AI

MDPI

Article

Improving Accuracy and Efficiency in DNNs with Approximate Multipliers: Insights from Information Bottleneck Theory

Salar Shakibhamedan [1,*], Nima Amirafshar [2], Axel Jantsch [1] and Nima TaheriNejad [1,2]

[1] Institute of Computer Technology, TU Wien, 1040 Vienna, Austria; axel.jantsch@tuwien.ac.at (A.J.); nima@uni-heidelberg.de (N.T.)
[2] Institute of Computer Engineering, Heidelberg University, 69117 Heidelberg, Germany; nima.amirafshar@uni-heidelberg.de
* Correspondence: salar.shakibhamedan@tuwien.ac.at

Abstract

Approximate multipliers have potential to improve energy efficiency in Deep Neural Networks but introduce computational errors that degrade accuracy. This paper introduces a novel method, leveraging approximate multipliers to enhance accuracy, while improving computational and energy efficiency. We propose a layer-wise heterogeneous approach using quantized approximate multipliers (INT8) in DNNs, applying varying levels of approximation across layers. This approach achieves estimated energy savings of up to 44.75% across all evaluated configurations, including up to 43.72% for the VGG models, while improving Top-1 accuracy by up to 2.07 percentage points relative to the corresponding exact INT8 baseline. Using Information Bottleneck (IB) theory, we analyze the enhanced information flow and feature extraction capabilities enabled by approximate multipliers. Through Information Plane (IP) analysis, we gain insights into DNN behavior and demonstrate how this method can overcome accuracy limitations. An analytical MAC-count comparison indicates that, under the experimental settings considered, the forward-pass-only GA search requires 146× to 22,500× fewer MAC operations than the corresponding gradient-based training schedules; this comparison reflects computational workload rather than an equivalent-objective algorithmic speedup.

Keywords: energy-efficient DNNs; information bottleneck theory; energy–accuracy trade-off; computational efficiency; layer-wise approximation

1. Introduction

Deep Neural Networks (DNNs) have revolutionized artificial intelligence, achieving state-of-the-art results in domains such as computer vision [1], natural-language processing [2], and generative AI [3]. As the scope of AI applications has expanded, the complexity and scale of DNNs have grown significantly [4]. This growth is fueled by the increasing availability of data and the demand for scalable, high-performing models [5,6]. However, these advances come with substantial computational and energy demands, which pose challenges, particularly in resource-constrained environments like edge devices [7]. Edge devices, such as smartphones and IoT systems, often operate under strict power and computational limitations, making it difficult to deploy modern DNNs effectively.

To address these challenges, approximate computing has emerged as a viable approach [8]. By introducing controlled inaccuracies, approximate computing reduces the computational complexity and energy consumption of DNNs [7]. Among the various techniques within this paradigm, approximate multipliers have shown promise for enhancing

Academic Editor: Demos T. Tsahalis

Received: 17 July 2026
Revised: 7 September 2026
Accepted: 13 September 2026
Published: 21 September 2026

efficiency without significantly degrading performance. These multipliers achieve energy efficiency by simplifying their hardware design, often introducing minor computational errors [9]. Recent advances, such as Signed Carry Disregard Multipliers (SCDM8), demonstrate the potential to balance accuracy and efficiency effectively, especially when combined with quantization techniques like INT8 [10].

Despite their advantages, approximate computing techniques are traditionally associated with a trade-off between accuracy and efficiency. Many studies have focused on optimizing efficiency, often at the expense of accuracy. In this work, we challenge this trade-off by proposing a novel approach that leverages approximate multipliers to improve not only the efficiency but also the accuracy of DNNs. By employing a heterogeneous configuration of SCDM8 multipliers, each layer of a model is assigned a specific multiplier variant tailored to its computational requirements and error tolerance. This approach harnesses the diversity of computational behaviors to enhance information flow within the model.

Additionally, DNN models are often complex and their behavior is sophisticated, frequently resembling a "black box" that is challenging to interpret and understand. This complexity complicates efforts to analyze and optimize DNNs. The Information Bottleneck (IB) theory, proposed by [11], provides a robust framework to understand the internal mechanisms of DNNs by analyzing their information processing capabilities [12]. By examining the trade-offs between compression and prediction, IB theory offers insights into the information flow and feature extraction processes that occur within these models.

Our experiments focus on image classification tasks, using well-known CNNs as case studies alongside other architectures, such as Multilayer Perceptrons (MLPs). The proposed method not only achieves significant energy efficiency and optimization but also surpasses the accuracy plateau often observed with traditional training processes. The proposed post-training configuration search requires substantially fewer estimated MAC operations than the corresponding gradient-based training schedules under the experimental settings considered. The selected heterogeneous configurations can additionally reduce estimated multiplier energy during inference while improving accuracy relative to the exact INT8 baseline. Overall, the proposed method provides a computationally efficient post-training configuration search and identifies heterogeneous multiplier assignments that improve accuracy relative to the corresponding exact INT8 baselines while reducing estimated multiplier energy during inference.

The remainder of this paper is structured as follows. Section 2 discusses the background and related work on approximate multipliers and Information Bottleneck theory. Section 3 presents the proposed method. Section 4 details the experimental setup, and Section 5 reports the results and evaluation. Section 6 provides a discussion and comparison, followed by a summary of the work in Section 7.

2. Background

The challenges associated with deploying DNNs in resource-constrained environments have spurred research into approximate computing and its applications in deep learning. Approximate computing techniques aim to improve energy efficiency and reduce computational demands by introducing controlled inaccuracies into computations. Among these techniques, approximate multipliers stand out as a practical solution to optimize the resource-heavy operations inherent in DNNs [13].

2.1. Approximate Multipliers

Approximate multipliers are hardware components designed to perform multiplication operations with reduced precision, enabling lower power consumption and faster

execution. By strategically simplifying their architecture, these multipliers achieve significant energy savings at the cost of minor computational errors. Quantization, a widely used approximation technique in deep learning, reduces computational bit width to lower energy consumption and memory usage [14]. Combining quantization with approximate multipliers amplifies efficiency gains, as the simplified compute units require fewer components and consume less power.

For instance, some approximate multipliers incorporate constant-truncated regions with error correction mechanisms, achieving substantial reductions in energy and area with minimal accuracy loss [15]. Designs like 4-2 approximate compressors balance hardware efficiency with computational precision [16], while Broken-Array Multipliers (BAMs) eliminate partial products to reduce power consumption further [17]. Libraries such as [18] provides a collection of approximate multiplier designs, catering to diverse requirements of hardware efficiency and accuracy.

A notable advancement in this field is the Signed Carry Disregard Multiplier (SCDM8) family, which introduces approximation by disregarding carry propagation from lower to higher bits [10]. This design significantly reduces hardware complexity and power usage. Each SCDM8 multiplier comprises two 8×4 units for the Most-Significant Bits (MSBs) and Least-Significant Bits (LSBs). By adjusting the number of approximate bits in the MSBs and LSBs, up to 100 configurations are possible. Studies have identified 20 configurations (SCDM8 xy, where x: 4→8 and y: 1→4) as optimal for convolutional neural networks (CNNs) [10]. These multipliers have demonstrated their suitability for image classification tasks, maintaining accuracy within acceptable limits while achieving substantial power savings.

Evaluations of SCDM8 multipliers often employ Post-Training Quantization (PTQ) on pre-trained DNNs. This combination of quantization and approximation underscores the potential of SCDM8 multipliers for energy-efficient DNN deployment, particularly on edge devices. The results highlight that the approximate multipliers achieved substantial energy savings and maintained acceptable accuracy levels across a range of DNN models and datasets. A summary of these findings, including energy savings and accuracy degradation compared to exact multipliers, is presented in Table 1.

Table 1. Energy–accuracy trade-off of approximate vs. exact multipliers in DNNs [10].

Model	DateSet	Accuracy (TOP-1) Degradation	Energy Saving
VGG16	ImageNet	0.49%	61%
ResNet152	ImageNet	0.23%	68%
MobileNetV2	ImageNet	0.10%	45%
ConvVeXt-T	ImageNet	0.37%	56%
LeNet5-Inspired	MNIST	0.07%	51%

2.2. Information Bottleneck Theory

The Information Bottleneck (IB) theory offers a rigorous framework for understanding the inner workings of deep neural networks. Proposed by [11], the IB principle seeks to maximize the mutual information between the input and output of a model while minimizing the mutual information between the input and intermediate representations. This trade-off enables the extraction of meaningful and discriminative features while compressing redundant information, thus improving the model's generalization capabilities. The IB framework conceptualizes DNNs as a series of transformations, where each layer acts as a bottleneck to distill relevant information. This is often visualized using the information plane, which plots the mutual information of hidden layers with respect to the

input and output. The information plane reveals distinct training phases: an initial fitting phase, where the model learns relevant patterns, followed by a compression phase, where redundant information is discarded.

While traditional IB methods have been extensively studied in low-dimensional settings, their application to high-dimensional models such as CNNs remains challenging [19]. Recent advancements, such as Mutual Information Neural Estimation [20], have addressed these limitations by providing scalable techniques for estimating mutual information in complex architectures. These methods have validated the existence of the compression phase in high-dimensional models like VGG-16, extending the relevance of IB theory to practical applications [21]. Moreover, IB-inspired approaches have been employed to enhance DNN performance. For instance, regularization techniques based on mutual-information constraints have been investigated to improve model performance and stability, particularly in tasks with limited training data [22]. By incorporating such techniques, this work explores how approximate multipliers influence the information flow within DNNs, leveraging the IB framework to optimize both efficiency and accuracy.

The insights gained from this analysis inform the design of heterogeneous multiplier configurations, offering a novel approach to reconcile the trade-offs between energy efficiency and model performance. These findings not only extend the applicability of IB theory to approximate computing but also provide a theoretical basis for designing next-generation models optimized for resource-constrained environments.

3. Proposed Method

3.1. Role of Approximate Multipliers

For our studies and experiments, we selected 20 approximate multipliers from the SCDM8 family, as named in the original study [10]. Using the naming convention SCDM8_XY, where X and Y denote the number of approximate bits (MSB, and LSB respectively), we chose the set SCDM8_XY ($X : 4 \rightarrow 8, Y : 1 \rightarrow 4$). These correspond to 20 approximate multipliers labeled Approx_Mult 1–20 in our experiments.

3.1.1. Unique Outputs of Approximate Multipliers

One notable characteristic of approximate multipliers is that their output values are not only distinct from those of exact multipliers but also vary significantly among the approximate multipliers themselves. This diversity arises from the inherent differences in their arithmetic operations, which are influenced by the levels of approximation in their designs. These variations in output play a critical role in influencing the information flow and computational behavior within DNNs. This diversity intrigued us to explore whether such differences could offer advantages in feature extraction, information encoding, and decoding. Specifically, we hypothesize that the distinct outputs of approximate multipliers can enrich the computational paradigms within DNNs, enabling novel behaviors and insights that are not achievable with traditional, exact multipliers. Our experiments aimed to analyze the effects of these diverse outputs on the overall performance and energy efficiency of DNNs.

3.1.2. Heterogeneous Architectures

Building on this observation, we propose a heterogeneous architecture in which different approximate multipliers are assigned to individual layers of a DNN. This heterogeneity, characterized by varying levels of approximation across layers, enhances the information flow within the model, ultimately improving accuracy, energy, and computational efficiency. By leveraging the unique outputs of each multiplier, we aim to exploit this diversity to achieve higher performance than traditional homogeneous configurations.

3.1.3. Optimization of Configurations

Identifying the optimal configuration for such heterogeneous architectures requires exploring a vast search space. For instance, in our case study, we utilized 20 approximate multipliers from the SCDM8 family. In a small CNN with 8 layers, assuming each layer uses one of the 20 multipliers, the total number of possible configurations becomes 20^8. This number increases exponentially with the number of layers or multipliers, rendering an exhaustive search impractical, even for models with relatively few layers. To address this challenge, we employed a genetic algorithm, a meta-heuristic optimization approach, to explore combinations of approximate multipliers across layers. Each configuration was evaluated based on its image classification accuracy, with higher accuracy serving as the merit function. This iterative process prioritized and retained high-accuracy solutions for subsequent generations, enabling the efficient identification of optimal configurations. It should be noted that 20^L represents the nominal combinatorial space rather than the number of configurations explicitly evaluated. The genetic algorithm explores only a small subset determined by the population size and number of generations.

The multiplier set could potentially be further reduced by clustering configurations with similar arithmetic-error, entropy, and energy characteristics. However, similarity in a single aggregate metric does not imply equivalent behavior within a DNN; for example, SCDM8_53 and SCDM8_63 exhibit relatively similar output entropy but different energy characteristics and model-level accuracy. Therefore, all 20 candidate multipliers were retained in the present study. Search-space pruning using multi-metric similarity is left for future investigation.

3.1.4. Computational Efficiency and Trade-Offs

Approximate multipliers inherently consume less power and exhibit shorter delays compared to exact multipliers. Consequently, any layer-wise combination of approximate multipliers results in a more efficient DNN than one using fully quantized exact INT8 multipliers. If the genetic algorithm successfully identifies a configuration that improves accuracy, it transcends the traditional trade-off paradigm, achieving simultaneous enhancements in accuracy and efficiency. Such a configuration represents a post-training optimization method that can improve accuracy and representation compression while reducing estimated multiplier energy during subsequent inference. Importantly, this approach eliminates the need for computationally expensive back-propagation, relying solely on the more efficient forward pass. This characteristic makes it particularly advantageous for scenarios requiring post-training configuration search/optimization, as it significantly reduces computational complexity while maintaining high performance.

3.2. Information Theory Analysis in DNNs

As a first step, we analyzed the entropy of the outputs generated by approximate multipliers and compared them to those of an exact multiplier. Entropy, introduced by [23], quantifies the uncertainty or intrinsic amount of information associated with a random variable. For a given discrete random variable X, with values drawn from a set of possible outcomes X and probability density function $p(x)$, the entropy is defined as:

$$H(X) = -\sum_{x \in X} p(x) \log p(x). \quad (1)$$

The approximations inherent in approximate multipliers cause their output entropy to differ from that of exact multipliers. These variations directly impact feature extraction, information encoding, and information flow within DNNs. The results of this entropy analysis are presented in Table 2.

Table 2. Output entropy for exact and approximate multipliers.

Multiplier Type	Output Entropy	Multiplier Type	Output Entropy
Exact Multiplier	12.856	SCDM8_63 (11)	13.516
SCDM8_41 (1)	13.183	SCDM8_64 (12)	13.632
SCDM8_42 (2)	13.261	SCDM8_71 (13)	13.513
SCDM8_43 (3)	13.372	SCDM8_72 (14)	13.525
SCDM8_44 (4)	13.580	SCDM8_73 (15)	13.502
SCDM8_51 (5)	13.325	SCDM8_74 (16)	13.637
SCDM8_52 (6)	13.379	SCDM8_81 (17)	13.625
SCDM8_53 (7)	13.458	SCDM8_82 (18)	13.633
SCDM8_54 (8)	13.600	SCDM8_83 (19)	13.634
SCDM8_61 (9)	13.438	SCDM8_84 (20)	13.641
SCDM8_62 (10)	13.436		

Motivated by these computational variations, we explored their effects on the information flow within DNNs, particularly their influence on model accuracy. We also investigated how heterogeneous computational flows, introduced by employing different approximate multipliers in various layers of a DNN, impact the information flow. Additionally, we studied the effects of approximate multipliers on the *fitting* and *compression* phases within DNNs.

To conduct this analysis, we utilized the Information Bottleneck (IB) concept, a framework proposed by [11,24] that studies and characterizes information flow in DNNs using information-theoretic principles. Beyond entropy, the IB framework relies on mutual information, which measures the shared dependency or mutual "knowledge" between two variables.

For a second random variable Y with probability distribution $p(y)$, as well as a joint probability distribution $p(x,y)$ representing the likelihood of x and y occurring together, the mutual information $I(X;Y)$ is expressed as:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right). \quad (2)$$

In the IB framework, each layer of a DNN is considered a random variable, with the input data and output forming the first and last layers, respectively. For a DNN with m hidden layers, the input X, hidden layers (illustrated in Figure 1), $T_1, T_2, \ldots, T_m$, and output $\hat{Y}$ form a Markov chain as follows:

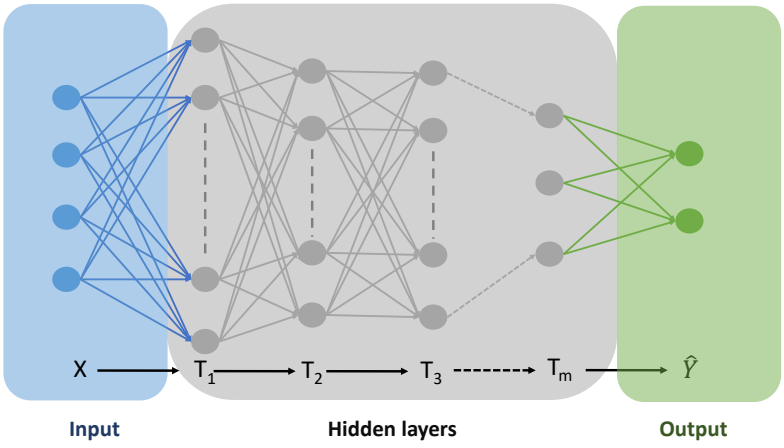


Figure 1. A DNN schematic with input X, output Y, and hidden layers T_1 to T_m for IB analysis.

$$Y \rightarrow X \rightarrow T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_m \rightarrow \hat{Y}. \tag{3}$$

Here, T_i represents the i-th hidden layer, X is the input, Y is the associated label, and $\hat{Y}$ is the model output. According to the IB principle [24], the hidden layers (T_i) serve as bottlenecks, compressing the input X while retaining information relevant to the target Y. The mutual information $I(Y;T)$ indicates the degree to which the representation T retains information about Y. The IB framework aims to optimize the representation T by maximizing $I(T;Y)$ (relevance) while minimizing $I(X;T)$ (compression). This dual objective is expressed as a Lagrangian optimization problem:

$$\mathcal{L} = I(Y;T) - \beta I(X;T), \tag{4}$$

where β is a regularization parameter balancing relevance and compression. Applying the IB principle and the Data Processing Inequality (DPI) [11] to DNNs yields the following inequalities for information flow:

$$I(Y;X) \geq I(Y;T_1) \geq \ldots \geq I(Y;T_m), \tag{5}$$

$$I(X;X) \geq I(X;T_1) \geq \ldots \geq I(X;T_m). \tag{6}$$

These inequalities set upper and lower bounds on the information flow through the layers of a DNN. The upper bound for the input X corresponds to the mutual information of the input with itself, $I(X;X)$, while the upper bound for the output Y is determined by the mutual information between the labels and the inputs, $I(Y;X)$. In our analysis, we reformulated the mutual information terms using entropy as follows:

$$I(X;T) = H(T) - H(T|X) = H(T), \tag{7}$$

$$I(T;Y) = H(T) - H(T|Y), \tag{8}$$

where $H(T|X)$ is the conditional entropy of T given X, which equals zero under the assumption that T is a deterministic function of X. Similarly, $H(T|Y)$ represents the conditional entropy of T given Y. To visualize the information flow, we employed the information plane, where $I(X;T_i)$ is plotted on the x-axis and $I(Y;T_i)$ on the y-axis. This representation provides detailed insights into the fitting and compression processes across the layers of a DNN during training.

As discussed by [25,26], distinct fitting and compression phases have been observed during training on various datasets and activations (e.g., ReLU and Tanh). These phases, particularly the compression phase, are hypothesized to enhance generalization and have been extensively studied in the literature [19,21,27–30].

3.3. Improving DNNs with Heterogeneous Computational Paradigms

Our approach leverages the diverse computational paradigms and output entropies of approximate multipliers to explore new possibilities for feature extraction and information flow within DNNs. These paradigms allow us to **achieve distinct points on the information plane that are otherwise unattainable with traditional computational methods, such as conventional post-training optimization approaches.**

A key observation is that the output entropy of each approximate multiplier differs from both exact multipliers and other approximate multipliers. This diversity enables a range of computational paradigms. Our method leverages this by assigning distinct approximate multipliers to each DNN layer, forming a heterogeneous computational framework. This heterogeneity, arising from varying levels of multiplication entropy,

positively influences the information flow on the information plane, leading to improved fitting (accuracy) and compression.

To identify optimal configurations for heterogeneous implementations, we utilized a genetic algorithm, as detailed in Algorithm 1. This approach efficiently navigates the extensive design space of multiplier configurations across DNN layers to find combinations that maximize accuracy while ensuring energy efficiency. Additionally, since approximate multipliers inherently consume less power than exact multipliers, any layer-wise configuration of approximate multipliers results in lower power consumption compared to fully quantized INT8 or Float32 DNNs.

Throughout this study, the exact INT8 quantized model is used as the primary accuracy baseline for evaluating the proposed heterogeneous approximate-multiplier configurations. This baseline was selected because both the exact and approximate multipliers operate in the INT8 domain, allowing the effect of multiplier approximation to be evaluated independently of the accuracy change caused by quantization. The FP32 model is reported separately as a reference for quantifying the effect of INT8 quantization, but it is not used as the primary baseline for the reported accuracy improvements.

Algorithm 1 Layer-wise Heterogeneous Approximate Multiplier Search

Input: DNN model with L computational blocks/layers.
Search space: Set of approximate multipliers $\{\texttt{SCDM8\_41}, \texttt{SCDM8\_42}, \ldots, \texttt{SCDM8\_84}\}$.
Hyperparameters: Population size P, number of generations N, mutation rate M, and validation dataset.
Output: Optimal layer-wise configuration of approximate multipliers, $\text{DS}_{\text{AMult}} = \{D_{\text{AMult}_1}, D_{\text{AMult}_2}, \ldots, D_{\text{AMult}_L}\}$.
Initialize population: Generate P random chromosomes of length L.
for $gen = 1$ **to** N **do**
    for each chromosome in the population **do**
        Replace the exact multipliers in the DNN layers with the approximate multipliers encoded in the chromosome.
        Evaluate fitness by running forward-pass inference on the validation dataset and computing Top-1 accuracy.
    end for
    Select parent chromosomes using tournament selection based on the highest Top-1 accuracy.
    Apply single-point crossover to generate offspring chromosomes.
    Randomly mutate genes in the offspring with probability M.
    Replace the old population with the new offspring generation.
end for
return the fittest chromosome DS_{AMult} from the final generation.

In summary, our heterogeneous computational framework, employing distinct multipliers in each layer, achieved improved accuracy (fitting), enhanced compression, and significant energy efficiency gains. This method introduces a novel GA-based configuration search, delivering notable benefits in accuracy, model compression, and estimated energy efficiency during subsequent inference.

4. Experimental Setup

As mentioned, we employed 20 quantized (INT8) approximate multipliers from [10], which have demonstrated effectiveness in homogeneous DNN implementations. The energy efficiency and output entropy of these multipliers are detailed in Tables 2 and 3, respectively. Table 3 reports the energy consumption of each approximate multiplier normalized to that of the exact multiplier. Specifically, the normalized energy coefficient of multiplier i is defined as $e_i = M_i/M_x$, where M_i and M_x denote the energy metrics of

the approximate and exact multipliers, respectively. Thus, a coefficient of 1.00 represents the exact multiplier, while a coefficient below 1.00 indicates reduced energy consumption. For example, a normalized energy coefficient of 0.40 corresponds to an energy saving of 60% at the multiplier level. Table 2 presents the entropy values of the outputs generated by both exact and approximate multipliers. Entropy, as introduced by Shannon, quantifies the uncertainty or the intrinsic amount of information contained within a variable. In this context, the output entropy reflects the variability of the results produced by each multiplier—where higher entropy indicates greater diversity in the output distribution. These variations provide insights into how different approximate multipliers influence the information content of computations within DNNs.

Table 3. Normalized energy consumption of the exact and approximate multipliers during inference [10]. The normalized energy of the exact multiplier is 1.00, and lower values indicate lower energy consumption.

Multiplier (Index)	Energy Usage (Normalized)
Exact Multiplier	1.00
SCDM8_41 (1)	0.78
SCDM8_42 (2)	0.76
SCDM8_43 (3)	0.74
SCDM8_44 (4)	0.69
SCDM8_51 (5)	0.69
SCDM8_52 (6)	0.64
SCDM8_53 (7)	0.61
SCDM8_54 (8)	0.58
SCDM8_61 (9)	0.66
SCDM8_62 (10)	0.61
SCDM8_63 (11)	0.58
SCDM8_64 (12)	0.55
SCDM8_71 (13)	0.63
SCDM8_72 (14)	0.54
SCDM8_73 (15)	0.49
SCDM8_74 (16)	0.43
SCDM8_81 (17)	0.58
SCDM8_82 (18)	0.52
SCDM8_83 (19)	0.46
SCDM8_84 (20)	0.40

When integrated into DNNs, these multipliers delivered significant improvements, some of which are summarized in Table 1. Inspired by the literature, our experiments included a CNN architecture (5 Conv layers followed by 3 FC-layers) [10] trained on the MNIST dataset [31] and Fashion MNIST dataset [32]. Additionally, we trained a Multilayer Perceptron (MLP) with a 784-200-100-10 architecture on both datasets. To expand the experimental scope, we evaluated more complex datasets, training VGG11 and VGG13 architectures [33] on CIFAR-10 and CIFAR-100 [34]. The hyperparameters used for training all models are detailed in Appendix A.1. The experiments, including the implementation and simulation of approximate multipliers, were conducted on a server equipped with 4× Intel Xeon Platinum 8160 processors and 2× NVIDIA Tesla V100 GPUs (16GB). The information plane computations were performed on a server cluster, with hardware specifications omitted for anonymity.

4.1. Training and Quantization

The models were initially trained in 32-bit floating-point (FP32) format on their respective datasets. To achieve optimal performance, extensive hyperparameter tuning was performed, including adjustments to learning rate, batch size, and regularization. Detailed training configurations are provided in the Appendix A.1. After FP32 training, full integer quantization (post-training quantization, PTQ) was applied using TensorFlow 2.17 [35], converting weights and activations to INT8. This quantization step ensures compatibility with the approximate multipliers, which operate in the INT8 domain. Moreover, as the activations/logits of each layer serve as inputs to subsequent layers, maintaining INT8 precision for activations is critical.

4.2. Heterogeneous Implementation Using Approximate Multipliers

To evaluate our method, we applied approximate multipliers heterogeneously by assigning different multipliers to each layer while maintaining the same multiplier within a given layer. A genetic algorithm was utilized to identify heterogeneous approximate-multiplier configurations capable of achieving higher accuracy than the corresponding exact INT8 baseline models. The validation and test sets were kept strictly disjoint throughout the experiments. During the GA search, the validation set was used exclusively for fitness evaluation and multiplier-configuration selection. The test set was not used for fitness calculation, GA hyperparameter tuning, or any other optimization decision. After the GA search was completed, the selected multiplier configurations were fixed and evaluated on the independent held-out test set. All final Top-1 accuracy values reported in Section 5 were obtained from this test set, and no test-set results were fed back into the GA. The algorithm's parameters, including population size, mutation rate, and number of generations, were carefully tuned to balance exploration, exploitation, and computational efficiency. The genetic algorithm parameters were carefully selected to balance exploration, exploitation, and computational efficiency. The number of generations was scaled according to the complexity of the models and datasets, with simpler datasets like MNIST requiring fewer generations (10–20) and more complex datasets like CIFAR-10/100 necessitating additional generations (35–40). Similarly, population size was adjusted based on model size, with larger models like VGG11/13 using 40–50 individuals and smaller models like MLP and CNN using 25. Mutation rates were set at 0.05 for VGG models and 0.1 for simpler models to introduce sufficient randomness for exploration without compromising convergence stability. Batch sizes of 64 for smaller models and 128 for larger ones were employed to optimize computational efficiency. These carefully tuned parameters allowed the algorithm to effectively identify heterogeneous configurations of approximate multipliers, achieving a balance between accuracy and efficiency. Detailed information on the genetic algorithm structure and implementation is provided in Table 4 and Algorithm 1.

Table 4. Parameter values for the genetic algorithm used in the case studies.

Model (Dataset)	Generation Number	Population Size	Mutation Rate	Batch Size
VGG11 (CIFAR10)	35	40	0.05	128
VGG11 (CIFAR100)	40	40	0.05	128
VGG13 (CIFAR10)	40	50	0.05	128
VGG13 (CIFAR100)	40	50	0.05	128
MLP (MNIST)	10	25	0.10	64
MLP (FMNIST)	15	25	0.10	64
CNN (MNIST)	20	25	0.10	64
CNN (FMNIST)	20	25	0.10	64

4.3. Information Plane Analysis

To evaluate the performance of DNNs on the information plane, we saved the output of each layer during training to compute mutual information $I(X;T)$ and $I(Y;T)$. The resulting information planes for this training procedure are presented in Figures A1 and A2 for the CNN and MLP case studies, respectively. For VGG11 and VGG13, due to their higher complexity, the information plane of the last layer is presented in Figure A5. Quantization was applied after each epoch to ensure that the outputs used for mutual information calculations adhered to the INT8 format. This approach maintained consistency throughout the training process. As noted by [26], mutual information calculations are more accurate in quantized models than in non-quantized models. For our CNN and MLP case studies, we utilized the tool proposed in [26] to compute mutual information values. For higher-dimensional and more complex models, such as VGG11 and VGG13, we employed the Mutual Information Neural Estimation (MINE) method [20,21] to achieve convergence and obtain a reasonable information plane. For the VGG-11 and VGG-13 experiments, MINE was implemented using a statistics network consisting of two fully connected hidden layers with 100 neurons each and ReLU activations, followed by a scalar output. The input variables to the statistics network were concatenated. The MINE estimator was optimized using Adam with a learning rate of 10^{-4}, a mini-batch size of 128, and an exponential-moving-average coefficient of $\alpha = 0.01$. Samples from the product of the marginal distributions were generated by randomly permuting one variable within each mini-batch, while the original paired samples represented the joint distribution. The estimator was optimized for 100 epochs without early stopping, and the mutual-information estimate obtained after the final optimization epoch was used for constructing the information plane. The same MINE configuration was used for VGG-11 and VGG-13 on both CIFAR-10 and CIFAR-100. For these experiments, the input and last-layer representations were flattened before being provided to the MINE estimator.

4.4. Final Experiments with Heterogeneous Multipliers

In the final step, we implemented layer-wise heterogeneous approximate multipliers to locate points on the information plane that offered both increased compression and higher accuracy. These configurations increased $I(Y;T)$ during the fitting phase while maintaining a high degree of compression. The genetic algorithm used in this step requires a lower estimated MAC workload than the corresponding gradient-based training schedules under the experimental settings considered (details in Appendix A.2). Using the genetic algorithm, we identified combinations of approximate multipliers that achieved both enhanced compression and improved accuracy. Furthermore, our proposed heterogeneous method is estimated to improve multiplier-energy efficiency during DNN inference. These details are elaborated in Section 5.

5. Results

5.1. Accuracy Improvement

By implementing the proposed heterogeneous approach for SCDM8 approximate multipliers, we identified four distinct sets of multiplier combinations for each case study. These configurations improve Top-1 accuracy relative to the corresponding exact INT8 baseline while also enhancing energy efficiency. Throughout this section, accuracy improvement refers to the absolute difference between the Top-1 accuracy of the heterogeneous approximate INT8 model and that of the exact INT8 model, expressed in percentage points. After completing the GA search on the validation set, the selected multiplier configurations were fixed and evaluated on the independent held-out test set. The resulting test-set Top-1 accuracies are summarized in Table 5. The exact INT8 model is used as the baseline for

calculating the reported accuracy improvements, while the FP32 model is included only as a reference for illustrating the effect of quantization. The identified sets of approximate multipliers for all evaluated models—CNN, MLP, VGG11, and VGG13—are detailed in Table A6 and summarized in Table 5. The Top-1 accuracies of four representative heterogeneous multiplier configurations, denoted as Set 1–Set 4, are reported in Table 5. Each set represents a fixed layer-wise assignment of approximate multipliers, and its reported accuracy is a point estimate obtained on the independent held-out test set. These values should therefore be distinguished from the statistical results obtained across independent GA executions reported in Table A2. Additional configurations that performed comparably to the exact INT8 baseline, or exhibited slight accuracy degradation while offering favorable efficiency–accuracy trade-offs, are not explicitly included.

Table 5. Top-1 accuracy (%) of the FP32 models, exact INT8 baseline models, homogeneous approximate-multiplier models, and proposed heterogeneous approximate-multiplier configurations. All reported accuracy improvements for the proposed configurations are calculated relative to the corresponding exact INT8 baseline and are expressed in percentage points. All final Top-1 accuracy values were measured on the independent held-out test set, which was not used during GA optimization or configuration selection.

Model (Dataset)	CNN (MNIST)	CNN (FMNIST)	MLP (MNIST)	MLP (FMNIST)	VGG11 (CIFAR10)	VGG11 (CIFAR100)	VGG13 (CIFAR10)	VGG13 (CIFAR100)
Original (FP32)	99.26	91.27	98.24	89.58	91.78	70.64	93.35	72.02
Exact Multiplier (INT8 Baseline)	99.25	90.22	98.18	87.89	90.55	68.59	92.83	70.28
SCDM8_41 (1)	99.25	90.21	98.16	87.87	90.55	68.59	92.81	70.25
SCDM8_42 (2)	99.24	90.16	98.13	87.85	90.54	68.59	92.79	70.20
SCDM8_43 (3)	99.04	90.05	97.98	87.63	90.26	68.32	92.62	69.98
SCDM8_44 (4)	98.27	88.95	97.06	86.68	89.33	67.47	91.90	69.02
SCDM8_51 (5)	99.2	90.11	98.10	87.80	90.52	68.48	92.72	70.10
SCDM8_52 (6)	99.18	90.05	98.09	87.80	90.51	68.41	92.66	70.09
SCDM8_53 (7)	98.91	89.64	97.84	87.47	90.25	68.08	92.29	69.86
SCDM8_54 (8)	98.72	89.48	97.62	87.30	89.98	67.91	92.12	69.72
SCDM8_61 (9)	99.12	90.10	98.02	87.56	90.36	68.10	92.30	69.94
SCDM8_62 (10)	99.21	90.14	98.10	87.70	90.45	68.18	92.34	68.98
SCDM8_63 (11)	98.61	89.43	97.55	87.10	90.10	67.47	91.80	68.25
SCDM8_64 (12)	98.44	89.26	97.30	86.90	89.91	66.84	91.48	68.02
SCDM8_71 (13)	98.70	89.55	97.48	86.98	90.15	67.62	91.66	68.14
SCDM8_72 (14)	98.62	89.46	97.46	86.82	90.10	67.25	90.96	67.98
SCDM8_73 (15)	98.98	89.60	97.70	86.88	90.48	67.88	91.78	68.12
SCDM8_74 (16)	97.24	86.72	95.88	85.10	88.70	66.32	90.55	66.55
SCDM8_81 (17)	89.06	75.19	88.13	80.43	79.02	55.82	83.69	54.65
SCDM8_82 (18)	94.60	81.55	91.26	84.61	82.08	57.10	88.93	56.59
SCDM8_83 (19)	94.57	81.04	91.06	83.51	81.88	56.92	88.82	55.92
SCDM8_84 (20)	94.47	80.78	90.90	82.45	79.95	55.91	88.16	53.58
Set1	**99.36**	**90.83**	**98.38**	**88.27**	**90.83**	**69.06**	**93.19**	**70.60**
Set2	**99.41**	**91.18**	**98.44**	**89.02**	**91.15**	**69.61**	**93.35**	**71.28**
Set3	**99.45**	**91.29**	**98.83**	**89.26**	**91.62**	**70.55**	**93.47**	**71.56**
Set4	**99.50**	**91.41**	**98.89**	**89.68**	**91.88**	**70.66**	**93.55**	**72.10**

Note: Bold values indicate the results obtained using the proposed heterogeneous approximate-multiplier configurations (Set1–Set4).

5.2. Impact of Approximate Multipliers on DNN Information Flow

For the CNN and MLP case studies, the information values and corresponding information planes were computed using the exact (IB) analysis method for quantized DNN models, as introduced in [26]. The resulting information planes for the CNN and MLP models are depicted in Figure A1 and Figure A2, respectively. To analyze the impact of approximate multipliers on these case studies, we evaluated their effects on Top-1 accuracy and information flow. Specifically, we computed the information values while replacing all multiplications in the models with approximate multipliers (using each of the 20 available approximate multipliers). Finally, to assess the effectiveness of our proposed method on

these case studies, we implemented the heterogeneous genetic-based approach using the genetic algorithm detailed in Algorithm 1. The obtained points on the information plane, demonstrating the performance of our method, are shown in Appendix A.4.

To further evaluate our method in more sophisticated and comprehensive case studies, we implemented it on VGG11 and VGG13, trained on CIFAR-10 and CIFAR-100. Given the complexity and high-dimensional nature of these models, we employed Mutual Information Neural Estimation (MINE) [20,21] to compute the information values and construct the information plane. MINE was selected because it provides a scalable approach for estimating mutual information in high-dimensional representations such as those of VGG11 and VGG13 [21]. Due to the larger number of layers in VGG models compared to the other case studies, the information plane for each layer is depicted individually to enhance visualization clarity and interpretability. Similar to CNN and MLP, the initial visualizations represent the information plane obtained during the training of VGG11 and VGG13 on CIFAR-10 and CIFAR-100, with results reflecting the best-obtained Top-1 accuracy. Alternative configurations are detailed in Appendix A.1. Finally, we implemented the proposed heterogeneous genetic-based approach to evaluate its effectiveness in VGG11 and VGG13. The obtained points on the information plane (last layer), representing the performance of our method, are visualized in Figure A6.

To improve the interpretability of the information-plane analysis, Appendix A.4 now presents both the heterogeneous configurations selected by the GA and representative homogeneous approximate-multiplier baselines, with numerical axis values and clearer legends. The additional homogeneous figures show that replacing exact multipliers with approximate multipliers uniformly across all layers generally shifts the information-plane points toward lower $I(X;T_i)$, which in our analysis is consistent with stronger compression. However, this homogeneous use of approximation also tends to reduce $I(T_i;Y)$, indicating lower retention of label-related information and aligning with the accuracy degradation observed for homogeneous approximate-multiplier implementations.

In contrast, the heterogeneous configurations identified by the GA (Set 1–Set 4) reach the shaded green region shown in Figures A3 and A6, which represents combinations of $I(X;T_i)$ and $I(T_i;Y)$ that were not reached by the exact multiplier baseline or by the representative homogeneous approximate-multiplier configurations considered in this study. These heterogeneous points combine lower $I(X;T_i)$ with higher $I(T_i;Y)$ relative to the exact baseline, suggesting more compressed yet more task-relevant internal representations. In contrast, the representative homogeneous configurations shown in Figures A4 and A7 generally reduce $I(X;T_i)$ but also reduce $I(T_i;Y)$. Thus, the homogeneous use of approximation is associated with stronger compression but also lower retention of label-related information, whereas heterogeneous approximation enables the simultaneous movement toward lower $I(X;T_i)$ and higher $I(T_i;Y)$. The corresponding numerical shifts relative to the exact INT8 multiplier baseline are summarized in Table A7, quantitatively showing that the heterogeneous configurations predominantly decrease $I(X;T_i)$ while consistently increasing $I(T_i;Y)$, whereas the representative homogeneous configurations generally decrease both quantities.

These interpretations should, however, be read with appropriate caution. The absolute values of $I(X;T)$ and $I(T;Y)$, as well as the observed degree of compression, depend on the mutual-information estimation method and its assumptions. In this work, CNN and MLP information values were computed using the quantized IB analysis method, while the VGG11 and VGG13 results were obtained using MINE. Therefore, we do not interpret the results as estimator-independent proof that reduced mutual information directly causes improved generalization. Rather, within each model and estimator setting, the observed reduction in $I(X;T)$ is interpreted as increased compression, while an increase in $I(T;Y)$

indicates better retention of task-relevant information. The heterogeneous configurations are thus best understood as providing evidence consistent with improved generalization and accuracy, rather than conclusive proof of a universal causal relationship.

5.3. Optimizing Energy Utilization

As discussed, one of the primary challenges of DNN models is their high energy consumption, particularly on resource-constrained hardware implementations. Since multiplication operations constitute a significant portion of the computations in DNNs—especially in convolutional layers and fully connected layers, which form the core components of our case studies—reducing the energy usage of these operations is critical. By employing approximate multipliers, energy efficiency during model inference can be significantly improved. It should be noted that the energy results reported in this work are analytical estimates rather than direct measurements from a complete hardware implementation. The estimates are based on the normalized energy coefficients of the individual SCDM8 multipliers reported in [10] and the number of MAC operations performed in each network layer. Equation (9) therefore estimates the energy contribution associated with multiplication operations using a MAC-weighted model. The reported values do not represent full accelerator- or system-level energy measurements and do not account for implementation-dependent energy contributions outside the multiplier operations. The estimated energy savings achieved through the heterogeneous, layer-wise use of approximate multipliers in our case studies are presented in Table 6. In the proposed heterogeneous method, energy and delay efficiencies are determined by the efficiency of the multipliers used in each layer, with their impact directly proportional to the number of MAC operations per layer. Since our case studies include both convolutional and fully connected layers, we provide a detailed breakdown of the computational operations required for each type (detailed in Table A3). In the proposed heterogeneous method, the contribution of each multiplier to the total network energy is proportional to the number of MAC operations performed in its corresponding layer. Therefore, the normalized energy consumption of the network is calculated as the MAC-weighted average of the normalized energy coefficients reported in Table 3. The resulting percentage energy saving, S_e, is calculated as follows:

$$S_e = \left(1 - \frac{\sum_{\ell=1}^{L} \mathrm{MAC}_\ell \left(\frac{M_{\mathrm{AMult}_\ell}}{M_x}\right)}{\sum_{\ell=1}^{L} \mathrm{MAC}_\ell}\right) \times 100 \tag{9}$$

Here, L denotes the number of computational layers in the DNN, MAC_ℓ is the number of MAC operations in layer ℓ, M_x is the energy metric of the exact multiplier, and M_{AMult_ℓ} is the energy metric of the approximate multiplier assigned to layer ℓ. The ratio $M_{\mathrm{AMult}_\ell}/M_x$ is the normalized energy coefficient reported in Table 3. Consequently, the fraction in Equation (9) represents the normalized energy consumption of the complete network, and subtracting it from one yields the corresponding energy-saving ratio. The corrected energy-saving results are presented in Table 6.

5.4. Computational Cost Analysis

The proposed approach differs fundamentally from conventional gradient-based training. Conventional training optimizes network weights by minimizing a loss function through forward and backward propagation, whereas our method operates on a previously trained model and searches over discrete layer-wise assignments of approximate multipliers using validation Top-1 accuracy as the fitness criterion. Consequently, the two procedures do not optimize identical quantities, and their computational costs should not be interpreted as a direct algorithmic speedup under equivalent optimization objectives.

Nevertheless, comparing their MAC requirements provides an indication of the computational characteristics of the two procedures. The proposed GA relies predominantly on forward-pass evaluations and does not perform weight updates or backpropagation. A MAC-count analysis under the experimental settings used in this work is provided in Appendix A.2.

Table 6. Estimated energy savings for the heterogeneous approximate-multiplier configurations evaluated in the case studies. The values are analytical estimates based on the normalized multiplier energy coefficients in Table 3 and the layer-wise MAC counts, rather than full hardware measurements.

Model (Dataset, Set)	Energy Saving	Model (Dataset, Set)	Energy Saving
CNN (MNIST, Set1)	35.55%	CNN (FMNIST, Set1)	40.58%
CNN (MNIST, Set2)	35.71%	CNN (FMNIST, Set2)	37.26%
CNN (MNIST, Set3)	33.21%	CNN (FMNIST, Set3)	42.92%
CNN (MNIST, Set4)	37.04%	CNN (FMNIST, Set4)	44.75%
Median	35.63%	Median	41.75%
MLP (MNIST, Set1)	24.25%	MLP (FMNIST, Set1)	23.57%
MLP (MNIST, Set2)	23.96%	MLP (FMNIST, Set2)	24.80%
MLP (MNIST, Set3)	25.75%	MLP (FMNIST, Set3)	26.11%
MLP (MNIST, Set4)	24.12%	MLP (FMNIST, Set4)	24.94%
Median	24.18%	Median	24.87%
VGG11 (CIFAR10, Set1)	43.72%	VGG11 (CIFAR100, Set1)	38.36%
VGG11 (CIFAR10, Set2)	38.69%	VGG11 (CIFAR100, Set2)	43.51%
VGG11 (CIFAR10, Set3)	39.48%	VGG11 (CIFAR100, Set3)	37.35%
VGG11 (CIFAR10, Set4)	43.37%	VGG11 (CIFAR100, Set4)	36.14%
Median	41.42%	Median	37.85%
VGG13 (CIFAR10, Set1)	41.49%	VGG13 (CIFAR100, Set1)	36.38%
VGG13 (CIFAR10, Set2)	36.28%	VGG13 (CIFAR100, Set2)	29.93%
VGG13 (CIFAR10, Set3)	28.40%	VGG13 (CIFAR100, Set3)	28.17%
VGG13 (CIFAR10, Set4)	38.65%	VGG13 (CIFAR100, Set4)	38.02%
Median	37.46%	Median	33.16%

5.5. Comparison

To provide a relevant comparison, we selected recent studies on approximate multipliers that employed the same model architectures, datasets, and INT8 data format where possible. The comparative results are summarized in Table 7.

The absolute accuracy values and accuracy changes reported across these studies should not be interpreted as a controlled head-to-head comparison, since the underlying baseline models were trained using different hyperparameters and may have different starting accuracies. Accordingly, each accuracy change in Table 7 is reported relative to the baseline defined within the corresponding study. Despite these differences, a qualitative distinction can be observed: the considered prior approximate-multiplier approaches report accuracy degradation relative to their respective baselines, whereas the proposed heterogeneous configurations achieve positive accuracy changes relative to their corresponding exact INT8 baselines while simultaneously reducing energy consumption.

Table 7. Comparison of accuracy changes and energy savings for the proposed approach and related works. The accuracy changes of the proposed configurations are absolute percentage-point differences relative to their corresponding exact INT8 baselines. The existing-work values are reported relative to the baselines defined in the cited studies. Energy-saving values are reported according to the hardware assumptions and evaluation procedures of the corresponding studies and therefore should be interpreted as indicative rather than as a strictly normalized hardware comparison.

Work	Model	Dataset	Accuracy Change (%)	Energy Saving (%)
Existing Works				
[36]	VGG11	CIFAR-10	−0.24	32
[37]	VGG11	CIFAR-100	−1.20	52
[38]	VGG13	CIFAR-10	−0.69	39
[38]	VGG13	CIFAR-100	−1.91	39
[10]	CNN	MNIST	−0.07	51
[39]	CNN	FMNIST	−0.17	42
[40]	MLP	MNIST	−0.30	17
[41]	MLP	FMNIST	−1.84	8
Proposed Approach				
Set3	VGG11	CIFAR-10	**+1.07**	**39.48**
Set3	VGG11	CIFAR-100	**+1.96**	**37.35**
Set1	VGG13	CIFAR-10	**+0.36**	**41.49**
Set4	VGG13	CIFAR-100	**+1.82**	**38.02**
Set4	CNN	MNIST	**+0.25**	**37.04**
Set2	CNN	FMNIST	**+0.96**	**37.26**
Set4	MLP	MNIST	**+0.71**	**24.12**
Set3	MLP	FMNIST	**+1.37**	**26.11**

Note: Bold entries indicate the selected configurations and corresponding results obtained using the proposed heterogeneous approximate-multiplier approach.

The proposed configurations also provide substantial reductions in energy consumption while producing positive accuracy improvements. Specifically, for VGG11 on CIFAR-100, the proposed method improves Top-1 accuracy relative to its exact INT8 baseline while reducing energy consumption. In comparison, ref. [37] reports an accuracy degradation relative to its own baseline together with energy savings. Similarly, for VGG13 on CIFAR-100, the proposed method achieves a positive accuracy change together with reduced energy consumption, whereas ref. [38] reports an accuracy degradation relative to its own baseline. These values are not used to calculate a direct numerical accuracy advantage between the methods because their underlying baselines differ. Rather, they illustrate the different accuracy–energy trade-offs achieved by the approaches.

Although some existing approaches report higher energy savings, the distinguishing characteristic of the proposed heterogeneous method is that it reduces energy consumption while simultaneously producing a positive accuracy change relative to the corresponding exact INT8 baseline. Across the evaluated models, these results indicate that heterogeneous approximate-multiplier configurations can provide a favorable accuracy–energy trade-off without the accuracy degradation observed in the considered prior approaches.

6. Discussion

Our proposed method addresses one of the primary challenges associated with using approximate multipliers in DNNs—maintaining accuracy while improving computational efficiency. Approximate multipliers have been widely explored as an efficiency enhancement technique, particularly because multiplication is one of the most computationally demanding operations in AI models. However, prior research has primarily implemented approximate multipliers homogeneously across all layers, often leading to accuracy degra-

dation. In contrast, our heterogeneous approach not only achieves significant energy savings but also improves accuracy relative to the corresponding exact INT8 baseline (comparison detailed in Section 5.5).

By analyzing the information plane, we observed that the heterogeneous configurations reach regions not reached by the exact multiplier baseline or by the representative homogeneous approximate-multiplier configurations evaluated in this study. In particular, homogeneous approximation generally results in lower estimated $I(X;T)$, consistent with stronger compression, but is also accompanied by lower estimated $I(T;Y)$, consistent with reduced retention of label-related information and the observed accuracy degradation. In contrast, the heterogeneous configurations combine lower estimated $I(X;T)$ with higher estimated $I(T;Y)$, consistent with stronger compression while preserving more task-relevant information.

These observations should not be interpreted as demonstrating a universal causal relationship between reduced $I(X;T)$ and improved generalization. Mutual-information estimates depend on the estimation method and its assumptions; in this study, different estimators are used for the CNN/MLP and VGG experiments. Therefore, we interpret the observed information-plane movements within each model and estimator setting. The results provide evidence consistent with the Information Bottleneck interpretation, rather than estimator-independent proof that compression directly improves generalization [42,43].

The approximate multipliers introduce deterministic, operand-dependent arithmetic errors that modify the intermediate computations and information flow of the network. Our results show that different layer-wise assignments of these approximate arithmetic operations produce distinct information-plane characteristics and model accuracies. By strategically selecting and combining approximate multipliers across layers, the proposed method identifies heterogeneous configurations that improve accuracy while reducing energy consumption. Our approach represents a novel and counterintuitive optimization strategy that has not been explored in prior studies. While conventional wisdom suggests that introducing computational inexactness degrades accuracy, our results indicate that carefully designed heterogeneous configurations can achieve both higher accuracy and improved efficiency. To identify these optimal configurations, we employed a genetic algorithm, though other optimization techniques could also be applied. To the best of our knowledge, this is the first work demonstrating that heterogeneous approximation in DNNs can simultaneously improve both accuracy and energy efficiency.

Furthermore, we examined the ratio $I(Y;T)/I(X;T)$ as an auxiliary information-plane indicator. Within each model and mutual-information estimator setting, the heterogeneous configurations exhibit a shift toward higher $I(Y;T)$ and predominantly lower $I(X;T)$, consistent with the quantitative results reported in Appendix A.4. These changes are interpreted descriptively and should not be considered evidence that the arithmetic errors introduced by approximate multipliers act as a stochastic regularization mechanism. The approximate multipliers considered in this study introduce deterministic, operand-dependent arithmetic errors. A systematic characterization of their error distributions and a controlled comparison with conventional regularization techniques such as dropout and weight decay are beyond the scope of this work and are left for future investigation. The observed information-plane changes indicate that approximate arithmetic can alter the learned representations in ways that depend strongly on the multiplier type and its layer-wise placement. In the heterogeneous configurations identified by the GA, these changes are associated with higher $I(T;Y)$ and predominantly lower $I(X;T)$ relative to the exact baseline. We therefore interpret the results in terms of changes in information flow rather than as evidence of a stochastic regularization mechanism. Despite these promising

results, obtaining the information plane remains computationally expensive due to its statistical nature, and this challenge is scaled-up for complex models like VGG11 and VGG13. Additionally, modeling and simulating the behavior of approximate multipliers is computationally intensive, making our experiments particularly demanding. These constraints limited the inclusion of even more complex architectures in our analysis. The optimization framework is architecture-agnostic in the sense that the same layer-wise GA procedure can be applied to different DNN models without modifying its fundamental formulation. However, the optimized multiplier configuration is model- and dataset-specific, and the GA search must therefore be performed separately for each target model–dataset pair.

This one-time offline optimization introduces an additional deployment-stage search cost when adapting the method to a new model or dataset, although it does not add computational overhead during inference.

As demonstrated in Table 1, homogeneous implementations of SCDM8 multipliers already achieve 45% to 68% energy savings with less than 0.5% accuracy degradation on modern, deep architectures such as ResNet152, MobileNetV2, and ConvNeXt-T on ImageNet. These results motivate evaluating the heterogeneous approach on larger architectures such as ResNet152, MobileNetV2, and ConvNeXt-T. However, because the optimized multiplier configurations are model- and dataset-specific, whether similar gains can be obtained on these architectures must be established empirically.

7. Conclusions

We introduced a novel approach for integrating approximate multipliers into DNN models, achieving simultaneous accuracy improvements and estimated multiplier-energy savings, thereby overcoming the conventional accuracy–efficiency trade-off observed in approximate computing. Through Information-Plane analysis, we observed estimator-dependent changes in the learned representations that are consistent with increased compression and improved retention of task-relevant information under heterogeneous approximation. As a post-training configuration-search method, our approach has a lower estimated MAC workload than the corresponding gradient-based training schedules under the experimental settings considered. A key contribution of this work is serving as an existence proof that approximate multipliers—typically associated with accuracy degradation—can, under specific configurations, enhance accuracy while improving resource efficiency. This challenges the prevailing assumption in the literature that approximation inherently leads to accuracy loss. Our findings suggest that controlled heterogeneity in multiplier selection can lead to novel optimization opportunities in DNNs.

Furthermore, our study highlights the growing importance of energy-efficient hardware in AI, as computational and energy constraints become increasingly critical. By leveraging hardware-aware optimizations such as approximate multipliers, AI models can achieve greater accuracy, sustainability, and scalability. While further validation across diverse architectures is needed, our findings open new avenues for efficient AI hardware design. Future work will focus on scaling the genetic algorithm and heterogeneous methodology to more advanced architectures, specifically evaluating the impact of approximate multipliers on networks with skip-connections (e.g., ResNets) and depthwise separable convolutions (e.g., MobileNets). To mitigate the computational costs associated with simulating forward-pass metrics and the Information Plane for deep models on large-scale datasets like ImageNet, future efforts will explore optimized hardware-in-the-loop simulations and the use of proxy datasets.

https://doi.org/10.3390/ai7090385

Author Contributions: Conceptualization, S.S. and N.T.; methodology, S.S.; software, S.S.; validation, S.S. and N.A.; formal analysis, S.S. and N.A.; investigation, S.S.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, S.S., N.T. and A.J.; visualization, S.S. and N.A.; supervision, N.T. and A.J. All authors have read and agreed to the published version of the manuscript.

Funding: European Union's Horizon 2020 (H2020) Marie Sklodowska-Curie Innovative Training Networks H2020-MSCA-ITN-2020 Research and Innovation Programme (Grant No. 956090).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this study—MNIST and Fashion-MNIST—are publicly available benchmark datasets. MNIST can be accessed at https://www.kaggle.com/datasets/hojjatk/mnist-dataset (accessed on 10 October 2024), and Fashion-MNIST is available at https://github.com/zalandoresearch/fashion-mnist (accessed on 10 October 2024).

Acknowledgments: The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme. Some resources and services used in this work were provided by the VSC (Vienna Scientific Cluster). 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). The authors at Heidelberg University gratefully acknowledge the support of the Hector Stiftung through project 2304191. During the preparation of this manuscript/study, the author(s) used OpenAI ChatGPT (GPT-5.6) for the purposes of language editing, grammatical correction, and improving the clarity and readability of the text. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Hyperparameter Tuning

To evaluate the reproducibility of the baseline model training, each model–dataset pair was trained repeatedly using the final selected hyperparameters and independent random seeds. The random seeds affected the model initialization and training-data order. Thirty independent training runs were performed for CNN on MNIST, while ten independent runs were performed for each of the remaining model–dataset pairs.

Table A1 reports the mean, standard deviation, and 95% confidence interval of the resulting FP32 Top-1 test accuracies. The final column reports the FP32 accuracy currently presented in Table 5, which corresponds to the best-performing FP32 model obtained during the repeated training experiments. This trained FP32 model was subsequently used to obtain the corresponding exact INT8 baseline and to conduct the approximate-multiplier experiments.

Pilot experiments were conducted to select batch size, epochs, and optimizer for each model–dataset pair. The repeated-training results are reported as mean ± standard deviation across the independent training runs summarized in Table A1. Learning rates were tuned within standard ranges (10^{-3}–10^{-2} for Adam, 10^{-3}–10^{-1} for SGD).

Appendix A.1.1. VGGs on CIFAR-10 and CIFAR-100

For VGG11 and VGG13 (SGD, momentum 0.9), performance improved with larger batch sizes and longer training. The best settings were: CIFAR-10 with batch sizes of 256 and 250 epochs (both models), and CIFAR-100 with batch sizes of 512 and 300 epochs. Adam further improved VGG11 on CIFAR-100 by 1–2%. Final configurations are: VGG11 (CIFAR-

10) 256/250/SGD, VGG11 (CIFAR-100) 512/300/Adam, VGG13 (CIFAR-10) 256/250/SGD, and VGG13 (CIFAR-100) 512/300/SGD.

Appendix A.1.2. MLP on MNIST and Fashion-MNIST

For MLP, increasing epochs improved convergence. On MNIST, extending training from 1000 to 2000 epochs improved performance, with SGD slightly outperforming Adam. On Fashion-MNIST, Adam benefited more from longer training and surpassed SGD. Final settings are: MLP (MNIST) 150/2000/SGD and MLP (FMNIST) 150/2000/Adam.

Appendix A.1.3. CNN on MNIST and Fashion-MNIST

For CNN, batch size was fixed at 256. Increasing epochs (1000–3000) improved accuracy, and Adam consistently outperformed SGD at higher epochs. Final settings are: CNN (MNIST) 256/3000/Adam and CNN (FMNIST) 256/3000/Adam.

These hyperparameters balance model complexity, dataset difficulty, optimizer behavior, and convergence requirements.

The statistics in Table A1 describe the variability of the FP32 model-training procedure. They do not describe repeated GA executions. The Set1–Set4 results in Table 5 represent fixed layer-wise approximate-multiplier configurations applied to the selected trained model and are therefore reported as individual point accuracies.

Table A1. Statistical results of repeated FP32 model training. The mean, standard deviation, and 95% confidence interval are calculated across independent training runs. The final column shows the FP32 accuracy reported in Table 5.

Model	Dataset	Training Runs	Top-1 Accuracy Mean ± SD	95% CI	FP32 Accuracy in Table 5
CNN	MNIST	30	99.21 ± 0.05	[99.19, 99.23]	99.26
CNN	Fashion-MNIST	10	91.10 ± 0.18	[90.97, 91.23]	91.27
MLP	MNIST	10	98.18 ± 0.06	[98.14, 98.22]	98.24
MLP	Fashion-MNIST	10	89.26 ± 0.34	[89.02, 89.50]	89.58
VGG11	CIFAR-10	10	91.58 ± 0.22	[91.42, 91.74]	91.78
VGG11	CIFAR-100	10	70.41 ± 0.30	[70.20, 70.62]	70.64
VGG13	CIFAR-10	10	93.35 ± 0.03	[93.33, 93.37]	93.35
VGG13	CIFAR-100	10	71.92 ± 0.14	[71.82, 72.02]	72.02

Appendix A.1.4. Robustness of the Genetic-Algorithm Search

To evaluate the robustness of the proposed search procedure with respect to stochastic initialization, the genetic algorithm was executed independently 10 times for each model–dataset pair using different random seeds. Each run started from an independently generated initial population, while the GA hyperparameters and evaluation protocol were kept unchanged. Table A2 reports the mean and standard deviation of the Top-1 accuracy obtained across the 10 independent GA runs. The relatively small standard deviations indicate that the GA repeatedly identifies high-performing regions of the heterogeneous multiplier-configuration space despite variations in its random initialization. These GA repetitions are distinct from the repeated FP32 training runs reported in Table A1.

Table A2. Robustness of the GA search across 10 independent runs with different random seeds.

Model	Dataset	Independent GA Runs	Top-1 Accuracy (Mean ± SD) (%)
CNN	MNIST	10	99.44 ± 0.05
CNN	Fashion-MNIST	10	91.26 ± 0.18
MLP	MNIST	10	98.68 ± 0.21
MLP	Fashion-MNIST	10	89.24 ± 0.38
VGG11	CIFAR-10	10	91.48 ± 0.07
VGG11	CIFAR-100	10	70.41 ± 0.46
VGG13	CIFAR-10	10	93.47 ± 0.08
VGG13	CIFAR-100	10	71.72 ± 0.34

A two-sided one-sample *t*-test was performed across the 10 independent GA executions for each model–dataset pair, using the corresponding exact INT8 accuracy as the reference value. The resulting differences were statistically significant for all evaluated cases ($p < 0.001$). Because these GA executions were performed on the same selected trained model, this analysis quantifies variability associated with GA initialization and should not be interpreted as accounting for variability across independently trained DNN models.

Appendix A.2. Computational Cost Analysis

We characterize the computational cost of the proposed GA search and conventional gradient-based training in terms of Multiply–Accumulate (MAC) operations in convolutional and fully connected layers. It is important to emphasize that this is not an equivalent-objective benchmark: conventional training optimizes network weights through loss minimization and backpropagation, whereas the proposed GA searches discrete approximate-multiplier configurations for a previously trained model using validation accuracy as its fitness criterion. The following analysis therefore compares the estimated MAC counts associated with the two procedures under the experimental settings used in this study, rather than establishing a direct algorithmic speedup.

For a convolutional layer, the number of MACs is $K(W - F + 1)(H - F + 1)F^2D$, where K is the number of filters, W and H are the input width and height, F is the filter size, and D is the input depth. For a fully connected layer, the number of MACs is $N \times M$, where N and M are the input and output dimensions. Using these expressions, we computed the layer-wise MAC counts shown in Table A3 (in millions).

The proposed method acts as an efficient GA-based configuration search, achieving higher accuracy with lower computational cost. Its complexity is governed by the genetic algorithm parameters: population size P, number of generations G, input size N, and fitness cost F, where F corresponds to the DNN forward pass for batch size T. Per generation, the fitness evaluation costs $P \times F$, selection costs $P\log(P)$, and crossover/mutation costs $P \times N$; therefore, the total complexity is $G(PF + P\log(P) + PN)$.

Table A3. Number of MACs in each layer of case studies.

Model (Dataset)	Conv MACs (M)	FC MACs (M)	Total (M)
VGG11 (CIFAR10)	[1.179, 1.179, 1.179, 2.359, 1.179, 2.359, 1.179, 1.179]	[2.097, 16.777, 0.040]	30.70
VGG11 (CIFAR100)	[1.179, 1.179, 1.179, 2.359, 1.179, 2.359, 1.179]	[2.097, 16.777, 0.410]	31.08
VGG13 (CIFAR10)	[1.179, 1.179, 1.179, 1.179, 1.179, 2.359, 1.179, 2.359, 1.179, 1.179]	[2.097, 16.777, 0.040]	33.07
VGG13 (CIFAR100)	[1.179, 1.179, 1.179, 1.179, 1.179, 2.359, 1.179, 2.359, 1.179, 1.179]	[2.097, 16.777, 0.410]	33.43
MLP (MNIST/FMNIST)	-	[0.15, 0.02, 0.00]	0.17
CNN (MNIST/FMNIST)	[0.05, 1.60, 0.40, 0.77, 0.16]	[0.29, 0.01, 0.00]	3.58

For traditional training, the cost of one epoch is $C_T^e = S_{TS}(C_F + C_B)$, where S_{TS} is the training-set size, C_F is the forward-pass complexity, and $C_B \approx 2C_F$ is the backward-pass complexity [44]. Combining these expressions with the experimental settings in Tables 4, A3 and A4, the final comparison is summarized in Table A5.

Under the experimental settings used in this study, the ratio between the estimated MAC counts of conventional training and the proposed GA search ranges from 146× to 22,500×, as summarized in Table A5. These values should be interpreted as MAC-count ratios rather than direct computational speedups, because the two procedures optimize different variables and employ different optimization mechanisms and stopping criteria. The lower MAC requirement of the GA mainly results from its forward-pass-only fitness evaluations and the absence of gradient computation and weight updates. Moreover, these analytical MAC counts do not account for implementation-dependent factors such as memory access, GA-management overhead, hardware utilization, or wall-clock execution time. Therefore, the comparison is intended to characterize the relative computational workload of the procedures under our settings rather than to establish an equivalent-task performance benchmark.

Table A4. Final hyperparameters selected from pilot experiments.

Model (Dataset)	Batch	Epochs	Opt.
VGG11 (CIFAR-10)	256	250	SGD
VGG11 (CIFAR-100)	512	300	Adam
VGG13 (CIFAR-10)	256	250	SGD
VGG13 (CIFAR-100)	512	300	SGD
MLP (MNIST)	150	2000	SGD
MLP (FMNIST)	150	2000	Adam
CNN (MNIST)	256	3000	Adam
CNN (FMNIST)	256	3000	Adam

Table A5. Estimated MAC counts of conventional training and the proposed GA search under the experimental settings used in this study. The reported ratio is a MAC-count ratio and should not be interpreted as a direct algorithmic speedup because the two procedures optimize different quantities.

Model (Dataset)	Traditional	Proposed	MAC-Count Ratio
VGG11 (CIFAR10)	1.24×10^{15}	5.96×10^{12}	208
VGG11 (CIFAR100)	1.51×10^{15}	6.89×10^{12}	219
VGG13 (CIFAR10)	1.28×10^{15}	8.77×10^{12}	146
VGG13 (CIFAR100)	1.55×10^{15}	8.77×10^{12}	177
MLP (MNIST)	6.12×10^{13}	2.72×10^{9}	22,500
MLP (FMNIST)	6.12×10^{13}	4.08×10^{9}	15,000
CNN (MNIST)	1.93×10^{15}	1.14×10^{11}	16,940
CNN (FMNIST)	2.56×10^{15}	1.14×10^{11}	22,460

Appendix A.3. Identified Approximate Multiplier Sets

This appendix reports the layer-wise configurations of approximate multipliers identified by the genetic algorithm. Table A6 lists the four highest-performing sets for each model–dataset pair (CNN, MLP, VGG11, VGG13).

Each set defines a heterogeneous assignment of SCDM8 approximate multipliers (indices as in Section 3.1), ordered according to layer sequence. All models use INT8 quantization, with a single multiplier applied uniformly per layer.

The reported configurations correspond to the best-performing solutions in Section 5, achieving improved Top-1 accuracy alongside reduced energy consumption. Multiple

sets per case highlight the presence of several competitive solutions, reflecting the non-uniqueness of optimal configurations.

Table A6. Obtained approximate multiplier sets for all case studies.

Model	Approximate Multiplier Set	Model	Approximate Multiplier Set
CNN (MNIST)		VGG11 (CIFAR10)	
Set1	[19, 10, 6, 3, 10, 6, 12, 10]	Set1	[8, 5, 6, 9, 18, 16, 9, 17, 1, 18, 7]
Set2	[18, 5, 14, 10, 3, 8, 6, 19]	Set2	[15, 5, 6, 1, 5, 8, 2, 11, 13, 11, 6]
Set3	[18, 5, 6, 13, 7, 3, 1, 14]	Set3	[19, 14, 1, 10, 12, 6, 1, 18, 2, 11, 1]
Set4	[12, 8, 14, 2, 5, 9, 15, 2]	Set4	[16, 5, 5, 17, 1, 2, 16, 15, 6, 18, 7]
CNN (FMNIST)		VGG11 (CIFAR100)	
Set1	[12, 15, 2, 9, 17, 1, 18, 16]	Set1	[12, 15, 2, 9, 13, 4, 1, 18, 2, 11, 7]
Set2	[16, 7, 4, 10, 18, 1, 18, 7]	Set2	[1, 15, 9, 18, 16, 13, 6, 8, 1, 18, 6]
Set3	[15, 14, 6, 13, 16, 11, 7, 5]	Set3	[3, 5, 1, 4, 12, 10, 9, 17, 1, 11, 1]
Set4	[19, 14, 9, 18, 15, 10, 7, 16]	Set4	[16, 15, 2, 4, 2, 1, 8, 17, 1, 7, 6]
MLP (MNIST)		VGG13 (CIFAR10)	
Set1	[1, 11, 1]	Set1	[19, 14, 6, 9, 1, 10, 12, 3, 5, 17, 1, 18, 7]
Set2	[1, 7, 5]	Set2	[3, 14, 18, 10, 12, 1, 3, 6, 9, 17, 1, 7, 6]
Set3	[2, 7, 6]	Set3	[10, 15, 2, 4, 11, 12, 2, 3, 8, 18, 1, 1, 7]
Set4	[2, 2, 12]	Set4	[16, 15, 6, 1, 1, 12, 15, 17, 2, 18, 2, 7, 6]
MLP (FMNIST)		VGG13 (CIFAR100)	
Set1	[1, 6, 1]	Set1	[16, 15, 2, 4, 1, 5, 18, 4, 3, 17, 1, 7, 6]
Set2	[1, 14, 7]	Set2	[10, 15, 10, 7, 2, 16, 10, 2, 8, 18, 1, 1, 7]
Set3	[2, 11, 7]	Set3	[10, 15, 2, 1, 11, 5, 5, 6, 8, 18, 1, 1, 7]
Set4	[1, 18, 2]	Set4	[16, 15, 2, 1, 2, 9, 7, 11, 15, 18, 2, 7, 6]

Appendix A.4. Information Planes

This section presents the Information-Plane (IP) results for all case studies. The information planes obtained during training for CNN and MLP are shown in Figure A1 and Figure A2, respectively. Figure A3 compares the exact multiplier baseline with the heterogeneous configurations (Set1–Set4), while Figure A4 presents representative homogeneous approximate-multiplier implementations using SCDM8_41 and SCDM8_84. For VGG11 and VGG13, Figure A5 presents the information planes obtained during training. Due to the higher dimensionality and complexity of the VGG models, the comparison focuses on the last fully connected layer. Figure A6 presents the heterogeneous configurations, whereas Figure A7 compares the exact multiplier with the representative homogeneous approximate-multiplier implementations. To quantify the shifts observed in the information-plane figures, we define the changes relative to the exact INT8 multiplier baseline as

$$\Delta I_X = I_{\text{config}}(X;T) - I_{\text{exact}}(X;T), \tag{A1}$$

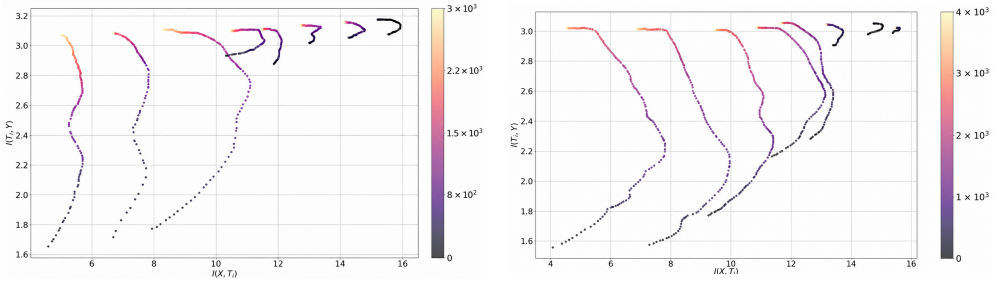
and

$$\Delta I_Y = I_{\text{config}}(T;Y) - I_{\text{exact}}(T;Y). \tag{A2}$$

For CNN and MLP, the reported shifts correspond to the final analyzed representation, while for VGG11 and VGG13 they correspond to the final fully connected layer, FC_3. The resulting ranges are summarized in Table A7.

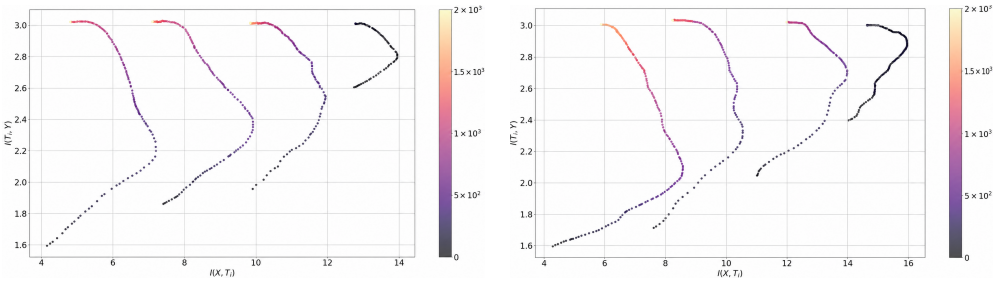
Table A7. Quantitative information-plane shifts relative to the exact INT8 multiplier baseline. For the homogeneous case, the reported ranges correspond to SCDM8_41 and SCDM8_84. For the heterogeneous case, the ranges correspond to Set1–Set4. Negative ΔI_X indicates a shift toward lower $I(X;T)$, whereas positive ΔI_Y indicates increased retention of label-related information within the adopted mutual-information estimator.

Model	Dataset	Homogeneous ΔI_X	Homogeneous ΔI_Y	Heterogeneous ΔI_X	Heterogeneous ΔI_Y
CNN	MNIST	$[-0.326, -0.009]$	$[-0.022, -0.008]$	$[-0.296, -0.074]$	$[+0.011, +0.024]$
CNN	Fashion-MNIST	$[-0.115, +0.004]$	$[-0.021, -0.006]$	$[-0.121, -0.010]$	$[+0.002, +0.019]$
MLP	MNIST	$[-0.286, -0.008]$	$[-0.061, -0.038]$	$[-0.107, -0.007]$	$[+0.001, +0.019]$
MLP	Fashion-MNIST	$[-0.195, -0.002]$	$[-0.057, -0.005]$	$[-0.162, +0.002]$	$[+0.003, +0.029]$
VGG11	CIFAR-10	$[-0.282, -0.072]$	$[-0.795, -0.206]$	$[-0.199, -0.045]$	$[+0.005, +0.017]$
VGG11	CIFAR-100	$[-0.152, -0.072]$	$[-0.332, -0.243]$	$[-0.165, -0.051]$	$[+0.042, +0.125]$
VGG13	CIFAR-10	$[-0.316, -0.032]$	$[-0.020, -0.007]$	$[-0.109, -0.030]$	$[+0.001, +0.016]$
VGG13	CIFAR-100	$[-0.365, -0.047]$	$[-0.100, -0.066]$	$[-0.305, -0.060]$	$[+0.046, +0.115]$



(**a**) CNN (MNIST) (**b**) CNN (FMNIST)

Figure A1. Information planes for CNN during training.



(**a**) MLP (MNIST) (**b**) MLP (FMNIST)

Figure A2. Information planes for MLP during training.

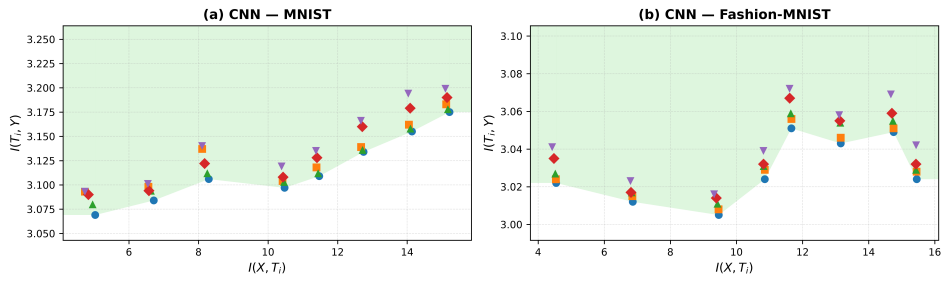


Figure A3. *Cont.*

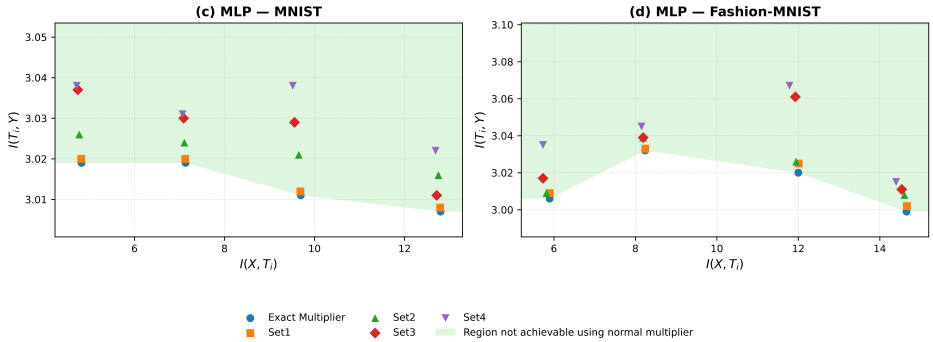


Figure A3. Information-plane locations of the exact multiplier baseline and the heterogeneous approximate-multiplier configurations (Set1–Set4) for CNN and MLP case studies. The shaded green region indicates points not attained by the exact multiplier baseline or by the representative homogeneous approximate-multiplier configurations considered in this work.

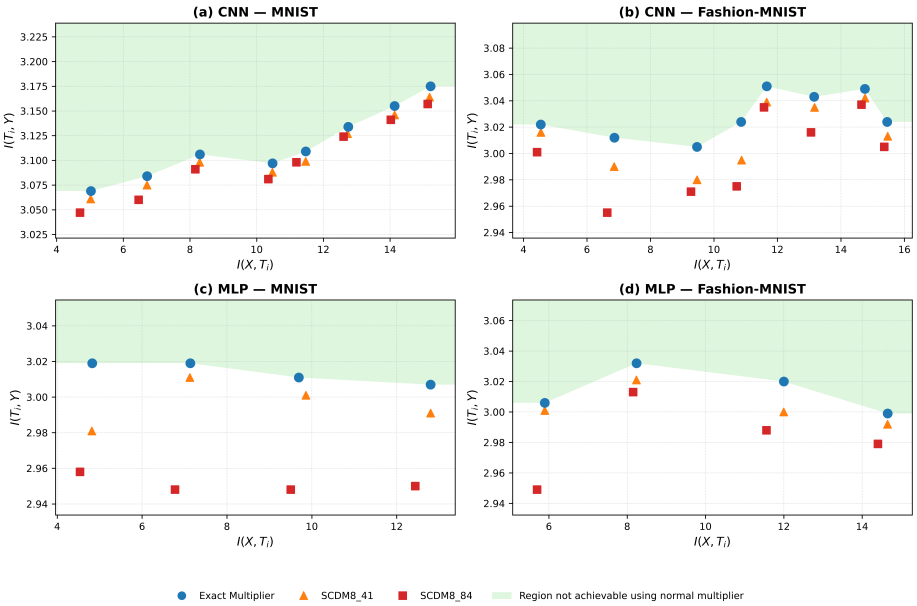
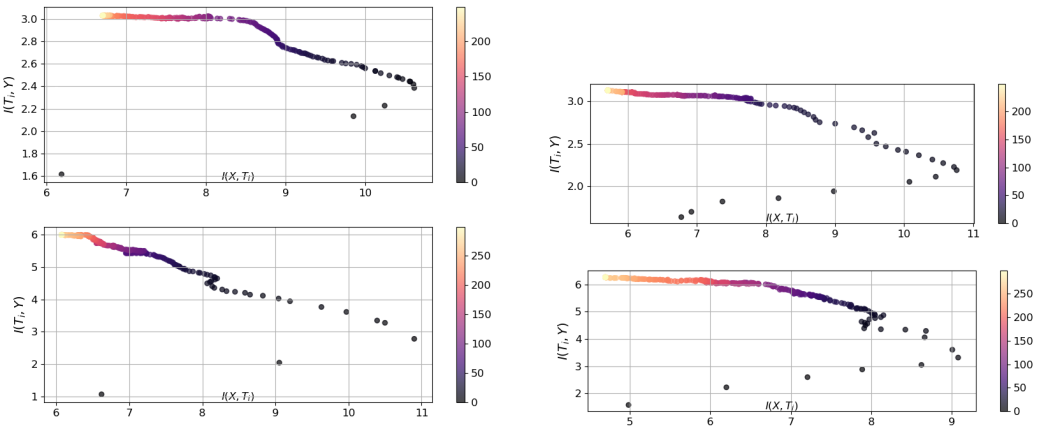


Figure A4. Information-plane comparison of the exact multiplier baseline and two representative homogeneous approximate-multiplier implementations (`SCDM8_41` and `SCDM8_84`) for CNN and MLP case studies. Homogeneous approximation generally shifts the points toward lower $I(X;T_i)$, but often also lower $I(T_i;Y)$, indicating increased compression together with reduced label-related information.



(**a**) VGG11 (**b**) VGG13

Figure A5. Information planes during training for VGG11 and VGG13 on CIFAR-10 and CIFAR-100: (**a**) VGG11 and (**b**) VGG13.

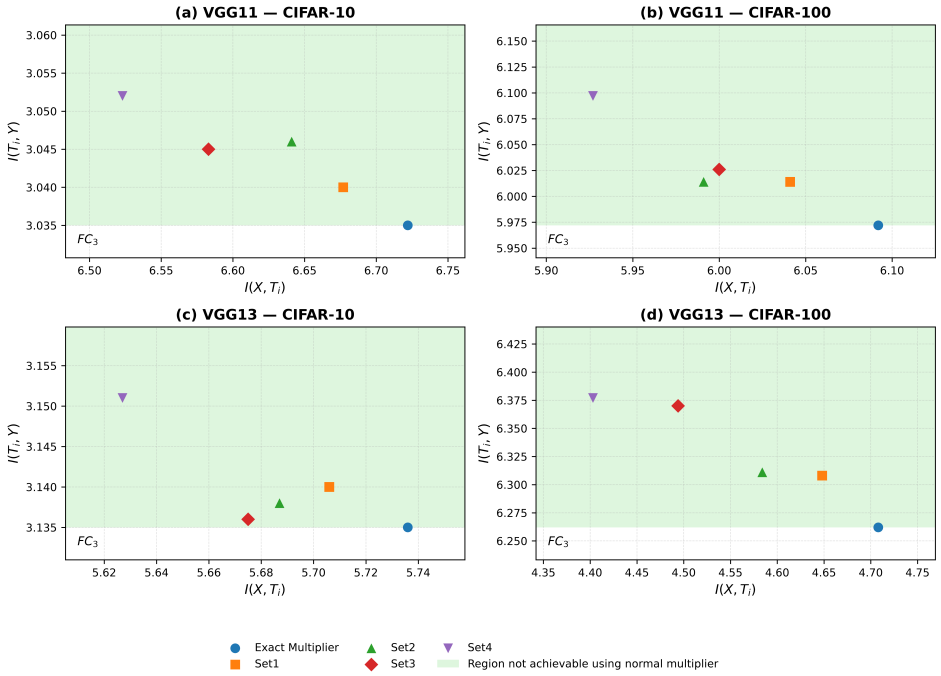


Figure A6. Information-plane locations of the exact multiplier baseline and the heterogeneous approximate-multiplier configurations (Set 1–Set 4) for the last fully connected layer (`FC_3`) of VGG11 and VGG13. The shaded green region indicates points not attained by the exact multiplier baseline or by the representative homogeneous approximate-multiplier configurations considered in this work.

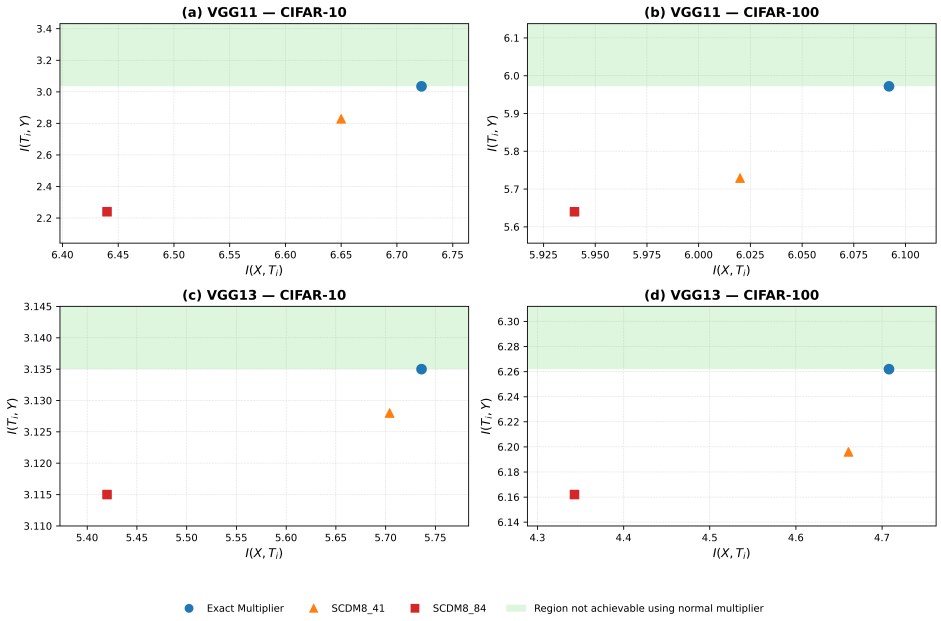


Figure A7. Information-plane comparison of the exact multiplier baseline and two representative homogeneous approximate-multiplier implementations (`SCDM8_41` and `SCDM8_84`) for the last fully connected layer (`FC_3`) of VGG11 and VGG13.

References

1. Chai, J.; Zeng, H.; Li, A.; Ngai, E.W. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **2021**, *6*, 100134. [CrossRef]
2. Min, B.; Ross, H.; Sulem, E.; Veyseh, A.P.B.; Nguyen, T.H.; Sainz, O.; Agirre, E.; Heintz, I.; Roth, D. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.* **2023**, *56*, 1–40. [CrossRef]
3. Sengar, S.S.; Hasan, A.B.; Kumar, S.; Carroll, F. Generative artificial intelligence: A systematic review and applications. *Multimed. Tools Appl.* **2024**, *84*, 23661–23700. [CrossRef]

4. Mall, P.K.; Singh, P.K.; Srivastav, S.; Narayan, V.; Paprzycki, M.; Jaworska, T.; Ganzha, M. A comprehensive review of deep neural networks for medical image processing: Recent developments and future opportunities. *Healthc. Anal.* **2023**, *4*, 100216. [CrossRef]
5. Deekshith, A. Scalable Machine Learning: Techniques for Managing Data Volume and Velocity in AI Applications. *Int. Sci. J. Res.* **2023**, *5*.
6. Freeda, A.R.; Kanthavel, R.; Anju, A. Scalability Issues in AI Computing in Large-Scale Networks. In *AI for Large Scale Communication Networks*; IGI Global: Hershey, PA, USA , 2025; pp. 395–414.
7. Damsgaard, H.J.; Ometov, A.; Nurmi, J. Approximation opportunities in edge computing hardware: A systematic literature review. *ACM Comput. Surv.* **2023**, *55*, 1–49. [CrossRef]
8. Younis, A.; Maheshwari, S.; Pompili, D. Energy-Latency Computation Offloading and Approximate Computing in Mobile-Edge Computing Networks. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 3401–3415. [CrossRef]
9. Amirafshar, N.; Baroughi, A.S.; Shahhoseini, H.S.; TaheriNejad, N. Carry Disregard Approximate Multipliers. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2023**, *70*, 4840–4853. [CrossRef]
10. Shakibhamedan, S.; Amirafshar, N.; Baroughi, A.S.; Shahhoseini, H.S.; Taherinejad, N. ACE-CNN: Approximate Carry Disregard Multipliers for Energy-Efficient CNN-Based Image Classification. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2024**, *71*, 2280–2293. [CrossRef]
11. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In *Proceedings of the 2015 IEEE Information Theory Workshop (ITW)*; IEEE: Piscataway, NJ, USA, 2015; pp. 1–5.
12. Goldfeld, Z.; Polyanskiy, Y. The Information Bottleneck Problem and its Applications in Machine Learning. *IEEE J. Sel. Areas Inf. Theory* **2020**, *1*, 19–38. [CrossRef]
13. Armeniakos, G.; Zervakis, G.; Soudris, D.; Henkel, J. Hardware approximate techniques for deep neural network accelerators: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–36. [CrossRef] [PubMed]
14. Rokh, B.; Azarpeyvand, A.; Khanteymoori, A. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Trans. Intell. Syst. Technol.* **2023**, *14*, 1–50. [CrossRef]
15. Sabetzadeh, F.; Moaiyeri, M.H.; Ahmadinejad, M. An Ultra-Efficient Approximate Multiplier with Error Compensation for Error-Resilient Applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 776–780. [CrossRef]
16. Park, G.; Kung, J.; Lee, Y. Design and Analysis of Approximate Compressors for Balanced Error Accumulation in MAC Operator. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2021**, *68*, 2950–2961. [CrossRef]
17. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 850–862. [CrossRef]
18. Mrazek, V.; Hrbacek, R.; Vasicek, Z.; Sekanina, L. EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), 2017, Lausanne, Switzerland, 27–31 March 2017; pp. 258–261. [CrossRef]
19. Yu, S.; Wickstrøm, K.; Jenssen, R.; Principe, J.C. Understanding convolutional neural networks with information theory: An initial exploration. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 435–442. [CrossRef] [PubMed]
20. Belghazi, M.I.; Baratin, A.; Rajeswar, S.; Ozair, S.; Bengio, Y.; Courville, A.; Hjelm, R.D. Mine: Mutual information neural estimation. *arXiv* **2018**, arXiv:1801.04062.
21. Jónsson, H.; Cherubini, G.; Eleftheriou, E. Convergence Behavior of DNNs with Mutual-Information-Based Regularization. *Entropy* **2020**, *22*, 727. [CrossRef] [PubMed]
22. Song, Z.; Chen, Z.; Xia, S. Novel Regularization Method Exploiting Mutual Information for Deep Neural Networks. In Proceedings of the 2023 2nd International Conference on Robotics, Artificial Intelligence and Intelligent Control (RAIIC), Mianyang, China, 11–13 August 2023; pp. 329–333. [CrossRef]
23. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
24. Hafez-Kolahi, H.; Kasaei, S. Information bottleneck and its applications in deep learning. *arXiv* **2019**, arXiv:1904.03743.
25. Shwartz-Ziv, R.; Tishby, N. Opening the black box of deep neural networks via information. *arXiv* **2017**, arXiv:1703.00810.
26. Lorenzen, S.S.; Igel, C.; Nielsen, M. Information bottleneck: Exact analysis of (quantized) neural networks. *arXiv* **2021**, arXiv:2106.12912.
27. Kirsch, A.; Lyle, C.; Gal, Y. Scalable training with information bottleneck objectives. In Proceedings of the International Conference on Machine Learning (ICML): Workshop on Uncertainty and Robustness in Deep Learning, Virtual, 17–18 July 2020; pp. 17–18.
28. Shwartz-Ziv, R.; Alemi, A.A. Information in infinite ensembles of infinitely-wide neural networks. In Proceedings of the 2nd Symposium on Advances in Approximate Bayesian Inference, Vancouver, BC, Canada, 8 December 2019; Volume 118, pp. 1–17.
29. Darlow, L.N.; Storkey, A. What information does a ResNet compress? *arXiv* **2020**, arXiv:2003.06254.
30. Goldfeld, Z.; Berg, E.v.d.; Greenewald, K.; Melnyk, I.; Nguyen, N.; Kingsbury, B.; Polyanskiy, Y. Estimating information flow in deep neural networks. *arXiv* **2018**, arXiv:1810.05728.
31. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]

32. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
33. Simonyan, K. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
34. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009; pp. 1–58. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 12 September 2026).
35. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://www.tensorflow.org/ (accessed on 12 September 2026).
36. Gu, F.Y.; Lin, I.C.; Lin, J.W. A Low-Power and High-Accuracy Approximate Multiplier with Reconfigurable Truncation. *IEEE Access* **2022**, *10*, 60447–60458. [CrossRef]
37. Zervakis, G.; Anagnostopoulos, I.; Salamin, S.; Spantidi, O.; Roman-Ballesteros, I.; Henkel, J.; Amrouch, H. Thermal-Aware Design for Approximate DNN Accelerators. *IEEE Trans. Comput.* **2022**, *71*, 2687–2697. [CrossRef]
38. Zervakis, G.; Spantidi, O.; Anagnostopoulos, I.; Amrouch, H.; Henkel, J. Control variate approximation for DNN accelerators. In *Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (DAC)*; IEEE: Piscataway, NJ, USA, 2021; pp. 481–486.
39. Gowda, B.G.; N, R.S.; C, P.H.; Nandi, P.; Rao, M. ApproxCNN: Evaluation of CNN with Approximated Layers Using In-Exact Multipliers. In *Proceedings of the 2023 IEEE 41st International Conference on Computer Design (ICCD), Washington, DC, USA, 6–8 November 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 46–53. [CrossRef]
40. Pei, H.; Yi, X.; Zhou, H.; He, Y. Design of Ultra-Low Power Consumption Approximate 4–2 Compressors Based on the Compensation Characteristic. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 461–465. [CrossRef]
41. Strollo, A.G.M.; Napoli, E.; De Caro, D.; Petra, N.; Saggese, G.; Di Meo, G. Approximate Multipliers Using Static Segmentation: Error Analysis and Improvements. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 2449–2462. [CrossRef]
42. Saxe, A.M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B.D.; Cox, D.D. On the information bottleneck theory of deep learning. *J. Stat. Mech. Theory Exp.* **2019**, *2019*, 124020. [CrossRef]
43. Cheng, H.; Lian, D.; Gao, S.; Geng, Y. Utilizing Information Bottleneck to Evaluate the Capability of Deep Neural Networks for Image Classification. *Entropy* **2019**, *21*, 456. [CrossRef] [PubMed]
44. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.