AxBy-ViT: Reconfigurable Approximate Computation Bypass for Vision Transformers

Dongning Ma, Xue Qin, Xun Jiao
Villanova University
{dma2, xue.qin, xun.jiao}@villanova.edu

***Abstract*—Vision Transformers, evolved from the attention based transformer architectures, have obtained elevated attention as a competitive alternative to convolutional neural networks in computer vision learning tasks. Although showing promising accuracy, vision transformers face challenges due to large model size and intensive computation workload with millions of parameters, inhibiting their practical deployment on resource constrained platforms.**

In this paper, we investigate trivial computations - a set of computations the results of which can be determined without actual computations - in vision transformers. Typical examples include multiplication with 0, +1/-1 and addition with 0. Existing studies have leveraged the existence of trivial computations to save energy or accelerate execution. Based on this, we present `AxBy-ViT`, a reconfigurable approximate computation bypass scheme for vision transformers that leverages approximate matching of bit representations of operands. Case study on vision transformers using the ImageNet dataset shows that with 2.4% accuracy loss, `AxBy-ViT` can match up to 28.69% of the operand for computation bypass. The code repository of `AxBy-ViT` can be found at `https://github.com/VU-DETAIL/AxBy-ViT`.

I. Introduction

Vision Transformers (ViT), inspired by the transformer architecture, have recently achieved tremendous success as an alternative method to traditional convolutional neural networks (CNNs) in image classification applications [1], [2], [3]. Although effective, vision transformers also require considerable amount of computations as well as significant size of model. For example, a very basic ViT implementation **ViT-16/B** in [4] contains around 80 million parameters. Thus, there is a dire demand to accelerate the processing or enhance the energy efficiency of ViT particularly for the implementation on edge computing, such as embedded systems and IoT devices.

Trivial computations (or trivial instructions), are those instructions that do not need actual computations, i.e., the result or output of the computation can be directly inferred from one or two of the operands [5], [6]. For a very straightforward example, if one operand inside a multiplication is 0, the result of this operation will always be 0 no matter what the other operand is. Such computations can be skipped, or bypassed by clock-gating processor units to enhance energy efficiency or achieve acceleration on computing platforms. Various computation-intensive application such as image processing and artificial neural networks have exhibited strong opportunities for trivial computation bypass [7], [8]. In addition, such applications often show strong robustness against errors and are able to tolerate errors which inspires the application of approximate computation bypass [9], [10]. Approximate computation bypass allows more bypass opportunities by enabling "approximate matching" of trivial operands. For example, $1.0001 * 5$ will be considered as $1 * 5$.

For the first time, this paper aims to explore the possibility of introducing approximate trivial computation bypass into ViT by proposing **`AxBy-ViT`**. **`AxBy-ViT`** targets the multiplications (which is the most computation-intensive operations in ViT) [11] and explore opportunities of trivial computation bypass of operands inside four major components in the vision transformer: patch embedding (PE), multi-head attention (MHA), multi-layer perceptron (MLP) and the final fully connected layer (FC).

Compared with existing schemes such as model (post-training) quantization [3], pruning [12], and memoization [13], **`AxBy-ViT`** does not require retraining and **`AxBy-ViT`** does not have specific data type requirement and can be applied in heterogeneous systems with diverse data types. Because **`AxBy-ViT`** operates at bit level, the approximate trivial computation bypass is viable as long as the precision of values inside the system is regulated by bit representation.

`AxBy-ViT` can co-exist and even benefit from other approximate computing techniques such as quantization or sparse matrix computing. Such techniques can potentially create more trivial computations in vision transformers, which in turns enhance the performance of **`AxBy-ViT`**.

The main contributions of this paper are as follows:

- We propose **`AxBy-ViT`**, a reconfigurable approximate trivial computation bypass scheme for ViT. **`AxBy-ViT`** leverages the strong data locality inside the ViT computations and explore the possibility of trivial computation bypass as well as using approximate matching to further enhance the bypass performance.
- We perform a case study of a baseline vision transformer model using the ImageNet dataset and record the amount of operands that can be matched for trivial computation bypass. Results show that **`AxBy-ViT`** is able to achieve up to 28.69% rate of bypassing operands with only 2.4% accuracy degradation.

II. `AxBy-ViT` Methodologies

A. *Overview of **`AxBy-ViT`***

In this section, we introduce the methodology of **`AxBy-ViT`** on developing a reconfigurable approximate com-

TABLE I
CATEGORIES OF TRIVIAL COMPUTATIONS IN **AxBy-ViT**

	description	expression	result
full	any operand equals 0	$p = 0$ or $q = 0$	0
semi (1)	any operand equals 1	$p = 1$ or $q = 1$	q or p
semi (-1)	any operand equals 1	$p = -1$ or $q = -1$	$-q$ or $-p$

putation bypass for the vision transformer. Fig. 1 shows the components and what data or parameters inside a transformer are targets of **AxBy-ViT** for approximate trivial computation bypass. This work focuses on the floating point multiplications which are the most common and power-consuming instructions inside four major components of vision transformer: the patch embedding (PE), the multi-head attention (MHA), the multi-layer perceptron (MLP) and the fully connected layer (FC) before the final output. Those components usually account for more than 75% of the total parameters and 85% of the total runtime of a vision transformer and thus are often the target of optimization [3], [11]. In **AxBy-ViT**, the bypassed data include both learnable parameters like weights as well as K, Q, and V matrices and input data. Each of the components owns its individual configuration and the accuracy from vision transformer inference can assist the reconfiguration of **AxBy-ViT**to achieve higher degree of approximate bypass within acceptable accuracy degradation.

B. Trivial Computations

The detailed definition of trivial computations in **AxBy-ViT** is listed in Table I. We classify the trivial computations in **AxBy-ViT** into two categories: full-trivial and semi-trivial, based on whether the computation results can be inferred by one or both the operands, respectively. Specifically, assume we have two operands p and q inside one multiplication $p \times q$, the computations with either of the operand equals to 0 ($p = 0$ or $q = 0$) are considered full-trivial as the results are always 0 regardless of the other operand. The computations with either of the operands equals to 1 or -1 ($p = \pm 1$ or $q = \pm 1$) are considered semi-trivial since the results are determined by the value of the other operand. Therefore, those trivial computations can be skipped during the inference stage to achieve acceleration or energy saving.

C. Data Locality

We perform a case study on the data locality inside Vision Transformers to highlight the possibility of introducing trivial computations. In Fig. 2, we randomly profile 500K operands and show the distribution inside the three components of a vision transformer (ViT-B/16 [4]). We can observe that for all the components, the operands are showing Gaussian-like distributions. For MHA, MLP and FC, more than 50% of the operands are concentrated around the interval near 0, indicating decent number of occurrences for full trivial computations. As PE is the first component inside ViT to take inputs, it has a more evenly distributed pattern for operands within its range that exhibits realistic distribution of (normalized) pixel values of images from a dataset. For semi-trivial computations, i.e., the value of ± 1, there is no significant clustering patterns like full-trivial. However, PE and FC both show a fair number of values (about 3-5%) that can be potentially considered as semi-trivial and then matched.

D. Approximate Operand Matching

The operand distribution in vision transformers inspires us to introduce the approximate matching of operands which allows values that are near 0 or ± 1 to be considered as trivial as well. This can further increase the number of trivial computations thus enhancing the energy efficiency and acceleration. **AxBy-ViT** provides an approximate matching solution based on the actual bit representations of the operands stored in the memory.

1) Bit Representation: As a case study, the vision transformer in **AxBy-ViT** uses 32-bit single precision floating point numbers based on IEEE 754 standard [14]. Note that, as **AxBy-ViT** matches operand by bit representations, it does not explicitly require the data to be floating point numbers and thus can be generalized for vision transformer models using different data types.

The IEEE 754 standard describes that a floating point number consists of three fields in their bit representation: the 1-bit sign, the 8-bit exponent field, and the 23-bit mantissa field. The value of a IEEE-754 number is computed as $sign \times 2^{exponent-127} \times mantissa$. Using the number 1.1875 as an example:

Binary: $0_1|01111111_{127}|00110000000000000000000_{1.1875}$
Decimal: $1 \times 2^{127-127} \times 1.1875 = 1.1875$.

In exact matching, a number must have a bit representation matching all the 32 bits with 0 or ± 1 to be considered trivial. However, with approximate matching, the bit representation can be considered as 0 or ± 1 within a reconfigurable approximation range.

2) Full Trivial: For full-trivial (0) matching, the approximation range is determined by the exponent field: if the bit representation has a smaller exponent field value, it is considered a match. Assume in one configuration, the exponent field of approximation range is set to 01111001_{121}, consider the number 0.01171875 as an example:

0.01171875: $0_1|01111000_{120}|10000000000000000000000_{1.5}$

As $01111000_{120} < 01111001_{121}$, 0.01171875 can subsequently be approximated into 0 as a match for the full trivial computation. In decimal, 01111001_{121} translates into 0.015625, thus with this configuration, every number within the range of ± 0.015625 is approximated into 0.

3) Semi Trivial: For semi-trivial (± 1) matching, the range is mostly confined by the mantissa field: certain number of mantissa bits can be truncated when matching. Assume in one configuration, the last 19 mantissa bits can be truncated, consider the number of 1.015625 as an example:

1.015625: $0_1|01111111_{127}|00000100000000000000000_{1.015625}$
1.015625 (19 bits truncated): $0_1|01111111_{127}|0000_{1.0}$
1.015625 (16 bits truncated): $0_1|01111111_{127}|0000010_{1.015625}$

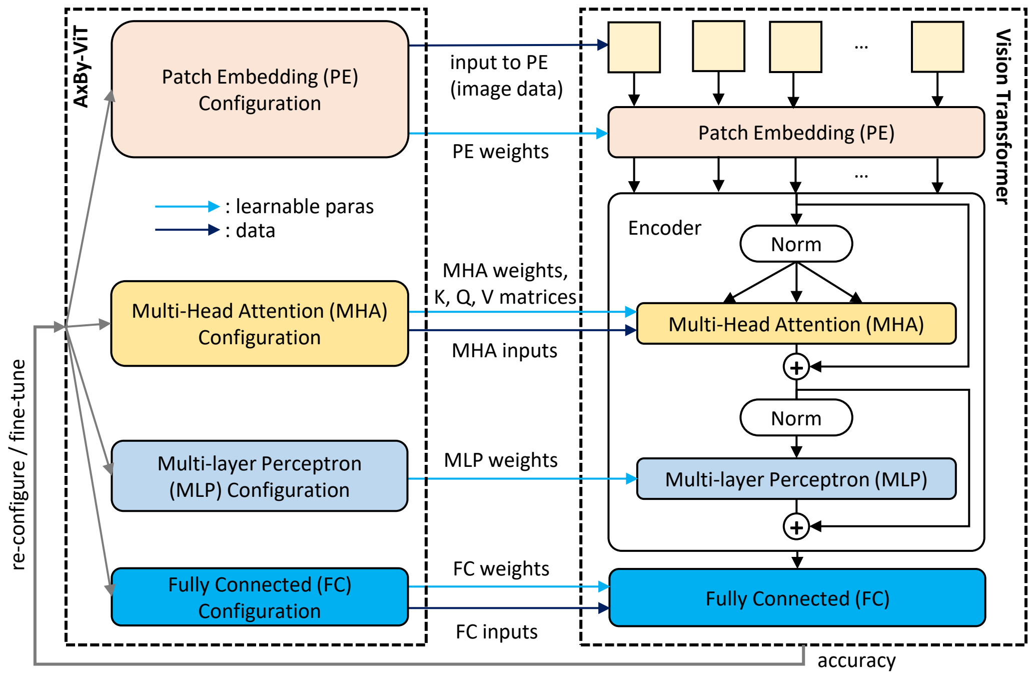

AxBy-ViT
Patch Embedding (PE) Configuration
Multi-Head Attention (MHA) Configuration
Multi-layer Perceptron (MLP) Configuration
Fully Connected (FC) Configuration
: learnable paras
: data
re-configure / fine-tune
input to PE (image data)
PE weights
MHA weights, K, Q, V matrices
MHA inputs
MLP weights
FC weights
FC inputs
accuracy
Vision Transformer
Patch Embedding (PE)
Encoder
Norm
Multi-Head Attention (MHA)
Norm
Multi-layer Perceptron (MLP)
Fully Connected (FC)

Fig. 1. Approximate trivial computation bypass targets of **AxBy-ViT** inside a vision transformer. **AxBy-ViT** focuses on four components of a vision transformer: patch embedding (PE), multi-head attention (MHA), multi-layer perceptron (MLP) and fully connected layer (FC).

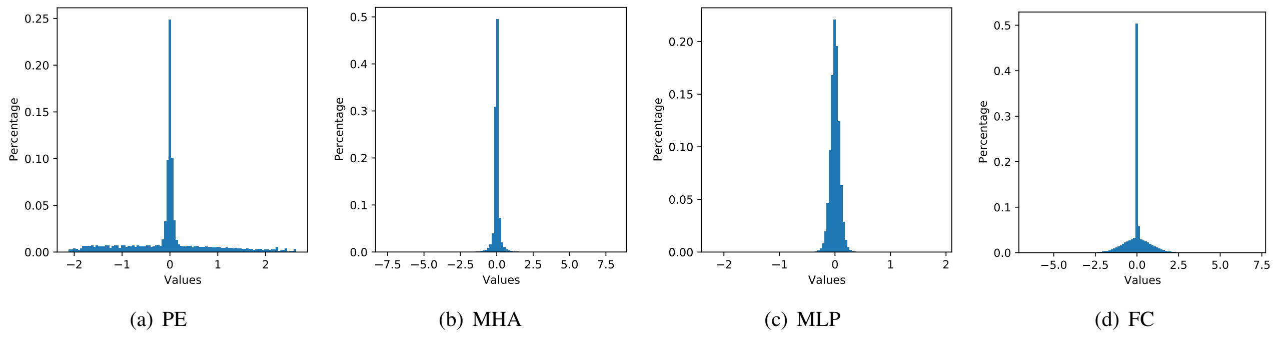


Fig. 2. Operand distribution within different components inside vision transformers.

By truncating the last 19 bits, the bit representation of 1.015625 is the same as 1 thus is considered as a semi-trivial computation. However, if the configuration only truncates 16 mantissa bits, then the bit representation of 1.015625 is not matching that of 1, thus is not considered as trivial.

E. Reconfigurability of **AxBy-ViT**

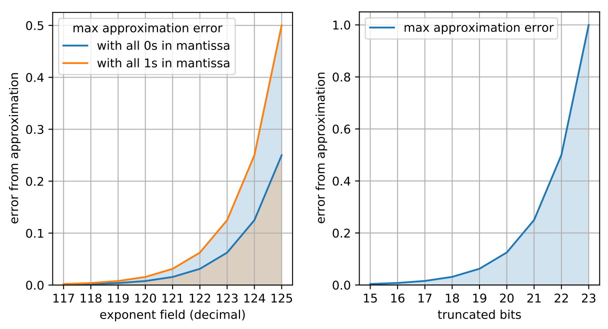
Approximation in operand matching increases the number of trivial computations, however, at the expense of introducing errors. In general, more reduced precision provides a broader range of approximation matching, which in turns increases the absolute possible errors as a side-effect. Therefore, the degree of approximation, namely the precision of operand matching for trivial computations inside **AxBy-ViT** is required be controlled to achieve reconfigurablility under different scenarios. For full-trivial matching, the precision is tuned by the value of exponent field: a larger exponent value means more reduced precision, thus higher approximation error. For semi-trivial matching, more truncated mantissa bits means more reduced precision and higher approximation error.

We plot how the configuration can affect the approximate matching in Fig. 3 by highlighting the max approximation error under different configurations. We can observe that the max approximation error exponentially grows as the precision is reduced. For full-trivial matching, we plot two curves that reflects both the mantissa field is filled by 0s or 1s.

III. Experimental Results

A. Experimental Setup

1) Vision Transformer and Dataset: We use the Vision Transformer Base model with 16×16 patch size (ViT-B/16) [4] pretrained on ImageNet dataset with reported accuracy of 0.779 [15]. In **AxBy-ViT**, to accelerate the evaluation, we select one image from each of the 1,000 classes in the



(a) Full-trivial (b) Semi-trivial

Fig. 3. Operand approximation error from **AxBy-ViT** approximate matching.

ImageNet inference set as our in house inference set. The ViT-B/16 model has an accuracy of 0.777 on this set, which is used as the baseline accuracy in the following experimental results on reporting accuracy degradation after approximate computation bypass.

2) Hit Rate: To quantitatively describe the performance of **AxBy-ViT**, we define the metric hit rate ζ as the percentage of matched operands out of all the values targeted for approximate bypassing per Eq. 1. Hit rate is also used as our metric to describe how **AxBy-ViT** can enhance efficiency of the vision transformer. A higher hit rate indicates more trivial computations are bypassed thus more enhancement over the efficiency of vision transformers that can translate into energy saving or acceleration dependent on the platforms of implementation.

$$\zeta = \frac{\#matched\_operands}{\#all\_operands} \tag{1}$$

3) Implementation: We evaluate **AxBy-ViT** performance using the Pytorch with GPU acceleration [16]. First, we compute the range of approximation under each configuration for both full-trivial and semi-trivial computation bypass. Then, during inference of each image, We obtain access into the operands inside the vision transformer model or the data from the image and iteratively check if the operand is within the range. If yes, we replace the operand with the trivial numbers (0 or ± 1). The number of matched operand is then recorded to calculate the hit rate ζ after the model finishes the inference of all the images along with the accuracy of the model.

B. ***AxBy-ViT*** *Performance*

Table II-V presents the hit rate as well as the accuracy on ImageNet under different configurations when using **AxBy-ViT**of 4 components inside the vision transformer: PE, MHA, MLP and FC. The results from each component are obtained without **AxBy-ViT** enabled on all the other 3 components. We define that accuracy degradation within 3% is acceptable and strike all the unacceptable accuracy in the table. On the other hand, we bold the accuracy of the recommended configuration during our experiment.

Firstly, we observe a general trend that with larger exponent value or more truncated bits, the hit rate increases. However, this introduces more error during inference and thus causing accuracy degradation. This observation aligns with our analysis on the approximation error at Section II-E.

Secondly, we notice that different components inside the vision transformer have different sensitivity for approximation particularly on the exponent configuration. As PE and FC are already experiencing lower-than-acceptable accuracy when exponent grows from 120 to 121, MHA and MLP exhibit much less accuracy drop which is mostly less than 3%. This indicate different components inside the transformer preserve different level of toleration against the error caused by approximate bypass. With such observation, we are able to perform a concise design exploration and locate the recommended configuration with relatively higher hit rate but acceptable accuracy.

Moreover, the hit rate reflects the operand distribution presented at Fig. 2. For example, MLP has most of the operands concentrated within ± 0.5, and there are hardly any significant number of operands near ± 1, therefore even if all the mantissa bits are truncated, the hit rate is still less than 0.01%.

TABLE II
HIT RATE AND IMAGENET ACCURACY UNDER DIFFERENT CONFIGURATIONS OF **AXBY-VIT**: PE

configuration		hit rate ζ		
exponent	truncated bits	full-trivial	semi-trivial	accuracy
120	20	20.69%	1.72%	0.764
120	21	20.69%	3.32%	0.761
120	22	20.69%	6.37%	**0.758**
120	23	20.69%	11.82%	~~0.733~~
121	20	33.70%	1.72%	~~0.642~~

TABLE III
HIT RATE AND IMAGENET ACCURACY UNDER DIFFERENT CONFIGURATIONS OF **AXBY-VIT**: MHA

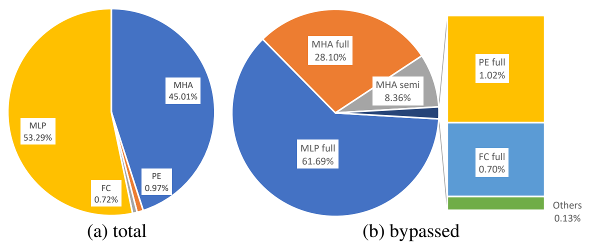
configuration		hit rate ζ		
exponent	truncated bits	full-trivial	semi-trivial	accuracy
120	20	9.19%	1.46%	0.777
121	20	17.88%	1.45%	0.776
121	21	17.89%	2.81%	0.774
121	22	17.90%	5.33%	**0.761**
121	23	17.88%	10.45%	~~0.726~~
122	20	32.82%	1.44%	~~0.343~~

TABLE IV
HIT RATE AND IMAGENET ACCURACY UNDER DIFFERENT CONFIGURATIONS OF **AXBY-VIT**: MLP

configuration		hit rate ζ		
exponent	truncated bits	full-trivial	semi-trivial	accuracy
120	20	17.11%	$\approx$0.00%	0.777
121	20	33.18%	$\approx$0.00%	0.769
121	21	33.18%	$\approx$0.00%	0.769
121	22	33.18%	$\approx$0.00%	0.768
121	23	33.18%	0.01%	**0.767**
122	20	59.55%	$\approx$0.00%	~~0.496~~

TABLE V
HIT RATE AND IMAGENET ACCURACY UNDER DIFFERENT CONFIGURATIONS OF **AxBy-ViT**: FC

configuration		hit rate ζ		
exponent	truncated bits	full-trivial	semi-trivial	accuracy
120	20	40.31%	0.66%	0.772
120	21	40.31%	1.48%	0.772
120	22	40.31%	2.96%	0.772
120	23	40.31%	5.18%	**0.772**
121	20	70.62%	10.63%	~~0.706~~



(a) total (b) bypassed

Fig. 4. Breakdown of total and bypassed operands in **AxBy-ViT**.

C. Discussion

We also compare **AxBy-ViT** hit rate with exact bypass. The exact bypass scheme requires all the bit representation matched, i.e., the operand must be identical to the trivial numbers. Exact bypass does not introduce errors because there is no approximation applied thus will not result in accuracy degradation. However, according to our observation, exact bypass can only match less than 0.1% of all the operands on average, which can barely bring anything than marginal benefits, while ssing approximate bypass, can achieve more than 1000X of computations bypassed.

To assess how **AxBy-ViT** can assist enhancing computation bypass globally, we further show the breakdown of the bypassed trivial computations in Fig. 4 using the recommended settings aforementioned. We can observe that MHA and MLP contribute most to the amount of bypassed computation in the vision transformer evaluated. We can estimate that with approximate trivial computation bypass enabled in all the four components, **AxBy-ViT** is able to obtain 28.69% hit rate overall with an accuracy of 0.753, i.e., only 2.4% accuracy degradation from the baseline 0.777.

Hit rate in trivial computation bypass can translate into multiple advantages such as energy saving and/or acceleration depending on the implementation platform. For example, in GPGPU platforms, bypassing trivial computation can reduce the energy cost [10]. On the other hand, computation bypass can also achieve speed-up as when matched, the trivial computations are not necessarily executed [6]. Moreover, there are already RTL-level implementation of trivial computation bypass circuits that are considered architectural support, using a clock-gate circuit of which the energy or area are comparatively negligible [8], [9]. This indicates that the overhead of implementing a trivial computation bypass circuit to the entire system can be minimal.

IV. CONCLUSION

In this paper, we present **AxBy-ViT**, an approximate computation bypass scheme for vision transformers. **AxBy-ViT** leverages the strong data locality in vision transformers to identify operands that are contributing to trivial computations based on the approximate matching bit representations. Such computations can subsequently be bypassed, i.e., without the actual operation or computation by the computation unit to save energy and/or to accelerate processing. **AxBy-ViT** can be reconfigured according to the different components inside a vision transformer to achieve optimal enhancement under different scenarios. Results on vision transformer with ImageNet dataset show that up to 28.69% of the operands can be matched to bypass trivial computations with only 2.4% accuracy loss.

REFERENCES

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," *arXiv preprint arXiv:2103.15691*, 2021.

[2] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Z. Jiang, Q. Hou, and J. Feng, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

[3] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] D. Ma and X. Jiao, "Energy efficient gpu applications through computation skip," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pp. 1–2, IEEE, 2019.

[6] Z. Shaikh and E. Atoofian, "Approximate trivial instructions," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*, pp. 1–9, 2020.

[7] E. Atoofian, "Trivial bypassing in gpgpus," *IEEE Embedded Systems Letters*, vol. 13, no. 1, pp. 25–28, 2020.

[8] D. Ma and X. Jiao, "Detecting and bypassing trivial computations in convolutional neural networks," in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 1–6, IEEE, 2019.

[9] D. Ma and X. Jiao, "Axby: Approximate computation bypass for data-intensive applications," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pp. 332–339, IEEE, 2020.

[10] E. Atoofian, Z. Shaikh, and A. Jannesari, "Reducing energy in gpgpus through approximate trivial bypassing," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 2, pp. 1–27, 2021.

[11] A. Nagarajan, S. Sen, J. R. Stevens, and A. Raghunathan, "Optimizing transformers with approximate computing for faster, smaller and more accurate nlp models," *arXiv preprint arXiv:2010.03688*, 2020.

[12] H. Yang, H. Yin, P. Molchanov, H. Li, and J. Kautz, "Nvit: Vision transformer compression and parameter redistribution," *arXiv preprint arXiv:2110.04869*, 2021.

[13] D. Ma, X. Yin, M. Niemier, X. S. Hu, and X. Jiao, "Axr-nn: Approximate computation reuse for energy-efficient convolutional neural networks," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, pp. 363–368, 2020.

[14] W. Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, p. 11, 1996.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.