Approximate Network-on-Chips with Application to Image Classification

Yuechen Chen
The George Washington University
Email: yuechen@gwu.edu

Ahmed Louri
The George Washington University
Email: louri@gwu.edu

Shanshan Liu
New Mexico State University
Email: ssliu@nmsu.edu

Fabrizio Lombardi
Northeastern University
Email: lombardi@ece.neu.edu

***Abstract*— Approximation is an emerging design methodology 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. 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. 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. Cycle-accurate simulation results show that ACT achieves 27% in network latency reduction and 28% in dynamic power reduction as compared to existing approximate communication techniques with less than 0.85% classification accuracy loss.**

***Keywords*— Image Classification, Network-on-Chips (NoCs), Approximation.**

I. Introduction

Image classification applications widely use deep convolutional neural networks (CNNs) and are deployed from cloud to edge computational frameworks for a variety of scenarios, such as search engines and self-driving cars [1], [2]. As the complexity of these applications and the resolution of images continue to increase, conventional homogeneous architectures (such as multicore CPU/GPU) are constrained due to excessive communication latencies and significant power dissipation [3]–[5]. To efficiently process these applications, heterogeneous architectures have been proposed with pre-processing and inference cores [3]–[8]. 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 [9]. Recent research [3], [10] 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 [11], [12]. Existing approximation techniques can be categorized as follows:

- Existing approximate communication techniques [13]–[17] 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 [18]–[22] reduce the size of the model using quantization or pruning. However, these techniques do not specifically target image classification. Moreover, as quantizing and 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 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 schemes: an approximate communication for the pre-processing cores (ACT-P) and an approximate communication for the inference cores (ACT-I). Each scheme 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.

- The proposed approximate communication technique (ACT) is utilized in the pre-processing cores (ACT-P) and the inference cores (ACT-I) to reduce network latency and dynamic power consumption for image classification applications by leveraging the error tolerance of the application.
- The ACT is implemented with software-hardware co-design.
- Performance evaluation results show that compared to the existing approximate communication techniques, ACT

978-1-6654-5408-7/22/$31.00 ©2022 IEEE

reduces network latency and dynamic power consumption by 27% and 28%, respectively, with less than 0.85% classification accuracy loss.

This paper is organized as follows. Section II presents a background for the proposed technique; Section III outlines the basic operational principles of the ACT. The implementation is presented in detail in Section IV, while Section V deals with its evaluation. Section VI concludes this manuscript.

II. Background

Approximation techniques are widely used to enhance the efficiency of image classification applications and CNNs [18]–[22]. Existing approximation techniques can be categorized into two types based on on-chip communication:

- 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 scheme.

A. Approximate Communication Techniques

Approximate communication is considered to be an effective approach to improve network performance when an application can tolerate errors [13]-[17]. With a reduced accuracy during communication, approximation techniques significantly reduce network latency and the power consumption for on-chip communication.

Fig. 1 shows an approximate communication NoC [13]-[17] implemented in a heterogeneous multicore system [3]-[8] with an L2 shared cache for CNN inference. Consider a cache miss during a memory load or store operation by a CPU for image pre-processing. 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. Then, the memory or shared cache uses a read reply packet to send the required data back to the core. When a cache miss occurs during a memory store operation, the data are incorporated into a write request packet and sent to the memory or shared cache through the NoC. After the memory or shared cache receives the data, a write reply is sent back to the core to confirm a successful memory write. The data approximation module in the network interface reduces the packet size by truncation or lossy compression according to the approximation information, which includes variable error tolerance and type (e.g., integer or floating-point). Various data approximation methods [14], [16], [17], [23] have been proposed to reduce the packet size according to the approximation information. 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.

B. CNN Approximation Techniques

Quantization and pruning methods are widely used for deep CNNs in image classification applications to reduce communication traffic and computation [20]. For example, in [22], the size of the deep neuron network is significantly reduced using quantization, pruning, and compression. Existing image classification applications [24]–[31] are implemented using Pytorch [32] and TensorFlow[33] frameworks, which

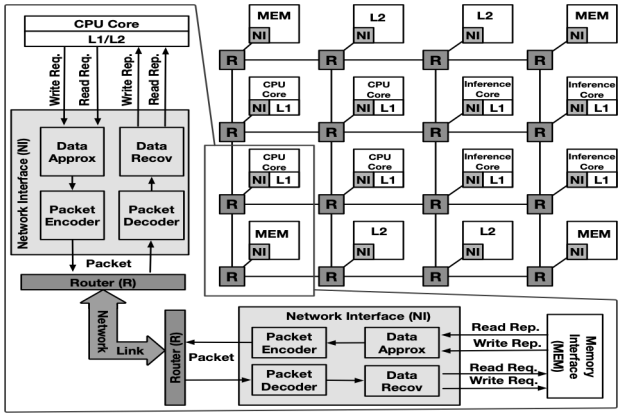

Fig. 1: Heterogeneous multicore architecture with an approximate communication NoC

support CNN quantization and pruning on generic inference cores (e.g., CPUs, GPUs, CNN Accelerators). However, existing model approximation techniques have two major limitations.

1. They are developed for generic CNN inference. The performance improvement methods are not designed specifically for image classification. Thus, system performance can be further improved with dedicated optimization techniques.
2. They require the model to be retrained or fine-tuned before classifying images, because these techniques incur 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 only a very limited impact on accuracy.

III. 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 [18], [19], [34]: the first type is image contrast reduction during 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.

1. 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.
2. 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*

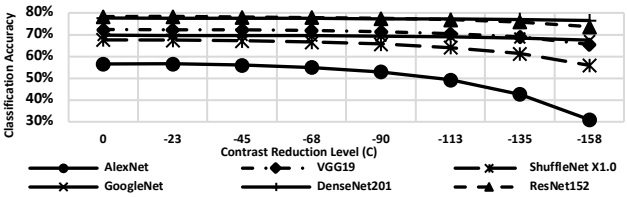

Fig. 2: Classification accuracy versus contrast reduction level (C).

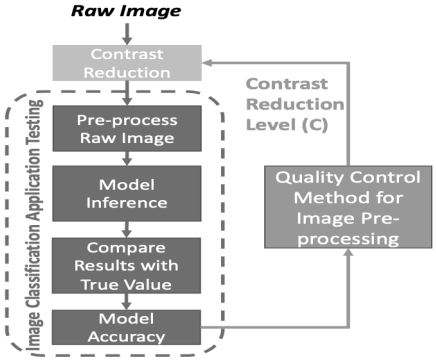

Fig. 3: Proposed quality control mechanism for image pre-processing.

approximation mechanism utilizes data quantization to reduce data size.

A. 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 [18], [34]. 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, 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.

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

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.

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

Fig. 2 shows the classification accuracy for a few widely-used image classification applications [2] 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. 2 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 [25] can tolerate 23 levels of contrast reduction, while VGG19 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.

1) Quality Control

A quality control mechanism for image pre-processing is utilized to maintain the accuracy of image classification. Fig. 3 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.

1. 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).
2. The model inference is then accomplished by fetching the processed data and the classification model.
3. 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%. The proposed quality control mechanism supports eight contrast reduction levels that are shown in the left column of Table I. Thus, the following novel procedure is proposed to determine the image contrast reduction level:

1. 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).

$$Classification\ Accuracy = \frac{Number\ of\ Correct\ Classifications}{Total\ number\ of\ Classificaitons} \quad (3)$$

   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 the expected answer (label).
2. The quality control mechanism gradually reduces the image contrast levels by choosing a contrast reduction level according to the left column of Table I until there is more than 1% loss (as threshold) in the classification accuracy compared to the estimated base accuracy in the next testing process.

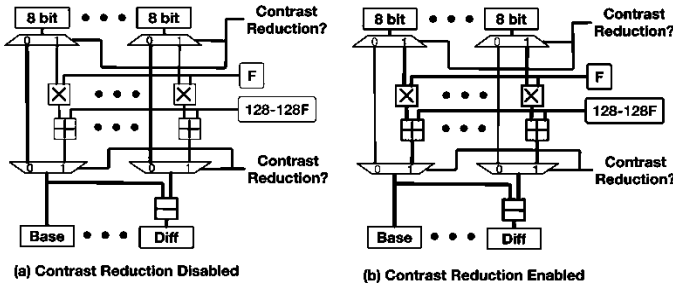

Fig. 4: Design of data approximation for image pre-processing.

TABLE I: RELATIONSHIPS BETWEEN QUALITY CONTROL SUPPORTED CONTRAST REDUCTION LEVEL (C) AND CONTRAST CORRECTION FACTOR (F).

Supported Contrast Reduction Level (*C*)	Contrast Correction Factor (*F*)
0	1
-23	0.835
-45	0.701
-68	0.581
-90	0.480
-113	0.388
-135	0.309
-158	0.236

3. The value prior to the last contrast reduction level is chosen for image classification. For example, if the classification error exceeds 1% at level -68, the lower consecutive level (-45 according to Table I) is selected for image classification. During image classification, the contrast reduction level is fixed and registered in the network interface.

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. 4 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 I shows the mapping of the conversion of the supported contrast reduction level (*C*) into the contrast correction factor (*F*). The first 8-bit data in a write request or read reply is chosen as the base; the remaining data is represented as the distance to the base. Fig. 4 (a) shows the approximation process when the image contrast reduction is deactivated (Contrast Reduction Level = 0). The data bypass the contrast reduction operation and is compressed with full accuracy. Fig. 4 (b) shows the approximation process when the image contrast reduction is activated.

B. Approximate Communication for Model Inference (ACT-I)

Existing work [21] has shown that the classification accuracy reduction in image classification applications is negligible after quantizing the parameters and activation of the fully connected layers (IEEE standard 32-bit floating-point) into 8-bit integers. As floating-point data type is widely used in image classification applications [24]-[31] 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).

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

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 are given as follows.

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

In Eq. (5), α and β are the minimum and maximum of the floating-point value, respectively. α_q and β_q are the minimum and maximum of the integer value (i.e., quantized floating-point value), respectively. c and d are the two variables that must be solved for the quantization process. Eq. (6) illustrates the solution for Eq. (5).

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

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. Therefore, a novel quality control mechanism is developed to estimate the range of inputs and parameters in the proposed scheme.

1) Quality Control

Fig. 5 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 [36] 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.

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

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

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).

$$i = log2(max(|\alpha|, |\beta|)) \tag{9}$$

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

TABLE II. 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

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.

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.

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.

The hardware design for the proposed data approximation mechanism for data quantization is illustrated in Fig. 6. 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 II) to further reduce the data size. Finally, the approximated data is sent to the packet encoder for transmission.

IV. IMPLEMENTATION OF THE APPROXIMATE COMMUNICATION TECHNIQUE (ACT)

An architecture based on hardware-software co-design is proposed in this section to implement ACT for image classification applications. The proposed implementation includes a software interface and an architectural design. The software interface is designed to identify the variables that need

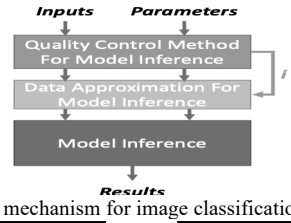

Fig. 5: Quality control mechanism for image classification model inference.

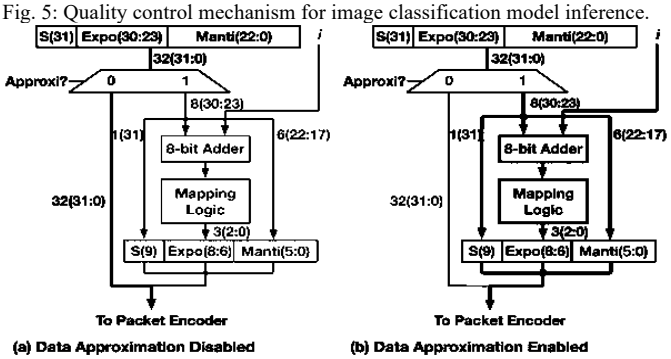

Fig. 6: Hardware design of data approximation for model inference.

to be monitored or approximated during image classification. The network interfaces in the heterogeneous architecture are augmented with data approximation and quality control.

A. Software Interface for Approximate Communication

ACT approximates pixels in the images when the pre-processing cores convert the raw image. 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. Specialized instructions are developed to identify these variables in the source code and the on-chip communication. During the execution of an application, these new instructions allow the network interface to identify these variables in the requests or replies.

B. Architecture Design of ACT

The ACT arguments the network interfaces (NIs) for the pre-processing cores, model inference cores, shared cache, and memory controller with specific hardware for approximation and recovery (Fig. 1). 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.

1) Approximate Network Interface (Pre-processing Cores)

To support the ACT-P, the data approximation logic approximates image pixels according to the contrast reduction level. Since images 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. 7 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 III.A.2. For clarity, only the control signal for the quality control logic is shown in Fig. 7. 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 level. If the write request cannot be approximated, the

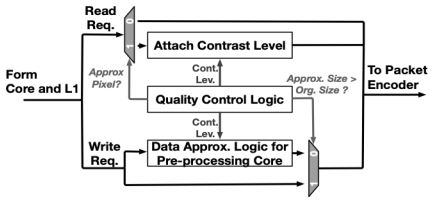

Fig. 7: Approximation logic for pre-processing cores.

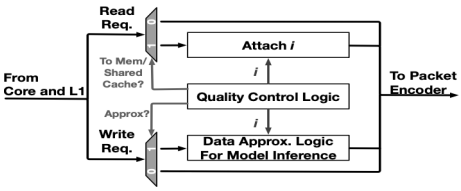

Fig. 8: Approximation logic for model inference cores.

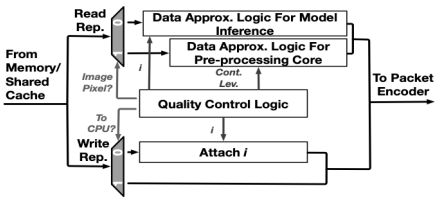

Fig. 9: Approximation logic for memory controllers and shared caches.

data approximation logic applies base-delta compression without contrast reduction (level 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.

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.

2) Approximate Network Interface (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 sent to the memory controller or shared cache. To support ACT-I, 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. 8 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. (1) 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. (2) 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.

As a model inference core needs to fetch images, parameters, and inputs, the data recovery logic contains two decompression functions to recover approximated data.

3) Approximate Network Interface (Memory Controller and Shared Cache)

Since the memory controller and shared cache handles requests from both pre-processing and model inference cores, this interface performs data approximation and recovery functions for both tasks.

Fig. 9 shows the approximation logic for the memory controller and shared cache. 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.

TABLE III: SIMULATION ENVIRONMENT.

Heterogeneous Architecture	**CPU/NDLA**
Pre-Processing Cores	X86 CPU * 8
Model-Inference Cores	NVIDIA Deep Learning Accelerator(NDLA) * 28 [37]
NoC Parameter	Network type: Garnet; Topology: 6 × 6 2D mesh; Data packet size: 5 Flits; Link width: 128 bits; Routing algorithm: X-Y routing; Flow Control: Wormhole Switching; Number of Router Pipeline Stage: 6
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]
Approximation Techniques	Approximate Communication Framework(ACF) [17]; Approx-NoC [14]; AxBA [16]; Proposed Technique

TABLE IV: IMAGE CLASSIFICATION APPLICATIONS

Name	Acc.	*C*	Name	Acc.	*C*
AlexNet [25]	56.55%	-45	DensNet169 [24]	77.20%	-158
VGG11 [26]	69.02%	-68	DensNet201 [24]	77.65%	-135
VGG13 [26]	69.93%	-68	ResNet101 [29]	77.37%	-45
VGG16 [26]	71.59%	-68	ResNet152 [29]	78.31%	-45
VGG19 [26]	72.38%	-90	NASNet-4A [30]	74.00%	-135
ShuffleNet X1.0 [27]	67.60%	-45	EfficientNet B0 [31]	76.30%	-68
GoogleNet [28]	69.78%	-113	EfficientNet B7 [31]	84.40%	-23

V. EVALUATION

In this section, the performance of the approximate communication technique (ACT) is evaluated by using the SMAUG [3] simulator. The SMAUG simulation model is modified to support the ACT for image classification. Table III shows the settings for the SMAUG simulator. The hardware for data approximation, data recovery, and quality control is implemented in the network interface. The CPU/NDLA is based on Simba [4], and all the cores in the system are connected using 6×6 2D mesh NoC. Table IV shows the executed image classification applications [24]-[31] with their original classification accuracy (Acc.) and the corresponding contrast reduction levels (C).

We evaluate the proposed technique by comparing it with approximate communication framework (ACF) [17], Approx-NoC [14], AxBA [16], and the baseline (i.e., NoC with no

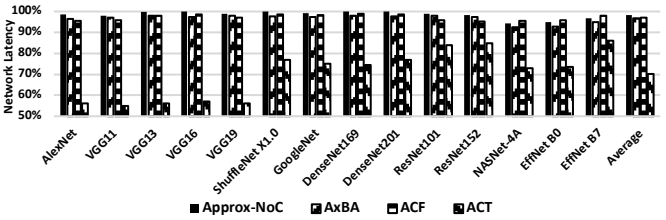

Fig. 10: Network latency for CPU/NDLA heterogeneous system (normalized to the baseline).

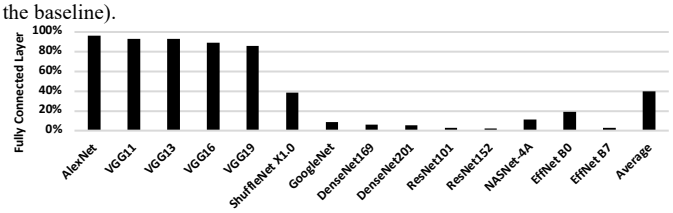

Fig. 11: The size of a fully connected layer in the image classification models.

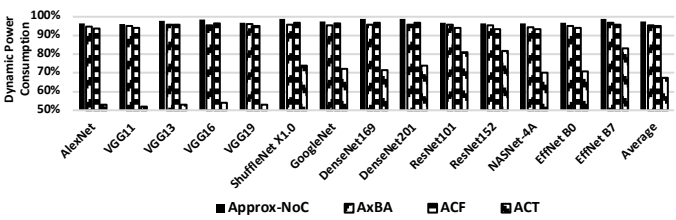

Fig. 12: Dynamic power consumption for CPU/NDLA heterogeneous system (normalized to the baseline).

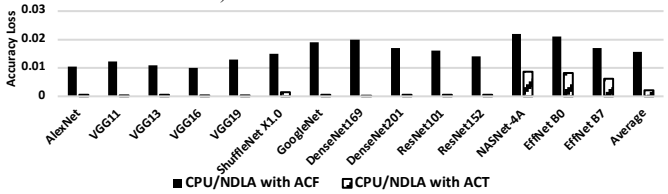

Fig. 13: Accuracy loss for image classification applications.

approximation) from the communication efficiency perspective including network latency and dynamic power consumption.

A. 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.

Fig. 10 shows the results for the network latency normalized with respect to the baseline. ACT achieves an average network latency reduction of 29% and 27% 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), while the smallest network latency improvement is obtained for EffcientNet B7 (14% reduction).

Compared to the baseline, existing approximate communication techniques 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. 11 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 IV shows that VGGs can tolerate -68 levels of contrast reduction ($C = -68$) with minimal accuracy loss, then the combined effect of two packet approximation mechanisms leads to a high reduction in packet size when VGG11 is executed on the heterogeneous system with ACT.

B. 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. 12 shows the dynamic power consumption for the CPU/NDLA heterogeneous system. ACT achieves an average dynamic power reduction of 32% and 28% 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. Therefore, ACT achieves a significant improvement in dynamic power consumption due to the effective packet approximation. The technique can significantly reduce packet size using the proposed data approximation mechanisms.

C. Accuracy Loss

Fig. 13 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 level. 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% for the ACT. However, ACF has a significantly higher quality loss compared to ACT. The highest accuracy loss (2.2%) is observed when NASNet-4A 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 (0.85%) is observed when NASNet-4A 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.

D. Overheads

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 μm^2. When the supply voltage is 1.0 Volt, the proposed technique incurs a static power overhead of 1.7 mW for each NI. For a 6×6 2D mesh NoC, the ACT modules occupy 1.7% of the total NoC area and consume 4.7% of the total static power consumption. 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. 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 level. Compared to the overhead of several epochs of retraining required by CNN approximation

techniques [18]-[22], 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 level.

VI. 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. Two approximate communication techniques are developed for pre-processing (ACT-P) and inference (ACT-I), respectively, thus reducing the transmitted data while maintaining the image classification accuracy. Novel approximate network interfaces for the pre-processing core, inference core, memory controller, and shared cache have been proposed to implement ACT in network-on-chips (NoCs). Compared to existing convolutional neural network (CNN) approximation techniques, ACT eliminates the retraining process, which is time and energy consuming. Compared to existing approximate communication techniques, ACT significantly reduces the transmitted data by efficiently approximating image classification applications. The detailed evaluation shows that compared to the state-of-the-art approximate communication techniques, the proposed approximate communication technique reduces dynamic power consumption and network latency by 28% and 27%, respectively, with less than 0.85% accuracy loss.

Acknowledgment

This research is supported by NSF grants CCF-1953961, CCF-1812467, CCF-1812495 and CCF-1953980.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, Art. no. 7553, May 2015.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-scale Hierarchical Image Database," in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[3] S. (Likun) Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, and D. Brooks, "SMAUG: End-to-End Full-Stack Simulation Infrastructure for Deep Learning Workloads," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, p. 39:1-39:26, Nov. 2020.

[4] Y. S. Shao *et al.*, "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture," in *Proceedings of the 52nd MICRO*, New York, NY, USA, Oct. 2019, pp. 14–27.

[5] D. Shin, J. Lee, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An Energy-Efficient Deep-Learning Processor with Heterogeneous Multi-Core Architecture," *IEEE Micro*, vol. 38, no. 5, pp. 85–93, Sep. 2018.

[6] N. Chandramoorthy *et al.*, "Exploring Architectural Heterogeneity in Intelligent Vision Systems," in *Proceedings of the 21st HPCA*, Feb. 2015, pp. 1–12.

[7] N. Bohm Agostini *et al.*, "Design Space Exploration of Accelerators and End-to-End DNN Evaluation with TFLITE-SOC," in *Proceedings of the 32nd SBAC-PAD*, Sep. 2020, pp. 10–19.

[8] S. Venkataramani *et al.*, "ScaleDeep: A Scalable Compute Architecture for Learning and Evaluating Deep Networks," in *Proceedings of the 44th ISCA*, New York, NY, USA, Jun. 2017, pp. 13–26.

[9] H. Zheng and A. Louri, "Agile: A Learning-enabled Power and Performance-Efficient Network-on-Chip Design," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 223-236, Jun. 2020.

[10] H. Zheng, K. Wang, and A. Louri, "Adapt-NoC: A Flexible Network-on-Chip Design for Heterogeneous Manycore Architectures," in *Proceedings of the 2021 HPCA*, Feb. 2021, pp. 723–735.

[11] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural Acceleration for General-Purpose Approximate Programs," in *Proceedings of the 45th MICRO*, Dec. 2012, pp. 449–460.

[12] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 1-33, Mar. 2016.

[13] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate Communication: Techniques for Reducing Communication Bottlenecks in Large-Scale Parallel Systems," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1-32, Jan. 2018.

[14] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "APPROX-NoC: A Data Approximation Framework for Network-On-Chip Architectures," in *Proceedings of the 44th ISCA*, Toronto, ON, Canada, Jun. 2017, pp. 666–677.

[15] V. Fernando, A. Franques, S. Abadal, S. Misailovic, and J. Torrellas, "Replica: A Wireless Manycore for Communication-Intensive and Approximate Data," in *Proceedings of the 24th ASPLOS*, New York, NY, USA, Apr. 2019, pp. 849–863.

[16] J. R. Stevens, A. Ranjan, and A. Raghunathan, "AxBA: An Approximate Bus Architecture Framework," in *Proceedings of the ICCAS*, San Diego California, Nov. 2018, pp. 1–8.

[17] Y. Chen and A. Louri, "An Approximate Communication Framework for Network-on-Chips," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1434–1446, Jun. 2020.

[18] S. F. Dodge and L. J. Karam, "Quality Robust Mixtures of Deep Neural Networks," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5553–5562, Nov. 2018.

[19] J. Qiu *et al.*, "Going Deeper with Embedded FPGA Platform for Convolutional Neural Network," in *Proceedings of the 2016 FPGA*, New York, NY, USA, Feb. 2016, pp. 26–35.

[20] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.

[21] T. J. Yang, Y. H. Chen, and V. Sze, "Designing Energy-Efficient Convolutional Neural Networks Using Energy-Aware Pruning," 2017, pp. 5687–5695.

[22] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *arXiv:1510.00149 [cs]*, Feb. 2016.

[23] S. Xiao, X. Wang, M. Palesi, A. K. Singh, and T. Mak, "ACDC: An Accuracy- and Congestion-aware Dynamic Traffic Control Method for Networks-on-Chip," in *Proceedings of the 2019 DATE*, Mar. 2019, pp. 630–633.

[24] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *arXiv:1608.06993 [cs]*, Jan. 2018.

[25] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv:1404.5997 [cs]*, Apr. 2014.

[26] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015.

[27] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *arXiv:1707.01083 [cs]*, Dec. 2017.

[28] C. Szegedy *et al.*, "Going Deeper with Convolutions," in *Proceedings of the 2015 CVPR*, Jun. 2015, pp. 1–9.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016, pp. 770–778.

[30] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," in *Proceedings of the 2018 CVPR*, Salt Lake City, UT, Jun. 2018, pp. 8697–8710.

[31] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the ICML*, May 2019, pp. 6105–6114.

[32] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[33] M. Abadi *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," 2016, pp. 265–283.

[34] S. Dodge and L. Karam, "Understanding How Image Quality Affects Deep Neural Networks," in *Proceedings of the 8th QoMEX*, Jun. 2016, pp. 1–6.

[35] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," *arXiv:1807.11164 [cs]*, Jul. 2018.

[36] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008*, pp. 1–70, Aug. 2008.

[37] "NVIDIA Deep Learning Accelerator." http://nvdla.org/.