. 2: Transformer models and related metrics on ImageNet

4-bit to 32-bit [35], [4], [39]. While 8-bit multipliers provide efficiency in low-power devices, they reduce accuracy impacting complex DNN models. However, 16-bit multipliers can balance models' accuracy and latency, making them ideal for computer vision tasks [44]. 32-bit multipliers provide the highest accuracy and precision but also increase latency and power consumption [16], [44]. Most of the benchmark models have 32-bit and 16-bit multiplier operations. Advances in multi-precision operations have also resulted in DNN models with mixed-precision arithmetic, improving overall efficiency [45], [53]. Techniques such as SmoothQuant implement post-training quantization by shifting quantization complexity from activations to weights, maintaining accuracy while reducing computational load and memory use during inference [50]. Additionally, approximate multipliers have shown potential in enhancing robustness against adversarial attacks, providing added reliability for security-sensitive applications [1], [38].

B. Probabilistic Approximation

Probabilistic approximation methods are efficiently used in hardware and software implementations, reducing the hardware form factor and enhancing energy savings [54], [23]. For software applications such as ML model deployment, these methods enable model compression, parameter reduction and energy savings during the training and inference phases using controlled approximations for computations (e.g., probability density function, convolution operation, multi-layer perceptron, feed-forward network). Stochastic rounding [54], [51] and probabilistic pruning [21] have been applied to ViTs and DNNs, resulting in more compact models with reduced memory footprints and computational requirements. [6]. Bayesian compression[57] and variational dropout [30] further contribute to creating efficient, low-power models suitable for edge devices and mobile applications.

C. Bit Reductions and Quantized Models

The rising computational demands of CNNs and Vision Transformers have also been addressed using Bit-by-bit reduction methods, explored for both hardware and software optimization [50], [38], [1], [54], [45]. Moons et al. [31] covered

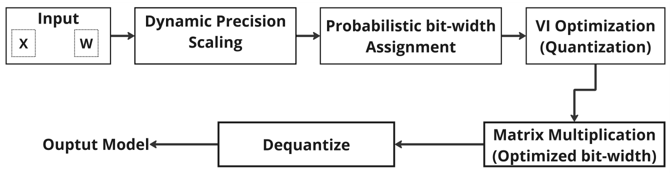

Input
X
W
Dynamic Precision Scaling
Probabilistic bit-width Assignment
VI Optimization (Quantization)
Matrix Multiplication (Optimized bit-width)
Dequantize
Ouptut Model

Fig. 3: Proposed Methodology for Optimized ViT Training

precision reduction methods focusing on energy saving while maintaining acceptable performance in CNNs. Kumar et al. [20] and Nguyen et al. [32] discuss bit-level optimizations that improve energy and latency metrics while preserving model accuracy. Louati et al. [28] proposed incremental quantization that dynamically adjusts weight bit-width in CNNs, showing energy savings with minimal impact on classification accuracy.

For Vision Transformers, Kuznedelev et al. [21] propose progressive quantization, which incrementally reduces precision in model parameters and activations and shows that ViTs are resilient to bit reduction and can achieve computational savings with minimal performance impact. This resilience enables ViTs to be deployed in resource-constrained environments. Additional approaches, such as training DNNs using 8-bit floating-point values, tackle the challenge of retaining gradient computation during back-propagation, contributing to efficient hardware training platforms [47]. Techniques like chunk-based accumulation and stochastic rounding further assist in reducing data and arithmetic precision. Addressing the deployment of Transformer models on edge TPU accelerators, Reidy et al. [37] provides a method for optimizing unsupported layers and computational graphs, enabling real-time inference on devices like Coral USB accelerator.

III. Proposed Approximation Schemes for Models

Deploying computationally complex vision models like CNNs, DNNs, and ViTs on edge devices requires a tradeoff that can balance performance and efficiency. Limited computing, memory resources, and battery capability can characterize these edge devices or environments. This section explores how adaptive approximation techniques can enhance and optimize these models for computing within a resource-constrained environment. We can reduce computational complexity and energy consumption by strategically applying approximate multipliers, variational inference (VI), training-aware quantization and post-training quantization while maintaining acceptable performance (an overview of the proposed methodology is shown in Figure 3). As CNNs, DNNs and ViTs architectures are supported by common and unique modules and operations such as convolution, fire modules, fully connected, and attention mechanisms, we first discuss approximation schemes suited for these modules and models by providing a high-level operational overview of proposed methods. Section IV then covers empirical tests and analyses of these mechanisms.

A. Approximate Multipliers

The *Approximate* operations are the fundamental component of this strategy, designed to facilitate multiplication with reductions in precision. We initialize lookup tables for 4-bit, 8-bit and 16-bit approximate multiplication to enhance the computation process. These tables, precomputed with the multiplication results for all possible pairs of 4-bit, 8-bit and 16-bit values, reduce computation time by eliminating the requirement for exact or precise multiplication calculations. To further optimize our approximation technique, we integrate the Signed Carry Disregard Multiplier (SCDM8) [41], [40], which uses 8-bit multiplication and strategically omits carry operations during various stages of computation. This operation reduces power consumption, aligning with our goal of achieving energy efficiency by trading computational accuracy. The lookup table is populated through nested loops: all possible combinations of two 4-bit numbers, ranging from 0 to 15, all 8-bit values (0 to 255), and the 16-bit values (0 to 65535). Each loop calculates and stores the approximate multiplication results in the corresponding lookup table. These tables provide resources for the computation steps, enhancing efficiency and speed. To efficiently compute approximate multiplications, we initialize an output matrix S, and apply precision-specific operations. For e.g., when either$A$ and B is 0, S is set to 0 to avoid extra computation. If A equals 1, S is assigned the value of B, utilizing the identity property of multiplication. For other values, the algorithm uses the lookup tables to use precomputed results, speeding up the computing operation.

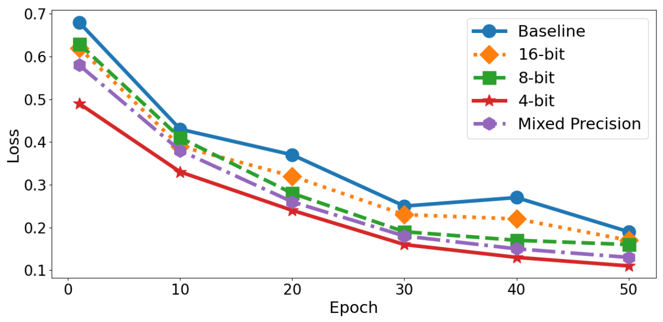


Fig. 4: Bit-wise comparison of training loss over epoch

The efficiency of the approximate strategy can be seen in Figure 4, using the model loss metric over epochs during training. The figure shows the training loss of the ResNet18 model for the first few epochs. We compare the baseline floating point precision of 32 bits with our proposed 16-bit, 8-bit, 4-bit and mixed precision. As shown in the figure, the model with baseline architecture and precision of floating point 32 starts with the highest loss values and shows a convergence between 40 and 50 epochs. Our proposed 8 and 16-bit precision shows similar convergence for the training loss. However, the optimal convergence can be seen in the mixed precision implementation. The overview presented in the plot for the model and the dataset can reflect alternate behaviour or characteristics, which can be optimized by performing reduced precision on sparse and non-contributing layers of model.
Mixed precision quantization optimizes deep neural network models for deployment across diverse computational platforms, particularly devices with limited computational resources like edge devices. As discussed above in the method

Algorithm 1 Multiplication for Different Precision Levels

```
0: procedure APPROXIMATE MULTIPLICATION($A$, $B$, precision)
0:   if precision = 16-bit then
1: $a_0 \leftarrow A[8:15]$, $a_1 \leftarrow A[0:7]$
2: $b_0 \leftarrow B[8:15]$, $b_1 \leftarrow B[0:7]$
3: $a_0b_0 \leftarrow$ appx_mul8x8$(a_0, b_0)$
4: $a_1b_0 \leftarrow$ appx_mul8x8$(a_1, b_0) \ll 8$
5: $a_0b_1 \leftarrow$ appx_mul8x8$(a_0, b_1) \ll 8$
6: $a_1b_1 \leftarrow$ appx_mul8x8$(a_1, b_1) \ll 16$
7: $S \leftarrow a_0b_0 + a_1b_0 + a_0b_1 + a_1b_1$
8: return $S$
8:   else if precision = 8-bit then
9: $a_0 \leftarrow A[4:7]$, $a_1 \leftarrow A[0:3]$
10: $b_0 \leftarrow B[4:7]$, $b_1 \leftarrow B[0:3]$
11: $a_0b_0 \leftarrow$ appx_mul4x4$(a_0, b_0)$
12: $a_1b_0 \leftarrow$ appx_mul4x4$(a_1, b_0)$
13: $a_0b_1 \leftarrow$ appx_mul4x4$(a_0, b_1)$
14: $a_1b_1 \leftarrow$ appx_mul4x4$(a_1, b_1)$
15: $S \leftarrow a_0b_0 + (a_1b_0 + a_0b_1) \ll 4 + a_1b_1 \ll 8$
16: return $S$
16:   else
17: return appx_mul4x4($A$, $B$)
17:   end if
17: end procedure
17: procedure APPX_MUL4X4($A$, $B$)
17:   if $A = 0$ or $B = 0$ then
18: return 0
18:   else if $A = 1$ then
19: return $B$
19:   else if $A > 1$ and $A$ is even then
19:     if $B < 8$ then
20: return 0
20:     else
21: return $32 \times (A/2)$
21:     end if
21:   else
21:     if $B < 8$ then
22: return $B$
22:     else
23: return $B + 32 \times (A-1)/2$
23:     end if
23:   end if
23: end procedure=0
```

and Algorithm 1 and 2, the concept is to use varying levels of precision across different neural network layers, adjusting the computational accuracy needed without causing a significant decline in overall model performance. This selective application of precision, utilizing lower precision arithmetic (such as 16-bit and 4-bit) for certain operations or layers while retaining higher precision (like 32-bit) for critical tasks, facilitates substantial reductions in memory usage, power consumption, and enhances computation speed due to fewer bits being processed. Algorithm 1 provides a detailed procedure for implementing approximate multiplications at various precision levels, ensuring each operation is executed with the optimal balance between speed and accuracy, which is crucial for maintaining performance in resource-constrained environments. These algorithms ensure the model remains within operational energy limits while achieving desired computational outcomes. Developing specialized hardware engineered to accelerate mixed precision matrix operations further supports the practical application of these algorithms, making them highly effective for deep-learning tasks. Algorithm 2 shows systematic process of multiplication with reduced energy consumption across different network layers by dynamically adjusting the precision level of multiplications as per predefined energy usage.

Algorithm 2 Energy-Efficient Multiplication for NN

```
Require: NN Model, Initial weights, Energy budget $E$
Ensure: Optimized NN with approximate multiplications
1: Initialize global energy usage to zero
2: Define energy costs for different precision levels (4-bit, 8-bit, 16-bit, 32-bit)
3: for each layer in the Neural Network do
4:   for each multiplication operation involving weights $A$ and $B$ do
5:     if operation requires 32-bit precision then
6:       Perform FP_appx_mul($A$, $B$)
7:       Convert $A$ and $B$ to binary floating-point values
8:       Compute sign, exponent, and mantissa
9:       Use appx_mul16x16 for mantissa multiplication
10:      Adjust exponent and construct final result
11:    else if operation can use 16-bit precision then
12:      Perform appx_mul16x16($A$, $B$)
13:    else if operation can use 8-bit precision then
14:      Perform appx_mul8x8($A$, $B$)
15:    else
16:      Perform appx_mul4x4($A$, $B$)
17:    end if
18:    Calculate energy usage for the operation
19:    if cumulative energy usage $\leq E$ then
20:      Update matrix product in layer
21:      Update global energy usage
22:    else
23:      Revert to higher precision operation
24:    end if
25:  end for
26:  Assess layer performance impact
27: end for
28: OptimizedNN $\leftarrow$ NN with approximate multiplications
29: return OptimizedNN =0
```

Algorithm 3 shows an approach for optimizing Vision Transformer (ViT) models by strategically reducing the precision of weights and operations to enhance energy efficiency while maintaining performance. The algorithm initializes the global energy usage and defining multiple precision levels along with their associated energy costs. For each transformer block in the model, it determines the block's position whether it is near the input, middle, or output layers and sets a base precision level accordingly. Within each block, the algorithm adjusts the precision of components based on their function. For the attention components, it assigns 8-bit precision to the Query (Q) and Key (K) matrices, 16-bit precision to the Value (V) matrices, and 4-bit precision to intermediate attention score computations, applying block-wise quantization for consistency. For the Multi-Layer Perceptron (MLP) components, it uses 16-bit precision for the first feed-forward network (FFN) layer to preserve essential features, and 8-bit precision with stochastic rounding for subsequent layers to reduce computational load.

During matrix operations involving weights A and B, the algorithm applies predefined precision rules when either A or B is zero, or when the layer is involved in patch embedding or positional encoding. Otherwise, it employs optimized multiplication based on the designated precision, updating the cumulative energy usage after each operation. If the energy usage exceeds the predefined budget E, it reverts the last operation to a higher precision to stay within limits. The algorithm further refines the model by applying structured sparsity to attention matrices and pruning redundant heads based on attention entropy measurements. LayerScale calibration is used to adjust layer scaling parameters for stability during training with reduced precision. Finally, the optimized model is validated using vision-specific performance metrics to ensure that accuracy remains acceptable, and the optimized ViT model with mixed-precision weights is returned.

Algorithm 3 Energy-Efficient Precision Reduction for ViTs

```
Require: ViT Model, 32-bit weights, Energy budget $E$
Ensure: Optimized ViT with mixed precision weights
 1: Initialize global energy usage to zero
 2: Define precision levels and energy costs
 2: for each transformer block do
 3: Determine block position (near input, middle, or output)
 4: Set base precision based on block position
 4:    for each component in [attention, MLP] do
 4:       if component is attention then
 5: Use 8-bit for Q, K; 16-bit for V
 6: Use 4-bit for attention score intermediates
 7: Apply block-wise quantization
 7:       else if component is MLP then
 7:          if first FFN layer then
 8: Use 16-bit precision
 8:          else
 9: Use 8-bit with stochastic rounding
 9:          end if
 9:       end if
 9:       for each matrix operation with weights $A$, $B$ do
 9:          if $A = 0$ or $B = 0$ or IsPatchEmbed(layer) or
    IsPosEncode(layer) then
10: Apply predefined precision rule
10:          else
11: Apply optimized multiplication based on precision
11:          end if
12: Update energy usage
12:          if energy usage $> E$ then
13: Revert to higher precision for this operation
13:          end if
13:       end for
13:    end for
13: end for
13: Apply structured sparsity to attention matrices
13: Prune redundant heads based on attention entropy
13: Apply LayerScale calibration for stability
13: Validate on vision-specific metrics
13: return Optimized Vision Transformer =0
```

B. Approximate multiplication for convolutions

One of the most common and computationally intensive components in neural network are convolutional layers [10], [17]. A convolution operation within CNN can be described as $Z = X \circledast W$. where X is the input image, and W is the kernel. The equation can be described as:

$$z_{m,n} = \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} x_{m+i,n+j} \cdot w_{i,j} \quad (1)$$

Here (m,n) are pixel coordinates of an image, and k_h, k_w is the respective height and width of the kernel or filter. As the operation involves multiplication and summation, an approximate operation of convolutional layers during training and inference can reduce computing and memory load from input and output feature maps. A form of approximation for convolution [17] with less multiplication can be described as:

$$z_{m,n} \approx \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} \mu_{|\hat{w}|} \cdot \min(x_{m+i,n+j}, \hat{w}_{i,j}) \quad (2)$$

As an approximation strategy, we propose to approximate the convolution operation (described in equation 2) using signum functions combined with bit-shifting techniques to reduce computational complexity, especially the multiplicative complexity. The revised approximate operation can be described as follows:

$$z_{m,n} \approx \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} \mathrm{sgn}(x_{m+i,n+j}) \times \mathrm{sgn}(\hat{w}_{i,j}) \times 2^{\mathrm{shift}} \quad (3)$$

- **Signum Multiplication:** The signum function, described as $\mathrm{sgn}(x)$, returns +1 for positive values, -1 for negative values. It is applied to both the image pixel $x_{m+i,n+j}$ and the kernel weight $\hat{w}_{i,j}$. The multiplication of two signum values simplifies to a sign check: if the signs are the same, the result is +1; if different then it is -1. This approach reduces complex floating-point multiplication to a integer operation, thus decreasing computational overhead.
- **Bit-Shifting:** The term 2^{shift} describes a left bit-shift operation, equivalent to multiplication by a power of 2. The 'shift' parameter is optimized during model training. This operation scales the output using only integer arithmetic, which is computationally less complex than floating-point multiplication. It allows for efficient adjustment of the magnitude of the convolution output without the exact multiplication operations.

This approach reduces the convolution operation to basic sign comparisons and bit shifting, thus lowering the overall computational demand. To implement this operation, the training process requires adaptations such as gradient clipping to accommodate the coarse approximation. The complexity analysis of the standard and approximate convolution methods shows interesting trade-offs between computational efficiency and potential accuracy. While all three approaches has a theoretical time complexity of O$(k_h * k_w * H * W)$, their practical performance on GPUs varies. The standard convolution utilizing full-precision floating-point operations serves as baseline. The min-based approximation method offers a moderate improvement in computational efficiency, from 1.2 to 1.5 times faster than the standard approach, by replacing multiplications with simpler operations. However, the signum and bit-shift based approximation shows 2.5 to 3.0 times speedup compared to baseline. This performance gain is also a result of using integer operations compared to complex floating-point operations, which are well supported in GPU architectures. Memory requirements is similar across all methods, with the signum-based approach potentially gaining from lower precision storage. It's important to note that while the approximate methods, especially the signum-based one, provide improve speedup, they may introduce accuracy trade-offs. The method is suitable for prioritizing efficiency and speed over precision in hardware-constrained environments like embedded systems or mobile devices, where computational resources are limited.

C. Variational Inference

In this strategy, we apply probabilistic approximation based on Occam's Razor to Vision Transformers with the goal of reducing computational complexity and improving speedup. The general goal of variational inference is to approximate the true posterior distribution of weights by optimizing an approximate distribution and focusing on estimating uncertainty linked to model predictions. We reformulate this objective to prioritize computational efficiency using Occam's Razor-inspired loss [43]. This approach balances model fit and complexity, directly addressing the trade-off between performance and efficiency in Vision Transformers. The weights in Vision Transformers are treated as random variables rather than fixed values. The objective is to closely approximate the posterior distribution of these weights based on observed data while penalizing excessive model capacity. Instead of traditional variational inference approach which is based on the Kullback-Leibler (KL) divergence and maximizing the Evidence Lower Bound (ELBO), in our proposed strategy, we optimize the objective as follows:

$$\mathcal{O} = -\log p(X|\hat{\theta}) + \log \int_{\mathcal{M}} \kappa(\theta) \exp\left(-\frac{N}{2}(\theta-\hat{\theta})^\top J(\hat{\theta})(\theta-\hat{\theta})\right) d\theta \quad (4)$$

Here X is the input data, $\hat{\theta}$ is the maximum likelihood estimate of the weights, and $J(\hat{\theta})$ is the Fisher Information Matrix, which approximates the curvature of the log-likelihood around $\hat{\theta}$. The $\kappa(\theta)$ is the capacity of the model, penalizing over-parameterization and ensuring efficient model representation. This objective ensures that the model achieves a balance between fitting the data and maintaining a compact representation. By optimizing O the model reduces computational complexity and improves robust performance in tasks such as object detection, segmentation, or classification. Using the strategy, we propose model training process for object detection enhanced with post-training bit quantization for improved computational efficiency:

Algorithm 4 Variational Inference with Post-Training Bit Quantization (VIPT: Occam's Razor-based)

```
Require: Training dataset $D$, learning rate $\alpha$, number of epochs $T$
Ensure: Trained object detection model $M$
 1: Initialize variational distribution $q(w)$
 2: Define optimizer with adjustable learning rate $\alpha$
 3: Use loss function $\mathcal{L}_{Occam}$ as described in Eq. (4)
 4: for $t = 1$ TO $T$ do
 5:     Shuffle $D$
 6:     Initialize total loss $\mathcal{L}_{total} = 0.0$
 7:     for $start = 0$ TO $len(D)$ STEP $batch\_size$ do
 8:         Compute loss $\mathcal{L}(M, D[start : start + batch\_size])$ using weight sampling
 9:         Backpropagate
10:         Update parameters $\rightarrow$ optimizer
11:         Add $\mathcal{L}$ to $\mathcal{L}_{total}$
12:     end for
13:     $\mathcal{L}_{avg} = \mathcal{L}_{total}/(len(D)/batch\_size)$
14:     Display $\mathcal{L}_{avg}$
15: end for
16: Quantize $M$ to 8 bits using an 8-bit quantizer
17: Replace F.P. multipliers in $M$ with 8-bit a.x. multipliers
18: Save trained model $M$ =0
```

Based on the algorithm 4, we propose training process that integrates the Occam's Razor-inspired loss combined with post-training bit quantization, with the goal of improving computational efficiency. During training, weights are optimized using the Occam's Razor-inspired loss $\mathcal{L}_{Occam}$, which balances model fit and complexity. This approach ensures efficient model convergence while maintaining the capacity to generalize from training data to unseen data. By penalizing excessive model complexity through the Fisher Information Matrix $J(\hat{\theta})$, the model achieves a compact representation, enhancing computational efficiency without significant loss of performance. The integration of 8-bit post-training quantization further reduces memory usage and computational processing requirements, providing a balance between efficiency and precision for accurate object detection. The impact of reduced precision on detection performance is systematically evaluated to ensure that any degradation in accuracy stays within acceptable limits.

The custom loss function (equation 4) includes both localization and classification components, ensuring the model learns to detect and classify objects effectively within an image. Fine-tuning the loss parameters allows the model to adapt to variations in object scale and shape, making it suitable for real-world scenarios where performance, robustness and efficiency are essential.

As an alternative approach and to have a detailed comparison with energy-aware training methods, we propose a strategy to optimize energy usage during the training of object detection models. This approach integrates probabilistic energy-efficient training with an Occam's Razor-inspired objective to reduce energy consumption while maintaining model accuracy. The method incorporates a customized loss function that balances model fit and complexity, penalizing excess capacity using the Fisher Information Matrix $J(\theta)$. The training process uses a variational distribution $q(w)$, representing the probabilistic distribution of model weights. Unlike traditional energy-aware methods, this approach dynamically adjusts model parameters and energy consumption metrics to balance performance and computational cost. The proposed loss function for energy-aware training is:

$$\mathcal{L}_{Occam-Energy} = -\log p(X|w) + \frac{\lambda}{2} w^{\top} J(w) w + \gamma EE_{stats}, \quad (5)$$

Here $\log p(X|w)$ represents the data likelihood, $J(w)$ is the Fisher Information Matrix approximating the curvature of the log-likelihood, λ controls the regularization strength on the model weights, and γ scales the energy consumption penalty EE_{stats}. Algorithm 5 covers the model training process, integrating real-time energy monitoring and loss-based parameter updates. The algorithm manages energy consumption through an energy statistics variable, EE_{stats}, which allows for dynamic adjustments to training parameters. This ensures that the model operates within predefined energy budgets. If energy usage exceeds the threshold E_{max}, the learning rate is reduced to minimize computational overhead while sustaining effective training progress. By incorporating the Fisher Information Matrix into the loss function, the model discourages over-complexity, promoting efficient resource utilization without compromising generalization. The energy statistics variable, EE_{stats}, is computed using real-time energy profiling, ensuring dynamic adjustments during training. Both regularization strength λ and energy penalty coefficient γ were optimized through a grid search over a validation set. **Regularization Strength (λ)**: Controls the impact of the regularization term. Smaller λ values allowed higher model capacity but increased the risk of overfitting, whereas larger values constrained the model, improving generalization at the cost of training accuracy. **Energy Penalty Coefficient (γ)**: determines the weight of the energy penalty. Higher γ values effectively reduced energy consumption but introduced minor accuracy degradation (typically within 1-2% mAP). Lower values balanced accuracy and energy use, but energy consumption was higher.

By fine-tuning λ and γ, the model achieves a balance between energy efficiency and performance. For example, models trained with higher γ shows significant energy savings but reduction in accuracy. Regularization through λ improved generalization while maintaining convergence stability. These trade-offs shows the flexibility of the proposed framework in adapting to different performance and resource constraints, making it suitable for real-world edge applications. These statistics are measured using a energy monitoring tool such as nvml or tegrastats, providing accurate feedback on power consumption for each training batch. The custom loss function, $\mathcal{L}_{Occam-Energy}$, integrates this information to penalize excessive energy usage while maintaining a balance between model complexity and fit. After training, the model can be further optimized through quantization, as discussed in the algorithm 4, ensuring that both training and inference are computationally efficient on edge devices.

Algorithm 5 Variational Inference-Based Energy-Efficient Training (VIET: Occam's Razor-Inspired)

```
Require: Training dataset $D$, learning rate $\alpha$, epochs $T$, batch size $b$, energy budget $E_{max}$
Ensure: Trained model $M$, Energy Stats $EE$
 1: Initialize variational distribution $q(w)$
 2: Set optimizer with learning rate $\alpha$
 3: Define custom loss function $\mathcal{L}_{Occam-Energy}$
 4: for $t = 1$ to $T$ do
 5:     Shuffle $D$
 6:     $\mathcal{L}_{total} \leftarrow 0$, $EE_{total} \leftarrow 0$
 7:     for $start = 0$ to len($D$) step $b$ do
 8:         Sample weights $w$ from $q(w)$
 9:         Compute loss $\mathcal{L}_{Occam-Energy}(w)$ on batch $D[start : start + b]$
10:         Backpropagate and update weights using $\nabla\mathcal{L}_{Occam-Energy}(w)$
11:         $EE_{batch} \leftarrow$ Compute energy usage for current batch
12:         Update energy statistics: $EE_{total} \leftarrow EE_{total} + EE_{batch}$
13:         $\mathcal{L}_{total} \leftarrow \mathcal{L}_{total} + \mathcal{L}_{Occam-Energy}(w)$
14:     end for
15:     Compute average loss $\mathcal{L}_{avg} = \mathcal{L}_{total}/(\text{len}(D)/b)$
16:     Compute average energy consumption $EE_{avg} = EE_{total}/(\text{len}(D)/b)$
17:     if $EE_{avg} > E_{max}$ then
18:         Update learning rate $\alpha \leftarrow 0.9 \times \alpha$
19:     end if
20:     Print $\mathcal{L}_{avg}$, $EE_{avg}$
21: end for
22: return $M$, $EE$ =0
```

IV. Implementation and Evaluation

We perform training and inference for object detection tasks to evaluate the proposed approximation strategies using the nuScenes and Waymo datasets. The evaluation focuses on balancing model performance and energy efficiency through the three strategies (*approximate multipliers*, *approximate convolution operations*, and *variational inference* (using both VIET and VIPT)) discussed in Section 3, including the Occam's Razor-inspired loss (Eq. 7). This section covers the test and experimental setup, including dataset preparation, hyperparameter tuning, and validation methodology.

A. Training Method

The nuScenes and Waymo datasets include diverse class and environmental conditions, making them suitable for assessing model performance and energy efficiency. For a fair comparison across all models, we initialized the hyperparameters as: learning rate of 0.001, batch size of 32, and weight decay of 0.0001 for CNN models. For ViT we configured the training with a batch size of 32 and an initial learning rate of 1e-3, following a linear decay to 1e-5. The model was trained for 100 epochs using the Adam optimizer. A dropout rate of 0.1 and a weight decay of 0.03 is applied to avoid overfitting. During training, key hyperparameters such as learning rate, batch size, and epochs are systematically tuned to ensure model stability and convergence. The iterative training process allows the model to progressively adapt to the dataset's complexity while maintaining an energy-efficient profile. Each of the three proposed methods was implemented with specific configurations to optimize performance and energy efficiency:

Method 1) Approximate Multipliers: In our evaluations using the nuScenes and Waymo datasets, we integrated Approximate Multipliers into the training and inference stages of both Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). This method utilizes precomputed multiplication outcomes from lookup tables for 4-bit, 8-bit, and 16-bit values, which speed up computations while minimizing energy usage. Lookup tables for 4-bit and 8-bit contain 256 and 65,536 approximately. For example, multiplying two 4-bit numbers, such as 3 (0011) and 5 (0101), obtains the result directly from the table, overlooking the need for real-time computation. Similarly, 8-bit and 16-bit tables offer rapid results for more complex multiplications, essential for faster computation times, especially in the deeper network layers.

We adapted mixed precision in CNNs and ViTs to optimize computational efficiency and accuracy. In CNNs, initial layers that process basic features such as edges and textures use lower precision (4-bit and 8-bit), while deeper layers that integrate complex features employ 16-bit precision to preserve detail and accuracy. For ViTs, which generally depend on self-attention mechanisms, the Query (Q) and Key (K) matrices use 8-bit computations to accelerate attention calculations, whereas the Value (V) matrix uses 16-bit precision to ensure the integrity of information processing. Intermediate calculations, such as attention scores, are performed with 4-bit precision to maximize efficiency. Our experimental results show that approximate multipliers can reduce computational time by up to 38% compared to traditional methods, with a minor reduction in accuracy, typically less than 4%. This approach lowered energy consumption and improved speed up in training, particularly for energy-intensive operations within deeper network layers. Table IIIa and Table IIIb and Figure 9, shows the balance between performance and power efficiency using precision reduction and energy saving.

Method 2) Approx multiplication for Convolution: As an approximation strategy, we previously proposed to approximate the convolution operation using signum functions combined with bit-shifting techniques. This approach can be efficiently implemented using a simple lookup table, as shown in Table I. Based on the look-up-table approach, the revised approximate operation can be described as follows:

$$z_{m,n} \approx \sum_{i=1}^{k_h} \sum_{j=1}^{k_w} LUT(x_{m+i,n+j} \cdot \hat{w}_{i,j}) \tag{6}$$

Here $LUT(x)$ refers to the lookup operation based on Table I. This approach effectively combines the signum function and bit-shifting into a single, efficient operation. *Signum Lookup:* The table simplifies the signum function to a binary decision based on the sign of the input, eliminating the need for explicit sign checks. *Integrated Bit-Shifting:* The 'bits' shift in the table output includes the 2^{shift} factor directly into the lookup result. This shift value is determined during the training process to optimize the approximation. This LUT-based implementation reduces the computational complexity of the convolution operation. It replaces the signum calculation and the bit-shifting with a single table lookup, which can be extremely efficient on most hardware architectures.

TABLE I: Look-up-table for simplified conv multiplication

Input Range	Output (Signum Output Shifted)
$-\infty < x < 0$	-1 shifted by appropriate bits
$0 \leq x < \infty$	$+1$ shifted by appropriate bits

Method 3) Variational Inference: As mentioned in section 3 includes two mechanisms: *VIET* (Variational Inference-Based Energy-Efficient Training) and *VIPT* (Variational Inference with Post-Training Bit Quantization). VIET implemented the customized loss function $\mathcal{L}_{Occam}$ (Equation 7) to balance classification and regression losses with a capacity penalty, thereby preventing over-parameterization. This approach utilized variational inference to model weight distributions and integrated energy-aware training to optimize resource utilization. Subsequently, VIPT applied 8-bit quantization post-training, as outlined in Algorithm 4, to further reduce memory usage and computational demands. This dual mechanism ensured the model achieved compression and energy efficiency without compromising detection performance.

TABLE II: Lookup Table for Activation Outputs in VI Strategy

Normalized Input Range	Probabilistic Output Distribution
$-1.0 \leq x < -0.8$	$\mathcal{N}(-0.9, 0.05)$
$-0.8 \leq x < -0.6$	$\mathcal{N}(-0.7, 0.05)$
$-0.6 \leq x < -0.4$	$\mathcal{N}(-0.5, 0.05)$
$-0.4 \leq x < -0.2$	$\mathcal{N}(-0.3, 0.05)$
$-0.2 \leq x < 0$	$\mathcal{N}(-0.1, 0.05)$
$0 \leq x < 0.2$	$\mathcal{N}(0.1, 0.05)$
$0.2 \leq x < 0.4$	$\mathcal{N}(0.3, 0.05)$
$0.4 \leq x < 0.6$	$\mathcal{N}(0.5, 0.05)$
$0.6 \leq x < 0.8$	$\mathcal{N}(0.7, 0.05)$
$0.8 \leq x \leq 1.0$	$\mathcal{N}(0.9, 0.05)$

Table II describes the probabilistic distributions with predefined ranges of normalized input values. Each range maps to a normal distribution defined by a mean (μ) and variance (σ^2), where the mean aligns with the midpoint of the input range. This mapping is particularly useful for models where activations are expected to vary within known limits based on the input they process. Using this table in the VIET and VIPT strategy, the models can utilize these predefined distributions to approximate neuron activations without recalculating the

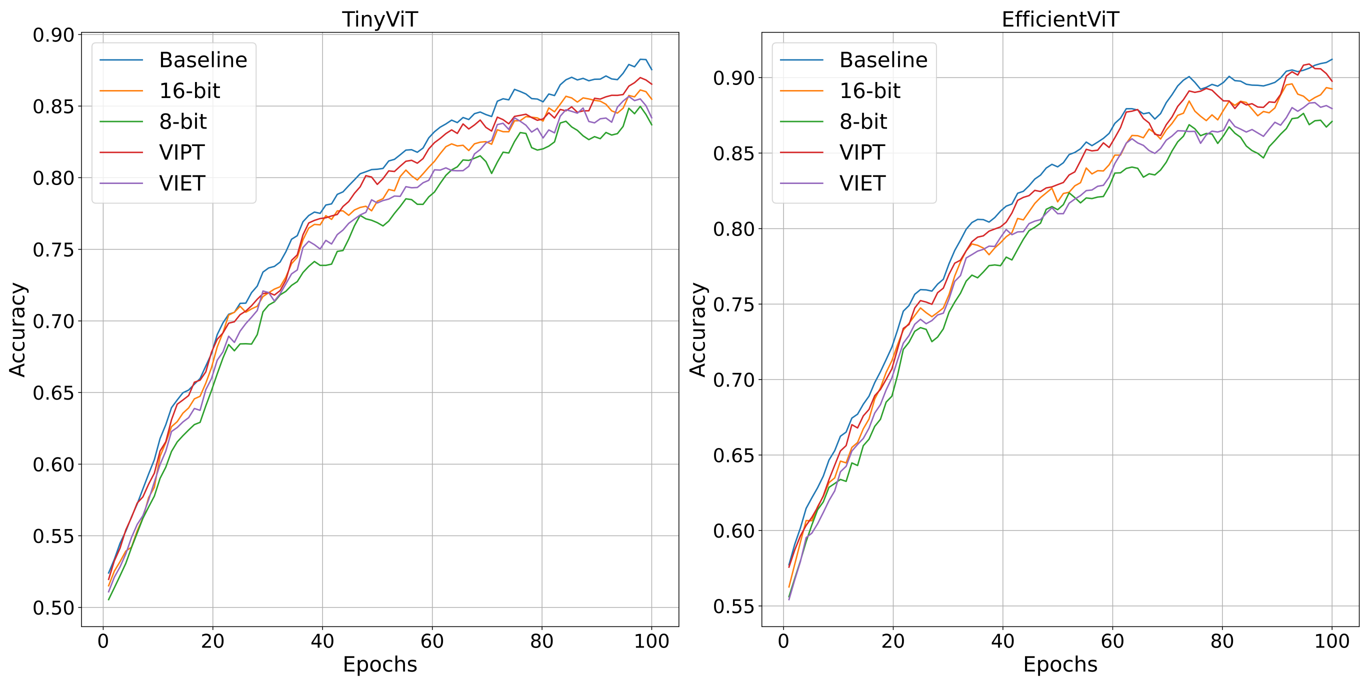


Fig. 5: ViTs model training accuracy using proposed approach

distributions for each input during runtime. This improves processing speed and optimizes energy usage, making the system ideal for deployment in environments with limited power and computational resources. For the VIPT strategy, this table guides the quantization process by suggesting which activations need higher precision based on their variance, thus preserving model accuracy. Utilizing this lookup table provides an efficient approach to balancing computational demands with the accuracy requirements of advanced neural network architectures. During model training, the overall training loss function used for the variational inference strategies, VIET (Variational Inference-Based Energy-Efficient Training) and VIPT (Variational Inference with Post-Training Bit Quantization) addresses classification accuracy, regression precision and model complexity while optimizing for energy efficiency, which complements the loss functions discussed in subsection (variational inference). The function is described as:

$$\mathrm{L}_{Occam}(y,t,\mu,\log(\sigma^2)) = \alpha\cdot\mathrm{CL}(y_{\text{cls}},t_{\text{cls}}) + \beta\cdot\mathrm{RL}(y_{\text{reg}},t_{\text{reg}}) + \lambda\cdot\text{CapacityPenalty}(\mu,J(\mu)) \quad (7)$$

Here $\mathrm{CL}(y_{\text{cls}},t_{\text{cls}})$ is the classification loss (cross-entropy). $\mathrm{RL}(y_{\text{reg}},t_{\text{reg}})$ is the regression loss (smooth L_1 loss). As mentioned earlier, $\text{CapacityPenalty}(\mu,J(\mu))$ penalizes over-parameterization using the Fisher Information Matrix $J(\mu)$. The hyperparameters α, β, and λ are fine-tuned to balance these components, ensuring that the model achieves high accuracy while conserving energy. The following subsections discuss quantitative results through proposed methods.

B. Training and Inference Results

We trained the DNN models and vision transformers for a detailed analysis using methods discussed in the previous subsection. We have trained the TinyViT and EfficientViT models using the methods discussed in the previous subsection. Since the second strategy (approximate multiplication for convolutions) can generally be used with the DNNs or vision transformers majorly consisting of convolutional layers, we interchangeably used the discussed approximation strategies during training and inference for evaluation and analysis. Figure 5 shows the training accuracy of two models, TinyViT and EfficientViT, for 100 epochs under different precision settings: Baseline (FP32), 16-bit, 8-bit, VIPT, and VIET. For both models, the Baseline configuration consistently shows the highest accuracy throughout the training; the 16-bit and 8-bit precision reduction maintain or are within the baseline range, with the 16-bit maintaining higher accuracy than the 8-bit multiplier, showing the expected trade-off between computational efficiency and accuracy due to reduced precision. VIPT and VIET strategies show a balanced tradeoff by optimizing model performance and computational resources. While these models have reduced accuracy compared to the baseline, they have a similar accuracy trend to 8-bit and 16-bit multipliers. This pattern suggests that these methods, involving variable or targeted precision techniques, offer a balanced approach for deployment on non-specialized hardware or edge devices.

Table IIIa shows precision metrics for vision models on a subset of the nuScenes dataset. Analyzed models include ResNet18, ResNet34, TinyViT, and EfficientViT under different conditions such as baseline, 8-bit and 16-bit, Ax-Conv (approximate multiplication for convolution) and methods applying variational inference like VIPT, and VIET. Precision values show transformer models (TinyViT and EfficientViT) perform better than ResNet models in baseline settings. When precision is reduced to 8-bit, all models show a drop in precision, with the least impact on EfficientViT. The 16-bit precision

TABLE III: Results for nuScenes subset using approximate multipliers and variational inference

(a) Precision results

Method	**ML Models for object detection tasks**			
	ResNet18	***ResNet34***	***TinyViT***	***EfficientViT***
Baseline	34.3	39.1	52.7	55.9
8-bit	22.7	29.5	44.1	49.9
16-bit	24.3	34.7	49.5	51.8
Ax-Conv	29.1	30.8	-	-
VIPT	33.9	38.2	51.3	54.7
VIET	33.4	37.4	49.7	52.1

[a]Ax-Conv: Variational inference on convolution operations
[b]VIPT: Variational inference with post-training quantization
[c]VIET: Variational inference with energy-aware training

(b) Model Performance

Model	**Metrics Measurements**			
	Param	***Flops (G)***	***MACs***	***Latency (ms)***
ResNet18	11.7	1.8	1.9	14
ResNet34	21.8	3.7	3.1	26
TinyViT-1	25.4	2.0	4.6	21
TinyViT-2	8.1	0.5	2.1	17
EfficientViT-1	40	1.5	3.4	39
EfficientViT-2	13.4	1	2.0	23

[a]Latency: Tested over Jetson Xavier NX
[b]Model-1: Trained using VIPT
[c]Model-2: Trained using VIET

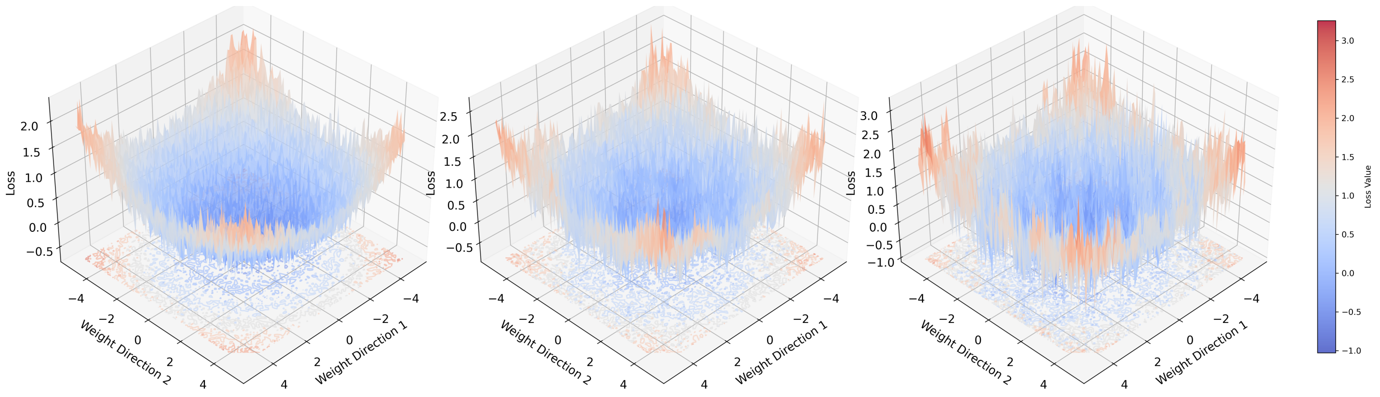


Fig. 6: EfficientViT models (baseline, variational inference) training loss on the nuScenes dataset

provides better results than 8-bit, showing less data loss. Ax-Conv performs better than both 8-bit and 16-bit settings by preserving more accuracy for ResNet models; however, this behaviour can be attributed to the heavy matrix multiplication and convolution operation. VIPT and VIET show results close to baseline, suggesting effectiveness in maintaining precision during quantization and training. Table IIIb compares model performance based on parameters, FLOPs, MACs, and latency metrics. Model complexities vary significantly, from TinyViT-2's lower parameter count to EfficientViT-1's higher count. ResNet34 shows the highest FLOPs, whereas TinyViT-2 is the most compute-efficient. MACs correlate with FLOPs across all models, with TinyViT-1 having the highest MACs value. For model inference overview, we measure the above-mentioned metrics and latency on the Jetson Xavier NX with a range from 14 ms for ResNet18 to 39 ms for EfficientViT-1, showing that higher model complexities do not necessarily correspond to increased latency, suggesting optimizations in architecture and further possibility of hardware-aware optimization.

Figure 6 shows the loss landscapes of the EfficientViT model under three different training strategies: baseline, Variational Inference with Post-Training Quantization (VIPT), and Variational Inference with Energy-Aware Training (VIET). The left plot is the baseline model, showing the loss landscape with few variations and peaks. These loss patterns are standard across optimized neural networks such as EfficientViT, showing areas where the model parameters adjust to minimize the loss effectively, with a balance between parameter stability and sensitivity. The middle plot shows the model trained using the VIPT strategy; the loss shows variation as compared to the baseline model. The plot also shows the effect of quantization as it stabilizes the loss landscape, though with a slight increase in peak values. This trend shows that while quantization helps in general stabilization, it also restricts the model's ability to reach the lowest possible loss values within the quantized parameter space. The right plot shows the loss landscape from the VIET training approach. Compared to the baseline and VIPT, this landscape has higher peaks and loss values, showing the challenges in achieving optimization. This trend or pattern can be associated with the trade-offs made for energy efficiency, where adjustments to model parameters or operational precision are made at the runtime cost of smooth optimization. Overall, the visualized loss landscapes in figure 6 provide insight into how each training method affects the model's optimization dynamics. VIPT provides a balance by smoothing out sharp gradients, whereas VIET introduces complexities that prevent loss minimization. Additional strategies such as adjusting regularization, exploring other energy-efficient modifications, or adapting learning rates might be necessary to optimize training further under VIET strategy.

Figure 7 compares the weight matrices of the EfficientViT model under three configurations: baseline, 16-bit quantization, and mixed precision using the first proposed approximation strategy (approximate multipliers: algorithm 3). The attention layer shows minimal visual change despite different precision levels, maintaining a clear diagonal pattern which signifies self-attention, with a little granularity under reduced precision. In the MLP layer, the transition from full to lower precision levels shows a more pronounced pixelation, showing the impact of the quantization process, with information loss

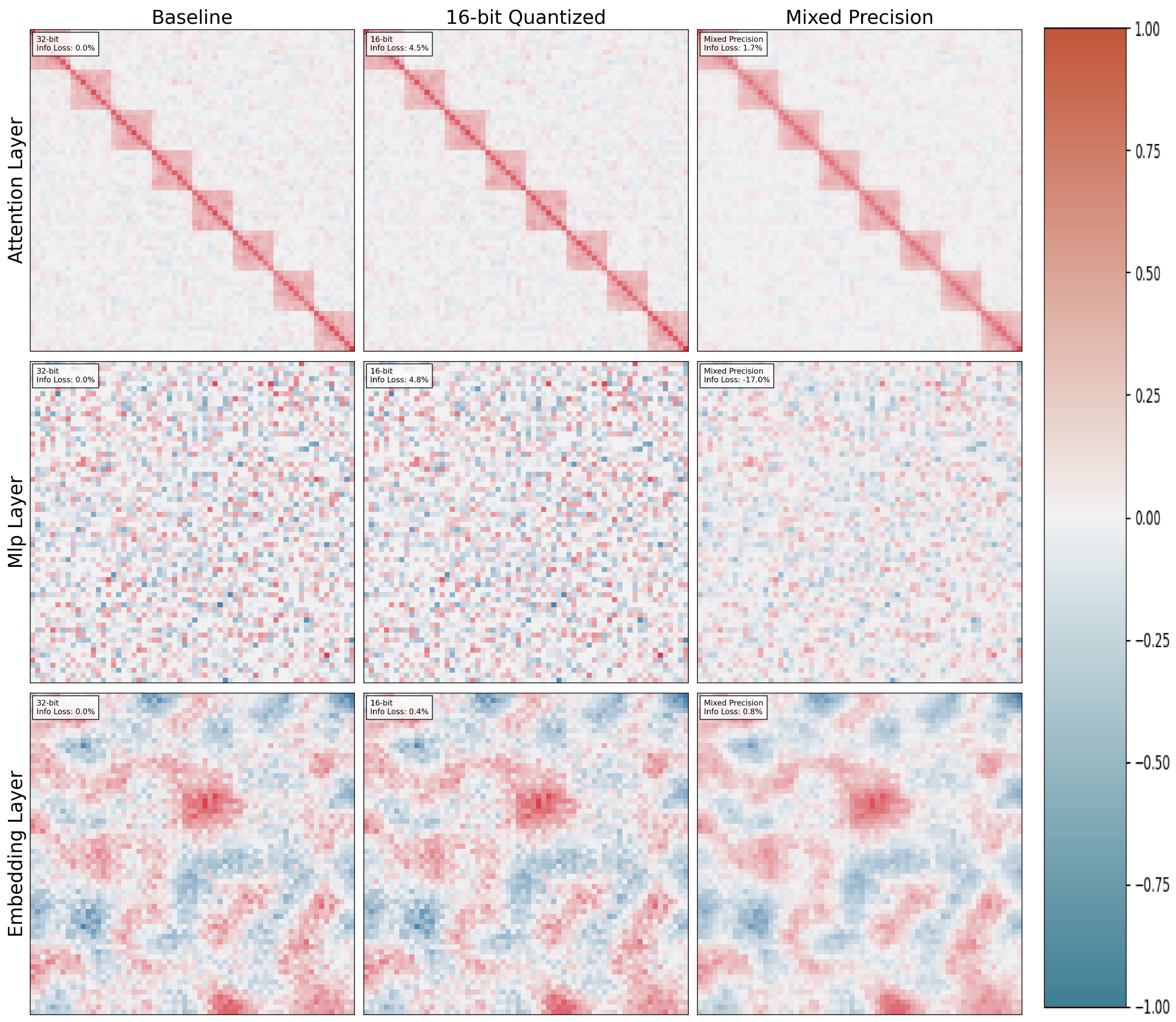


Fig. 7: EfficientViT weights distribution on different precision levels and layers

increasing in mixed precision. The embedding layer has the most differences, where lower precision results in visibly coarser gradients, affecting the representation of spatial correlations within the weights. Information loss is also higher in mixed precision, confirming a more substantial impact on the layer's pattern. This comparison from full precision through various quantization levels visually shows the impact of precision reduction on weight matrix structure across different model components.

V. Results Analysis

This section analyses multiple benchmark models using approximation strategies over nuScenes and Waymo validation set. We also run models for a single inference on Xavier NX to capture energy consumption, model performance and system metrics for a valid comparison.

Energy Consumption: When considering the energy consumption of these models on a Xavier NX, we must account for the device's power draw and the time taken to perform a single inference. Under the assumption that the Jetson Xavier NX draws an average power of 15 watts during inference, which is a mid-range estimate for this device—the energy usage for each model can be calculated by multiplying this power with the inference time (latency). The energy consumed E can be:

$$E = P \times t$$

Where P is the power usage in watts, and t is the time in seconds. As shown in Figure 9, the ResNet18 model trained using the approximation consumes about 0.21 microjoules of energy per inference with a latency of 14 milliseconds. Similarly, ResNet34, with a longer latency of 26 milliseconds, uses approximately 0.39 microjoules per inference. We observe a difference for TinyViT models (Model-1 is trained using

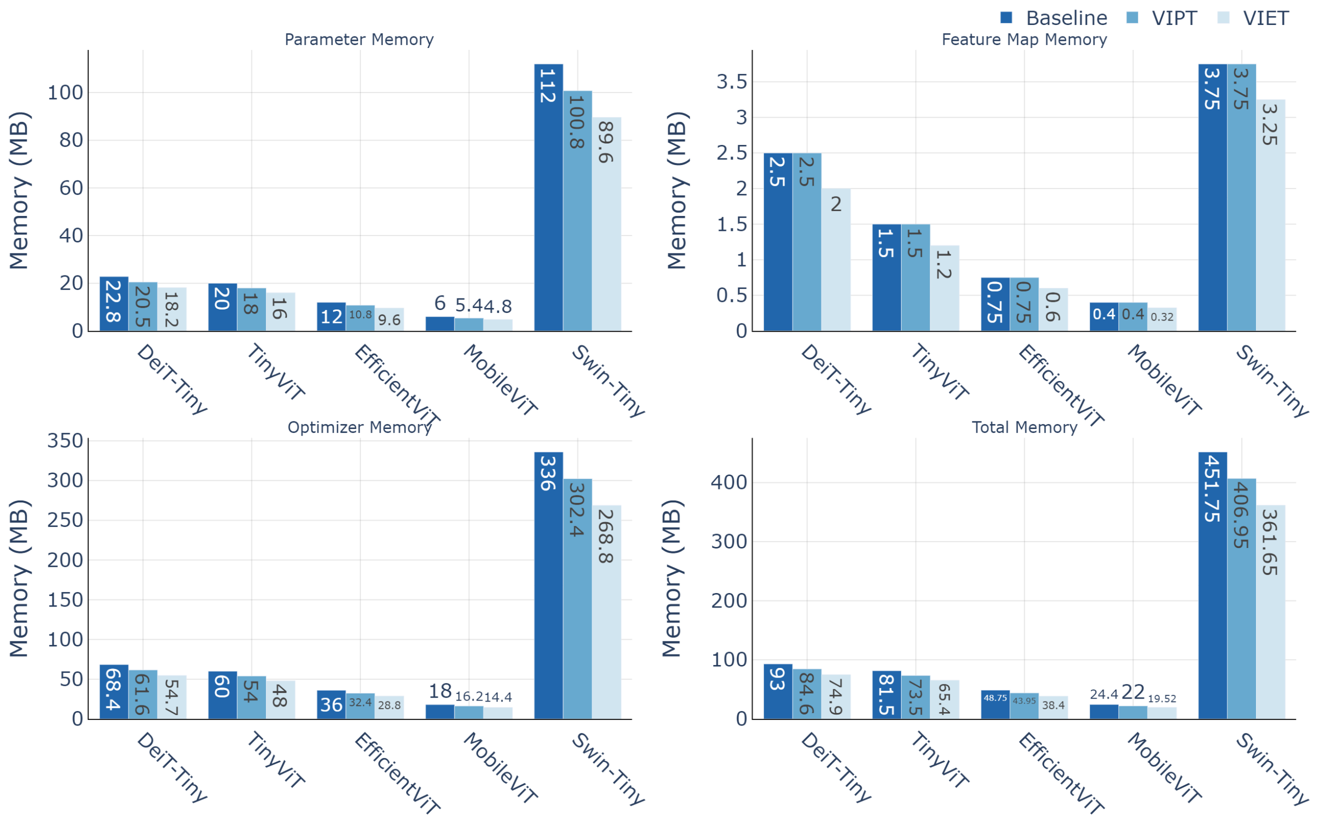


Fig. 8: Memory comparison of vision transformer models with variational inference strategies

VIPT, and Model-2 is trained using VIET). TinyViT-1, with a 21-millisecond latency, consumes around 0.315 joules, while the more efficient TinyViT-2, with its 17-millisecond latency, only use about 0.255 joules per inference. These results show that even within the same family of models, architectural differences and optimization approaches can lead to on-board energy savings. The EfficientViT-1 shows a higher latency of 39 milliseconds, resulting in an energy consumption of about 0.585 joules per inference, higher than any of the models discussed. EfficientViT-2, with a 23-millisecond latency, uses about 0.345 joules, showing a similar trend as the TinyViT-2 model. It's important to note that these energy results are based on static power draw values recorded by nvml and tegrastats manager of the device, while actual energy consumption would fluctuate because of the computational and memory load at any given moment.

Memory Footprint: Figure 8 gives an overview of the memory footprint across different models and optimization strategies and shows improvements in memory efficiency through the application of VIPT (Variational Inference with Post-Training Quantization) and VIET (Variational Inference with Energy-Aware Training) strategies. The baseline measurements show the actual memory demands of each model, with Swin-Tiny having the highest memory requirement at approximately 451-452 MB due to its extensive parameter and optimizer requirements. Models like MobileViT (xx-small) have minimal memory usage, around 24-25 MB, suitable for

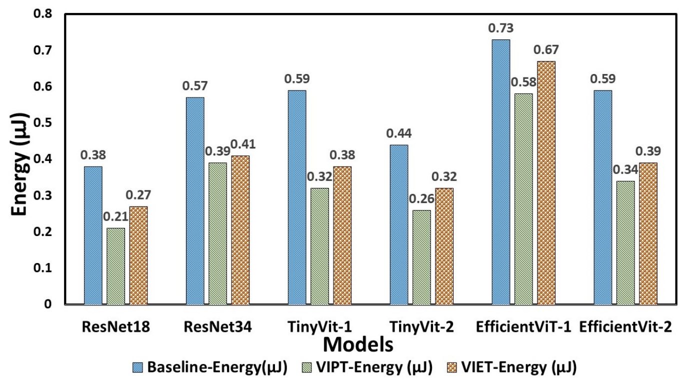


Fig. 9: Energy usage comparison on Nvidia Xavier NX

environments like smartphones or edge devices with strict memory constraints. After applying the VIPT strategy, we observe a reduction in memory usage across all models. For e.g., Swin-Tiny total memory usage decreases from 452 MB to approximately 406-407 MB. This reduction is primarily driven by optimized memory management in parameter storage and the optimizer's memory footprint, aligning with VIPT's objective to enhance post-training quantization efficiency without architectural changes. The VIET strategy, focused on energy-aware optimizations, further reduces the memory footprint. It achieves higher savings, as seen in Swin-Tiny, reducing the total memory requirement to about 360.9-361.9 MB. This strategy optimises parameter storage and also reduces the optimizer memory, making it the first choice for deployment in power-sensitive or memory-constrained environments. Overall, these results show the effectiveness of VIPT and VIET in reducing the memory footprint and enhancing the deployability of models in diverse operational contexts, particularly where memory efficiency is important. Such optimizations facilitate the broader adoption of advanced models like Swin-Tiny and EfficientViT in edge devices, highlighting the practical benefits of these advanced optimization techniques in real-world applications.

TABLE IV: Performance comparison of models [12], [11].

Model	#Para	GFLOPs	Latency (ms)	mIoU(%)
TransFusion	27.8M	38.3	268.2	69.3
DeiT-Ti	5.7M	1.3	125.1	62.4
SSP-DeiT-Ti	4.2M	1.0	120.9	61.1
UVC-DeiT	-	0.7	115.2	60.2
SPViT-DeiT-Ti	4.8M	1.0	122.4	61.3
DeiT-S	22.1M	4.6	210.5	68.6
EViT-DeiT-S	22.1M	3.0	210.8	68.1
eTPS-DeiT-S	23.5M	3.0	214.6	68.2
dTPS-DeiT-S	23.8M	3.0	223.1	68.2
MDC-DeiT-S	-	2.9	208.4	67.7
ToMe-DeiT-S	22.1M	2.7	206.6	67.9
UVC-DeiT-S	-	2.5	206.9	67.5
SPViT-DeiT-S	15.9M	3.3	182.4	66.2
DeiT-B	86.4M	17.5	250.1	70.4
VTP-DeiT-B	67.3M	13.8	247.6	69.8
IA-RED2-DeiT-B	86.4M	11.8	257.3	70.2
S2ViTE-DeiT-B*	56.8M	11.7	232.9	68.3
EViT-DeiT-B	86.4M	11.6	252.6	70.1
eTPS-DeiT-B	86.4M	11.4	248.7	70.7
dTPS-DeiT-B	87.0M	11.4	241.5	70.4
MDC-DeiT-B	-	11.2	246.4	69.2
VTP-DeiT-B	48.0M	10.0	225.4	67.4
SPViT-DeiT-B	41.6M	8.4	222.1	66.1
Swin-Ti	28.3M	4.5	215.9	68.2
STEP-Swin-Ti	23.6M	3.5	190.4	65.5
SPViT-Swin-Ti	25.8M	3.4	192.7	65.6
Swin-S	49.6M	8.7	230.1	69.7
STEP-Swin-S	36.9M	6.3	218.2	67.3
SPViT-Swin-S	38.9M	6.1	212 .4	67.5
TinyViT	21.4M	27.0	210.2	69.5
EfficientViT	49M	9.1	170.4	70.8
MobileViT	38.9M	6.1	212.4	67.8
TinyViT-1	21.4M	35.4	188.2	65.1
EfficientViT-1	25.5M	38.6	212.5	69.0
MobileViT-1	27.9M	39.4	236.9	74.6
TinyViT-2*	16.7M	32.1	145.1	63.8
EfficientViT-2*	19.1M	33.7	155.2	64.6
MobileViT-2*	10.3M	24.4	122.9	72.9

Model Metrics: Table IV provides a detailed performance analysis of several vision transformer models, compared with TinyViT, EfficientViT, and MobileViT trained using VIPT and VIET strategies on nuScenes validation set. The model adapted for training are described by suffixes *Model-1* for VIPT and *Model-2* for VIET. *TinyViT-1* model has 21.4M parameters, utilizing 35.4 GFLOPs, resulting in a latency of 188.2 ms and a mIoU of 65.1%. This model balances processing speed and analytical performance, which is ideal for edge devices. *EfficientViT-1* has a higher resource demand with 25.5M parameters and 38.6 GFLOPs. It also has a higher latency value at 212.5 ms, with an mIoU of 69.0%. The model is suitable for a design choice to enhance image understanding at the expense of higher resource use. *MobileViT-1*, with 27.9M parameters and 39.4 GFLOPs, experiences an increase in latency to 236.9 ms. However, it has a higher accuracy with an mIoU of 74.6%, highlighting its capability to have better model performance. Overall, these models show two approaches to balance the trade-offs between resource demands, processing time, and accuracy, highlighting the effectiveness of VIPT and VIET strategies in optimizing vision transformer architectures for specific application needs.

TABLE V: Validation Results on Waymo Dataset

	3D mAP			
Method	**Overall**	**0-30m**	**30-50m**	**50m-Inf**
ResNet18	71.5	87.8	69.3	47.3
BevFusion	69.7	91.3	68.5	41.3
FusionFormer	75.7	91.5	74.1	51.3
PointPillar	72.1	88.5	69.9	48.0
MV3	62.9	86.3	60.0	36.9
Pillar-OD	69.8	88.5	60.5	42.6
PV-RCNN	70.3	91.9	69.2	42.2
CenterNet	76.5	92.0	74.8	53.0
CenterPoint	77.0	92.0	76.8	56.1
MobileViT-1	83.8	86.9	78.1	64.6
MobileViT-2	81.5	83.3	71.4	56.3
Efficient-ViT-1	79.2	92.4	78.2	59.7
Efficient-ViT-2	78.7	92.2	77.5	58.9
TinyViT-1	76.1	91.6	76.4	55.2
TinyVit-2	78.5	92.8	75.3	56.3

Table V shows the validation results of various benchmark models on the Waymo dataset, focusing on the performance metrics of 3D mean Average Precision (mAP) and 3D mean Average Precision with Heading (mAPH), which are critical for assessing the accuracy of 3D object detection models. Among the traditional DNN models, PointPillar has a benchmark accuracy with a 72.1% overall 3D mAP, showing better results in closer ranges (0-30m) with an 88.5% score but dropping in performance at distances beyond 50m. MV3 shows a 62.9% overall 3D mAP, with a performance decline after the 30m mark. Pillar-OD shows 69.8% overall 3D mAP, comparable close-range performance to PointPillar but reduced effectiveness at medium and long ranges. From our proposed method, MobileViT-1 shows higher performance with an 83.8% overall 3D mAP, consistent accuracy across all ranges, and the highest long-range (50m-Inf) detection at 64.6%, shows efficiency in detecting objects across varying distances with high precision. The TinyVit-1 model trained using VIPT shows reduction in average precision as compared to any other models on the waymo validation set.

Edge Inference: Table VI shows the inference speedup and energy savings for various Vision Transformer models using batch size. The first row is the baseline (1.00×) and the following next rows are the different quantization configurations: 8/8/4, 4/8/4, and 8/4/4, which are the precision levels for weights, activations, and attention, respectively. TinyViT-1 shows speedup of up to 1.92× and energy savings of up to 1.88× compared to the baseline. TinyViT-2 trained using VIET shows higher speedups and energy efficiencies, reaching up to 1.96× and 1.93×, respectively. For the MobileViT models the speed up and energy savings are minimal on the tested devices. The key takeaway from the test and evaluation can be summarized as:

TABLE VI: Results for Inference Speedup and Energy Saving on ViT models

Speedup (×)				**Energy Saving (×)**			
TinyViT-1	**TinyViT-2**	**MobileViT-1**	**MobileViT-2**	**TinyViT-1**	**TinyViT-2**	**MobileViT-1**	**MobileViT-2**
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.92	1.96	1.80	1.83	1.88	1.93	1.75	1.80
1.62	1.53	1.43	1.39	1.51	1.47	1.38	1.32

- Optimized Balance: The approximation methods like 8-bit multipliers and variational inference effectively balance model accuracy with significant improvements in energy efficiency, showcasing only a minimal drop in accuracy for substantial gains in GFlops and latency.
- Adaptive Techniques for Diverse Hardware: Tested across various architectures, the approximation techniques offer adaptable solutions tailored to match specific hardware capabilities, enhancing both the scalability and practicality of AI models on devices ranging from CPUs to GPUs.
- Practical Implementation for Edge AI: By integrating these techniques into distributed edge AI frameworks, the research advances the deployment of AI models that are energy-efficient and capable of real-world application, setting a foundation for future AI developments in edge computing and cyber-physical systems.

VI. Conclusion

This paper explores the training and on-device deployment of data and memory-intensive AI models used in perception tasks. The focus is on balancing model performance metrics against energy consumption using optimized software approximation techniques, such as 8-bit multipliers and variational inference. Our evaluations show that these approximation methods balance model accuracy and energy efficiency. The different architectures used for training and testing provide valuable insights into choosing an appropriate approximation scheme for a balanced trade-off. While comparing the model performance, we observe an improved GFlops and latency gain of 5x while reducing accuracy with a margin of 3-4% for the transformer models. The fundamentals from these approximation schemes can be incorporated within a distributed edge AI framework to match the device resources against the AI models for on-device training and inference. Adopting such techniques is essential for sustainable and feasible AI deployment as the AI landscape evolves towards edge computing and cyber-physical systems. By focusing on the trade-offs involved in training and inference on edge devices, this research contributes to the ongoing efforts to make AI models more accessible and practical in real-world applications.

Acknowledgment

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

References

[1] Ehsan Atoofian. Increasing robustness against adversarial attacks through ensemble of approximate multipliers. In *2022 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 1–8, 2022.

[2] Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17302–17313, 2023.

[3] Zekai Chen, Fangtian Zhong, Qi Luo, Xiao Zhang, and Yanwei Zheng. Edgevit: Efficient visual modeling for edge computing. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 393–405. Springer, 2022.

[4] Hans Jakob Damsgaard, Antoine Grenier, Dewant Katare, Zain Taufique, Salar Shakibhamedan, Tiago Troccoli, Georgios Chatzitsompanis, Anil Kanduri, Aleksandr Ometov, Aaron Yi Ding, et al. Adaptive approximate computing in edge ai and iot applications: A review. *Journal of Systems Architecture*, page 103114, 2024.

[5] Chunhua Deng, Siyu Liao, and Bo Yuan. Permcnn: Energy-efficient convolutional neural network hardware architecture with permuted diagonal structure. *IEEE Transactions on Computers*, 70(2), 2020.

[6] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

[7] Aaron Yi Ding, Ella Peltonen, Tobias Meuser, Atakan Aral, Christian Becker, Schahram Dustdar, Thomas Hiessl, Dieter Kranzlmüller, Madhusanka Liyanage, Setareh Maghsudi, et al. Roadmap for edge ai: A dagstuhl perspective, 2022.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[9] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019.

[10] Gousia Habib and Shaima Qureshi. Optimization and acceleration of convolutional neural networks: A survey. *Journal of King Saud University - Computer and Information Sciences*, 34:4244–4268, 2022.

[11] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

[12] Haoyu He, Jianfei Cai, Jing Liu, Zizheng Pan, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Pruning self-attentions into convolutional layers in single path. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[15] Shashikant Ilager, Vincenzo De Maio, Ivan Lujic, and Ivona Brandic. Data-centric edge-ai: A symbolic representation use case. In *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, pages 301–308. IEEE, 2023.

[16] E Jagadeeswara Rao and P Samundiswary. A review of approximate multipliers and its applications. *Advances in Automation, Signal Processing, Instrumentation, and Control: Select Proceedings of i-CASIC 2020*, pages 1381–1392, 2021.

[17] Dewant Katare and Aaron Yi Ding. Energy-efficient edge approximation for connected vehicular services. In *2023 57th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2023.

[18] Dewant Katare and Mohamed El-Sharkawy. Autonomous embedded system enabled 3-d object detector:(with point cloud and camera). In *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 1–6. IEEE, 2019.

[19] Dewant Katare, Diego Perino, Jari Nurmi, Martijn Warnier, Marijn Janssen, and Aaron Yi Ding. A survey on approximate edge ai for energy efficient autonomous driving services. *IEEE Communications Surveys & Tutorials*, 2023.

[20] Vinay Kumar and Ravi Kant. Approximate computing for machine learning. In *Proceedings of 2nd International Conference on Communication, Computing and Networking: ICCCN 2018, NITTTR Chandigarh, India*, pages 607–613. Springer, 2019.

[21] Denis Kuznedelev, Eldar Kurtic, Elias Frantar, and Dan Alistarh. ovit: An accurate second-order pruning framework for vision transformers. 2022.

[22] Steffen Lange, Johanna Pohl, and Tilman Santarius. Digitalization and energy consumption. does ict reduce energy demand? *Ecological economics*, 176:106760, 2020.

[23] Jelin Leslin, Antti Hyttinen, Karthekeyan Periasamy, Lingyun Yao, Martin Trapp, and Martin Andraud. A hardware perspective to evaluating probabilistic circuits. In *Proceedings of The 11th International Conference on Probabilistic Graphical Models*, volume 186 of *Proceedings of Machine Learning Research*, pages 349–360. PMLR, 05–07 Oct 2022.

[24] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Ds-net++: Dynamic weight slicing for efficient inference in cnns and vision transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4430–4446, 2023.

[25] Xiangtai Li, Henghui Ding, Haobo Yuan, Wenwei Zhang, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-based visual segmentation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–24, 2024.

[26] Di Liu, Hao Kong, Xiangzhong Luo, Weichen Liu, and Ravi Subramaniam. Bringing ai to edge: From deep learning's perspective. *Neurocomputing*, 485:297–320, 2022.

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[28] Hassen Louati, Slim Bechikh, Ali Louati, Chih-Cheng Hung, and Lamjed Ben Said. Deep convolutional neural network architecture design as a bi-level optimization problem. *Neurocomputing*, 439:44–62, 2021.

[29] Vishwanadham Mandala, Srinivas Naveen Reddy Dolu Surabhi, Phani Durga Nanda Kishore Kommisetty, Bala Maruthi Subba Rao Kuppala, and Roopak Ingole. Towards carbon-free automotive futures: Leveraging ai and ml for sustainable transformation. *Educational Administration: Theory and Practice*, 30(5):3485–3497, 2024.

[30] Aryan Mobiny, Pengyu Yuan, Supratik K Moulik, Naveen Garg, Carol C Wu, and Hien Van Nguyen. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 2021.

[31] Bert Moons, Bert De Brabandere, Luc Van Gool, and Marian Verhelst. Energy-efficient convnets through approximate computing. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[32] Duy Thanh Nguyen, Hyun Kim, and Hyuk-Jae Lee. Layer-specific optimization for mixed data flow with mixed precision in fpga design for cnn-based object detectors. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(6):2450–2464, 2020.

[33] Keng-Boon Ooi, Garry Wei-Han Tan, Mostafa Al-Emran, Mohammed A Al-Sharafi, Alexandru Capatina, Amrita Chakraborty, Yogesh K Dwivedi, Tzu-Ling Huang, Arpan Kumar Kar, Voon-Hsien Lee, et al. The potential of generative artificial intelligence across disciplines: perspectives and future directions. *Journal of Computer Information Systems*, pages 1–32, 2023.

[34] Lorenzo Papa, Paolo Russo, Irene Amerini, and Luping Zhou. A survey on efficient vision transformers: Algorithms, techniques, and performance benchmarking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–20, 2024.

[35] Ratko Pilipović, Vladimir Risojević, Janko Božič, Patricio Bulić, and Uroš Lotrič. An approximate gemm unit for energy-efficient object detection. *Sensors*, 21(12):4195, 2021.

[36] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[37] Brendan C Reidy, Mohammadreza Mohammadi, Mohammed E Elbtity, and Ramtin Zand. Efficient deployment of transformer models on edge tpu accelerators: A real system evaluation. In *Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023)*, 2023.

[38] Sohrab Sajadimanesh and Ehsan Atoofian. Eam: Ensemble of approximate multipliers for robust dnns. *Microprocessors and Microsystems*, 98:104800, 2023.

[39] Salar Shakibhamedan, Amin Aminifar, Nima Taherinejad, and Axel Jantsch. Ease: Energy optimization through adaptation–a review of runtime energy-aware approximate deep learning algorithms. *Authorea Preprints*, 2024.

[40] Salar Shakibhamedan, Nima Amirafshar, Ahmad Sedigh Baroughi, Hadi Shahriar Shahhoseini, and Nima TaheriNejad. Ace-cnn: Approximate carry disregard multipliers for energy-efficient cnn-based image classification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2280–2293, 2024.

[41] Salar Shakibhamedan, Anice Jahanjoo, Amin Aminifar, Nima Amirafshar, Nima TaheriNejad, and Axel Jantsch. An analytical approach to enhancing DNN efficiency and accuracy using approximate multiplication. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024.

[42] Raghubir Singh and Sukhpal Singh Gill. Edge ai: a survey. *Internet of Things and Cyber-Physical Systems*, 2023.

[43] Ke Sun and Frank Nielsen. A geometric modeling of occam's razor in deep learning. *arXiv preprint arXiv:1905.11027*, 2019.

[44] Nima TaheriNejad and Salar Shakibhamedan. Energy-aware adaptive approximate computing for deep learning applications. In *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 328–328. IEEE, 2022.

[45] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *European Conference on Computer Vision*, pages 259–275. Springer, 2022.

[46] Rosalie van Oosterhout, Peter Striekwold, and Meng Wang. On data-induced co2 emissions of vehicle automation: An overlooked emission source. *Sustainable Horizons*, 9:100082, 2024.

[47] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.

[48] Yangtao Wang, Xi Shen, Yuan Yuan, Yuming Du, Maomao Li, Shell Xu Hu, James L. Crowley, and Dominique Vaufreydaz. Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(12):15790–15801, 2023.

[49] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision*, pages 68–85. Springer, 2022.

[50] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[51] Tong Xie, Yixuan Hu, Renjie Wei, Meng Li, Yuan Wang, Runsheng Wang, and Ru Huang. Ascend: Accurate yet efficient end-to-end stochastic computing acceleration of vision transformer. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.

[52] Guanyu Xu, Zhiwei Hao, Yong Luo, Han Hu, Jianping An, and Shiwen Mao. Devit: Decomposing vision transformers for collaborative inference in edge devices. *IEEE Transactions on Mobile Computing*, 2023.

[53] Peng Xu, Xiatian Zhu, and David A. Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132, 2023.

[54] Lingyun Yao, Martin Trapp, Karthekeyan Periasamy, Jelin Leslin, Gaurav Singh, and Martin Andraud. Logarithm-approximate floating-point multiplier for hardware-efficient inference in probabilistic circuits. In *The 6th Workshop on Tractable Probabilistic Modeling*, 2023.

[55] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[56] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6575–6586, 2023.
[57] Chenglong Zhao, Shibin Mei, Bingbing Ni, Shengchao Yuan, Zhenbo Yu, and Jun Wang. Variational adversarial defense: A bayes perspective for adversarial training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3047–3063, 2024.
[58] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.

Dewant Katare received his B.Sc. degree in electrical and electronic engineering in 2016 and M.S. degree in electrical and computer engineering in 2019 from Purdue University, USA. He is currently a Ph.D. candidate researching on the topics of energy efficient approximate Edge-AI for automated driving services at the department of engineering systems and services, Delft University of Technology, The Netherlands. His research interests include computer vision, edge computing, distributed ML systems, Edge AI and model approximations.

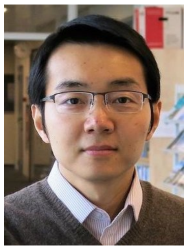
Salar Shakibhamedan was born in Tehran, Iran, in 1992. He received the B.Sc. Degree in electrical engineering from the K. N. Toosi University of Technology, Tehran, Iran, in 2015 and the M.Sc. Degree in electrical engineering, focused on multi-modal signal processing from the same university, in 2018. Currently, he is working as a Ph.D. Student at the APROPOS (EU-funded project) focuses on approximate computing in embedded machine learning and deep learning.

Nima Amirafshar received the B.Sc. degree in electrical engineering from the Ferdowsi University of Mashhad (FUM), Mashhad, Iran, in 2019, and the M.Sc. degree in electrical engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2023. He is currently a PhD candidate at Ruprecht-Karls-Universität Heidelberg, Germany. His research interests include computer architecture, Approximate computing, Hardware acceleration, and Multicore systems.

Prof. Nima TaheriNejad (S'08-M'15) received his Ph.D. degree in electrical and computer engineering from The University of British Columbia (UBC), Vancouver, Canada, in 2015. He is currently a Full Professor at Heidelberg University, Heidelberg, Germany and affiliated with TU Wien (formerly known also as Vienna University of Technology), Vienna, Austria. His areas of work include in-memory computing, cyber-physical and embedded systems, systems on chip, memristor-based circuit and systems, self-*systems, and health-care. He has published three books, three patents, and more than 90 articles. Dr. Taherinejad has served as a reviewer and an editor of many journals and conferences. He has also been an organizer and a chair of various conferences and workshops. He has received several awards and scholarships from universities, conferences, and competitions he has attended. This includes the Best University Booth award at DATE 2021, First prize in the 15th Digilent Design Contest (2019) and in the Open-Source Hardware Competition at Eurolab4HPC (2019) as well as Best Teacher and Best Course awards at TU Wien (2020).

Prof. Axel Jantsch (Senior Member, IEEE) received the Dipl.Ing. and Ph.D. degrees in computer science from TU Wien, Vienna, Austria, in 1987 and 1992, respectively. From 1997 to 2002, he was an Associate Professor with the KTH Royal Institute of Technology, Stockholm. From 2002 to 2014, he was a Full Professor of electronic systems design at the KTH. Since 2014, he has been a Professor of systems on chips with the Institute of Computer Technology, TU Wien. He has published five books as an editor and one as the author and over 300 peer-reviewed contributions in journals, books, and conference proceedings. He has given over 100 invited presentations at conferences, universities, and companies. His current research interests include systems on chips, selfaware cyber-physical systems, and embedded machine learning.

Prof. dr. Marijn Janssen is a Full Professor in ICT Governance in the Technology, Policy and Management Faculty of Delft University of Technology, head of the Engineering Systems Servies (ESS) department, and (honorary) visiting professor at Bradford University, KU Leuven and Universiti Teknologi Mara. He was ranked as one of the leading e-government researchers in surveys in 2009, 2014, and 2016. He was nominated in 2018 and 2019 by Apolitical as one of the 100 most influential people in the Digital Government worldwide: https://apolitical.co/lists/digital-government-world100. More information: www.tudelft.nl/staff/m.f.w.h.a.janssen/.



Dr. Aaron Yi Ding is a tenured Associate Professor and leading the Cyber–Physical Intelligence (CPI) Laboratory at TU Delft, Netherlands. He has over 16 years of research and development experience across EU, U.K., and USA, including TU Munich, University of Cambridge, and Columbia University. His research focuses on edge AI solutions for cyber–physical systems in smart health, mobility, and energy domains. He has 80+ peer reviewed publications, receiving best paper awards and recognition from ACM SIGCOMM, ACM EdgeSys, ACM SenSys CCIoT, IEEE INFOCOM, and the esteemed Nokia Foundation Scholarships. He is the scientific director and coordinator for the EU Horizon project SPATIAL. He is an Associate Editor of ACM Transactions on Internet of Things (TIOT) and IEEE Open Journal of the Intelligent Transportation Systems.