A Technique for Approximate Communication in Network-on-Chips for Image Classification

Yuechen Chen[a,*], Shanshan Liu[b], Fabrizio Lombardi[b], Ahmed Louri[a]

[a]*Department of Electrical and Computer Engineering, School of Engineering and Applied Science, The George Washington University, 800 22nd St. NW, Washington DC 20052, US*
[b]*Department of Electrical and Computer Engineering, College of Engineering, Northeastern University, 360 Huntington Ave., Boston MA 02115, US*

Abstract

Approximation is an effective technique for reducing power consumption and latency of on-chip communication in many computing applications. However, existing approximation techniques either achieve modest improvements in these metrics or require retraining after approximation, such when convolutional neural networks (CNNs) are employed. Since classifying many images introduces intensive on-chip communication, reductions in both network latency and power consumption are highly desired. In this paper, we propose an approximate communication technique (ACT) to improve the efficiency of on-chip communications for image classification applications. The proposed technique exploits the error-tolerance of the image classification process to reduce power consumption and latency of on-chip communications, resulting in better overall performance for image classification computation. This is achieved by incorporating novel quality control and data approximation mechanisms that reduce the packet size. In particular, the proposed quality control mechanisms identify the error-resilient variables and automatically adjust the error thresholds of the variables based on the image classification accuracy. The proposed data approximation mechanisms significantly reduce packet size when the variables are transmitted. The proposed technique reduces the number of flits in each data packet as well as the on-chip communication, while maintaining an excellent image classification accuracy. The cycle-accurate simulation results show that ACT achieves 23% in network latency reduction and 24% in dynamic power reduction compared to the existing approximate communication technique with less than 0.99% classification accuracy loss.

Keywords: Image Classification, Network-on-Chips (NoCs), Approximation

1. Introduction

Image classification applications widely use deep convolutional neural networks (CNNs) and are deployed from cloud to edge computational frameworks for varieties of scenarios, such as search engines and self-driving cars [1, 2, 3, 4, 5, 6]. As the complexity of these applications and the resolution of images continue to increase, conventional homogeneous architectures (such as multi-core CPU/GPU) are constrained due to an excessive long latency and significant power dissipation [7, 8, 9]. To efficiently process these applications, heterogeneous architectures have been proposed with pre-processing and inference cores [7, 8, 9, 10, 11, 12, 13].

- Pre-processing cores are designed to prepare data by resizing the raw image and then normalizing the value for each pixel into a specific range.
- Inference cores are designed to fetch the processed data and parameters of the CNN model to perform inference.

Network-on-chips (NoCs) have been widely used to efficiently connect cores, memory interfaces, and caches in these architectures [14, 15, 16]. Recent research [7, 17, 18, 19] has shown that with a heterogeneous architecture, data transfer can account for up to 34% of the execution time and up to 40% of the overall chip power consumption. Since image classification applications can tolerate errors in the parameters and the inputs, approximation techniques have been proposed for reducing data transfer, thus reducing network latency and power consumption [20, 21, 22, 23]. Existing approximation techniques can be categorized as follows:

- Existing approximate communication techniques [22, 24, 25, 26, 27, 28, 29, 30, 31, 32] reduce communication latency and power consumption by utilizing packet approximation in NoCs. However, existing techniques only rely on the relative error for data approximation. Since relative error tolerance is limited for image classification applications, only few packets can be approximated using existing approximate communication techniques.
- Existing CNN approximation techniques [33, 34, 35, 36, 37] reduce the size of the model using quantization or pruning. However, these techniques do not specifically target image classification. Moreover, as quantizing and

[*]Corresponding author
Email addresses: `yuechen@gwu.edu` (Yuechen Chen), `ssliu@coe.neu.edu` (Shanshan Liu), `lombardi@ece.neu.edu` (Fabrizio Lombardi), `louri@gwu.edu` (Ahmed Louri)

pruning the parameters can significantly reduce the classification accuracy, existing techniques require the model to be retrained prior to inference. The retraining process requires substantial time to complete, while incurring in considerable power consumption.

To address the above issues, an approximate communication technique (ACT) that enhances communication efficiency for image classification is proposed for heterogeneous systems; it leverages the error-tolerance of the image classification application to reduce the transmitted packet size, thus reducing power consumption and network latency. ACT utilizes two approximate communication techniques: an approximate communication for the pre-processing cores (ACT-P) and an approximate communication for the inference cores (ACT-I). Each technique includes quality control and data approximation mechanisms to leverage the error tolerance in multiple steps of the image classification process. Specifically, the contributions of this paper are as follows:

- An approximate communication technique (ACT) is proposed to reduce network latency and dynamic power consumption for image classification applications by leveraging the error tolerance of the application itself.
- Two techniques, ACT-P and ACT-I, are developed to reduce the data transmission during image pre-processing and model inference.
- The ACT is implemented with software-hardware co-design.
- Performance evaluation results show that compared to the existing approximate communication technique, ACT reduces network latency and dynamic power consumption by 23% and 24%, respectively, with less than 0.99% classification accuracy loss.

This paper is organized as follows. Section 2 presents a background for the proposed technique; Section 3 outlines the basic operational principles of the ACT. The implementation is presented in detail in Section 4, while Section 5 deals with its extensive evaluation. Section 6 concludes this manuscript.

2. Background

Approximation techniques are widely used to enhance the communication efficiency of image classification applications. Existing approximation techniques can be categorized into two types:

- Approximate communication techniques to reduce power and latency of communication during the execution of an application
- Approximation techniques for the CNN model to reduce the model size prior to the execution

These techniques will be reviewed next as relevant to the proposed technique.

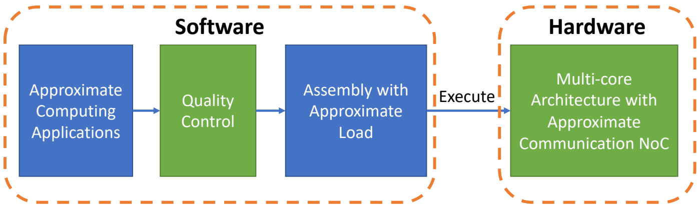

Software
Hardware
Approximate Computing Applications
Quality Control
Assembly with Approximate Load
Execute
Multi-core Architecture with Approximate Communication NoC

Figure 1: High-Level Workflow of Current Approximate Communication Techniques.

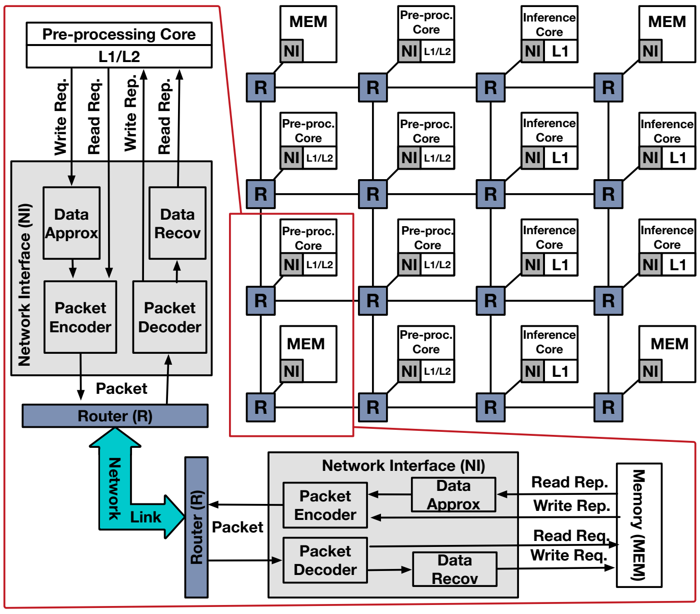


Figure 2: Heterogeneous Multicore Architecture with an Approximate Communication NoC.

2.1. Approximate Communication Techniques

Approximate communication is considered to be one of the most effective approaches to improve network performance when an application can tolerate errors [22, 24, 25, 26, 27, 28, 29, 30, 31, 32]; with a reduced accuracy during communication, the approximation techniques significantly reduce the network latency and power consumption for on-chip communication.

Fig. 1 shows the general process of current approximate communication techniques; such techniques include a software-based control method to ensure result quality and a hardware-based data approximation method to reduce packet size. Prior to the execution of an application, a quality control method identifies error-resilient variables and calculates the error tolerance of each variable based on the application requirements in terms of the result quality. Then, load and store instructions for the error-resilient variables are replaced with approximate counterparts for packet approximation during execution. These approximate load and store instructions contain approximation information, including the approximation level for the corresponding variable and the variable type (e.g., integer or floating-point). The approximation level indicates the amount of relative error that a variable can tolerate. Then, the application is usually executed on a multi-core architecture with an approximate communication network-on-chip (NoC).

Fig. 2 shows an approximate communication NoC [22, 24, 25, 26, 27, 28, 29, 30, 31, 32] implemented in a heterogeneous multi-core system [7, 8, 9, 10, 11, 12, 13] with an L2 shared

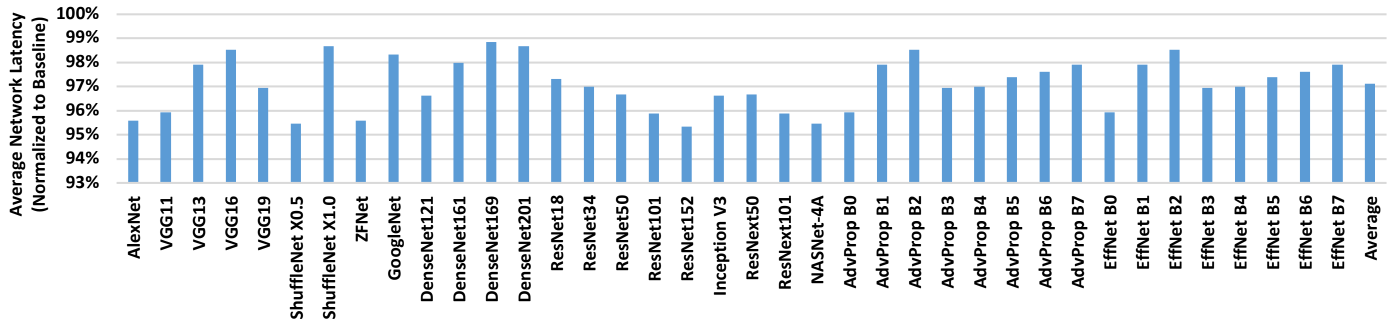


Figure 3: Network Latency Reduction by Employing Existing Approximate Communication Technique to Different Image Classification Applications.

cache for CNN inference. The data approximation module in the network interface reduces the packet size by truncation or lossy compression according to the approximation information. Consider a cache miss during a memory load or store operation by an X86 CPU for image pre-processing.

- *Miss on load operation:* When a cache miss occurs during a memory load operation, a read request packet is sent to the memory or the shared cache through the NoC. The memory or shared cache uses a read reply packet to send the required data back to the core. If the memory load can be approximated, then the read request also carries approximation information; subsequently, the data approximation module at the memory/shared cache node reduces the size of the read reply packet accordingly. The approximated read reply packet carries the approximated data and the approximation information to the core. When the approximated read reply reaches the core, the data recovery module recovers the approximated data to the original length in accordance with the approximation information.

- *Miss on store operation:* When a cache miss occurs during a memory store operation, the data is incorporated into a write request packet and sent to the memory or shared cache through the NoC. The data approximation module reduces the size of the write request packet according to the approximation information if the memory store can be approximated. The approximated write request packet carries the approximated data and approximation information to the memory or shared cache node. When the approximated write request reaches the memory or shared cache node, the data recovery module recovers the approximated data to the original length in accordance with the approximation information. After the memory or shared cache has received the data, a write reply is sent back to the core to confirm a successful memory write.

Various data approximation methods have been proposed to reduce the packet size according to the approximation information. [27] and [38] have proposed lossy data compression techniques to further reduce the size of the error-resilient data based on the relative error. [28, 39] have proposed techniques to reduce network congestion by dropping data in the packet prior to injecting it into the network. [39] has investigated the application error threshold and proposed an approximation method to drop data.

However, existing techniques achieve a limited improvement when CNNs are utilized for image classification applications because the parameters of the model and the inputs cannot be approximated using methods based on the relative error. Fig. 3 shows the network latency reduction normalized to the network with no approximation (baseline) for the existing approximate communication framework (ACF) [26, 38]. This figure indicates that ACF only reduces network latency by less than 5% when applied to various state-of-the-art image classification applications [3, 4, 5, 6, 40, 41, 42, 43, 44, 45, 46, 47] as executed on heterogeneous multi-core architectures with an NoC.

2.2. *CNN Approximation Techniques*

Quantization and pruning methods are widely used for deep CNNs in image classification applications to reduce communication traffic and computation [35]. For example, in [47], the size of the deep neuron network is significantly reduced using quantization, pruning, and Huffman coding. [48] has proposed to quantize the activations and the parameters of the CNNs to a single bit (i.e., so with a binary value of 0 or 1). [36] has proposed a CNN pruning method to reduce the model size based on the user requirement of energy consumption. Existing image classification applications [3, 4, 5, 6, 40, 41, 42, 43, 44, 45, 46, 47] are implemented using Pytorch [48] and TensorFlow [49] frameworks, which support CNN quantization and pruning on generic inference cores (e.g., CPUs, GPUs, CNN Accelerators). However, existing model approximation techniques have two major limitations.

(i) They are developed for generic CNN inference. The classification system performance improvement methods have not been explored specifically designed for image classification. Thus, system performance can be further improved with dedicated optimization techniques.
(ii) They require the model to be retrained or fine-tuned before classifying images, because these techniques incur in a significant reduction in classification accuracy.

This paper aims to approximate the image classification application during the execution process for communication efficiency enhancement by incurring in only a very limited impact on accuracy.

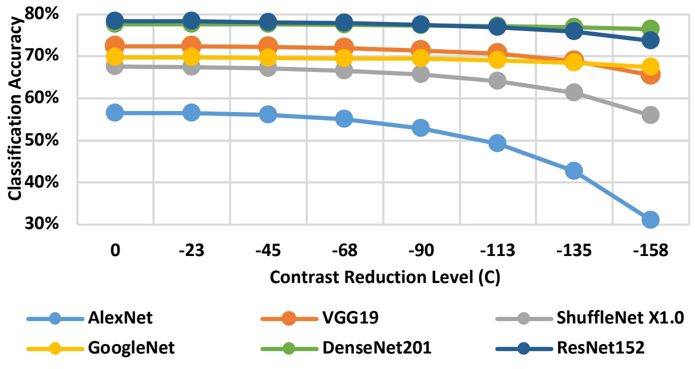
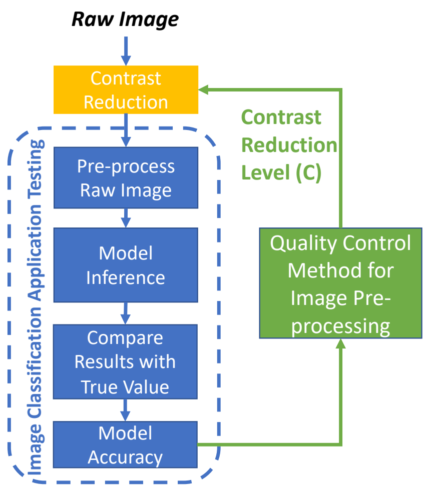


Figure 4: Classification Accuracy V.S. Contrast Reduction Level (C).



Raw Image
Contrast Reduction
Image Classification Application Testing
Pre-process Raw Image
Model Inference
Compare Results with True Value
Model Accuracy
Contrast Reduction Level (C)
Quality Control Method for Image Pre-processing

Figure 5: Proposed Quality Control Mechanism For Image Pre-processing.

3. Proposed Approximate Communication Technique

The proposed approximate communication technique (ACT) reduces network latency and power consumption of on-chip communication in NoCs. This is mainly accomplished by reducing the size of each packet and exploiting the error-tolerant features of image classification applications. The image classification applications tolerate two types of errors [33, 34, 50]: the first type is image contrast reduction during the image pre-processing; the second type is quantization errors in the fully connected layer during model inference. Thus, ACT includes two sets of approximate communication techniques to leverage two types of error tolerance.

(i) The *approximate communication for image pre-processing* (ACT-P) includes quality control and data approximation mechanisms.

- The *quality control mechanism* dynamically adjusts the image contrast and monitors the accuracy of the application to balance it with the communication efficiency.
- The *data approximation mechanism* for image pre-processing reduces the data size by reducing the image contrast.

(ii) The *approximate communication for model inference* (ACT-I) includes quality control and data approximation mechanisms.

- The *quality control mechanism* monitors the values of the variables when a fully connected layer is processed.
- After recording the maximum/minimum values of the variables by the quality control mechanism, the *data approximation mechanism* utilizes data quantization to reduce data size.

Section 3.1 describes the design of ACT-P, while Section 3.2 describes the design of ACT-I

3.1. Approximate Communication for Image Pre-processing (ACT-P)

Recent research has shown that image classification applications are resilient to contrast reduction on the raw image prior to inference [33, 50]. In this paper, it is assumed that the level of contrast C ranges from −255 to infinity: when $C = 0$, there is no adjustment to the image, but when $C \in (-255, 0)$, the image contrast is reduced. When $C = -255$, all values of the pixels (R, G, B) in an image are 128, so making the image of a solid grey color; hence, Eq. 1 describes the relationship between the contrast correction factor F and the level of contrast C.

$$\mathrm{F} = \frac{(259(\mathrm{C} + 255))}{(255(259 - \mathrm{C}))} \tag{1}$$

As per F above, the contrast reduction for each pixel is performed by Eq. 2, in which the variable P is the value of a color of a pixel (in a range from 0 to 255), and P' represents the corresponding value with contrast reduction.

$$P' = round(F(P - 128) + 128) \tag{2}$$

Fig. 4 shows the classification accuracy for a few widely-used image classification applications versus the level of contrast reduction; image classification applications can tolerate 23 levels of contrast reduction (i.e., $C = -23$) with negligible accuracy reduction (0.07% accuracy reduction on average). Fig. 4 also shows that different image classification applications have different accuracy tolerance for image contrast reduction; for example, for a classification accuracy loss of up to 1%, AlexNet [5] can tolerate 23 levels of contrast reduction, while VGG19 [6] can tolerate 90 levels. Thus, a quality control mechanism is needed to select the appropriate contrast reduction level for the different image classification applications to avoid a significant loss in classification accuracy.

3.1.1. Quality Control

A quality control mechanism for image pre-processing is utilized to maintain the accuracy of image classification. Fig. 5 shows the proposed design of the quality control for image classification. This mechanism adjusts the contrast level during the testing process, this is the last step prior to the classification of the images by the application. Testing includes three phases.

(i) The raw images in the test data set are processed by the core. Different from the images that the application processes, the data set contains the query data (raw image) and the true value for each image (label).

Table 1: Relationships Between Contrast Reduction Modes, Contrast Reduction Level (C), and Contrast Correction Factor(F).

Contrast Reduction Modes	Contrast Reduction Level (C)	Contrast Correction Factor (F)
0	0	1
1	-23	0.835
2	-45	0.701
3	-68	0.581
4	-90	0.480
5	-113	0.388
6	-135	0.309
7	-158	0.236

(ii) The model inference is then accomplished by fetching the processed data and the classification model.

(iii) Finally, the generated result is processed by the core to compare it with the true value. The model accuracy is calculated by comparing the predictions generated by the model with the true value.

The quality control mechanism utilizes the accuracy calculated by the core to adjust the image contrast. When considering the potential accuracy reduction caused by applying approximate communication for model inference, the accuracy reduction due to the image contrast reduction is limited to less than 1%. Thus, the following novel procedure is proposed to determine the image contrast reduction level:

(i) During the first phase of the test process, the classification accuracy of the image application is calculated with no image contrast reduction. The classification accuracy is calculated as in Eq. 3. The correct classification is defined as the image category (e.g., cat, dog, car) with the highest probability (as predicted by the model); this must be exactly the same as expected answer (label).

$$Classification\ Accuracy = \frac{Number\ of\ Correct\ Classifications}{Total\ Number\ of\ Classifications} \tag{3}$$

(ii) The quality control mechanism gradually reduces the image contrast levels by choosing a contrast reduction mode according to Table 1 until there is more than 1% loss (as threshold) in the classification accuracy compared to the estimated base accuracy in the next testing process.

(iii) The mode before the last contrast reduction mode is chosen for image classification. For example, if the classification error exceeds 1% at mode 3, mode 2 is selected for image classification.

The proposed quality control mechanism is implemented in software and supports eight contrast reduction modes that are shown in Table 1. Section 5.5 discusses the impact on classification accuracy if different accuracy loss thresholds are chosen. During image classification, the true value for each query image is not available, so the contrast reduction mode is fixed and registered in the network interface prior to the classification.

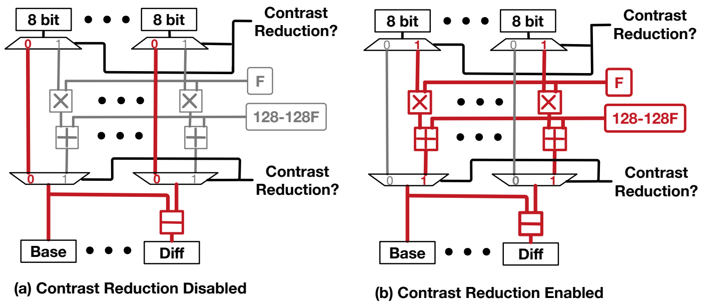


Figure 6: Design of Data Approximation for Image Pre-processing.

3.1.2. Data Approximation

The data approximation mechanism reduces the amount of transmitted data for image contrast reduction; the so-called base-delta approximation mechanism is proposed to take advantage of the reduced image contrast for data reduction. Since the difference in values between pixels in an image is small, the base-delta compression mechanism can significantly reduce the number of bits needed to represent each pixel. Moreover, the proposed image contrast reduction process further reduces the difference between pixels, so the data size can be substantially reduced by only transmitting the difference between pixels. Fig. 6 shows the design of the proposed base-delta approximation mechanism for image pre-processing. The data approximation process consists of two steps.

Step 1: The image contrast reduction operation is activated with a contrast reduction level.

Step 2: The multipliers and adders then adjust the value for each pixel based on Eq. 1 and Eq. 2.

To reduce its complexity, ACT-P supports eight levels of image contrast reduction. Table 1 shows the mapping of the conversion of the contrast reduction level (C) into the contrast correction factor (F). The first 8-bit data is chosen as the base; the remaining data is represented as the distance to the base. Fig. 6 (a) shows the approximation process when the image contrast reduction is deactivated (Contrast Reduction Level = 0, Mode 0). The data bypass the contrast reduction operation and being compressed with full accuracy. Fig. 6 (b) shows the approximation process when the image contrast reduction is activated.

For example, a packet that contains three pixels with values 128(10000000), 192(11000000), and 100(1100100). Suppose these values can tolerate -68 levels of contrast reduction (Mode 3). According to Eq. 2, the values after the contrast reduction are 128(100000), 166(10100110), and 122(1111010), respectively. After base-delta compression, the packet only contains 128(10000000), -38(1100110), and 6(110), respectively. Compared to the original data (24 bits), the compressed data (18 bits) is 25% smaller. If the image is sent with no contrast reduction, the compressed data consists of 21 bits, so only 12.5% smaller than the original data. After the compressed data arrives at the destination, the data is recovered by adding the delta and the base.

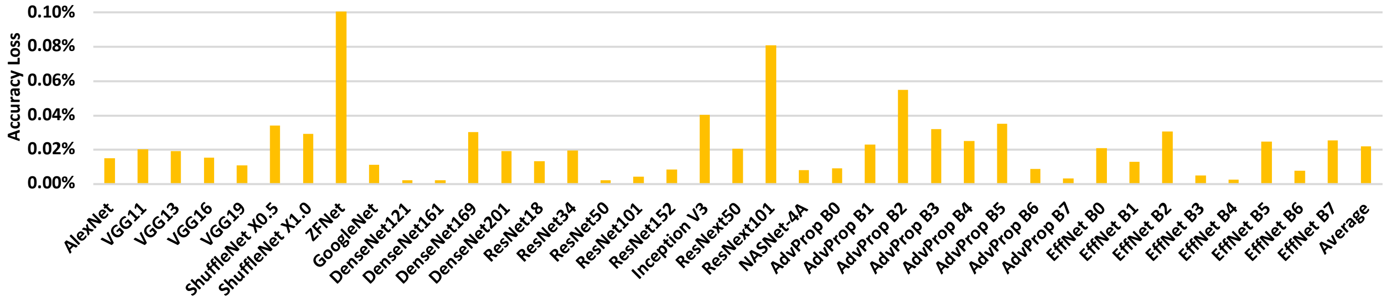


Figure 7: Quantization of fully connected layer affects classification accuracy.

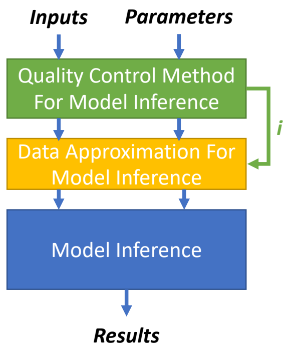

Inputs
Parameters
Quality Control Method For Model Inference
i
Data Approximation For Model Inference
Model Inference
Results

Figure 8: Quality Control Mechanism For Image Classification Model Inference.

3.2. *Approximate Communication for Model Inference (ACT-I)*

[36] has shown that the classification accuracy reduction in image classification applications is negligible after applying quantization to the parameters and activation of the fully connected layers (inputs). As floating-point data type is widely used in image classification applications [3, 4, 5, 6, 40, 41, 42, 43, 44, 45, 46, 47] to represent parameters and activation, the quantization process consists of mapping a floating-point value $x \in [\alpha, \beta]$ to a b-bit integer $x_q \in [\alpha_q, \beta_q]$; this is computed as per Eq. 4 (where c and d are variables).

$$x_q = round\ down(\frac{1}{c}x - d) \tag{4}$$

Note that when performing quantization, the floating-point 0 must be mapped to a b-bit integer 0. Thus, the relationship between c, d and the ranges of x and x_q is given as follows.

$$\begin{cases} \beta = c(\beta_q + d) \\ \alpha = c(\alpha_q + d) \end{cases} \tag{5}$$

The solution of Eq. 5 yields the values of c and d (i.e., Eq. 6).

$$\begin{cases} c = \frac{\beta - \alpha}{\beta_q - \alpha_q} \\ d = \frac{\alpha\beta_q - \beta\alpha_q}{\beta - \alpha} \end{cases} \tag{6}$$

[36] has shown that the accuracy reduction caused by quantizing the IEEE standard 32-bit floating-point data into 8-bit integers in fully connected layers is negligible. This is shown in Fig. 7 for different image classification applications in which the parameters and inputs of the fully connected layers are quantized. The quantization process only reduces the classification accuracy by 0.022% on average. This observation suggests that data quantization is an attractive solution to significantly reduce on-chip communication when the model has fully connected layers.

However, as per Eq. 6, the range of x (i.e., α and β) must be considered when performing data quantization. Since data (i.e., x) exceeding the range is basically clipped (by truncation) during the quantization process, the range must be dynamically determined for different data items. Otherwise, many data items could be clipped, thus negatively impacting the model accuracy. Therefore, a novel quality control mechanism is developed to estimate the range of inputs and parameters in the proposed scheme.

Also, once the data range is determined, floating-point operations (such as multiplication, division, and subtraction) must be performed to map the data (as per Eqs. 4 and 6). These operations are not always acceptable because they could incur significant communication overheads for latency and power consumption. Therefore, the proposed data approximation mechanism uses a variable i to quantize data, as described next.

3.2.1. *Quality Control*

Fig. 8 shows the proposed process of quality control for model inference. The proposed quality control mechanism constantly monitors the parameters and inputs of the fully connected layer. To reduce the complexity of data quantization, a new variable i is introduced based on the following observations.

Observation 1: Quantization maps data from the original range to another range with different granularity, thus causing quantization errors. For example, when quantizing 32-bit floating-point data into 8-bit integers, the granularity of the data range increases from 1/16777216 to 1/255; in this case, the error originates from the decimal part.

Observation 2: For an integer, a deviation within (0, 1) (i.e., adding the integer with a decimal value) is only reflected on a few lower mantissa bits in its floating-point representation, and it has an almost negligible impact on all upper bits. Moreover, the changed mantissa bits are separated from the upper bits related to the sign and the integer part of the data.

Observation 3: The expansion of a floating-point value by 2^i times only changes its exponent bits (where i is a positive integer). Consider Eqs. 7 and 8 that calculate the value of a 32-bit floating-point data D [51] and the corresponding enlarged value with 2^i times respectively; only the exponent value increases by i, while the sign and mantissa remain the same.

$$D = (-1)^{sign} \cdot 2^{exponent-127} \cdot (1 + mantissa) \tag{7}$$

$$D \cdot 2^i = (-1)^{sign} \cdot 2^{(exponent+i)-127} \cdot (1 + mantissa) \qquad (8)$$

Therefore, as per the above observations, a conventional data quantization approach can be replaced by expanding the original data by 2^i times and then rounding it down (i.e., mapping the floating-point data to an integer).

Thus, when the quality control mechanism receives the minimum (α) and the maximum (β) values of the weights and biases, i is calculated based on α and the β using Eq. 9.

$$i = log2(max(|\alpha|, |\beta|) \qquad (9)$$

However, the dynamic range of the inputs is not fixed because the query image is changed after each image classification. Thus, the inputs of the fully connected layer are constantly monitored during the image classification to establish the dynamic range of the input to calculate i.

To reduce the hardware overhead, i is limited to 8 bits, and the initial i values for the inputs and parameters are calculated during the image classification application testing and registered in the network interface before processing images. Since the increase of the value range leads to a decrease of i, the quality control mechanism automatically reduces i by 1 when an input value exceeds the dynamic range during the classification. Then, the data approximation mechanism reduces the data size according to i.

3.2.2. Data Approximation

The proposed data approximation mechanism quantizes data by enlarging the original data by 2^i times and then rounding it down. Thus, the quantization error is bounded within a few lower mantissa bits. Only the sign bit, the exponent bits, and a few upper mantissa bits need to be transmitted because they are separated from the lower bits. Moreover, to perform the multiplication with 2^i, a binary sequence of i needs to be added to the exponent part, i.e., only a binary addition operation is required, rather than floating-point arithmetic operations. As the ranges of inputs and parameters of the fully connected layers can be determined by utilizing the quality control mechanism proposed in the previous section, then the value of i is adjusted to guarantee that the quantized data belong to an integer range with an acceptable granularity (so to provide a good classification accuracy, such as the 8-bit integer range [34]).

To further reduce the size of the transmitted data, the exponent part is compressed by mapping the data patterns into symbols with a shorter length. Since all integers within the range of $[2^j, 2^{j+1})$ share the same exponent pattern as per Eq. (7) (where $j = 1, 2, ..., 127$), then only a few patterns are used for representing the quantized data that belongs to a range significantly smaller than the entire floating-point field. For example, when quantizing data into the 8-bit integer range, only eight exponent patterns may appear (i.e., 00000000 for value 0, 01111111 for value 2^0, 10000000 for values within $[2^1, 2^2)$, 10000001 for values within $[2^2, 2^3)$, etc.). In this case, 3-bit symbols that provide eight different combinations can be used for mapping all possible exponent patterns (and so transmitted), thus reducing the size for each exponent from 8 to 3 bits.

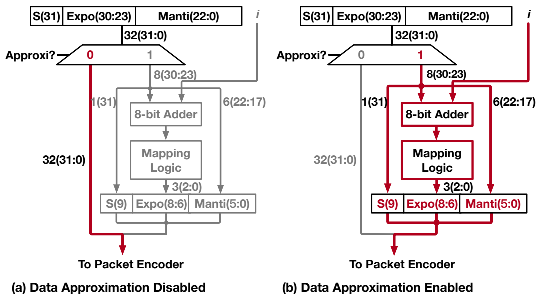


Figure 9: Hardware Design of Data Approximation for Model Inference.

Table 2: Mapping Between 8-bit Exponent Patterns and 3-bit Symbols.

Integer	Exponent Patten	Symbol
0	00000000	000
2^0	01111111	001
$[2^1,2^2)$	10000000	010
$[2^2,2^3)$	10000001	011
$[2^3,2^4)$	10000010	100
$[2^4,2^5)$	10000011	101
$[2^5,2^6)$	10000100	110
$[2^6,2^7)$	10000101	111

Next, an example of quantizing 32-bit floating-point values within $[-0.0903250, 0.0882086]$ (i.e., the range of parameters for one fully connected layer of the trained VGG11 [6]) into 8-bit integers is shown as an application of the proposed mechanism. To perform quantization, the original data is enlarged by 2^{10} (i.e., $i = 10$) times because $i = 11$ makes the quantized data exceeding the range of $[-127, 127]$. Therefore, only 15 bits, including 1 sign bit, 8 exponent bits, and 6 upper mantissa bits are sufficient for all quantized data. Then, the exponent part is further compressed by utilizing 3-bit symbols as per the mapping given in Table 2 because only eight exponent patterns are used to represent the quantized data. Overall, the data size is reduced from 32 bits to 10 bits by employing the proposed mechanism, so achieving a reduction of 68.75%.

The hardware design for the proposed data approximation mechanism for data quantization is illustrated in Fig. 9 Once data approximation is enabled, an 8-bit adder is utilized to perform the binary addition between the original exponent and the binary sequence of i obtained by the quality control logic for quantization (i.e., enlarge the data by 2^i times); then the mapping hardware compresses the quantized exponent (Table 2) to further reduce the data size. Finally, the approximated data is sent to the packet encoder for transmission. Note that once the data arrives for neuron computation, the compressed exponent is decompressed according to Table 2, and a few bits with 0 are padded to the mantissa to recover the format of the quantized data back into the standard 32-bit floating-point format for subsequent computation.

4. Implementation of the Approximate Communication Technique (ACT)

An architecture based on hardware-software co-design is proposed in this section to implement the approximate communication technique (ACT) for image classification applications; the proposed implementation includes a software interface and

Table 3: Approximate Instructions.

Core Function	Instruction Function	Original Instruction	Approximate Instruction
Pre-processing Core	Load/Store Image Pixel	*mov dist, src*	*amov dist, src*
Model Inference Core	Load/Store Fully Connected Layer	*ld dist, src*	*ald dist, src*

an architectural design. The software interface is designed to identify the variables that need to be monitored or approximated during image classification. The network interfaces in the heterogeneous architecture are augmented with data approximation and quality control.

4.1. Software Interface for Approximate Communication

ACT approximates pixels in the images when the raw image is converted by the pre-processing cores. Also, ACT quantizes the inputs and parameters when the inference cores process the fully connected layers. Hence, ACT monitors and approximates the pixels, inputs, and parameters when the image classification application is executed on the heterogeneous architecture. Two specialized instructions are developed to identify these variables in the source code and the on-chip communication (Table 3). When the application designer programs the pre-processing cores, the variables (which store the images) are separately annotated in the application. For example, if the pre-processing cores are X86 CPUs, once the program is compiled into X86 instructions, then the load-and-store of an image pixel (*mov dist*, *src*) is replaced with (*amov dist*, *src*) for the network interface to identify the image pixels, that can be approximated. Similarly, the parameters and the inputs of the fully connected layer are annotated using specialized instructions (*alddist*, *src*). During the execution of an application, these new instructions allow the network interface to identify these variables in the requests or replies; for example, when the *amov* instruction is executed, the network interface identifies the data of the requested packet that can be approximated and apply data approximation prior to transmission.

4.2. Architecture Design of Approximate Communication

Fig. 10 shows the architecture design of the proposed approximate communication technique. The network interfaces (NIs) for the pre-processing cores, model inference cores, shared cache, and memory controller are augmented with specific hardware for approximation and recovery. Since the approximation logic needs to handle different data at different nodes, the approximation and recovery logics are specifically designed according to the functionality of the node, such as pre-processing or model inference.

4.2.1. Approximate Network Interface (Pre-processing Cores)

The proposed approximate network interface consists of approximation and recovery logics specifically designed for image pre-processing. To support the ACT-P proposed in Section 3.1, the data approximation logic approximates image pixels according to the contrast reduction mode. Since images

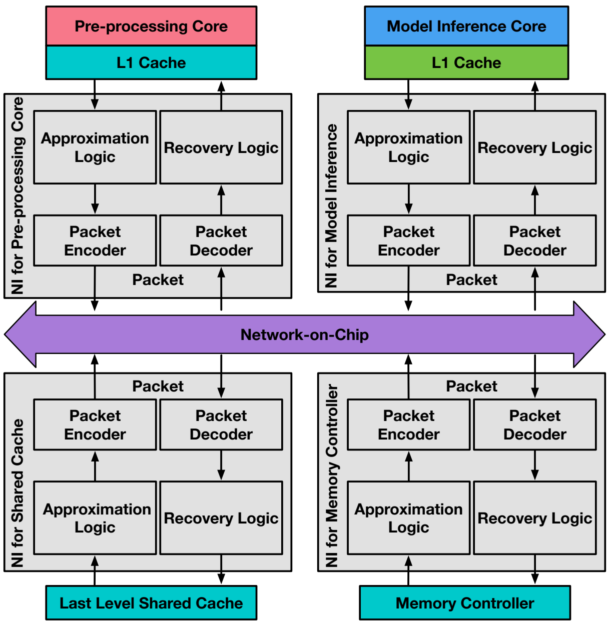


Figure 10: Architecture for the Approximate Communication Technique (ACT).

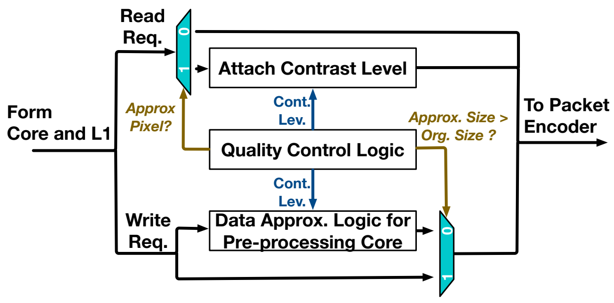


Figure 11: Approximation Logic for Pre-processing Core.

must be processed by the pre-processing core, the write requests and read replies carry image pixels and data in these packets can be approximated.

Fig. 11 shows the proposed approximation logic for the pre-processing core; the approximation logic includes the data approximation logic and the quality control logic to adjust the image contrast.

- The design of the data approximation logic for a pre-processing core is described in Section 3.1.2; for clarity, only the control signal for the quality control logic is shown in Fig. 11.
- The quality control logic provides the contrast reduction level (Cont. Lev.) according to the contrast reduction mode if a packet contains image pixels.

The quality control logic monitors the write requests. If the write requests contain raw images, then the quality control logic instructs the data approximation logic to approximate the requests according to the current contrast reduction mode. The proposed data approximation logic for image pre-processing supports 8 contrast reduction models (0 to -158), so 3 bits are used to represent the contrast reduction mode. If the write request cannot be approximated, the data approximation logic applies base-delta compression without contrast reduction (Mode

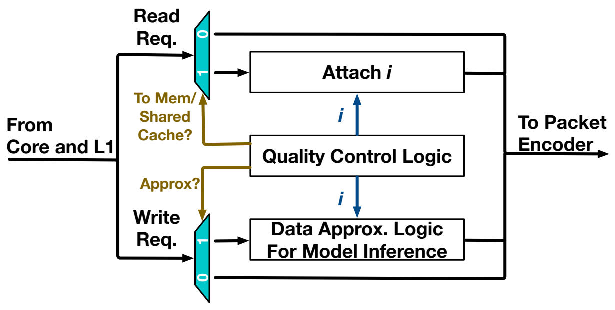


Figure 12: Approximation Logic for Model Inference Core.

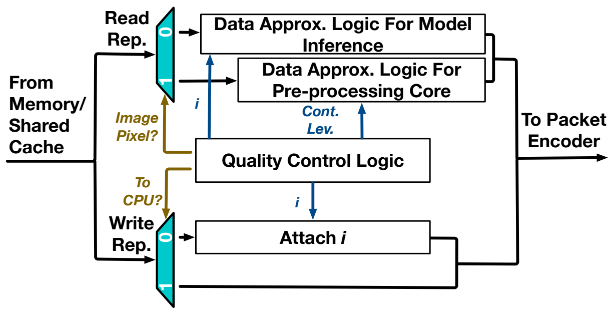


Figure 13: Approximation Logic for Memory Controller and Shared Cache.

0). Then, the quality control logic checks the length of the write requests. If the length is larger than the original write request ($Approx.\,Size > Org.\,Size$), the original request is sent to the packet encoder; once the memory or shared cache has received the data, a write reply is sent back to the core to confirm a successful memory write.

During the image load, the quality control logic attaches the information of contrast reduction mode (3 bits) to the read requests. Once the read reply packet arrives at the core, the data recovery logic recovers the data into its original form if the packet is compressed. Otherwise, the data recovery logic directly sends the read reply to the core. The data recovery logic for the pre-processing cores decompresses the data by adding the delta back to the base.

4.2.2. *Approximate Network Interface (Model Inference Cores)*

The proposed approximate network interface incorporates the approximation logic and recovery logic for model inference cores. Since the core directly loads and stores data from/to memory or shared cache, the read and write requests are generated by the node and send to the memory controller or shared cache. To support ACT-I proposed in Section 3.2, the data approximation logic monitors the write requests and read replies to update the dynamic range of the parameters and the inputs for the fully connected layer.

Fig. 12 shows the proposed approximation logic for the model inference. The quality control logic monitors all requests and replies to update i for the inputs; it also controls two demultiplexers and the data approximation logic. Since the destination of the write request could be another node for model inference or a memory controller or a shared cache, i (monitored at a specific node) can be the dynamic range of a section of the inputs for the fully connected layer. To find the dynamic range of the inputs for the entire layer, the following procedure is proposed.

(i) The quality control logic attaches i of the inputs to the read request packet if the destination of the packet is the memory controller or shared cache.

(ii) The quality control logic constantly monitors the i of the write reply packets from the memory controller or shared cache. If the received i is smaller than the current i, the value of i for the inputs in the current node is updated.

Therefore, the i in the memory controller and shared cache node has the dynamic range of all the inputs when the core loads the data. When the core stores the result, the i in each node is updated with the i in the memory controller and shared cache.

As a model inference core needs to fetch images, parameters, and inputs, the data recovery logic contains two decompression functions. The decompression function for images is the same function used in the pre-processing core. The decompression function for the parameters and inputs recovers the data based on Table 2, and a few bits (of values 0's) are padded to the mantissa to recover the format of the quantized data back into the standard 32-bit floating-point format for subsequent computation.

4.2.3. *Approximate Network Interface (Memory Controller)*

The proposed approximate network interface incorporates the approximation logic and recovery logic for the memory controller. Since the memory controller handles requests from both pre-processing and model inference cores, this interface performs data approximation and recovery functions for both tasks. Also, the network interface carries the quality control logic for both pre-processing and model inference.

Fig. 13 shows the approximation logic for the memory controller. The approximation logic consists of the data approximation and quality control logic. The quality control logic monitors the read request packets for the i value from the node for inputs. If the i value is smaller than the value stored in the quality control logic, the stored i is updated. The updated i is attached to the write replies to update i stored in the network interface at the node for model inference. The quality control logic also monitors the read request packets for receiving the contrast level for the read reply packet approximation. When the read reply has the data for image pre-processing or model inference, the corresponding data approximation logic is activated to approximate the data based on the contrast level or i. Similar to the quality control logic in the pre-processing core, the quality control logic checks the length of the read reply to the pre-processing core; if the length is greater than the original read reply after base-delta compression, the original reply is sent to the packet encoder.

Since the traffic contains the pixels, model parameters, and inputs, the data recovery logic has the recovery functions for both model inference and pre-processing.

4.2.4. *Approximate Network Interface (Shared Cache)*

The proposed approximate network interface incorporates the approximation and recovery logic for the shared cache. As the shared cache handles requests from pre-processing cores and model inference cores [7], the approximation and recovery logics are used to approximate and recover image pixels,

Table 4: Simulation Environment

Heterogeneous Architectures	**CPU/NDLA**	**ASIC/ACC**
Pre-Processing Cores	X86 CPU * 8	ASIC * 7 [9]
Model-Inference Cores	NVIDIA Deep Learning Accelerator(NDLA) * 28 [52]	CNN Processor * 7 [9]
NoC Parameter	Network type: Garnet Topology: 6 × 6 2D mesh Data packet size without approximation: 5 flits Link width: 128 bits Routing algorithm: X-Y routing	
System Parameter	32 kB L1 instruction cache 32 kB L1 data cache 8-bank fully shared 16 MB L2 cache	
Data Set	ImageNet Large Scale Visual Recognition Challenge [2]	
Approximate Communication Techniques	Approximate Communication Framework(ACF) [26, 38]; Approx-NoC [27]; AxBA [31]; Proposed Technique	

model parameters, and inputs. Thus, the approximate network interface for shared cache uses the same design as the memory controller (Fig. 13).

5. Evaluation

In this section, the performance of the approximate communication technique (ACT) is evaluated by using the SMAUG [7] simulator. The SMAUG simulation model is modified to support the ACT and heterogeneous architectures for image classification. Table 4 shows the settings for the SMAUG simulator; the hardware for data approximation, data recovery, and quality control is implemented in the network interface. Two heterogeneous architectures (i.e., CPU/NDLA [7, 8] and ASIC/ACC [9]) are modified for image classification to show the wide applicability and effectiveness of the proposed technique. The CPU/NDLA is based on Simba [8], and the ASIC/NDLA is based on DNPU [9]. The CPU/NDLA system contains 8 X86 CPUs as pre-processing cores and 28 NDLA cores as model-inference cores. The ASIC/ACC system contains 7 ASIC pre-processing cores, an X86 CPU as controller, and 7 CNN processors. Each CNN processor includes an aggregation core and 3 convolution cores. All the cores in the two architectures are connected using 6×6 2D mesh NoC. Table 5 shows the executed image classification applications with their classification accuracy [3, 4, 5, 6, 40, 41, 42, 43, 44, 45, 46, 47]. These applications achieve the best accuracy on the ImageNet large-scale visual recognition challenge [2] from 2012 to 2020.

We evaluate the proposed technique by comparing it with approximate communication framework (ACF) [26, 38], Approx-NoC [32], AxBA [31], and the baseline (i.e., NoC with no approximation) from the communication efficiency perspective, which includes network latency and dynamic power consumption. The ACF includes a network interface packet approximation method using data truncation and a frequent pattern compression technique [38] with a learning-based quality management system [26]. Approx-NoC [27] includes an approximated frequent pattern compression technique and a quality control method that requires a designer to manually annotate the approximable values and their error tolerance. AxBA [31] includes a base-delta data approximation scheme with a quality control method requiring manual annotation.

5.1. Network Latency

The network latency is defined as the number of clock cycles elapsed between sending a packet at the source node and the successful delivery of the packet to the destination. Thus, the network latency includes the time of three procedures: packet generation at the source node, packet transmission in the network, and data extraction at the destination node. The packet generation process includes data approximation and packet encoding at the source network interface. Data extraction includes packet decoding and data recovery at the destination network interface. Next, ACT is compared with the baseline, ACF, Approx-NoC, and AxBA.

- **The Heterogeneous system with CPUs for image pre-processing and NDLA for model inference:** Fig. 14 shows the results for the network latency normalized with respect to the baseline; ACT achieves an average network latency reduction of 26% and 23% compared to the baseline and ACF, respectively. This occurs because image classification applications have limited tolerance to the relative error for a smaller reduction in data size compared to ACT. The largest network latency reduction achieved by ACT in the experiment is VGG11 (45% reduction compared to the baseline), while the smallest network latency improvement is obtained for EffcientNet B7 (14% reduction compared to baseline).

- **The Heterogeneous system with ASIC for image pre-processing and CNN processor for model inference:** Fig. 15 shows the results for the average network latency normalized with respect to baseline for the heterogeneous system which uses ASIC for image pre-processing and CNN processor for model inference. The ACT achieves an average network latency reduction of 22% compared to the baseline. Compared to the heterogeneous system that uses CPUs for pre-processing, the ACT achieves different improvement in network latency when ASICs are used for pre-processing. This occurs due to the better pre-processing data flow. The CPUs' cache coherence protocol incurs in more traffic injected into the network, whereas each ASIC pre-processing core is designed to directly load images from shared cache or memory. Therefore, ASIC has fewer packets carrying image pixels; thus, more packets can be approximated for the CPUs, leading to more improvement in network latency than for ASIC.

Compared to the baseline, existing approximate communication techniques (e.g., Approx-NoC, AxBA, and ACF) achieve marginal improvement in network latency (less than 5% on average), as these techniques only rely on the relative error to approximate data. As a result, existing techniques miss the opportunity of data approximation for image classification applications; however, ACT can achieve a significant latency reduction due to the dual approximate communication scheme. Moreover, the proposed technique significantly reduces the network latency when the model frequently uses the fully connected layer and can tolerate a significant image contrast loss. For example, Fig. 16 shows the size of the fully connected layer in the image classification models. VGG11 uses 86% of the data, which includes inputs and parameters for the fully connected layers. As Table 5 shows that VGGs can tolerate 68 levels of contrast reduction with minimal accuracy loss, then the combined effect of two packet approximation mechanisms leads to a high packet compression rate (2.4 as shown in Fig. 17) when VGG11 is executed on the heterogeneous system with ACT. The high compression rate leads to a significant improvement in network latency; however, 3% of the data in EfficientNet B7 (EffNet B7 in Fig. 16) is occupied by the fully connected layer and can tolerate 23 levels of contrast reduction. The ACT yields 1.7 in compressing rate for CPU/NDLA heterogeneous architecture; therefore, the proposed technique achieves the smallest network latency improvement when EfficientNet B7 is executed on the heterogeneous system.

5.2. *Dynamic Power Consumption*

Dynamic power includes the power consumed by the switching activity for all transistors in the NIs and routers. For all on-chip communication, the results are normalized with respect to the baseline. Fig. 18 shows the dynamic power consumption for the CPU/NDLA heterogeneous system. ACT achieves an average dynamic power reduction of 29% and 24% compared with the baseline and ACF, respectively. The power reduction for the rest of the applications is between 48% and 17% compared to the baseline. Fig. 19 compares the dynamic power consumption for two heterogeneous systems with ACT. As ASIC improves data flow for pre-processing, the proposed technique achieves the best dynamic power saving for CPU/NDLA compared to an ASIC/ACC system. Depending on the size of the fully connected layer in each application and the tolerance for image contrast reduction, the dynamic power reduction compared to the baseline for ASIC/ACC ranges between 45% to 14%.

Fig. 20 shows the breakdown of the dynamic power saving of the ACT to the approximation for the model inference (ACT-I) and image pre-processing (ACT-P). Since the EfficientNet B7 is sensitive to contrast reduction, ACT-P achieves the smallest improvement to dynamic power saving (16%). However, the AdvProps can tolerate significant image contrast reduction; thus, ACT-P yields a 24% reduction in dynamic power. Overall, the average dynamic power reduction achieved by ACT-P is 21%, while the dynamic power saving for ACT-I is determined by the size of the fully connected layer. ACT-I contributes 30% in dynamic power reduction, as AlexNet contains the largest fully connected layer (96% as per Fig. 16) compared to other image classification applications. The ACT-I achieves the smallest contribution (1%) on EfficientNet B7 due to the small size of the fully connected layer (3%, as shown in Fig. 16). Overall, the average dynamic power reduction achieved by ACT-I is 9%.

Therefore, ACT achieves a significant improvement in dynamic power consumption for the two considered heterogeneous systems due to the effective packet approximation. The technique can significantly reduce packet size using the proposed data approximation mechanisms; thus, less activity is observed in the NoC, leading to significant dynamic power reduction.

5.3. *Accuracy Loss*

Fig. 21 shows the accuracy loss (i.e., loss of classification accuracy) for different image classification applications when ACT and ACF are applied to different heterogeneous systems. The classification accuracy is measured using the testing data set of ImageNet [2]. 512 randomly selected images from the testing data set are used for testing and setting the contrast reduction mode. The rest of the images are used to measure the accuracy loss of the application. The accuracy loss for all applications is less than 0.99% across all considered heterogeneous systems for ACT. However, ACF, which utilizes a learning-based quality control method [26], has a significantly higher quality loss compared to ACT. The highest accuracy loss (2.3%) is observed when EfficientNet B1 is executed on heterogeneous systems with ACF. This is mainly due to the low relative error tolerance of the image classification application. The highest accuracy loss (0.99%) is observed when EfficientNet B3 is executed with ACT. Moreover, the incurred accuracy loss is consistent across all systems, thus indicating that the proposed quality control mechanisms are effective in maintaining a low accuracy loss during approximate communication.

Fig. 22 shows the impact on the best and worst classification accuracies across all batches for the CPU/NDLA heterogeneous system with ACT when the batch size is 256. The average accuracy loss for the best classification accuracy (Highest Accuracy) is 0.8%, while the average accuracy loss of the worst classification accuracy (Lowest Accuracy) is 0.9%. Compared to the typical image classification accuracy (57% to 84%), the accuracy loss caused by approximate communication is extremely small.

5.4. *Overall System Performance Evaluation*

The ACT is implemented using Verilog to evaluate the area, static power, and latency; the entire system is synthesized with 32 *nm* technology using Synopsys Design Vision software. The synthesis results show that for each NI, the proposed hardware implementation incurs in an area of $4.79\mu m^2$. When the supply voltage is 1.0 *Volt*, the proposed technique incurs a power overhead of 1.7 *mW* for each NI. As for the latency, the approximation process and data recovery for pre-processing cores require one cycle each. Also, the approximation process and data recovery for the mode-inference cores require one cycle each. Therefore, two cycles are added for each write request and read reply. As for the overhead of this process, 5 iterations of testing are needed on average for the quality control mechanism to choose the appropriate contrast reduction mode. Compared to the overhead of several epochs of retraining required by CNN approximation techniques [33, 34, 35, 36, 37], testing is very efficient. Moreover, testing overhead can be further reduced for the proposed technique by using a small test data set or a predetermined contrast reduction mode.

Fig. 23 illustrates the speedup in execution time for image classification applications when ACT is implemented. The execution time is defined as the number of clock cycles elapsed

Table 5: Image Classification Applications

Application Name	Classification Accuracy	Contrast Reduction Mode	Application Name	Classification Accuracy	Contrast Reduction Mode
AlexNet [5]	56.55%	2 ($C = -45$)	ResNext50 [42]	77.8%	3 ($C = -68$)
VGG11 [6]	69.02%	3 ($C = -68$)	ResNext101 [42]	78.8%	2 ($C = -45$)
VGG13 [6]	69.93%	3 ($C = -68$)	NASNet-4A [45]	74.0%	6 ($C = -135$)
VGG16 [6]	71.59%	3 ($C = -68$)	AdvProp B0 [46]	77.6%	7 ($C = -158$)
VGG19 [6]	72.38%	4 ($C = -90$)	AdvProp B1 [46]	79.6%	7 ($C = -158$)
ShuffleNet X0.5 [4]	57.7%	2 ($C = -45$)	AdvProp B2 [46]	80.5%	7 ($C = -158$)
ShuffleNet X1.0 [4]	67.6%	2 ($C = -45$)	AdvProp B3 [46]	81.9%	7 ($C = -158$)
ZFNet [44]	61.6%	3 ($C = -68$)	AdvProp B4 [46]	83.3%	7 ($C = -158$)
GoogleNet [40]	69.78%	5 ($C = -113$)	AdvProp B5 [46]	84.3%	7 ($C = -158$)
DensNet121 [3]	74.65%	6 ($C = -135$)	AdvProp B6 [46]	84.8%	7 ($C = -158$)
DensNet161 [3]	76.00%	7 ($C = -158$)	AdvProp B7 [46]	85.2%	7 ($C = -158$)
DensNet169 [3]	77.20%	7 ($C = -158$)	EfficientNet B0 [47]	76.3%	3 ($C = -68$)
DensNet201 [3]	77.65%	6 ($C = -135$)	EfficientNet B1 [47]	78.8%	3 ($C = -68$)
ResNet18 [41]	69.76%	3 ($C = -68$)	EfficientNet B2 [47]	79.8%	3 ($C = -68$)
ResNet34 [41]	73.30%	3 ($C = -68$)	EfficientNet B3 [47]	81.1%	4 ($C = -90$)
ResNet50 [41]	76.15%	3 ($C = -68$)	EfficientNet B4 [47]	82.6%	3 ($C = -68$)
ResNet101 [41]	77.37%	2 ($C = -45$)	EfficientNet B5 [47]	83.3%	2 ($C = -45$)
ResNet152 [41]	78.31%	2 ($C = -45$)	EfficientNet B6 [47]	84.0%	3 ($C = -68$)
Inception V3 [43]	81.23%	2 ($C = -45$)	EfficientNet B7 [47]	84.4%	1 ($C = -23$)

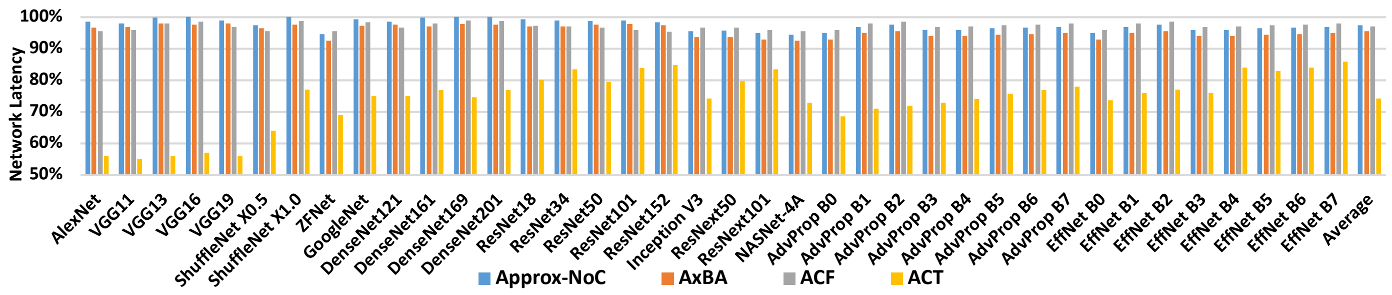


Figure 14: Network Latency for CPU/NDLA Heterogeneous System (Normalized to the Baseline).

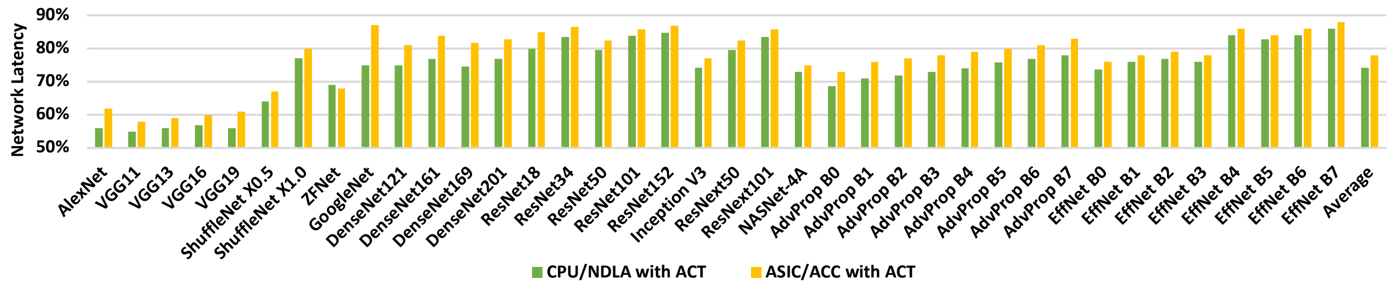


Figure 15: Network Latency for Two Heterogeneous Systems with ACT. The results of CPU/NDLA and ASIC/ACC with ACTs are normalized with the corresponding baseline systems.

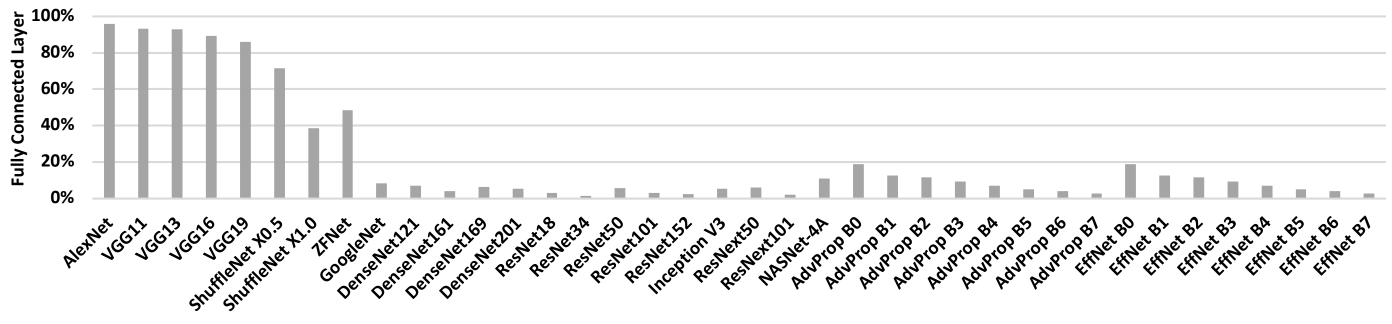


Figure 16: The Size of Fully Connected Layer in The Image Classification Models.

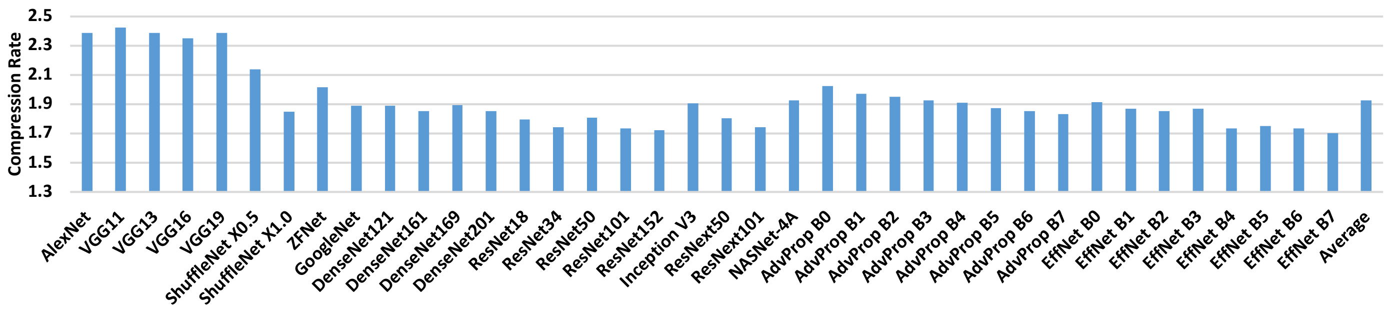


Figure 17: The Compression Rate of the ACT for CPU/NDLA Heterogeneous Systems.

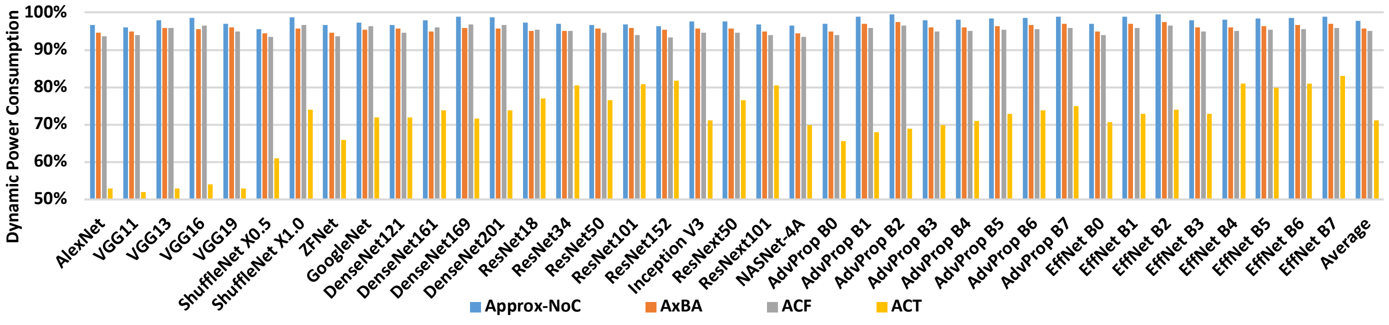


Figure 18: Dynamic Power Consumption for CPU/NDLA Heterogeneous System (Normalized to The Baseline).

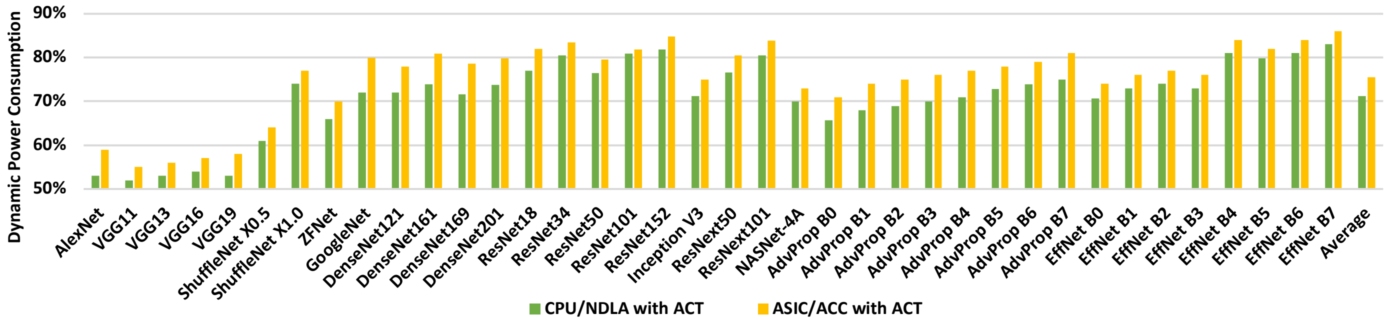


Figure 19: Dynamic Power Consumption for Two Heterogeneous Systems. The results of CPU/NDLA and ASIC/ACC with ACTs are normalized with the corresponding baseline systems.

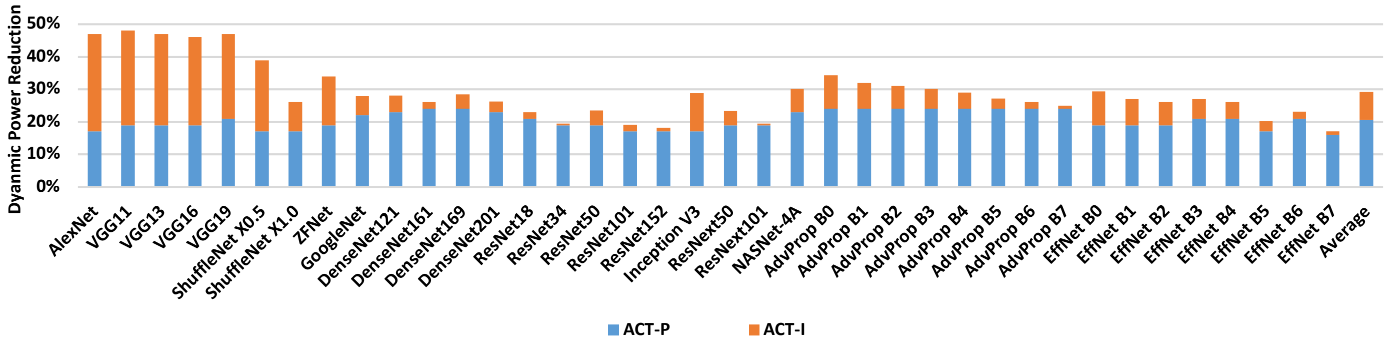


Figure 20: The Breakdown of the Dynamic Power Reduction on Baseline CPU/NDLA for ACT-P and ACT-I.

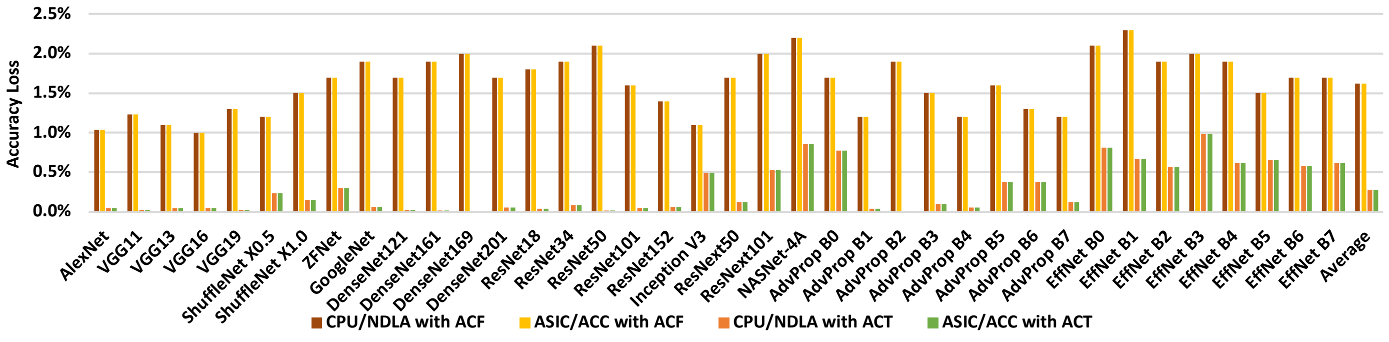


Figure 21: Accuracy Loss for Image Classification Applications.

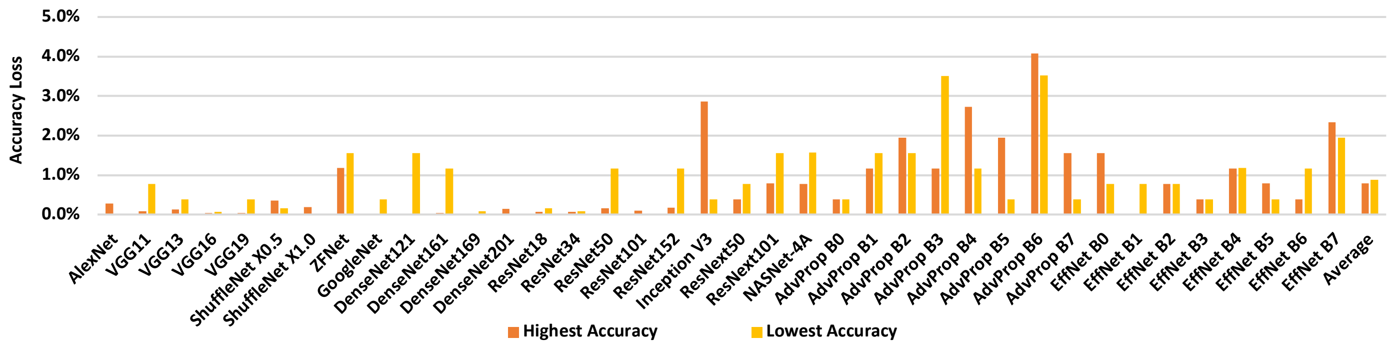


Figure 22: Impact on the Highest and Lowest Accuracy of All the Batches for CPU/NDLA Heterogeneous Architecture with ACT.

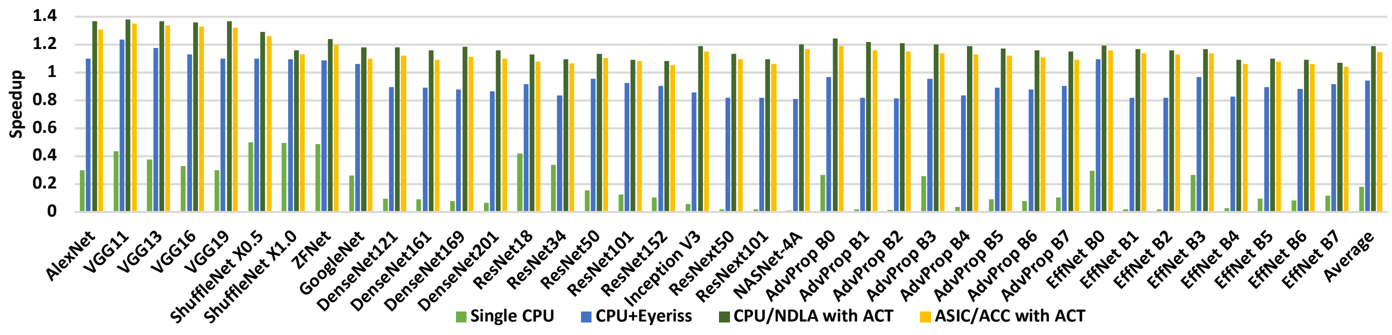


Figure 23: Speedups in Execution Time for Image Classification Applications. Single CPU and CPU+Eyeriss are normalized to baseline CPU/NDLA.

between the CPU receiving the image classification request and when the result is computed. A single-core CPU and CPU+ Eyeriss architectures are added to the comparison to show the need of using heterogeneous architecture for image classification. Eyeriss [53] is a standalone machine-learning accelerator that relies on an off-chip system interconnect (i.e., PCI Express) to communicate with CPUs. The results of single CPU and CPU+Eyeriss are normalized to the baseline CPU/NDLA. Due to the high latency in off-chip communication, CPU+Eyeriss is 5% slower on average compared to the baseline CPU/NDLA heterogeneous system. The improvement in the execution time for the ACT ranges from 9% to 38% for the two heterogeneous architectures; especially, the execution times for VGG11 are reduced by 38% and 35% for CPU/NDLA and ASIC/ACC with ACTs, respectively.

Therefore, ACT improves system performance for the two heterogeneous architectures by implementing the data approximation mechanism in the on-chip network.

5.5. *Sensitivity Study*

Fig. 24 shows the accuracy loss of the image classification applications when the threshold for accuracy loss is changing from 1% to 7%. According to Section 3.2 and Fig. 21, the type of pre-processing or model-inference core (e.g., NDLA or CNN Processor) used in a heterogeneous architecture has no effect on the classification accuracy loss. Thus, the experiment on the sensitivity analysis is based on the CPU/NDLA with ACT, as shown in Fig. 24 . If the threshold accuracy loss is more than 1%, then the approximation mechanism incurs more than a 1%

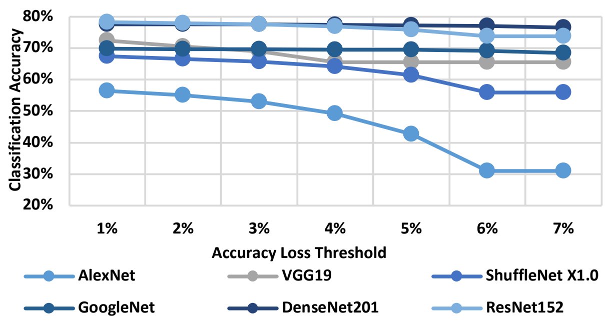


Figure 24: Image Classification Accuracy V.S. Accuracy Loss Threshold.

accuracy loss for the applications that are sensitive to image contrast reduction. For example, AlexNet has a 1.5% accuracy loss when the quality loss threshold is 2%. Thus, 1% is chosen to be the threshold for the quality control mechanism for the image pre-processing to maintain current accuracy.

6. Conclusion

In this work, we have proposed an approximate communication technique (ACT) to enhance on-chip communication efficiency for image classification applications. The proposed technique leverages the error tolerance of image classification applications to enhance communication efficiency during the execution of an application. ACT-P and ACT-I are developed for pre-processing and inference, respectively, so reducing the transmitted data while maintaining the image classification accuracy. Novel approximate network interfaces for the inference core, pre-processing core, memory controller, and shared cache have been proposed to implement ACT in NoCs. Compared to existing approximate communication techniques [27, 31, 38], ACT significantly reduces the transmitted data by efficiently approximating image classification applications. Compared to existing CNN approximation techniques [33, 34, 35, 36, 37], ACT eliminates the retraining process, which is time and energy-consuming. The detailed evaluation shows that compared to the state-of-the-art approximate communication technique (ACF) [26, 38], the proposed approximate communication technique reduces dynamic power consumption and network latency by 24% and 23%, respectively, with less than 0.99% classification accuracy loss.

Acknowledgment

The work of S. Liu and F. Lombardi is supported by NSF grants CCF-1953961 and CCF-1812467. The work of Y. Chen and A. Louri is supported by NSF grants CCF-1812495 and CCF-1953980.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[3] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, arXiv:1608.06993 [cs].
URL http://arxiv.org/abs/1608.06993

[4] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, arXiv:1707.01083 [cs].
URL http://arxiv.org/abs/1707.01083

[5] A. Krizhevsky, One weird trick for parallelizing convolutional neural networks, arXiv:1404.5997 [cs].
URL http://arxiv.org/abs/1404.5997

[6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 [cs].
URL http://arxiv.org/abs/1409.1556

[7] S. L. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, D. Brooks, Smaug: End-to-end full-stack simulation infrastructure for deep learning workloads, ACM Transactions on Architecture and Code Optimization 17 (4) (2020) 39:1–39:26. doi:10.1145/3424669.

[8] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, C. T. Gray, B. Khailany, S. W. Keckler, Simba: Scaling deep-learning inference with multi-chip-module-based architecture, MICRO '52, Association for Computing Machinery, New York, NY, USA, 2019, p. 14–27. doi:10.1145/3352460.3358302.

[9] D. Shin, J. Lee, J. Lee, J. Lee, H.-J. Yoo, Dnpu: An energy-efficient deep-learning processor with heterogeneous multi-core architecture, IEEE Micro 38 (5) (2018) 85–93. doi:10.1109/MM.2018.053631145.

[10] N. Chandramoorthy, G. Tagliavini, K. Irick, A. Pullini, S. Advani, S. Al Habsi, M. Cotter, J. Sampson, V. Narayanan, L. Benini, Exploring architectural heterogeneity in intelligent vision systems, 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, pp. 1–12. doi:10.1109/HPCA.2015.7056017.

[11] N. Bohm Agostini, S. Dong, E. Karimi, M. Torrents Lapuerta, J. Cano, J. L. Abellán, D. Kaeli, Design space exploration of accelerators and end-to-end dnn evaluation with tflite-soc, 2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2020, pp. 10–19. doi:10.1109/SBAC-PAD49847.2020.00013.

[12] S. Venkataramani, A. Ranjan, S. Banerjee, D. Das, S. Avancha, A. Jagannathan, A. Durg, D. Nagaraj, B. Kaul, P. Dubey, A. Raghunathan, Scaledeep: A scalable compute architecture for learning and evaluating deep networks, ISCA '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 13–26. doi:10.1145/3079856.3080244.

[13] R. Iyer, S. Srinivasan, O. Tickoo, Z. Fang, R. Illikkal, S. Zhang, V. Chadha, P. Stillwell, S. E. Lee, Cogniserve: Heterogeneous server architecture for large-scale recognition, IEEE Micro 31 (3) (2011) 20–31. doi:10.1109/MM.2011.37.

[14] L. Yang, W. Liu, W. Jiang, C. Chen, M. Li, P. Chen, E. H. M. Sha, Hardware-software collaboration for dark silicon heterogeneous many-core systems, Future Generation Computer Systems 68 (2017) 234–247. doi:10.1016/j.future.2016.09.012.

[15] A. Dehghani, A design flow for an optimized congestion-aware application-specific wireless network-on-chip architecture, Future Generation Computer Systems 106 (2020) 234–249. doi:10.1016/j.future.2020.01.001.

[16] T. Maqsood, K. Bilal, S. A. Madani, Congestion-aware core mapping for network-on-chip based systems using betweenness centrality, Future Generation Computer Systems 82 (2018) 459–471. doi:10.1016/j.future.2016.12.031.

[17] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, The raw microprocessor: a computational fabric for software circuits and general-purpose programs, IEEE Micro 22 (2) (2002) 25–35. doi:10.1109/MM.2002.997877.

[18] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-ghz mesh interconnect for a teraflops processor, IEEE Micro 27 (5) (2007) 51–61. doi:10.1109/MM.2007.4378783.

[19] H. Wang, L.-S. Peh, S. Malik, Power-driven design of router microarchitectures in on-chip networks, Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36., 2003, pp. 105–116. doi:10.1109/MICRO.2003.1253187.

[20] T. Yeh, P. Faloutsos, M. Ercegovac, S. Patel, G. Reinman, The art of deception: Adaptive precision reduction for area efficient physics acceleration, 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007), 2007, pp. 394–406. doi:10.1109/MICRO.2007.9.

[21] H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, Neural acceleration for general-purpose approximate programs, 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, 2012, pp. 449–460. doi:10.1109/MICRO.2012.48.

[22] Q. Xu, T. Mytkowicz, N. S. Kim, Approximate computing: A survey, IEEE Design Test 33 (1) (2016) 8–22. doi:10.1109/MDAT.2015.2505723.

[23] S. Mittal, A survey of techniques for approximate computing, ACM Computing Surveys 48 (4) (2016) 62:1–62:33. doi:10.1145/2893356.

[24] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, U. Karpuzcu, Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems, ACM Computing Surveys 51 (1) (2018) 1:1–1:32. doi:10.1145/3145812.

[25] Y. Chen, A. Louri, An online quality management framework for approximate communication in network-on-chips, ICS '19, Association for Computing Machinery, Phoenix, Arizona, 2019, p. 217–226. doi:10.1145/3330345.3330365.

[26] Y. Chen, A. Louri, Learning-based quality management for approximate communication in network-on-chips, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 39 (11) (2020) 3724–3735. doi:10.1109/TCAD.2020.3012235.

[27] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, E. J. Kim, Approx-noc: A data approximation framework for network-on-chip architectures, ISCA '17, Association for Computing Machinery, Toronto, ON, Canada, 2017, p. 666–677. doi:10.1145/3079856.3080241.

[28] L. Wang, X. Wang, Y. Wang, Abdtr: Approximation-based dynamic traffic regulation for networks-on-chip systems, 2017 IEEE International Conference on Computer Design (ICCD), 2017, pp. 153–160. doi:10.1109/ICCD.2017.31.

[29] M. F. Reza, P. Ampadu, Approximate communication strategies for energy-efficient and high performance noc: Opportunities and challenges, GLSVLSI '19, Association for Computing Machinery, New York, NY,

USA, 2019, p. 399–404. doi:10.1145/3299874.3319455.

[30] V. Fernando, A. Franques, S. Abadal, S. Misailovic, J. Torrellas, Replica: A wireless manycore for communication-intensive and approximate data, ASPLOS '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 849–863. doi:10.1145/3297858.3304033.

[31] J. R. Stevens, A. Ranjan, A. Raghunathan, Axba: an approximate bus architecture framework, ICCAD '18: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ACM, San Diego California, 2018, pp. 1–8. doi:10.1145/3240765.3240782.

[32] Y. Chen, M. F. Reza, A. Louri, Dec-noc: An approximate framework based on dynamic error control with applications to energy-efficient nocs, 2018 IEEE 36th International Conference on Computer Design (ICCD), 2018, pp. 480–487. doi:10.1109/ICCD.2018.00078.

[33] S. F. Dodge, L. J. Karam, Quality robust mixtures of deep neural networks, IEEE Transactions on Image Processing 27 (11) (2018) 5553–5562. doi:10.1109/TIP.2018.2855966.

[34] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, H. Yang, Going deeper with embedded fpga platform for convolutional neural network, FPGA '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 26–35. doi:10.1145/2847263.2847265.

[35] L. Deng, G. Li, S. Han, L. Shi, Y. Xie, Model compression and hardware acceleration for neural networks: A comprehensive survey, Proceedings of the IEEE 108 (4) (2020) 485–532. doi:10.1109/JPROC.2020.2976475.

[36] T.-J. Yang, Y.-H. Chen, V. Sze, Designing energy-efficient convolutional neural networks using energy-aware pruning, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5687–5695.

[37] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, arXiv:1510.00149 [cs].
URL http://arxiv.org/abs/1510.00149

[38] Y. Chen, A. Louri, An approximate communication framework for network-on-chips, IEEE Transactions on Parallel and Distributed Systems 31 (6) (2020) 1434–1446. doi:10.1109/TPDS.2020.2968068.

[39] S. Xiao, X. Wang, M. Palesi, A. K. Singh, T. Mak, Acdc: An accuracy- and congestion-aware dynamic traffic control method for networks-on-chip, 2019 Design, Automation Test in Europe Conference Exhibition (DATE), 2019, pp. 630–633. doi:10.23919/DATE.2019.8715189.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9. doi:10.1109/CVPR.2015.7298594.

[41] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[42] S. Xie, R. Girshick, P. Dollar, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, 2017, pp. 5987–5995. doi:10.1109/CVPR.2017.634.

[43] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 2818–2826. doi:10.1109/CVPR.2016.308.

[44] M. D. Zeiler, R. Fergus, Visualizing and Understanding Convolutional Networks, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision – ECCV 2014, Vol. 8689, Springer International Publishing, Cham, 2014, pp. 818–833. doi:10.1007/978-3-319-10590-1_53.

[45] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning Transferable Architectures for Scalable Image Recognition, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, UT, 2018, pp. 8697–8710. doi:10.1109/CVPR.2018.00907.

[46] C. Xie, M. Tan, B. Gong, J. Wang, A. Yuille, Q. V. Le, Adversarial examples improve image recognition, arXiv:1911.09665 [cs].
URL http://arxiv.org/abs/1911.09665

[47] M. Tan, Q. Le, Efficientnet: Rethinking model scaling for convolutional neural networks, International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems 32.

[49] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, Tensorflow: A system for large-scale machine learning, 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[50] S. Dodge, L. Karam, Understanding how image quality affects deep neural networks, 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016, pp. 1–6. doi:10.1109/QoMEX.2016.7498955.

[51] W. Kahan, Ieee standard 754 for binary floating-point arithmetic (1996) 30.

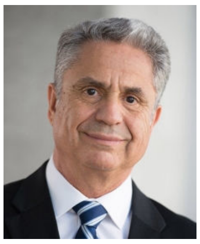
[52] Nvidia deep learning accelerator.
URL http://nvdla.org/

[53] Y.-H. Chen, J. Emer, V. Sze, Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks, ACM SIGARCH Computer Architecture News 44 (3) (2016) 367–379. doi:10.1145/3007787.3001177.

Yuechen Chen received his master's degree in Electrical Engineering from the George Washington University, Washington, District of Columbia, in 2016. He is currently pursuing Ph.D. with the Department of Electrical and Computer Engineering at the George Washington University. His research interests include approximate computing and NoCs.

Shanshan Liu received the M.Sc. degree and Ph.D. degree in Microelectronics and Solid-State Electronics from Harbin Institute of Technology, Harbin, China, in 2012 and 2018, respectively. She is currently a Post-doctoral faculty researcher with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA. She is a Guest Editor for the IEEE Transactions on Circuits and Systems-I and the IEEE Transactions on Emerging Topics in Computing, an Associate Editor for the IEEE Transactions on Nanotechnology, and a technical program committee member for the IEEE International Symposiums on DFT and IOLTS. Her research interests include fault tolerance design in high performance computer systems, VLSI design, dependable machine learning, stochastic computing, error correction codes.

Fabrizio Lombardi received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, U.K., in 1977, the master's degree in microwaves and modern optics and the Diploma degree in microwave engineering from the Microwave Research Unit, University College London, in 1978, and the Ph.D. degree from the University of London in 1982. He is currently the International Test Conference (ITC) Endowed Chair Professorship with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books. He was the Editor-In-Chief of the IEEE TRANSACTIONS ON COMPUTERS from 2007 to 2010 and the inaugural Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING from 2013 to 2017, IEEE TRANSACTIONS ON NANOTECHNOLOGY from 2014 to 2019. He is currently the Vice President for Publications of the IEEE Computer Society, the 2021 President-elect of the IEEE Nanotechnology Council and a member of the IEEE Publication Services and Products Board.



Ahmed Louri is the David and Marilyn Karlgaard Endowed Chair Professor of Electrical and Computer Engineering at the George Washington University, which he joined in August 2015. He is also the Director of the High Performance Computing Architectures and Technologies Laboratory. Professor Louri received the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California in 1988. From 1988 to 2015, he was a professor of Electrical and Computer Engineering at the University of Arizona. From 2010 to 2013, he served as a program director in the National Science Foundation's (NSF) Directorate for Computer and Information Science and Engineering. Professor Louri conducts research in the broad area of computer architecture and parallel computing, with emphasis on interconnection networks and network on chips for multicores, and the use of machine learning techniques for energy-efficient, reliable, high-performance and secure many-core architectures and accelerators. Professor Louri was recently selected to be the recipient of the IEEE Computer Society 2020 Edward J. McCluskey Technical Achievement Award, "for pioneering contributions to the solution of on-chip and off-chip communication problems for parallel computing and manycore architectures". Professor Louri is a Fellow of IEEE, and is currently serving as the Editor-in-Chief of IEEE TRANSACTIONS ON COMPUTERS.