Learning the Error Features of Approximate Multipliers for Neural Network Applications

Hai Mo*, Yong Wu*, Honglan Jiang, *Member, IEEE,* Zining Ma, Fabrizio Lombardi, *Fellow, IEEE,* Jie Han, *Senior Member, IEEE,* and Leibo Liu, *Senior Member, IEEE*

***Abstract*—Approximate multipliers (AMs) have widely been investigated to pursue high-performance and energy-efficient hardware designs for error-tolerant applications, such as neural networks (NNs). The computing accuracy of an AM has been evaluated by using statistical error features; however, it is difficult to estimate the quality of a specific application using AMs. Thus, it is of a great challenge to select or design appropriate AMs for an accuracy-constrained application. This paper proposes an application-oriented error evaluation framework for AMs with the aim of exploring the correlation between statistical error features of AMs and the accuracy degradation in AM-based NN applications. Specifically, based on the Dropout Feature Ranking technique, statistical error features of AMs are extensively studied and ranked by their importance to the accuracy of AM-based NN applications. The three most informative features are obtained to construct error models to predict the accuracy loss of AM-based NN applications. The constructed classification models show a probability higher than 96% for correctly classifying the AMs into three categories in accordance with the induced accuracy loss in AM-based NN applications. Furthermore, regression models can predict the accuracy of NN applications using an AM with a deviation as low as 6%. These results show that the proposed error evaluation framework can guide an efficient selection of AMs for NN applications by using just several AM error features, instead of running time-consuming and complicated hardware simulation. The obtained statistical error features can also provide a guidance for the design or generation of application-oriented AMs. Moreover, the proposed framework is applicable for quickly analyzing and selecting other approximate circuits for error-tolerant applications.**

***Index Terms*—approximate multiplier, statistical error feature, error model, accuracy degradation, machine learning**

I. Introduction

With the unprecedented development of big data and artificial intelligence, hardware with high performance and low power is required for processing massive amount of data. Due to the limitation of CMOS technology, it has become increasingly difficult to further improve performance and reduce power of the computing circuits. Meanwhile, some errors can be neglected in the computational outcome in numerous applications, such as image processing, audio processing, and machine learning (ML) [1]. Thus, approximate or low-precision computing has been extensively considered to improve hardware efficiency with a reasonable accuracy degradation in some error-resilient applications [2].

Over the last few years, approximate computing has appeared as a potential paradigm to improve performance and energy efficiency of hardware for compute-intensive applications with intrinsic error tolerance. For a microprocessor, circuit designs of approximate adders and multipliers were proposed nearly two decades ago [3]. In the past decade, attention has been paid to designs of approximate multipliers (AMs) due to their high complexity and common utilization as basic arithmetic circuits [4]. Except for the ad hoc designs, algorithms have been proposed for automatically generating and synthesizing AMs, such as SALSA [5], CGP [6], SABER [7], and BLASYS [8].

Although many manually and automatically designed AMs have been evaluated using some statistical error features such as the error distance (ED) and mean relative ED (MRED), it remains a problem to determine the accuracy degradation when using them to substitute exact multipliers in some error-tolerant applications. Thus, selecting suitable AMs for these specific applications with a limited accuracy loss from a large AM library is difficult and time-consuming, as it may need a large amount of software and hardware simulation.

To address this challenge, methods for quickly evaluating the error effects of approximate arithmetic circuits on applications have been investigated by using learning-based methodologies [9], [10]. In [9], the probability mass function (PMF) of the inputs for each arithmetic operation in an application is used to obtain its weighted mean error distance (WMED). The WMEDs for all operations are then utilized as the input features of a learning-based model to predict the quality of the result. However, this model is not suitable for complex applications that require numerous operations such as neural networks (NNs), due to the difficulty in acquiring the PMF of the inputs for each operation. Moreover, the error model can be very complicated when the number of required operations is large; it can result in a long training time and a large degradation in the predicted accuracy. Using nine conventional statistical error features, [10] has presented several learning-based classifiers for a multi-layer perceptron (MLP) and a

This work was supported in part by the National Key Research and Development Science and Technology under Grant 2022YFB4500200, the National Natural Science Foundation of China under Grant 62104127, a project under Grant 2022-XXXX-ZD-005-00, and the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RES0048688.

*Hai Mo and Yong Wu contributed equally to this work.

Corresponding authors: Honglan Jiang; Leibo Liu.

H. Mo, Y. Wu, Z. Ma and L. Liu are with the School of Integrated Circuits, Tsinghua University, Beijing 100084, China. (e-mail: haihai.self@outlook.com, wuyong20@mails.tsinghua.edu.cn, mzn20@mails.tsinghua.edu.cn, liulb@tsinghua.edu.cn)

H. Jiang is with the Department of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai 200240, China. (e-mail: honglan@sjtu.edu.cn)

F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA. (e-mail: lombardi@ece.neu.edu)

J. Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. (e-mail: jhan8@ualberta.ca)

small convolutional neural network (CNN) on MNIST and SVHN datasets. The classifiers distinguish the AMs (a set of 600 designs that are also included in this work) that can result in an accuracy for image classification close to the accurate design. However, the considered networks and datasets in [10] are very simple; thus, the devised classifiers are hard to be adapted to more complex NN applications. In addition, the predicted accuracy of the classifiers is below 86%. The mathematical analysis in [11] shows that the AMs with low variance of relative error likely result in a low accuracy loss in AM-based CNN applications. However, only AMs with specific attributes are analyzed and verified in [11]; it is unclear whether the conclusion is applicable to arbitrary AMs. Hence, an effective framework is urgently needed to quickly and precisely select and design proper AMs for complex error-tolerant applications with a limited accuracy loss.

This paper evaluates the error characteristics of AMs with respect to complex NN applications; hence, AMs can be quickly selected (from a design library) and designed to satisfy different accuracy constraints in NN applications. First, the statistical error features of AMs are ranked as per their importance to the accuracy of NN applications. Based on several informative error features, error models are then established to predict the accuracy loss of an AM-based NN application. Since only three statistical error features of an AM are utilized, time-consuming software and hardware simulations are avoided for the applications. The major contributions of this paper are listed as below.

- A framework is proposed for constructing the error models of AMs to predict the accuracy loss of AM-based applications. This framework can be generally replanted to other error-tolerant applications and approximate circuits.
- An AM library with over 1,200 AMs (including 500 automatically generated AMs [12] and more than 700 manually designed AMs) is constructed. For each AM, its lookup table implementation, error measurements evaluated by 16 statistical error features, and accuracy loss when used on three common datasets are generated for error feature analysis and model construction.
- Three out of 16 statistical error features contributing most to the accuracy loss of the applications are obtained by using the Dropout Feature Ranking approach. This reduces the complexity of the error models and provides a guidance for the design of AMs.
- Several error models are established using ML methods for predicting the accuracy degradation of an AM-based NN application based on the statistical error features. These models can quickly select suitable AMs for NN applications with a true positive rate of over 96%.
- The selected statistical error features of AMs are further analyzed with respect to the accuracy of the AM-based NN applications and the hardware efficiency. These features provide a guidance for the design and generation of application-oriented AMs.

This paper is organized as follows. Section II introduces some preliminaries, including the generation of AMs, some commonly used statistical error features, and neural networks as the applications of AMs. Section III-B presents the overall framework for constructing error models, and introduces three feature selection approaches and four prediction model construction methods. The early experimental preparations are presented in Section IV. Section V gives the experimental results of classification and regression models. Section VI discusses the error and hardware characteristics of the AMs with respect to the accuracy of the NN applications. Conclusion is drawn in Section VII.

II. Preliminaries

In this section, we first introduce some AM design methodologies including manual and automated designs. Then, statistical error features used to evaluate the quality of AMs are presented and the relationship between them is also explored. For error-tolerant applications, several common NNs and datasets for image classification are studied; they are briefly introduced at the end of this section.

A. Approximate Multiplier Generation

In general, AMs are designed by modifying the original (exact) designs at the circuit and algorithm levels. At the circuit level, an AM is obtained by deliberately removing some logic gates from the accurate design [13], or simplifying the truth tables [14] and Karnaugh maps [15], [16], [17]. At the algorithm level, the binary logarithmic algorithm [18] approximately implements the multiplication by using simple adders [19], [20], [21]. Some automated processes and synthesis algorithms can automatically generate AMs, usually under a given error constraint [12], [8]. This method produces hundreds of AMs and a large amount of time is needed to evaluate their usability in specific applications when considering both the accuracy and hardware requirements. These AMs show deterministic output errors; in other words, their outputs are fixed for the same inputs. On the contrary, AMs based on voltage over-scaling [22] show uncertain probabilistic errors, which are not considered in this paper.

B. Statistical Error Features

The error characteristics of AMs depend on their approximation schemes. For each AM, the computing errors are usually given by several error metrics that statistically measure the differences between the approximate and accurate outputs. However, when an AM is applied to substitute the exact multiplier in an application, it is not clear how many and which error features can precisely indicate the accuracy loss in the application. Therefore, all commonly used statistical error features for evaluating AMs are studied in this paper.

The statistical error features of an AM are usually computed by using exhaustive simulations with all possible inputs, or Monte Carlo simulations with random inputs from a specific distribution, such as uniform and normal distributions. Let M and M' be the accurate and approximate results of the AM, respectively. The two basic metrics for computing the values of the statistical error features are the error calculated

TABLE I: Error features of approximate arithmetic circuits.

Feature	Description	Computation formula
μ_E	The mean of all possible Es.	$\sum_{i=1}^{N} p(E_i)E_i$
σ_E	The variance of the E.	$\sum_{i=1}^{N} p(E_i)(E_i-\mu_E)^2$
μ_{ED}	The mean value of EDs.	$\sum_{i=1}^{N} p(E_i)ED_i$
NMED	The μ_{ED} normalized by the maximum accurate result.	μ_{ED}/M_{max}
σ_{ED}	The variance of the ED.	$\sum_{i=1}^{N} p(E_i)(ED_i-\mu_{ED})^2$
μ_{RE}	The mean value of REs.	$\sum_{i=1}^{N} p(E_i)RE_i$
σ_{RE}	The variance of the RE.	$\sum_{i=1}^{N} p(E_i)(RE_i-\mu_{RE})^2$
μ_{RED}	The mean value of REDs.	$\sum_{i=1}^{N} p(E_i)RED_i$
σ_{RED}	The variance of RED.	$\sum_{i=1}^{N} p(E_i)(RED_i-\mu_{RED})^2$
rms_E	The root mean square of EDs.	$\sqrt{\sum_{i=1}^{N} p(E_i)ED_i^2}$
rms_{RE}	The root mean square of REDs.	$\sqrt{\sum_{i=1}^{N} p(E_i)RED_i^2}$
W_E	The maximum value of EDs.	$\max_{i\in[1,N]} ED_i$
W_{RE}	The maximum value of REDs.	$\max_{i\in[1,N]} RED_i$
ER	The probability of error.	$p(M_i' \neq M_i), i \in [1,N]$
E_{ss}	The errors are single-sided.	$E_{ss}=\begin{cases}1 & \forall i\in[1,N],\\ & E_i<0 \text{ or } E_i>0\\ 0 & \text{otherwise}\end{cases}$
E_{zo}	Errors may occur when the accurate result is zero.	$E_{zo}=\begin{cases}1 & \exists i\in[1,N],\\ & M_i'\neq 0 \text{ when } M_i=0\\ 0 & \text{otherwise}\end{cases}$

as $E = M' - M$ and the relative error to the accurate result, $RE = E/M$. The absolute values of E and RE are also widely used to show the error magnitude, denoted as ED and RED, respectively. When these metrics are evaluated on average, variance, mean squared deviation, normalization and maximization, we can obtain the error features shown in Table I. $p(E_i)$ is the probability of generating an error of E_i. N is the number of inputs used for error evaluation. Also, the probability of producing an error, ER, is a commonly used metric for evaluating how often the computing result is incorrect.

Moreover, we propose two Boolean metrics to consider the accuracy performance of an approximate circuit in a relatively complex application. As shown in Table I, E_{ss} denotes whether the errors in a design are single-sided, i.e., E_{ss} is 1 when all Es are negative or positive; otherwise, E_{ss} is 0. This metric is introduced because the frequent accumulation operations are sensitive to single-sided errors [4]. Thus, E_{ss} implies whether an approximate circuit can be used for accumulation. As the data in deep learning are usually sparse, many multiplication results are zero; thus, ensuring an accurate result for the zero output is of great significant relevance. Therefore, E_{zo} is considered in this paper to specify whether an error occurs when the accurate output is zero. E_{zo} is an effective metric to assess if an approximate circuit is appropriate for the processing of sparse data.

C. Neural Networks and Datasets for Image Classification

Over the past few decades, many NN architectures have been proposed to handle some basic tasks in machine learning, e.g., classification. In ISLVRC 2014, the VGG [23] prototype consisting of several convolutional layers improves the accuracy of image classification. To solve the issue of vanishing gradient, ResNet [24] introduces the concept of residual, which further expands the classification capability of deep CNNs. In ISLVRC 2015, ResNet won the competition and became a prevalent NN for image classification. These two architectures can be constructed with various numbers of layers, such as VGG-16, VGG-19, ResNet-18, ResNet-34, etc.

To train and test the classification capabilities of NNs, several classical datasets have been constructed. MNIST is a relatively simple dataset of handwritten numbers from 0 to 9 that contains 60,000 28 $\times$ 28 gray images for training and 10,000 images for testing. CIFAR-10 [25] is a more complex dataset consisting of ten types of different objects and each type has 6,000 32 $\times$ 32 GRB images, among which 5,000 images are for training. To further increase the complexity of datasets, CIFAR-100 [25] consists of 100 classes of images and each class contains 600 32 $\times$ 32 RGB images, i.e., 500 images for training and the remaining 100 for testing. Generally, the more complex the dataset is, the more complex an NN is needed to obtain an acceptable classification accuracy. A large NN results in a huge hardware overhead in computing, especially for multiplication. The utilization of AMs can obtain significant improvements in hardware efficiency.

III. An Error Model Construction Framework

In this section, we first introduce the overall framework for constructing the application-oriented error models. Then the implementation details including three types of feature selection strategies and four prediction model construction methods are presented.

A. Overall Framework

Fig. 1 shows the overall framework to quickly evaluate the error effects of AMs on NN applications. It consists of three stages, data generation, model construction, and model verification.

At the data generation stage, an AM library is built based on different approximation methodologies as discussed in Section II-A. Two files for each AM, .m file and .bin file, are generated. A .m file is used to produce the 16 statistical error features via Matlab. A .bin file stores the results of an AM in the form of look-up tables, which is used to substitute the exact multiplier in the inference of the pre-trained NNs, thereby obtaining the accuracy of the AM-based NNs on the aforementioned datasets in Section II-C [26].

After data generation, the obtained data are then divided into training and testing sets. Using the training set, the statistical error features of AMs are ranked as per their importance to the application accuracy loss. As the correlations between the AM error features and the accuracy of an AM-based application are complex, we propose to adapt the feature selection approaches in ML to learn these correlations, including the Filter, Wrapper, and Embedded methods. Several most informative error features out of the 16 candidates are then obtained to establish error models based on classical prediction models, e.g., MLP, support vector machine (SVM), decision tree (DT), among others. Thus, simple error models using a few AM error features can be obtained for predicting the accuracy loss of AM-based applications. The testing dataset is then used to evaluate the quality of the constructed error models, where the models with a lower complexity and higher prediction accuracy are preferred.

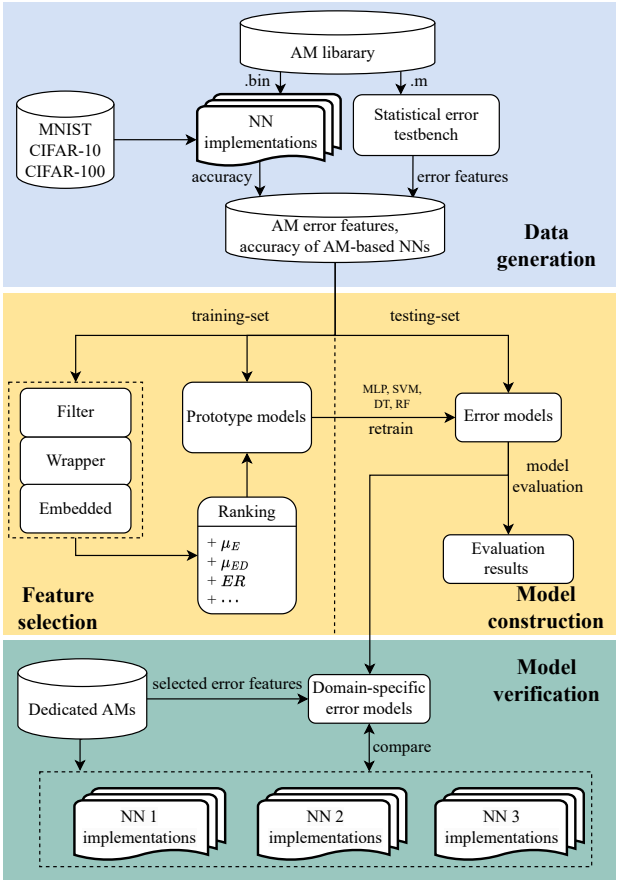


Fig. 1: The proposed framework for constructing the error models of AMs with respect to NN applications.

To further verify the versatility of the error models, a dataset for some dedicated AMs designed by using the methods different from those of the training and testing AM sets are generated. The selected statistical error features of the dedicate AMs are used in the error models as inputs to predict the accuracy loss of the NN applications. The prediction results are then compared with the actual results obtained by using these AMs in the considered applications.

When the error models are obtained from this framework, the effectiveness of newly designed AMs in NN applications can be predicted. As shown in Fig. 2, only the selected error features of newly generated AMs need to be calculated and used in the error models. The accuracy loss of the AM-based NN application predicted by the error models is then compared with the provided accuracy constraint, thus determining the effectiveness of the AM. This process can be efficient for AM selection and generation for specific NN applications such that time-consuming simulations for AM-based NN applications are avoided.

B. Feature Selection Strategies

The statistical error features of an AM are generally obtained from the difference between the approximate and accurate results, so they are highly correlated. However, the use of correlated features in the construction of a learning-based error model can lead to overfitting, which impacts the prediction

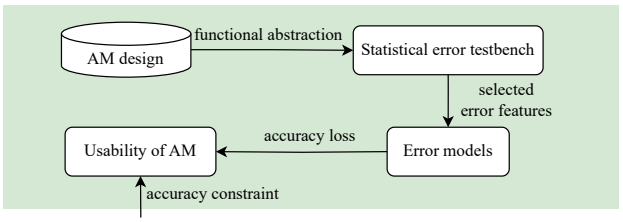


Fig. 2: The utilization of error models for predicting the effectiveness of new AMs in NN applications.

accuracy of the model. Moreover, when considering too many features, it may result in a complex data collection process. To determine those that have the most impact on the application accuracy, we need to select some informative error features in terms of their importance to the application accuracy.

In this paper, we propose to adapt feature ranking and selection methodologies in ML for evaluating the importance of the AM error features to the accuracy of an AM-based application. To select proper input features for the model of an ML application, three categories of feature ranking and selection approaches have been developed, i.e., Filter, Wrapper and Embedded [27].

1) Filter Method: In this method, the features are selected without considering model construction. The amount of information for each feature is calculated. The more informative features are then used to train the model. Various criteria have been developed to evaluate the information carried by features. Considering the efficiency and suitability, the variance selection (σ^2) [28], Chi-square verification (χ^2) [29] and max relevance and min redundancy (mRMR) [30] methods are considered in this paper.

For the variance selection, the feature information is assessed by the variance that is given by [28]

$$\sigma_i^2 = \sum_{j=1}^{n} (x_{ij} - E_i)^2, \tag{1}$$

where x_{ij} denotes the value of the ith feature in sample j and E_i is the average value of all n samples. A larger σ_i^2 indicates that the distribution of the ith feature is more scattered; thus, containing more information. However, using variance for feature selection is not always effective; when the distributions of the features are concentrated, less information can be expressed by the variance. In addition, it is not accurate to use variance to measure the amount of information when the distributions of features are different.

χ^2 is proposed to verify the independence between two events, so it describes the difference between a hypothetical value and an actual measured value [29]. It is calculated as

$$\chi_i^2 = \sum_{j=1}^{n} \frac{(x_{ij} - m_{ij})^2}{m_{ij}} = \sum_{j=1}^{n} \frac{(x_{ij} - E_i)^2}{E_i}, \tag{2}$$

where m_{ij} and x_{ij} denote the hypothetical and measured values respectively. The mean E_i is usually used for m_{ij}. Once χ_i^2 is achieved, the corresponding p-value is obtained by looking up the table. A smaller p-value indicates a higher independence of the two events. Although χ^2 provides a way to solve the problem of inconsistent distribution in variance

selection by using a division, it only describes the amount of information that a feature carries, but the relationship between features and application results are not revealed.

The mRMR method selects the features that have minimum-redundant information and maximum-relevance to the considered target result [30]. As a common method to measure the correlation between two variables, the mutual information (MI) [31] is used to calculate the correlations in the mRMR method. MI describes the degree of dependence between two features. Let $k-1$ features be chosen from the feature set X, represented by S_{k-1}; then, the next step is to choose the kth feature from the remaining features in the subset of $X-S_{k-1}$ by using

$$\max_{\mathrm{x}_j \in \mathrm{X}-S_{k-1}} \left[I\left(\mathrm{x}_j; c\right) - \frac{1}{k-1} \sum_{\mathrm{x}_i \in S_{k-1}} I\left(\mathrm{x}_i; \mathrm{x}_j\right) \right], \quad (3)$$

where x_i and x_j are the sample values of the ith and jth features in the subset S_{k-1} and $X-S_{k-1}$ respectively, and c is the target class for each sample. $I(\mathrm{x};\mathrm{y})$ is the MI between feature x and feature y. When the features are discrete values, MI is calculated as

$$I(\mathrm{x}_i;\mathrm{y}) = \sum_{x_{ij} \in \mathrm{x}_i} \sum_{y_j \in \mathrm{y}} p(x_{ij}) \log \frac{p(x_{ij}, y_j)}{p(x_{ij}) p(y_j)}, \quad (4)$$

where $p\left(x_{ij}\right)$ and $p\left(y_j\right)$ are the probabilities of x_{ij} and y_j, respectively. $p\left(x_{ij}, y_j\right)$ is the joint probability of x_{ij} and y_j. The mRMR method comprehensively considers the correlation between features as well as features and targets; however, it only evaluates the correlation between two features, which does not show the influence of multiple features on the targets.

2) Wrapper Method: In the Wrapper method, a subset of features is continuously selected from the feature set in the training of a specific model. The feature subset is then evaluated on the test dataset; the subset returning the best test result is finally selected. As an efficient Wrapper method, the Las Vegas method (LVM) [32] is discussed next.

LVM can be described by Algorithm 1, where E_v and E_v' are the evaluation results of the optimal and current iterations, and d and d' are the numbers of features in the current optimal (X^*) and randomly selected (X') feature subsets, respectively. At each iteration, a subset of features is randomly selected for training the model, and the evaluation results of the trained model on the test dataset are saved as E_v'. If E_v' is smaller than the optimal result E_v, or if $E_v' = E_v$ and $x' < d$, E_v and X^* are updated. Otherwise, this process repeats until the iteration number reaches T. Δ is a prediction model such as SVM [33], MLP, DT, or random forest (RF) [34], among others. Although the Wrapper method directly finds the optimal subset of features for the models, training of the model is time-consuming and likely to find a local optimal subset.

3) Embedded Method: In the embedded method, the input features are selected along with the model training. In this paper, we propose to use a relatively advanced approach, the dropout feature ranking (DFR) [35], for the ranking and selection of the AM error features. DFR is derived from the

Algorithm 1 LVM Algorithm

Inputs: dataset D; feature set X; model Δ; parameter T;
Outputs: subset feature X^*

$E_v = \infty$, $d = |X|$, $X^* = X$, $t = 0$
while $t < T$ **do**
  Randomly generate a subset of features X'
  $d' = |X'|$
  $E_v' = -CrossValidation(\Delta(D^{X'}))$
  if $(E_v' < E_v) \vee ((E_v' = E_v) \wedge (d' < d))$ **then**
    $t = 0$, $E_v = E_v'$, $d = d'$, $X^* = X'$
  else
    $t = t + 1$
  end if
end while

variational dropout [36] mechanism in NNs and can be regarded as a reinforcement learning technique. Specifically, the dropout is performed by adding masks to neurons in the hidden layers. The mask is then sampled from a Bernoulli sequence consisting of 0s and 1s; so, the neurons are probabilistically removed by setting a node to its original value or 0. To enhance the dropout operation, a variational dropout is explored by introducing the dropout rate θ that can be optimized via a loss objective function. In DFR, the following loss objective function is utilized when training the model M with a mini-batch size of B_N and F_N features.

$$L(\boldsymbol{\theta}) = -\frac{1}{B_N} \sum_{i=1}^{B_N} log(p_M(\mathbf{y_i}|f(\mathbf{x_i}, \mathbf{z_i}))) + \frac{\lambda}{B_N} \sum_{i=1}^{B_N} \sum_{j=1}^{F_N} z_{ij} \quad (5)$$

where $\mathbf{x_i}$ is the input vector containing the feature values, $\mathbf{y_i}$ is the target output, and $log(p_M(\mathbf{y_i}|f(\mathbf{x_i}, \mathbf{z_i})))$ is the corresponding loss. $\boldsymbol{\theta}$ represents the dropout rate vector for the F_N features, i.e., $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_{F_N})$. λ is a constant that depends on cross validation. Rather than obtained from a discrete Bernoulli distribution, $\mathbf{z_i}$ is relaxed by sampling from a Concrete distribution, i.e., $\mathbf{z_i}$ is given by

$$\mathbf{z_i} = \text{sigmoid}(\frac{1}{t}(log\theta) - log(1-\theta) + log(u) - log(1-u)) \quad (6)$$

where $\mathbf{u}$ is a vector containing F_N random numbers within [0,1] following a uniform distribution, i.e., $u_j \sim \mathrm{U}(0,1)$, $j = 1, \cdots, F_N$. t is an empirical coefficient that is fixed to 0.1.

The minimum loss can be reached when smaller dropout rates are achieved for more informative features; thus, the importance of the features can be ranked by comparing the values of their dropout rates. As DFR not only learns the contributions of the input features to the target task, but also considers the correlations among features, it is well suited for the aim of this paper. Thus, DFR is mainly used to select the most informative AM error features as the inputs of the error models. In addition, the other feature selection methods are tested for comparison.

C. Prediction Models

To map the statistical error features of AMs to the accuracy loss of the applications, several classical prediction models are utilized to construct error models. Specifically, we build classification and regression models to predict the accuracy loss of

the AM-based NN applications. In this paper, four prediction models, the SVM, MLP, DT and RF, are investigated.

1) SVM Model: SVM aims at mapping the input vector $\mathbf{x} = \{x_i, i = 1, 2, \cdots, l\}$ into a higher dimensional space $\mathbf{Z}$ [37]. A linear hyperplane can be constructed in $\mathbf{Z}$ space to classify $\mathbf{Z}$ into two classes. The linear hyperplane is determined by maximizing the margin between the two classes of vectors to be classified; the vectors with the minimum distance to the hyperplane are referred to as support vectors. For the multi-classification problem, using one-versus-one method, classifiers for each two classes are constructed, and the final decision is made by the voting results of the two-class classifiers. Different from the classification tasks, the input samples for regression tasks are constrained to both sides of the hyperplane as much as possible to obtain the smallest mean squared error.

2) MLP Model: MLP consists of an input layer, the hidden layers and an output layer. In this work, the input layer takes the AM error features as inputs; the hidden layers are fully connected, and the activation function is ReLu; the output layer is determined by the target task. In the classification task, the output is processed using softmax, such that the activation of each neuron represents the probability of the input sample that belongs to a class. In the regression task, sigmoid is used as the activation function to predict the application accuracy using the AM. The regression task is more complex than the classification task and requires more hidden layers.

3) DT Model: DT describes the classification of a sample using a tree structure. A DT has one root with no incoming edges and certain leaves without outstretched branches. All nodes between the root and leaves are internal nodes, which partition the temporary space into at least one subspace as per one or more features. The root or an internal node often represents a feature, while a leave indicates a class. A sample can be classified as one class from the root to the leaves as per its features [38]. Commonly used algorithms for DT include ID3, C4.5 and CART [39]. The CART specializes in complex and unbalanced data, and has a pruning method to lower the model complexity. Hence, CART is utilized in this work. This algorithm is based on the Gini index (GI) criteria; however, DT can only classify discrete features. For continuous features, discretization such as the dichotomy method can be used.

4) RF Model: RF is a special case of the integrated learning model. Using DT as a base learner, RF decides the final classification of a sample by the voting result of base learners. Different from DT, a base learner in RF is trained by using k samples from the entire training dataset, and different training datasets are utilized for different base learners. The diversity of the base learners enhances the generalization of RF.

IV. Model Construction Preparations

This section prepares for the model construction. First, to measure the quality of the constructed error models, some model evaluation metrics are shown. Second, an explicit AM library is built to generate data for model construction. The required data including the AM error features and the accuracy results of the AM-based applications are then obtained via simulations. Finally, the AM error features are ranked according to their importance to the application accuracy using the training dataset. Three most informative error features are selected by using each selection method, for constructing efficient error models.

A. Model Evaluation Metrics

For classification models, a common evaluation metric is the accuracy. In the 3-class classification cases, the accuracy rate is calculated by top-1 and top-2 criteria; they are given by

$$\text{top-1} = \frac{\sum_{i=1}^{n}(\hat{y_i}^1 = y_i)}{n}, \tag{7}$$

and

$$\text{top-2} = \frac{\sum_{i=1}^{n}(\hat{y_i}^1 = y_i) \vee (\hat{y_i}^2 = y_i)}{n}, \tag{8}$$

where n is the number of samples, y_i is the class of the ith sample, and $\hat{y_i}^1$ and $\hat{y_i}^2$ denote the predicted classes with the highest and second highest probabilities, respectively.

In this work, the goal of the model is to find the AMs that result in a reasonable accuracy loss in the NN task, i.e., the first class of AMs leading to a loss less than a certain threshold. Therefore, it is important for the first class of samples to be accurately identified and predicted by the model; thus, recall-1 is calculated to show the recall rate of the first class. Also, the average recall rates of the three classes are reported, referred to as macro-tpr. They are calculated as

$$\text{recall-1} = \frac{TP_1}{(TP_1 + FN_1)}, \tag{9}$$

and

$$\text{macro-tpr} = \frac{1}{3}\sum_{i=1}^{3} TPR_i, \tag{10}$$

where recall-i is also known as the true positive rate (TPR) of the i-th class. TP_i denotes the number of samples that are predicted as the i-th class and truly belonging to the i-th class, while FN_i denotes the number of samples that are predicted as not in the i-th class but belonging to the i-th class.

For the regression model, the gap between the actual result of the AM-based application and the predicted result (obtained by the model) must be evaluated. In this paper, the mean absolute percentage error (MAPE) is used to show the deviation between the predicted result and the actual result. It is given by

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - \hat{p_i}|}{p_i} \times 100\%, \tag{11}$$

where $\hat{p_i}$ and p_i are the predicted and actual results of the application using the ith AM. In addition, the coefficient of determination (R^2) is reported to show the causal relationship between the predicted result and the features. It is given by

$$\text{R}^2 = \frac{\sum_{i=1}^{n}(\hat{p_i} - \bar{p})^2}{\sum_{i=1}^{n}(p_i - \bar{p})^2}, \tag{12}$$

where $\bar{p}$ is the mean of all p_i.

B. Data Generation

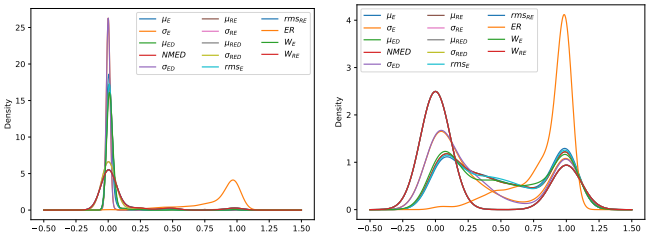
1) AM library: By using an advanced quantization scheme, the weights and activations of many NN models can be quantized as 8-bit integers with only a marginal loss in accuracy. The quantization enables efficient hardware implementations with simple arithmetic circuits [40]. Thus, 8×8 unsigned multipliers are considered in this work. To study the effects of AM error features on AM-based NN applications, an 8×8 unsigned AM library is built. In this work, we analyze only ASIC implementations for AM, other design technologies such as emerging technologies and FPGA are not considered because their underlying hardware is different. Three distinct methodologies are utilized for generating the AM library; 1237 AMs are involved in model construction. The open-source 500 AMs (automatically generated by using CGP) referred to as mul1 to mul500 [12] are included. In addition, two AM sets are further generated based on an accurate Wallace tree multiplier.

- In the first set, several full adders (FAs) in the Wallace tree of an accurate multiplier are removed and the corresponding sums are replaced by constant 0s or 1s, resulting in 352 AMs. Different AMs are obtained by varying the location and the number of the removed FAs. These AMs are denoted as DAM_s_j_k_c, where s and j represent the stage and starting column of the removed FAs; k and c denote the number of removed FAs and the constant substituting the sums of the removed FAs, respectively.
- The second set contains 385 AMs generated by replacing some FAs by approximate FAs (AFAs). Three AFAs are respectively utilized. Let a and b be the two input bits, C_{in} be the carry input, and Sum and $Cout$ be the sum and carry outputs, respectively. The first AFA is implemented as $Sum = a+b+C_{in}$ and $Cout = 0$. To compensate for the errors, $Cout$ of the most significant AFA is generated by using an AND gate, i.e., $Cout = ab$. The second AFA is given by $Sum = a \oplus b \oplus C_{in}$ and $Cout = \overline{Sum}$. The third AFA is implemented as $Sum = a \oplus b \oplus C_{in}$ and $Cout = a$. These AMs are denoted as RAM_i_s_j_k, where i indicates that the ith AFA is used, s and j are the stage and starting column of the replaced FAs; k denotes the number of AFAs used in the AM.

2) Data processing: For each AM, two types of files are generated, the .m file for calculating the AM statistical errors and the .bin file for obtaining the accuracy of the AM-based NN applications. To compute the statistical errors, the exhaustive inputs in the range of $[0, 255]$ are tested for each AM. The product from each AM is stored in a look-up table (.bin) to substitute the accurate product in the considered NNs [26]. Specifically, the pre-trained VGG-16, ResNet-18, and ResNet-34 are implemented by replacing the accurate 8×8 unsigned multipliers by the AMs and then tested on the MNIST, CIFAR-10, and CIFAR-100 datasets. Note that the cross entropy is utilized as the loss function in the training of NN models; the optimization algorithm is Adam with hyperparameters set as the default values in [41]. The considered combinations of NNs and datasets used in this work are listed in Table II. The classification accuracy is then reported for these AM-based NN applications.

TABLE II: The combinations of NNs and datasets.

NN	MNIST	CIFAR-10	CIFAR-100
VGG-16	√	√	×
ResNet-18	√	√	×
ResNet-34	√	√	√



(a) normalization without pre-processing

(b) normalization after pre-processing

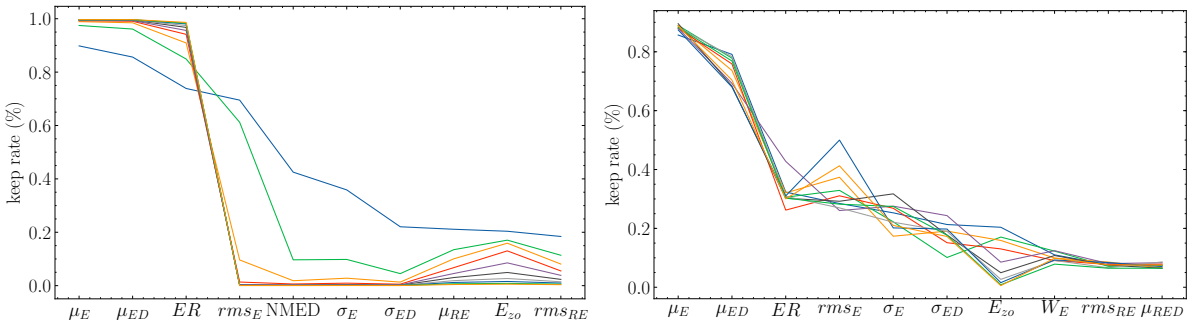
Fig. 3: Distribution density of the normalized error features with and without pre-processing.

Consider the case in which a different application accuracy may be required; the AMs are divided into three categories as per their accuracy loss in the NN applications. In this work, two accuracy thresholds are set to classify the AMs; the AM resulting in an accuracy loss lower than or equal to 3% is labeled as "class 0"; it is labeled as "class 2" for a loss higher than 8% and "class 1" for a loss between the two thresholds. The reason for choosing 3% and 8% as the thresholds is that the numbers of AMs change drastically at these values. To generate and evaluate the error models, the AMs are randomly divided into training and testing sets accounting for 75% and 25% of the entire library, respectively. In addition, for a better data fitting, the dataset is pre-processed and normalized.

Since the AMs are randomly generated by using various methods, some AMs produce very large errors. Thus, the direct normalization of the error features without pre-processing would likely lead to a loss of valuable information. The distribution of the error features after using the maximum-minimum normalization is shown in Fig. 3. Without pre-processing, 90% of the data are normalized to a value near 0, resulting in a large information loss, as shown in Fig. 3(a). To preserve useful information, we have pre-processed the error features by clipping them at 73%, i.e., setting values larger than the 73% quantile to 73%. Fig. 3(b) shows the probability density of the normalized error features after pre-processing. Compared to the results in Fig. 3(a), they are more uniformly distributed in the interval [0, 1].

C. Selected AM Error Features

Based on the training dataset obtained in IV-B, the ranking of the 16 statistical error features obtained by the DFR method is shown in Fig. 4; the importance of features is sorted as per their average keep rates (1-$p_{dropout}$) from 10 trials. For both the classification and regression models, the most important 3 features are μ_E, μ_{ED}, and ER. The keep rates of the first 3 and 2 features are significantly higher than those of the other features for classification and regression tasks, respectively.



(a) classification model (b) regression model

Fig. 4: Keep rates ($1 - p_{dropout}$) of the error features from 10 DFR trials. The error features with relatively low keep rates are not shown in the figures.

TABLE III: Selection of the statistical error features.

Method		classification features	regression features
Filter	σ^2	σ_{RE}, σ_{RED}, μ_{RE}	
	χ^2	μ_{ED}, NMED, μ_E	
	M_c	W_{RE}, W_E, μ_E	
	M_d	E_{ss}, ER, E_{zo}	
	M_q	E_{zo}, E_{ss}, W_{RE}	
Wrapper	l_{dt}	rms_{RE}, W_{RE}, μ_{ED}	μ_{ED}, rms_{RE}, μ_E
	L_{rf}	σ_E, σ_{RE}, μ_E	μ_E, σ_E, μ_{RED}
	L_{svm}	ER, μ_E, σ_{ED}	W_E, ER, μ_{RED}
	L_{mlp}	σ_{RED}, ER, rms_E	W_E, ER, μ_{RED}
Embedded	DFR	μ_E, μ_{ED}, ER	

For comparison, the ranking and selection results using the other methods are also shown in Table III. M_c, M_d and M_q are the mRMR method for continuous sample values, discretized samples with equal distance and quantity, respectively. L_{mlp}, L_{dt}, L_{svm} and L_{rf} denote the LVM method using MLP, DT, SVM and RF models, respectively. For a fair comparison, the first three features are selected for the ranking. Note that the LVM method generates two different optimal feature subsets for classification and regression models. In this paper, the number of features in each subset is constrained to three for the LVM methods.

As shown in Table III, different feature selection methods result in rather diverse feature subsets; however, some error features appear more frequently than the others in the obtained 14 subsets. Specifically, the most frequently occurred three error features are ER, μ_E, μ_{ED}; they are the same as the features obtained by using DFR. Thus, this confirms that DFR is a very effective method for selecting the statistical error features to show the impacts on the accuracy of AM-based applications. The effectiveness of all considered feature selection approaches is then tested by using the respective selected features in Table III, where the considered NN and dataset are VGG-16 and CIFAR-10, respectively.

Both classification and regression prototype models are established based on SVM, MLP, DT and RF for each feature ranking method, as shown in Fig. 5. For DT and RF, the classification and regression models with the features selected by DFR show the highest accuracy, while the ones using the features selected by M_d and M_q perform the worst. For the MLP- and SVM-based models (except for the SVM-based regression models), DFR is the second best, leading to classification and prediction accuracy very close to the best selection method. Overall, DFR performs better than the other feature selection approaches for both classification and regression tasks. Thus, the AM error features selected by using DFR method are utilized to construct the error models next.

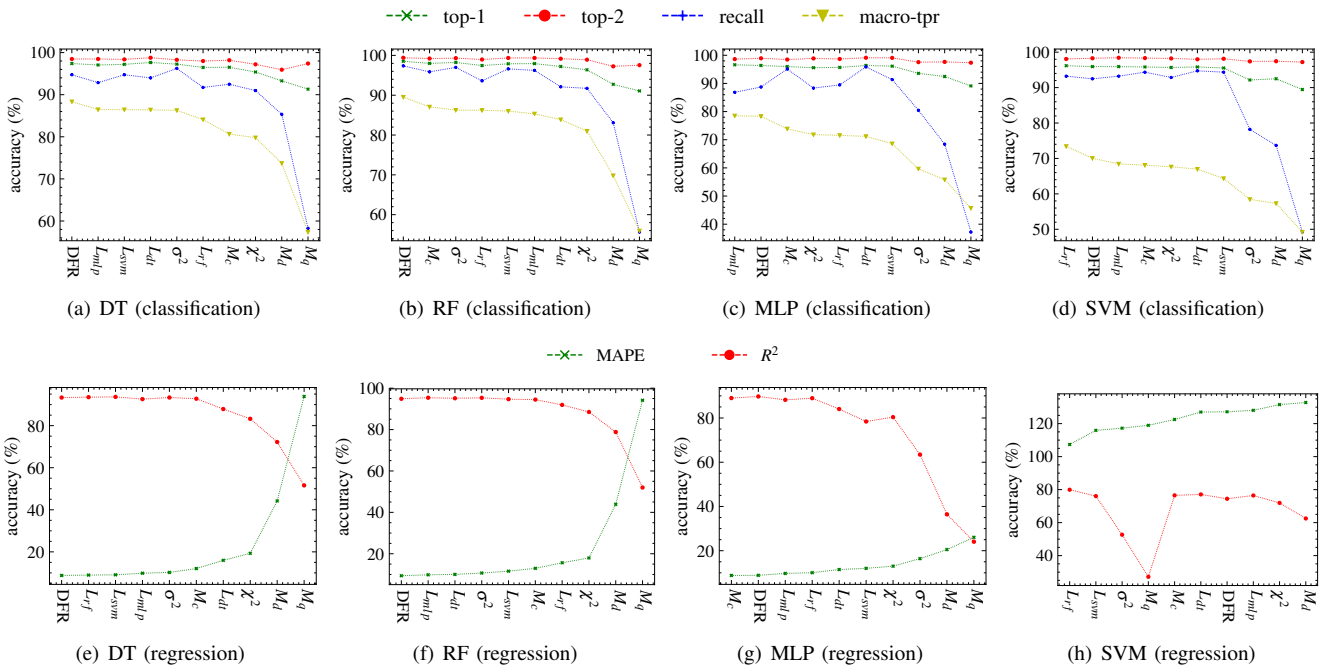
V. Model Construction and Evaluation

Using the selected AM error features as the inputs, the classification and regression models are constructed to predict the accuracy of an AM-based application, by training the prototype models. The output labels in the training dataset are generated based on the accuracy of the NN applications. The models are assessed by reporting their evaluation metrics with the testing dataset. Finally, a dedicated AM set is used to verify the versatility of the models.

A. Classification Models

Three-class classification models based on DT, RF, MLP, and SVM are constructed with two accuracy loss thresholds, 3% and 8%. The specific models for a particular NN and dataset denoted as "NN-dataset" (e.g., vgg16-mnist denotes the model designed for the VGG-16 running on MNIST). Also, a more general error model for all the considered NNs and datasets are established, denoted as "domain". To further verify the effectiveness of DFR, different numbers of error features are utilized to build classification models as per the feature ranking results of DFR.

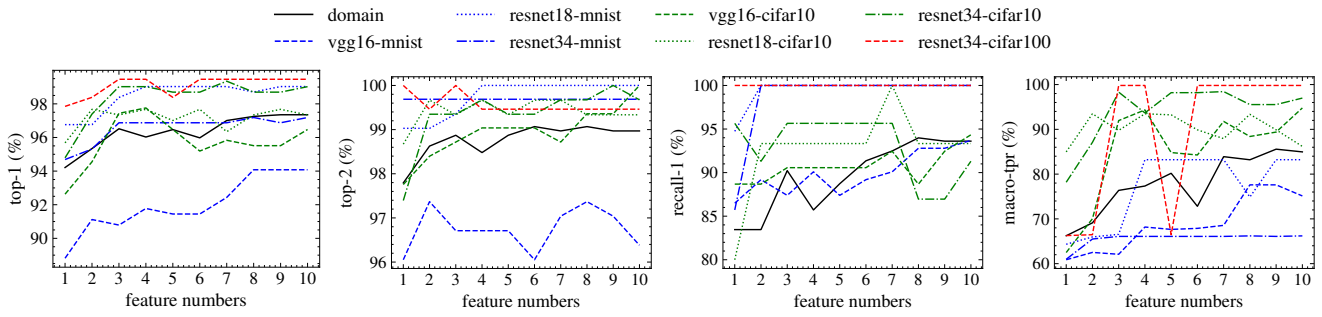
Consider MLP as an example, the NN of the specific classification model is given by x-128-256-128-3, where x most informative error features are utilized as the inputs. As shown in Fig. 6, the accuracy of most classification models increases sharply until three features are utilized; this is consistent with the ranking of error features. When the number of utilized features reaches three, the classification accuracy stabilizes at a high level. The accuracy of the model may deteriorate if less important features are added to the inputs. By using three error features, the specific error models (except for vgg16-mnist) result in a 96%-99% top-1 accuracy, 98%-100% top-2 accuracy, and 88%-100% recall-1. This indicates that specific error models reach about 96% probability of correctly selecting an AM within a 3% accuracy loss in the considered applications, and higher than 98% for the accuracy loss within



(a) DT (classification) (b) RF (classification) (c) MLP (classification) (d) SVM (classification)

(e) DT (regression) (f) RF (regression) (g) MLP (regression) (h) SVM (regression)

Fig. 5: Accuracy of prototype models using the selected features by different feature selection methods.



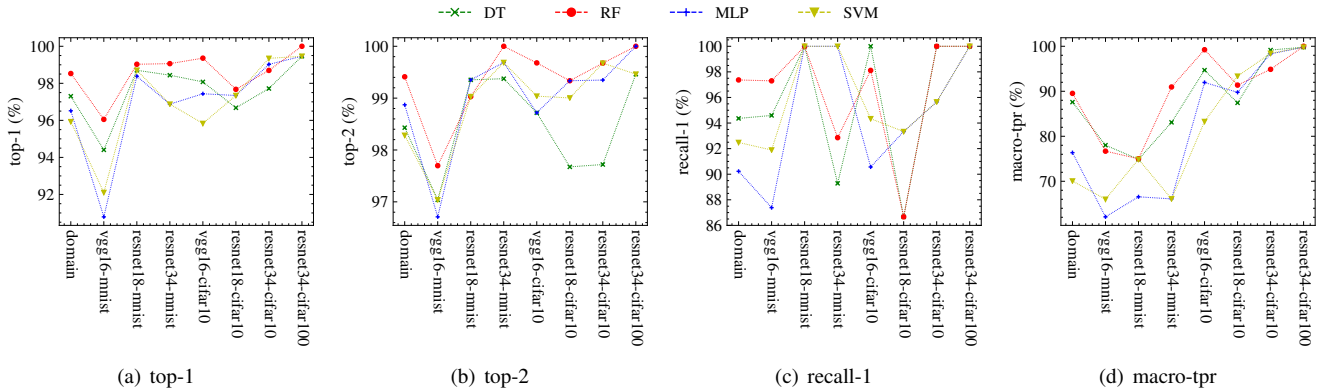
(a) top-1 (b) top-2 (c) recall-1 (d) macro-tpr

Fig. 6: The accuracy of the MLP-based classification models with different number of input error features.

8%. vgg16-mnist shows a relatively low accuracy because it can tolerate more errors due to the simple NN and dataset; therefore, more features or larger prototype models should be used to learn the relationship between the error characteristics of AMs and the accuracy of this application. The values of macro-tpr are much lower than those of recall-1 due to the low classification accuracy of the second class TPR_2 (see (10)). This occurs because the number of samples belonging to "class 1" is significantly smaller than those of "class 0" and "class 2", resulting in a very low TP_2 and hence TPR_2.

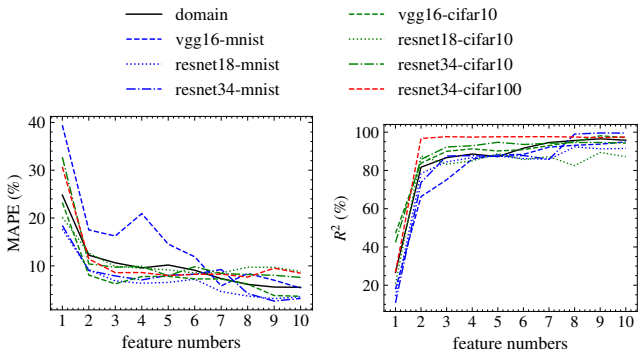
To increase the generality, domain-specific models dealing with all the considered NNs and datasets are constructed. To this end, three extra features are added to the specific error models; they are the NN type, the dataset type, and the cross feature between NN and dataset. Thus, the form of the utilized MLP is (x+3)-128-256-128-3. As shown in Fig. 6, the domain-specific error models achieves 96% top-1 accuracy, 98.8% top-2 accuracy, and about 93% recall-1. This may benefit from the larger dataset, i.e., the dataset for the domain-specific models is 7× as large as that for a specific model. Overall, the accuracy of the domain-specific models tends to be higher after the utilized number of error features reaches three. To summarize, DFR effectively ranks the importance of error features, and the top three ranked features μ_E, μ_{ED}, ER are more important than the other features.

Finally, the classification models using DT, RF, SVM and MLP are established respectively with the three most informative error features obtained by DFR; the evaluation results are shown in Fig. 7. Among the four models, the RF-based model attains the best classification accuracy with higher than 96% top-1 and higher than 86.2% recall-1 for all applications. However, the RF-based models are the most complex: an RF model is made of 80-380 DTs, and each DT has 380-970 nodes. Thus, the connection complexity of the RF models is at a level of 10^6. The DT-based models are simpler with a complexity at a level of 10^3 connections, while the accuracy rate is only 2% lower than that for RF-based models. For the MLP-based models, the accuracy is about 5% lower than that



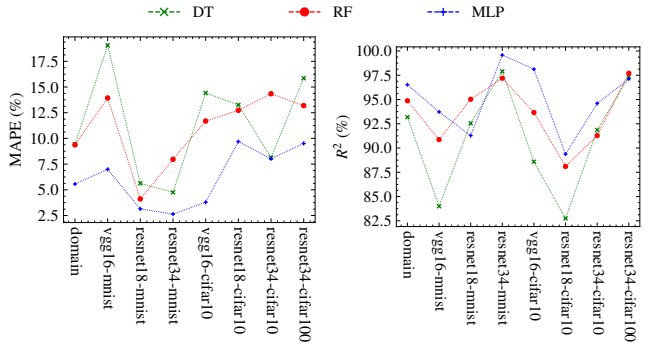
(a) top-1 (b) top-2 (c) recall-1 (d) macro-tpr

Fig. 7: The accuracy of classification models with 3 error features.



(a) $MAPE$ (b) R^2

Fig. 8: The $MAPE$ and R^2 of the MLP regression models.



(a) $MAPE$ (b) R^2

Fig. 9: Regression error models with three input features.

of the RF-based models, with a complexity of 10^5 connections. Thus, according to these metrics, DT is the most effective model for the AM classification in terms of the accuracy loss of the AM-based NN applications.

B. Regression Models

Regression models are proposed to predict the accuracy of image classification using AMs. A specific error model with a structure of x-128-256-521-1024-128-1 MLP is utilized for the regression models; more complex models are used to deal with the regression task with a higher complexity. Fig. 8 shows the $MAPE$ and R^2 of the MLP-based regression models using different numbers of error features. Like the classification models, a major improvement occurs in $MAPE$ and R^2 when the number of utilized features increases to 2. This is consistent with the feature ranking result in which the first 2 features have significantly larger keep rates than others (see Fig. 4(b)). The accuracies of the regression models for resnet18-cifar10, resnet34-cifar10, and resnet34-cifar100 tend to be stable after the number of error features reaches three.

Fig. 9 shows the $MAPE$ and R^2 of the regression models based on DT, RF, and MLP using the three error features selected by DFR. Compared with the RF-based models, the MLP-based models have a higher prediction accuracy and a lower complexity. The DT-based model consists of 1030-7300 nodes, and the RF-based models consist of 20-170 DTs. The connection complexity of the RF-based models is at a level of 10^7, larger than that of the MLP-based models (10^6). The DT-based models show the lowest complexity, yet resulting in relatively high $MAPEs$ and low R^2. Overall, among the regression models, MLP achieves the best prediction results for all applications. In addition, the NN of the MLP model can be flexibly adjusted according to the accuracy requirements; a balance between model complexity and prediction accuracy is then achieved.

C. Model Verification

To verify the versatility of the obtained error models, 74 dedicated AMs proposed in the technical literature that are comprehensively surveyed in [4] are tested. These AMs are manually designed by using different approximation techniques that are not involved in the training and testing sets for constructing models. Note that these AMs cover most typical approximation techniques, including the dynamic segmentation [42], the logarithmic approximation [19], [20], and using approximate 4:2 compressors for the partial product accumulation [43], [44], among others. As a small dataset is not sufficient for a reasonable model evaluation of specific models, only the domain-specific models with 518 input samples are tested. As per the simulation results, the models with three error features provide the best tradeoff between the number of utilized features and accuracy. Thus, the classification and regression models with three input features are evaluated. The

TABLE IV: The verification of domain-specific models.

models	Classification			Regression	
	Top-1	Top-2	Recall-1	$MAPE$	R^2
MLP	81.67%	95.75%	86.48%	**5.73%**	75.3%
RF	**98.88%**	99.44%	**86.20%**	7.56%	77.9%
DT	96.28%	97.40%	79.32%	7.41%	72.8%
SVM	97.03%	99.44%	72.41%	125.04%	66.2%

results are shown in Table IV; the macro-tpr is not included due to the small number of samples in "class 1." Consistent with expectations, the best classification results are obtained by RF, resulting in 98.88% top-1 and 86.2% recall-1, whereas the MLP model shows the best prediction results with a $MAPE$ of 5.73%. These results indicate that the proposed error models can effectively predict the error effects of AMs on the quality of AM-based image classification tasks. Moreover, this verification experiment demonstrates that the obtained models are useful to assess arbitrary AMs with various error features and thus they support the evolution of AM design.

D. Comparison with Related Works

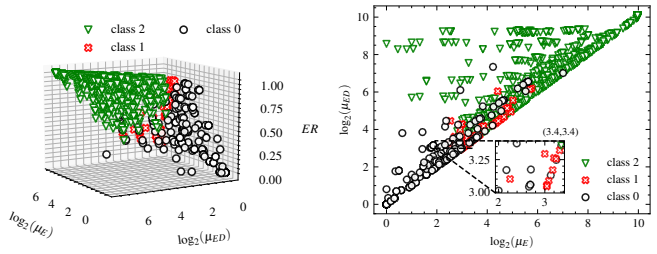
The proposed framework exhibits significant advantages over state-of-the-art learning-based methodologies for evaluating the error effects of approximate arithmetic circuits on applications [9], [10]. Compared to [9], specifically, this framework is capable of dealing with more complex applications, e.g., ResNet34 on CIFAR-100. Several simple NN applications are studied in [10], but only two-class classifiers are constructed for 600 AMs with an 86% accuracy, whereas our framework is validated in three-class classifiers and regression tasks for more than 1200 AMs with much higher accuracy. Moreover, this framework is applicable to arbitrary AMs rather than AMs with specific attributes [11].

VI. Further Discussion

This section analyzes the error characteristics of the AMs in terms of the three selected error features and their effects on the accuracy of the AM-based NN applications. The VGG-16 on CIFAR-10 with a moderate complexity is considered as an example. The hardware improvements of AMs that result in acceptable NN implementations are discussed.

A. Error Analysis for AM Designs

Fig. 10 shows the classification results of the AMs with respect to their error features of μ_E, μ_{ED} and ER. Without retraining, "class 0" denotes the AMs resulting in less than 3% accuracy loss in vgg16-cifar10, "class 2" refers to those AMs that show an accuracy loss greater than 8%; otherwise, the AMs are marked as "class 1." Fig. 10(a) shows that an AM performs well in the application of vgg16-cifar10 when its ER is smaller than 30%, i.e., the AMs with low values for the z-axis belong to "class 0." Similarly, Fig. 10(b) indicates that, with a μ_{ED} smaller than 8, the AM must belong to "class 0." Considering the μ_E shown in the x-axis, all AMs of "class 0" and "class 1" resulting in less than 8% accuracy loss in vgg16-cifar10 have small μ_E values; however, a small μ_E does not guarantee a small loss in application accuracy. This means that a small μ_E is necessary but not sufficient to indicate an



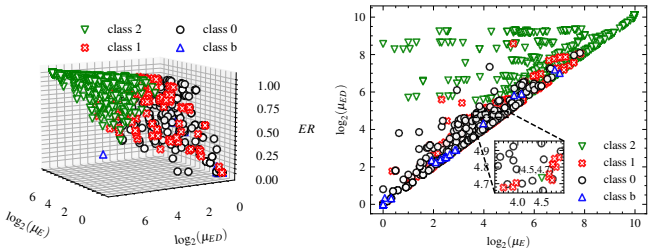
(a) classification results vs. μ_E, μ_{ED} and ER (b) classification results vs. μ_E and μ_{ED}

Fig. 10: Classification results of AMs in pre-trained vgg16-cifar10 vs. the three most informative error features.

appropriate AM for the application of image classification. As the ER of an AM is generally high, it is better to limit μ_{ED} to a reasonable range when designing AMs.

To sum up, the accuracy of an AM-based NN application is closely related to the μ_E, μ_{ED}, and ER of the AM. The specific correlations between these error features and the accuracy of AM-based NN applications are analyzed as follows. As accumulation dominates the computations in NN, the output error of a neuron is highly correlated with the mean error μ_E of the AM, especially for the case with a large number of inputs. Thus, μ_E is a key factor determining the accuracy of AM-based NN applications. However, a smaller μ_E does not guarantee a more accurate accumulation when the number of inputs is not large enough. In this case, a small μ_{ED} is necessary for achieving a relatively accurate accumulation. In addition, the AM with a very small ER (e.g., smaller than 30%) often results in small errors in accumulation operations and hence a low accuracy loss in AM-based NN applications.

Note that the image classification results shown in Fig. 10 are obtained by directly replacing the accurate multipliers in the inference of a pre-trained VGG-16 with the corresponding AMs. Taking advantages of the learning ability, retraining after the replacement would adjust the weights to compensate some errors due to AMs, which can improve the accuracy of AM-based NN applications. Thus, retraining is attempted next for the AM-based vgg16-cifar10. Note that only the inference of the NN is implemented by using AMs, while the backward propagation for retraining the NN is implemented by using accurate single-precision floating-point arithmetic operations. The classification results of AMs as per their effects on the retrained vgg16-cifar10 are shown in Fig. 11. Due to the retraining, some AMs result in a higher accuracy than using the exact design; they are denoted as "class b" in Fig. 11. This probably occurs because training with AM is equivalent to adding noise to the NN, which acts as a regularization that can result in a higher accuracy. Compared to the accuracy without retraining, vgg16-cifar10 tolerates more errors in the multiplication. Thus, the threshold values for the ER and μ_{ED} (ensuring less than 8% accuracy loss in the AM-based vgg16-cifar10) are increased to 45% and 33, respectively. This indicates that retraining an AM-based NN relaxes the error constraints for AM designs, which may result in more efficient hardware design when the same accuracy is required for the applications.



(a) retrained classification results vs. μ_E, μ_{ED} and ER (b) retrained classification results vs. μ_E and μ_{ED}

Fig. 11: Classification results of AMs in retrained vgg16-cifar10 vs. their three most informative error features.

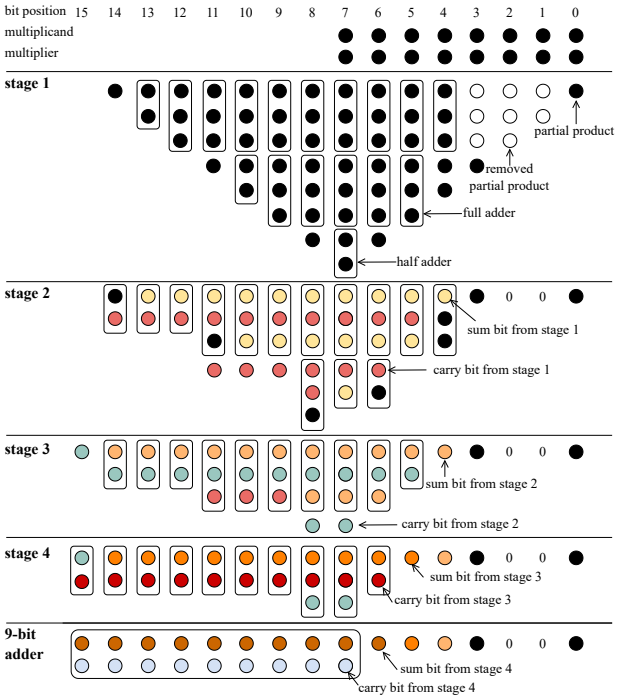


Fig. 12: The functional diagram of DAM_1_2_2_0.

B. Hardware Improvements

Table V shows the circuit and error measurements for the accurate and approximate multipliers. Note that two classical structures for the accurate multiplier are considered, the carry-save array multiplier and the Wallace tree multiplier, and denoted as accurate_array and accurate_Wallace, respectively. The average values of the circuit measurements for these two designs are considered as the baseline in the comparison. The circuits are synthesized by using Synopsys Design Compiler under HLMC 40 nm process with a supply voltage of 1.1 V at 25°. The clock period is 2 ns. The syntheses for all circuits are subject to area optimization. For a similar quality loss in the NN application, the AMs with the largest μ_{ED} are synthesized to obtain the largest hardware gain due to approximation. Both scenarios with and without retraining are studied.

As shown in Table V, without retraining, DAM_1_2_2_0 results in the most significant reduction in power-delay product (PDP) (by 21.88%) and in area-delay product (ADP) (by 27.52%) compared with the accurate design, with about 3% accuracy loss in vgg16-cifar10. DAM_1_2_2_0 is obtained by discarding some full and half adders in the accurate_Wallace multiplier, as shown in Fig. 12. Compared with accurate_Wallace, DAM_1_2_2_0 saves 4 full adders and 5 half adders for the partial product accumulation, and utilizes a 9-bit instead of 11-bit carry-propagate adder for the final addition. By introducing retraining, the AM named mul8_069 [12] achieves as much as 58.07% and 52.78% reductions in PDP and ADP with less than 3% accuracy loss. Moreover, the retrained vgg16-cifar10 using mul8_431 [12] reaches a higher accuracy than the one using an accurate multiplier. The PDP and ADP gains for mul8_431 [12] are both higher than 43%. Thus, retraining enhances the tolerance of the application to errors due to multipliers. Compared with the truncation-based AMs (i.e., BAM [13][1] and TruM[2]), the AMs using other approximation techniques achieve higher accuracy with less hardware complexity.

VII. Conclusion

Based on a feature ranking and selection strategy, this paper reveals the relationship between the statistical error features of AMs and the accuracy of NN applications using AMs. By using DFR, three most informative error features, the error bias (μ_E), the mean error distance (μ_{ED}), and the error rate (ER), are obtained. The error models are then constructed by using the selected error features to predict the impact of AMs on NN applications. Specifically, classification and regression models for some particular NNs and datasets are developed and verified; the obtained models show a high classification accuracy (over 96% top-1). Given the statistical error features, the proposed framework can construct effective error models to predict the accuracy of an application using approximate arithmetic units. This framework, thus, guides the selection and design of a proper approximate design for an application under a particular accuracy constraint.

The analysis shows that a low ER or μ_{ED} for AMs guarantees the accuracy of the NN applications. Although a small error bias (μ_E) in the multiplication is not a sufficient condition for ensuring a high-quality NN application, it can relax the requirements for ER and μ_{ED}. As an AM with a low ER is generally hard to be hardware-efficient, a priority in designing an efficient AM for NN applications lies in lowering the error bias and distance. Last but not the least, retraining the AM-based NNs can significantly improve their tolerance for errors in the multiplication. Some AMs can achieve improvements in PDP and ADP by over 40% compared with an exact multiplier while achieving no accuracy loss in NN applications with retraining. Thus, retraining is highly effective to ensure the accuracy of AM-based NN applications and allows for more significant hardware improvements in AM design.

References

[1] B. Liu, H. Cai, Z. Wang, Y. Sun, Z. Shen, W. Zhu, Y. Li, Y. Gong, W. Ge, J. Yang, and L. Shi, "A 22nm, 10.8 μw/15.1 μw dual computing

[1]BAM_k is the approximate array multiplier that truncates the k least significant partial product bits.

[2]TruM_k denotes the truncated multiplier whose k least significant bits of the two input operands are truncated.

TABLE V: The error and circuit characteristics of accurate and some efficient approximate multipliers.

AM	With retraining	μ_E	μ_{ED}	ER (%)	Accuracy loss (%)	Delay (ns)	Power (μW)	Area (μm^2)	PDP reduction (%)	ADP reduction (%)
accurate_array	–	0.00	0.00	0.00	0.00	1.85	42.59	276.07	–	–
accurate_Wallace	–	0.00	0.00	0.00	0.00	1.75	48.04	289.30	–	–
mul8_065 [12]	no	17.75	161.3	95.7	-1.28	1.52	42.10	275.54	21.42	17.64
mul8_453 [12]	no	128.0	128.0	12.5	-2.44	1.26	51.12	319.28	20.09	20.88
DAM_1_2_2_0	no	10.00	10.00	77.3	-3.07	1.44	44.18	255.96	**21.88**	**27.52**
RAM_1_1_1_7	no	27.43	34.72	39.5	-5.20	1.34	40.40	243.61	33.52	35.80
BAM_3 [13]	no	4.250	4.250	68.7	-1.24	1.84	39.86	253.66	9.958	8.212
TruM_1	no	127.3	127.3	74.6	-87.4	1.55	29.16	202.68	44.50	38.22
mul8_069 [12]	yes	224.2	270.8	98.2	-2.41	1.77	19.29	135.65	**58.07**	**52.78**
mul8_486 [12]	yes	210.4	256.1	98.2	-4.96	1.84	19.53	140.24	55.87	49.25
DAM_1_4_3_0	yes	90.00	90.00	83.9	-4.87	1.28	37.62	239.90	40.87	39.61
mul8_431 [12]	yes	111.4	145.3	96.9	**+0.49**	1.75	25.45	164.93	**45.31**	**43.24**
mul8_160 [12]	yes	44.13	58.76	93.5	+0.24	1.97	29.73	186.45	28.09	27.76
RAM_1_1_1_8	yes	55.43	74.03	43.7	-3.8	1.34	39.34	237.96	35.28	37.29
BAM_4 [13]	yes	12.25	12.25	81.2	-3.73	1.78	38.22	240.08	16.47	15.96
TruM_1	yes	127.3	127.3	74.6	-66.8	1.55	29.16	202.68	44.50	38.22

modes high power-performance-area efficiency domained background noise aware keyword-spotting processor," *IEEE TCAS-I*, vol. 67, no. 12, pp. 4733–4746, 2020.

[2] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," in *ETS*, 2013, pp. 1–6.

[3] S. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, March 2004.

[4] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[5] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: Systematic logic synthesis of approximate circuits," in *DAC Design Automation Conference 2012*, 2012, pp. 796–801.

[6] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 432–444, 2015.

[7] D. Sengupta, F. S. Snigdha, Jiang Hu, and S. S. Sapatnekar, "SABER: Selection of approximate bits for the design of error tolerant circuits," in *DAC*, 2017, pp. 1–6.

[8] S. Hashemi, H. Tann, and S. Reda, "BLASYS: Approximate logic synthesis using boolean matrix factorization," in *DAC*, 2018, pp. 1–6.

[9] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "autoAx: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *DAC*, 2019, pp. 1–6.

[10] M. Ansari, V. Mrazek, B. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *TVLSI*, vol. 28, no. 2, pp. 317–328, 2020.

[11] M. S. Kim, A. A. Del Barrio, H. Kim, and N. Bagherzadeh, "The effects of approximate multiplication on convolutional neural networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 904–916, 2022.

[12] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "EvoApprox8b: library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *DATE*, 2017, pp. 258–261.

[13] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *TCAS I*, vol. 57, no. 4, pp. 850–862, 2010.

[14] M. Ahmadinejad and M. H. Moaiyeri, "Energy- and quality-efficient approximate multipliers for neural network and image processing applications," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–12, 2021.

[15] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSID*, 2011, pp. 346–351.

[16] K. Bhardwaj, P. S. Mane, and J. Henkel, "Power- and area-efficient approximate wallace tree multiplier for error-resilient systems," in *Fifteenth International Symposium on Quality Electronic Design*, 2014, pp. 263–269.

[17] S. Venkatachalam, E. Adams, H. J. Lee, and S.-B. Ko, "Design and analysis of area and power efficient approximate booth multipliers," *IEEE Transactions on Computers*, vol. 68, no. 11, pp. 1697–1703, 2019.

[18] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.

[19] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi, and F. Lombardi, "Design and evaluation of approximate logarithmic multipliers for low power error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[20] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 614–625, 2021.

[21] R. Murillo, A. A. Del Barrio Garcia, G. Botella, M. S. Kim, H. Kim, and N. Bagherzadeh, "PLAM: a posit logarithm-approximate multiplier," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–6, 2021.

[22] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys, "Pushing the limits of voltage over-scaling for error-resilient applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, pp. 476–481.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of Systemic Autoimmune Diseases*, vol. 1, no. 4, 2009.

[26] F. Vaverka, V. Mrazek, Z. Vasicek, and L. Sekanina, "TFApprox: towards a fast emulation of dnn approximate hardware accelerators on GPU," in *DATE*, 2020, pp. 294–297.

[27] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[28] B. Kaddar, H. Fizazi, and D. El Kefel Mansouri, "Convolutional neural network features selection based on analysis of variance," in *International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, vol. 1, 2019, pp. 1–5.

[29] N. L. Kotz, Samueland Johnson, *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY: Springer New York, 1992.

[30] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[31] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[32] H. Liu and R. Setiono, "Feature selection and classification-a probabilistic wrapper approach." in *IEA/AIE*, 1996, pp. 419–424.

[33] J. Dass, Y. Narawane, R. N. Mahapatra, and V. Sarin, "Distributed training of support vector machine on a multiple-fpga system," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 1015–1026, 2020.

[34] A. F. Ajirlou and I. Partin-Vaisband, "A machine learning pipeline stage for adaptive frequency adjustment," *IEEE Transactions on Computers*, vol. 71, no. 3, pp. 587–598, 2022.

[35] C. H. Chang, L. Rampasek, and A. Goldenberg, "Dropout feature ranking for deep learning models," *arXiv e-prints*, 2017.

[36] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, pp. 2575–2583, 2015.

[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, p. 273–297, 1995.

[38] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
[39] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers - a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 4, pp. 476–487, 2005.
[40] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.
[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations, ICLR*, 2015, pp. 1–15.
[42] S. Hashemi *et al.*, "Drum: A dynamic range unbiased multiplier for approximate applications," in *ICCAD*, 2015, pp. 418–425.
[43] A. G. M. Strollo *et al.*, "Comparison and extension of approximate 4-2 compressors for low-power approximate multipliers," *TCAS-I*, vol. 67, no. 9, pp. 3021–3034, 2020.
[44] L. Sayadi *et al.*, "Two efficient approximate unsigned multipliers by developing new configuration for approximate 4:2 compressors," *TCAS-I*, vol. 70, no. 4, pp. 1649–1659, 2023.

Hai Mo received B.S. degree in Computer Science from Huazhong University of Science and Technology, Wuhan, Hubei, China in 2019 In 2022, he obtained a M.S. degree in the Integrated Circuits at Tsinghua University, Beijing, China. Currently, he is employed as an engineer at Huawei Technologies Co., Ltd, engaged in R&D of CPU.

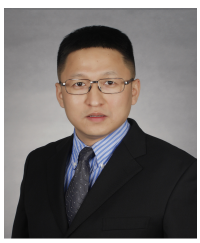
Yong Wu received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2020 and the Master degree from Tsinghua University, Beijing, China, in 2023. His current research interests include approximate computing and system-on-chip design.

Honglan Jiang (S'14-M'18) received the B.Sc. and Master degrees from Harbin Institute of Technology, Harbin, in 2011 and 2013, respectively. In 2018, she received the Ph.D. degree in integrated circuits and systems from University of Alberta, Edmonton, Canada. From 2018 to 2021, she worked as a postdoctoral fellow with the School of Integrated Circuits, Tsinghua University, Beijing, China. She is currently an associate professor with the Department of Micro-Nano Electronics, Shanghai Jiao Tong University, Shanghai. Her research interests include approximate computing, reconfigurable computing, and stochastic computing. She serves as an Associate Editor for the Microelectronics Reliability.

Zining Ma received the B.S. degree in microelectronics from the Harbin Institute of Technology, Harbin, China in 2019. He is currently studying forward the master's degree at the School of Integrated Circuits, Tsinghua University, Beijing, China. His current research interests include stochastic computing and reconfigurable computing.

Fabrizio Lombardi (M'81-SM'02-F'09) graduated from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. He joined the Microwave Research Unit at University College London, where he received the Master degree in Microwaves and Modern Optics, and the Ph.D. from the University of London. He is the holder of the International Test Conference Endowed Chair at Northeastern University, Boston. Dr. Lombardi is the 2022-2023 President of the IEEE Nanotechnology Council (NTC); he was 2nd VP of the IEEE Computer Society (CS) (2021), the VP of Publications of the IEEE CS (2019-2020) and the NTC (2020). Dr. Lombardi has been the Editor-in-Chief (2007-2010), Associate Editor-in-Chief (2000-2006) and Associate Editor (1996-2000) of the IEEE Transactions on Computers, the inaugural Editor-in-Chief of the IEEE Transactions on Emerging Topics in Computing (2013-2017) and Editor-in-Chief of the IEEE Transactions on Nanotechnology (2014-2019). Dr. Lombardi has received many Best Paper Awards; since inception, he has been always included in the list of World's Top 2% Scientists, as compiled by Stanford University. His research interests are emerging technologies, memory systems, VLSI design and fault/defect tolerance of digital systems.

Jie Han (S'02-M'05-SM'16) received the B.Sc. degree in electronic engineering from Tsinghua University, China, in 1999 and the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2004. He is currently a Professor and Director of Computer Engineering in the Department of Electrical and Computer Engineering at the University of Alberta, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for nanoscale and biological applications. Dr. Han was a recipient of the Best Paper Awards at NANOARCH 2015 and DATE 2023, and Best Paper Nominations at GLSVLSI 2015, NANOARCH 2016, ISQED 2018 and DATE 2022. He was nominated for the 2006 Christiaan Huygens Prize of Science by the Royal Dutch Academy of Science. He serves (or has served) as an associate editor for the IEEE Transactions on Emerging Topics in Computing (TETC), the IEEE Transactions on Nanotechnology, the IEEE Circuits and Systems Magazine, the IEEE Open Journal of the Computer Society, Microelectronics Reliability and the Journal of Electronic Testing: Test and Application (JETTA).



Leibo Liu (M'10-SM'17) received the B.S. degree in electronic engineering and the Ph.D. degree with the Institute of Microelectronics, both from Tsinghua University, Beijing, China, in 1999 and 2004, respectively. He is currently a Full Professor with the School of Integrated Circuits, Tsinghua University. His current research interests include software defined chips, crypto-chip design and hardware security, and VLSI digital signal processing.