Zero Aware Configurable Data Encoding by Skipping Transfer for Error Resilient Applications

Chandan Kumar Jha, Shreyas Singh, Riddhi Thakker, Manu Awasthi, and Joycee Mekie

***Abstract*—Data transfer across DRAM channels accounts for nearly a quarter of the total energy consumption of DDR4 DRAMs. Modern applications with high bandwidth requirements further increase channel energy consumption. However, channel energy consumption is dependent on data being transferred. Pseudo Open Drain (POD) asymmetric termination, used in current DDR4 systems, consumes energy only when 1's are being transmitted over the channels. Many modern applications, including AI/ML ones are resilient to errors in data, and can work well with approximate data. This resilience can vary widely across and within applications, which provides a number of ways for exploiting these characteristics to save data transfer energy across the DRAM channel. However, all DRAM data encoding schemes have been targeted towards applications that require exact data and are not approximation resilient.**

In this paper, we propose Zero Aware Configurable Data Encoding by Skipping Transfer (ZAC-DEST), a data encoding scheme to reduce the energy consumption of DRAM channels, specifically targeted towards approximate computing and error resilient applications. ZAC-DEST exploits the similarity between recent data transfers across channels and information abut error resilience behaviour of applications to reduce on-die termination and switching energy by reducing the number of 1's transmitted over the channels. ZAC-DEST also provides a number of knobs for trading off application's accuracy for energy savings, and vice versa, and can be applied to both training and inference.

We apply ZAC-DEST to five machine learning applications. On average, across all applications and configurations, we observed a reduction of 40% in termination energy and 37% in switching energy as compared to the state of the art data encoding technique BD-Coder with an average output quality loss of 10%. We show that if both training and testing are done assuming the presence of ZAC-DEST, the output quality of the applications can be improved upto 9× as compared to when ZAC-DEST is only applied during testing leading to energy savings during training and inference with increased output quality.

***Index Terms*—Data Encoding, DRAM Channels, Approximate Computing, Machine Learning.**

I. Introduction

DRAMs are an integral component of memory systems [1]–[4]. DRAM energy accounts for approximately 46% of the total system energy consumption [5], [6]. The energy consumption of the DRAM I/O channel contributes 25% of the total DRAM energy due to off-chip communications [7], [8]. To reduce DRAM I/O energy consumption, asymmetric I/O termination mechanisms like Pseudo Open Drain (POD) and Low Voltage Swing Terminated Logic (LVSTL) have been implemented [9]–[12]. These mechanisms help in reducing energy consumption of DRAM channels [9]–[12]. This happens because asymmetric termination mechanisms dissipate energy for only one of the bits during data transfer, i.e. for bit-0 in LVSTL and bit-1 in POD [9], [13], [14].

Error resilient applications in the domain of machine learning, object recognition, image/video processing etc. have opened up a plethora of possibilities to optimize current computing and memory systems [15]–[17]. The error resilience of applications is exploited by introducing approximation in computation or data to reduce energy and/or improve performance. As a result, the applications are able to achieve the same level of performance and accuracy, with sometimes significant amount of approximation introduced into the data, which enables us to explore trade-off between accuracy and energy savings. Previous research explored approximate data encoding for serial data transfer in embedded systems [18]–[20]. Recent works have also explored approximate compression and decompression of data [21], [22].

DRAM I/O energy consists of two components, namely termination and switching. Termination energy is consumed in DRAM channels as a result of on-die termination. Switching energy is consumed due to charging of DRAM channels during data transfer. Termination energy in POD, used in DDR4 DRAMs, is directly proportional to the number of 1's being sent over the DRAM channel. Bit value 1 is sent using 0V and bit value 0 is sent using V_{dd}V [13], where V_{dd} is the supply voltage. The number of 1's in a data word, also called its hamming weight, has a positive correlation with termination energy [9]. On the other hand, switching energy is proportional to the number of 1 to 0 (charging) transitions. For 0 to 1 (discharging) transitions, no current is drawn from the supply voltage [9], [13], [14]. In most cases, reducing hamming weight also leads to a reduction in switching count, thus reducing switching energy as well [14]. In modern DRAMs, which deploy one of the two termination schemes, the termination energy has become the dominant source of energy consumption in DRAM channels [9], [14]. Thus, in recent

This work was done when Riddhi Thakker was at the Indian Institute of Technology Gandhinagar.

Chandan Kumar Jha and Joycee Mekie are with the Discipline of Electrical Engineering, Indian Institute of Technology Gandhinagar. e-mail: chandan.jha@iitgn.ac.in and joycee@iitgn.ac.in

Shreyas Singh is with the Discipline of Computer Science and Engineering, Indian Institute of Technology Gandhinagar. e-mail: shreyas.singh@iitgn.ac.in

Riddhi Thakker was with the Discipline of Information and Communication Technology from Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar. She is now with Oracle India. e-mail: 201601124@daiict.ac.in

Manu Awasthi is with the Department of Computer Science, Ashoka University, Sonepat. e-mail: manu.awasthi@ashoka.edu.in

years, research has focused on reducing termination energy by reducing the number of 1's sent across the channel [13], [14]. However, to the best of our knowledge, there exists no prior research which looks into encoding the data approximately between DRAM channels and processors. This is the first work to exploit approximate computing to benefit while performing data transfers.

In this paper, we propose **ZAC-DEST** (Zero Aware Configurable Data Encoding by Skipping Transfer), an energy efficient data encoding scheme for DRAM channels for approximate computing applications. DEST extends existing data encoding schemes for exact data transfers - Bitwise Difference Encoding (BD-Coder) [14] and Dynamic Bus Inversion (DBI) [23] with additional optimizations options provided by error and approximation tolerant applications to provide a further reduction in termination and switching energies of DRAM I/O.

Transfer of approximate data across the DRAM channel leads to energy savings at the cost of output quality loss in applications. An error resilient application can tolerate varying amounts of approximation, i.e. there is a trade-off between output quality and energy consumption. ZAC-DEST introduces three tuning features – i) *Similarity Limit*, ii) *Truncation*, and iii) *Tolerance* which allows it to introduce a varying range of approximations in data sent across the DRAM channel, and as a result, allows for interesting trade-offs to be made by the architect or the application programmer. In most applications increasing the approximation leads to an increase in energy savings with a reduction in output quality [24].

Depending upon the application, the acceptable output quality may vary. The tuning features in ZAC-DEST allow it to be tailored for obtaining acceptable output quality, while achieving significant energy contributions. Overall, in this paper, we make the following contributions.

- To the best of our knowledge, ZAC-DEST is the first proposal for an encoding mechanism for transferring data across DRAM channels geared specifically towards error resilient applications. ZAC-DEST extends the data encoding schemes designed for exact applications to provide and average savings of 40% in termination and 37% in switching energy off-chip DRAM channels across five machine learning applications.
- We augment the existing encoding mechanisms (BD-Coder) with two additional policies that improves the BD-Coders' table update mechanisms. In addition, ZAC-DEST handles transfer of zeros across the channel separately, which is useful for reducing data transfer energy when data to be transferred has majority zeros. On average the modified BD-Coder consumes 25% lesser energy as compared to the original BD-Coder.
- ZAC-DEST incorporates multiple knobs to trade off accuracy and DRAM channel transfer energy in error resilient applications. : Similarity Limit (exploits similarity between recent data transfers), Truncation(removing bits that do not affect output quality), and Tolerance (masking bits that cannot tolerate approximation), making it an ideal candidate for use in approximate processors. These knobs can be varied to obtain the desired accuracy and we have explored them in detail. We have also implemented ZAC-DEST design in UMC 65nm. The area overhead of ZAC-DEST over BD-coder is 15%.

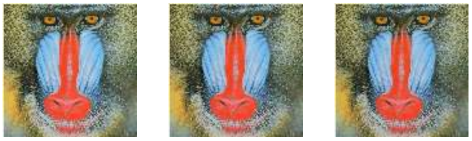

(a) PSNR=Inf (b) PSNR=36 (c) PSNR=32

Fig. 1: (a) Original (b) 20% 1's flipped in LSBs (c) 40% 1's flipped in LSBs

- We developed a framework that allows for the evaluation of DEST on error resilient machine learning applications. We also evaluated five different machine learning applications namely: i) ImageNet inferencing, ii) CIFAR-100 training and inferencing, iii) Eigenfaces, iv) Color quantization using K-Means and v) SVM. For each of the applications, we observe a reduction of hamming energy by 39%, 34%, 44%, 47%, and 36% respectively.
- Finally, we demonstrate that inference accuracy of image classification of CIFAR-100 dataset using ResNet-110 can increase on an average 24% (by upto 9×) when ZAC-DEST is applied to data transfer from DRAM during both training and inference phases, as opposed to application of ZAC-DEST to only the inference process data transfers. Thus, not only can ZAC-DEST be exploited to provide energy savings during both training and testing, but also improve the output accuracy.

II. Motivation

Error Resilient Applications: Various recognition, mining, and synthesis applications are resilient to some degree of approximation in data and computations [21], [25]–[27]. Machine learning applications for object detection, image recognition, etc. have also shown robustness towards errors in data and computations [28]–[35]. Thus, there exists a wide variety of applications where approximation can be traded off for energy savings. We demonstrate error resilience in images with the help of an example image, shown in Fig. 1a. Every pixel of the image is of an 8-bit entry. To introduce approximation in data, the 1's in the last 4-bits of pixels were flipped to 0's. The percentage of flipped 1's in Fig. 1b is 20% and Fig. 1c is 40%. Peak signal to noise ratio (PSNR), a quality metric used to measure the similarity between images, is 36 for Fig. 1b and 32 for Fig. 1c, higher PSNR is better [36]. For most images, PSNR $\geq$ 30 is acceptable [36] as it is indifferentiable to the human eye. This shows the error resilience of images towards bit flips. In later sections, we will show that when these kinds of approximated images are fed as input to error resilient applications, the output quality loss is minimal. Hence, there is an opportunity to reduce energy by approximating the data.

Energy Consumption by DRAM I/O: A breakdown of energy consumption of various DDR4 DRAM components was provided in [14], and is shown in Fig. 2. We observe that

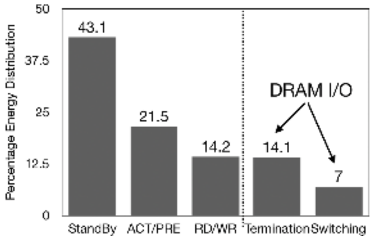


Fig. 2: Energy dissipation breakdown in DDR4 DRAM sub-system [14]

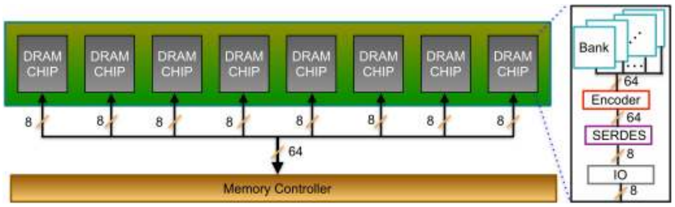


Fig. 3: Overall Data encoding-decoding structure

DRAM I/O energy (termination + switching) accounts for 21% of the total DRAM energy consumption. The termination energy accounts for 67% of the total DRAM I/O energy while the switching energy accounts for the rest. Prior research has focused mostly on reducing termination energy [13], [14]. Furthermore, DRAM I/O energy is predicted to worsen in the future as it is unaffected by scaling [13], [14]. This makes it crucial to devise techniques to reduce the DRAM I/O energy consumption.

III. Background and Prior Work

DRAM Data Transfer: The data over the DRAM channel is transferred in 64 byte (*cache line*) granularity. DRAM data bus is 64-bit wide, i.e. there are 64 physical lines from DRAM DIMM to the memory controller for the transfer of data. There are other physical lines for the transfer of error correction codes, control commands, etc. The 64 byte cache line is transferred in 8 bursts of 64 bits each (*assuming each chip is* x8) [37]. For a 64-bit burst, the overall structure for data encoding is shown in Fig. 3. The encoder is situated between the DRAM chip and the I/O bus, while the decoder is located between the I/O bus and the memory controller. In DRAMs, while transferring the bit value 0, the DRAM channel is connected to V_{dd} and for bit value 1, connected to GND [13].

DRAM I/O Termination and Switching Energy: POD I/O termination is a widely used termination scheme in DDR4 DRAMs [9]. The termination energy is a result of the POD I/O scheme. Due to asymmetric design, it consumes a significant amount of energy, which depends on termination resistance, while transferring a bit value 1 [13], [14]. This is due to the direct path between V_{dd} and GND as shown in Fig. 4 (❶). This current accounts for the termination energy in DRAM I/O's. When bit value 0, is transmitted there is no current flow as shown in Fig. 4 (❷). Transferring bit value 1 can consume 13.75 mA additional current as compared to bit value 0 [9]. Thus termination energy is directly proportional to the number of 1's transferred over the DRAM channel. Switching energy is proportional to the number of 1 to 0 (charging) transitions. The energy consumption as a result of switching is obtained using $E = CV_{dd}^2$, where C is the capacitance and V_{dd} is the supply voltage. The typical value of C per channel is $15pF$ [14].

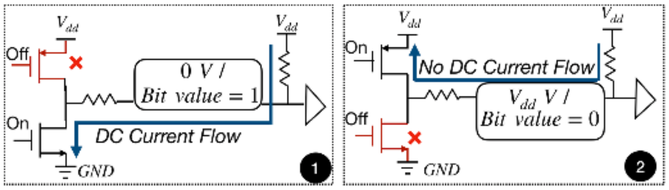
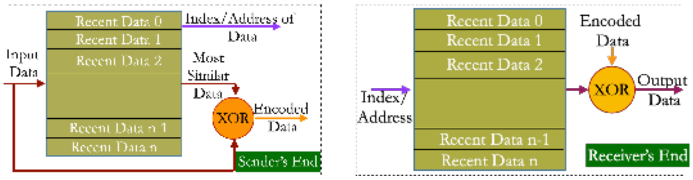


Fig. 4: Pseudo Open Drain I/O Termination



(a)

(b)

Fig. 5: BD-Coder (a) Encoder (b) Decoder

*Bitwise Difference Coder (*BD-Coder*):* BD-Coder [14] exploits data similarity between recent data transfers to reduce DRAM I/O energy consumption. It maintains a table (*data table*) of recent data transfers at sender's (*DRAM*) as well as receiver's (*memory controller*) end. The data to be sent is first compared to all the entries in the data table to find the most similar entry. To find the most similar entry, the data to be sent is bitwise XORed with all the data table entries (*XORing the same numbers gives a 0*) to reduce the number of 1's. This new number of 1's (*hamming weight*) in the XORed output is now compared to that of the original data. If the XORed output has a smaller hamming weight, the address/index of the most similar data (*using a separate line per chip*), along with the XORed output is transferred over data lines. Otherwise, the original data is sent over the data lines and the index lines send the address. If encoded data is received at the receiver's end, it is XORed with the data table entry pointed by the received address. Otherwise, the original data is passed to the memory controller and the data table is updated with this data at both the sender and the receiver's end. The overall structure of the encoder and decoder in BD-Coder is shown in Fig. 5a and Fig. 5b, respectively.

*Dynamic Bus Inversion (*DBI*):* DBI is widely used in DDR4 systems to reduce energy consumption of the DRAM I/O [23]. It is applied at a granularity of 8-bits. If more than 4-bits out of 8-bits are 1's, DBI inverts the data being transferred. An additional line is added per chip, i.e. 8 lines total, to convey if DBI has been applied. Thus, the transmitted data always has at most four 1's leading to a reduction in termination energy.

In the next section we will discuss ZAC-DEST, our proposed encoding scheme.

IV. Zero Aware Configurable Data Encoding by Skipping Transfer (*ZAC-DEST*)

Most data encoding schemes try to reduce the DRAM I/O energy during data transfer. In this section, the mechanisms to exploit the error resilience when transferring data over the DRAM channel to reduce DRAM I/O energy are explained. Since termination energy is proportional to the 1's being transmitted, ZAC-DEST focuses reducing the number of 1's. ZAC-DEST is built on top of current state of the art data encoding schemes for data transfer: a) BD-Coder [14] and b) DBI [23], which allows ZAC-DEST to be easily integrated with any encoding schemes built using similar design principles.

A. *Leveraging Error Resilience*

The similarity between recent data accesses remains the same irrespective of whether the application is error resilient to input data. We leverage the error resilience of applications by introducing approximations with the goal of reducing the number of 1's. The naive approach to introduce approximations will be to change all 1's to 0's when a request for approximate data is received. The key goal is to reduce the number of 1's being sent over the channel keeping the degree of approximation under check.

The amount of approximation that can be tolerated not only varies widely across applications but also within the same application. Thus, we need to provide a variety of configurations that control the degree of approximation introduced in data. Note that data transfers pertaining to instructions are *never* approximated. Also, among the data, only the accesses that are known to be error resilient a priori are approximated. The information related to approximation, can be transferred over the already existing address lines of the DRAM while transferring the column address as column address have lesser bits as compared to the row address and it leaves some address lines unutilized [14].

ZAC-DEST uses the same data table as in BD-Coder, shown in Fig. 5. Each data table, one per chip, holds 'n' recent entries of 64-bits transferred over the DRAM channel [14]. We assume that the degree of approximation that can be tolerated by an application will be known a priori and can be encoded in the applications. The output quality for each workload will be defined in Section VII-A. The data to be sent is compared with all the entries in the data table. So, if an application can tolerate a 25% approximation in data, 16 out of 64-bits can be approximated. The data to be sent is compared to all the entries in the table to find the most similar entry. The most similar entry is now checked to see if it differs from the original data by not more than 16-bits. If true, in place of actual data, all 0's are sent over the DRAM channel along with the index of the most similar entry, which is already present at the receiver's end. Note that this is the same as best case scenario since we are not transmitting any 1's. The only overhead is sending the index of the receiver's data table at which the most similar entry is stored. Here, the assumption is that number of 1's in the index is very small as compared to the data. If it would have been the case that the most similar entry has less than 48 similar bits, we would have applied BD-Coder on it i.e, the data would be sent without approximation. Thus, this encoding scheme fits very well on top of the existing data encoding scheme. BD-Coder updates data table after every transfer, which can lead to multiple entries having the same value. In ZAC-DEST, we update the data table only when the *exact* data is transferred. This ensures no duplicate entries are present in the table. Since there are no duplicate data entries in ZAC-DEST, the probability of finding a most similar entry is higher, leading to further energy savings.

B. *Using the Unused*

In frequent value (FV) encoding, the frequent values are encoded and sent as a one-hot encoded address and was targeted towards reducing switching energy [38]. ZAC-DEST differs from FV encoding as we have a separate encoding scheme and target termination energy. We will show how we exploit one hot encoding to further reduce the termination energy for ZAC-DEST. ZAC-DEST allows us to skip data transfer when a similar entry is found in the data table. The only hiccup now is of transferring the index (location) of the most similar entry. In BD-coder, a separate line was used to transfer the index to the receiver. However, when ZAC-DEST is true, the skipping of data transfer during ZAC-DEST leaves the data lines unused. These lines are used to our benefit for sending the index of the entry in the one-hot encoded (OHE) format. For example, in the worst case scenario which occurs when transferring the index value 111111 (i.e. 63 in decimal) causes six 1's to be sent. If the same is encoded in 64-bit OHE, the index sent will be '$0x8000000000000000$'. This reduces the number of 1's down from six to one. Also, no additional lines are required since existing lines for data transfer are used to send the OHE index.

V. ZAC-DEST Optimizations

In this section, we discuss a separate addressing technique for zeros. We also discuss the support provided by ZAC-DEST for allowing configurability in approximation within each data transfer.

A. *Handling All Zeros*

Without any data encoding scheme, the transfer of 0's consumes the least amount of energy [13]. Hence, we must ensure that there are no overheads while transferring 0's. Thus, whenever a 64-bit data containing all 0's needs to be transferred, neither ZAC-DEST nor BD-coder applied to it. Also, unlike BD-Coder, which would update the table after every data transfer we do not add an entry in the data table when 0's are transferred which allows us to store unique data in the data table.

B. *Configurability in ZAC-DEST*

Similarity Limit: *Similarity Limit*, as the name suggests refers to the number of bits that needs to same, between the data to be sent and the most similar entry, for ZAC-DEST to be true. We have included 4 different similarity limits in ZAC-DEST for evaluation purposes. These are 7, 13, 16, and 20 out of 64 bits which corresponding to 90%, 80%, 75%, and 70% similarity limit respectively. ZAC-DEST can be tuned to use any of the similarity limit values required by the application.

Tolerance: Tolerance refers to the bits which *cannot* be approximated. Even though we are proposing an encoding scheme for approximate applications, it may happen that approximating most significant bits (MSBs) may cause large errors in applications. These bits need to be transferred without approximation, irrespective of the similarity limit. Thus, the number of bits that can tolerate errors, in this case, will reduce. For example, if the data is 64-bits and a tolerance of 16 is required, the most significant 16 bits of data cannot be approximated. This will put a tighter constraint on approximation and ZAC-DEST will be applied a fewer number of times. Support for a wide range of values that can be selected to tune required tolerance depending upon the data width is provided in ZAC-DEST. It is important to note that while having higher tolerance reduces energy savings, it does increase the output quality of the application.

Truncation: Truncation refers to the removal of a fixed number of bits from the original data. In approximate computing, one of the most widely used approximation methodology is the removal of least significant bits (LSBs). Thus, it is useless to transfer these bits over the DRAM channel. For example, if we have an 8-bit data of the form 01101111 and we had to truncate 4 LSBs the data would change to 01100000. In ZAC-DEST we incorporate truncation in the following way. If we have a 64-bit data and a truncation of 16 bits is required, the least significant 16 bits of the data will be ignored while finding the most similar entry. These bits will be replaced by 0's hereafter. The rest of the steps remain the same as that of ZAC-DEST. The overall algorithm for BD-Coder and ZAC-DEST is shown in Algorithm 1 and Algorithm 2.

Algorithm 1 BD-Coder Algorithm

```
Definitions
DCD- DRAM Chip Data
DS - Data sent over DRAM channels
DR - Data reconstructed at receiver end
MSE - Most similar entry
BD-Coder- BDE
for all chip do
  Find MSE w.r.t DCD
  Check for BDE
  Condition for BDE to be True:
  Hamm(DCD) > Hamming Count of (MSE XOR DCD)
  if BDE condition TRUE then
    DS : (MSE xored DCD) and Index of MSE
    DR: DCD
  else
    DS : DCD
    DR: DCD
    Table Updated with DCD
  end if
end for
```

Algorithm 2 ZAC-DEST Algorithm

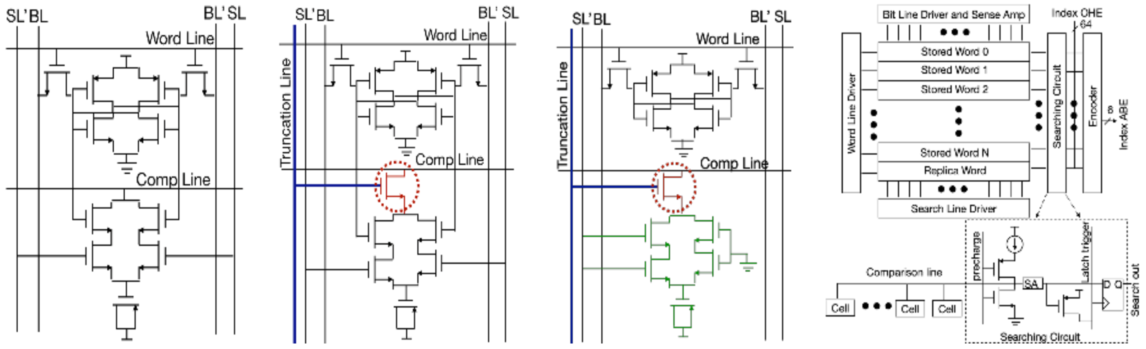
```
Definitions
DCD- DRAM Chip Data
DCDT- DRAM Chip Data after Truncation
DS - Data sent over DRAM channels
DR - Data reconstructed at receiver end
MSE - Most similar entry
MSET - Truncated most similar entry
ZAC-DEST - Zero Aware Configurable Data Encoding by Skipping Transfer
MBDC - Modified Bitwise Difference Coder
DBI - Dynamic Bus Inversion
for all chip do
  Find MSE w.r.t DCDT
  {Truncated bits are not used for comparison}
  Check for Zeros
  if DCDT == 0 then
    return 0
  end if
  Check for ZAC-DEST
  Condition for ZAC-DEST to be True:
  Hamming Count of (MSET XOR DCDT) < Threshold and Tolerance bits are same
  if ZAC-DEST condition TRUE then
    DS : OHE index of MSE
    DR: MSET
  else
    Check for MBDC
    Condition for MBDC to be True:
    Hamm(DCDT) > Hamming Count of (MSET XOR DCDT) added to Hamming count of Index
    if MBDC condition TRUE then
      DS : DBI (MSET xored DCDT) and Index of MSE

      DR: DCDT
    else
      DS : DBI (DCDT)
      DR: DCDT
    end if
    Table Updated with DCDT
  end if
end for
```

VI. ZAC-DEST Circuit Implementation

The detailed circuit implementation of ZAC-DEST will be shown in this section.

ZAC-DEST Data Table: We start with modifying the BD-Coder design. Fig. 6a shows the NOR based binary content addressable memory (CAM) used to implement the data table. The data table in BD-Coder does the following **i)** stores recent data transfers, and **ii)** finds the most similar entry (MSE). **(i)** A 6-transistor based SRAM is used for storing the data in the CAM cell as shown in Fig. 6a. This allows reading and writing data into the data table using BL and BL'. For **(ii)** a 5-transistor comparator is used and search is performed using SL and SL'. The most similar entry is obtained using the



(a) (b) (c) (d)

Fig. 6: (a) Original CAM cell [14] (b) Modified CAM cell (c) Replica Word CAM Cell (d) Modified BD Coder (MBDC) Data Table for ZAC-DEST

comparator. For ZAC-DEST we have added one more feature **(iii)** for supporting truncation. For **(iii)** we add 1-transistor and an additional line, *truncation line* in the CAM module as shown in Fig. 6b. When the truncation line goes to 0, the NMOS connected to the line turns off and disconnects the comparator from the comparison line. This bit value connected to this line will then not be used for comparison. An additional row called the replica cell row is used in BD-Coder to count the number of 1's in the input data. The replica row will be used when the number of 1's is lesser than the MSE. We modify this row similarly as shown in Fig. 6c. The overall structure is shown in Fig. 6d is called the Modified BD-Coder (MBDC).

Zero Checker: The zero checker circuit is used to detect, in advance, if the input data to be sent is all 0's. The zero checker gives an output 1, only when all the 64 input bits are 0's. This is achieved using the NOR gate as shown in Fig. 7 (❶).

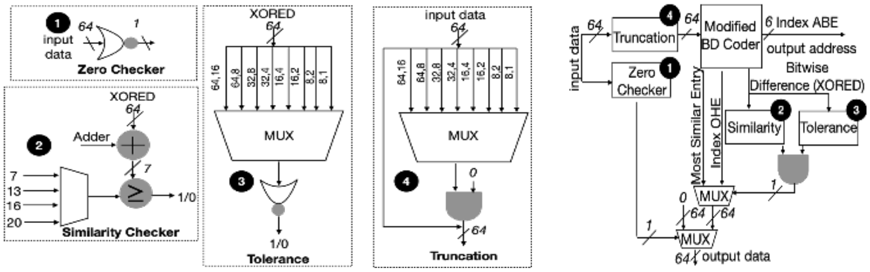
Similarity Checker: The similarity checker is shown in Fig. 7 (❷). It sums up the count of the number of dissimilar bits between input data and the most similar data. Depending upon the required similarity percentage, i.e. the percentage of bits which are supposed to be equal, 90%, 80%, 75% and 70% the numbers of dissimilar bits, irrespective of the bit positions, can be 7, 13, 16 and 20 respectively for 64-bit data. Hence, if an application requires a 90% similarity, the sum of the bitwise difference should be less than 7 for ZAC-DEST and so on.

Tolerance: The circuit design for introducing tolerance is shown in Fig. 7 (❸). At a time, we can transfer 64-bits of data. If we assume that this data contains eight chunks of 8-bit values, then tolerance will be applied to the MSB of each chunk. For a tolerance of 16, 2bit MSBs from each chunk cannot be approximated as shown in Fig. 8 (❶). Similarly if the values were of 16-bit each, i.e. there will be 4 chunks, then 4-bit MSBs of each of the chunks cannot be approximated as shown in Fig. 8 (❷).

Mostly the most significant bits (MSBs) are the ones that cannot tolerate approximation as described in Section VIII. So, for 64-bit data, ZAC-DEST allows for the introduction of tolerance in 8 or 16 bit granularities. Depending upon the bit-width the tolerance bits can be distributed. The tolerance bits can be selected as per need using MUX as shown in fig. 7a (❸). For a bit-width of N, ZAC-DEST can have tolerance in first N/4 or N/8 MSBs, where N can have values of 8, 16, 32 and 64. A single mismatch (*between data and the most similar entry*) in the tolerant bits will make the NOR gate output go low so that ZAC-DEST encoding is not applied and exact data is sent as shown in Fig. (7).

Truncation: The circuit design for introducing truncation is shown in Fig. 7 (❹). Similar to tolerance we allow support for a various bit-widths. ZAC-DEST allows a choice of N/4 and N/8 bit truncation for N equal to 8, 16, 32 and 64. The crucial difference is that truncation will make the bits to go to 0. For truncation of 16 and two different chunk sizes of 8 and 16, how the bits are approximated is shown in Fig. 8 (❸) and (❹).

Overall ZAC-DEST Encoding Scheme: The block diagram of ZAC-DEST encoder is shown in Fig. 7b. The input data to be sent over the channel is sent to zero checker. If the data is all 0's the zero checker output is 1 and all 0's are sent over the channel. At the receiver's end, all 0's are identified as such. The data is then forwarded to the MBDC to obtain MSE. MBDC also provides the One Hot Encoded (OHE) address and the Address Binary Encoded (ABE) address of the most similar data. The most similar data is XORed with the original data to get the bitwise difference, which is then provided as input to the similarity and the tolerance blocks. The similarity block checks for the required similarity criteria and will output a 1 if the criteria is satisfied. The tolerance checker will output a 1 only if *all* the bit positions selected for tolerance do not have a mismatch. If both similarity and tolerance criteria meet (ANDed output is 1), the One Hot Encoded Address is sent over the data lines (*ZAC-DEST Output*), else the MSE is sent (*BD-Coder Output*). Not if the original data has a lesser hamming weight that the MSE, the MBDC output the original data in place of MSE. A bit that informs the receiver whether the bits on the data lines represent the data or address. BD-Coder uses a single index line per chip to transfer the address. Since data table size is 64, a maximum of 6-bits are required to address the entire data table. The final output is sent after applying DBI.



(a) (b)

Fig. 7: (a) Sub Modules for ZAC-DEST (b) ZAC-DEST Encoder Circuit

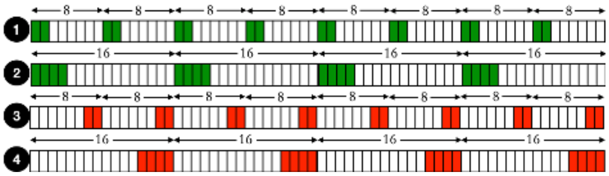


Fig. 8: Bits for Tolerance and Truncation 8, 2 and 16, 4. (Chunk Size = 8, 16, Bits within Chunk = 2, 4)

MBDC Overheads: We have derived energy values of BD-Coder in 65nm to be 7 pJ from [14]. The modification introduced in the data table is a single transistor that does not increase the energy significantly. We implemented the additional modules for ZAC-DEST in Verilog. We used 10,000 random inputs to generate the switching activity file (SAIF) using Synopsys VCS tool. We used the SAIF file generated in Synopsys Design Compiler to obtain the power consumption of the hardware. The energy consumption overhead of the entire sub module is 9% higher than that of the BD-Coder. The ZAC-DEST submodules, combined with BD-Coder consume 7.66 pJ per access. The latency for the data table in BD-Coder was 2.4 ns, while the entire ZAC-DEST sub module combined with BD-Coder has a latency of 3.4 ns. Even though the latency of MBDC increases as compared to BD-Coder, this is minimal as compared to the DRAM latency as also shown in [14]. The area overheads of the submodules are 15% higher as compared to the BD-Coder. The receiver of both ZAC-DEST and BD-Coder is similar. Thus, the energy consumption, latency, and area of ZAC-DEST receiver is similar to that of BD-Coder's receiver. These overheads are per DRAM chip, but overall overheads are still negligible as compared to DRAM as also shown in [14].

VII. Methodology

TABLE I: Encoding Schemes Under Evaluation

OHE	One-Hot Encoding of ZAC-DEST
BDE_ORG	Original Bitwise Difference Coder
BDE	Modified Bitwise Difference Coder
DBI	Dynamic Bus Inversion
ORG	Original Unencoded Data (Baseline)

ZAC-DEST improves channel-energy efficiency by transmitting approximate data in error resilient applications. Therefore we must choose a set of workloads that are amenable to approximation and have a quantifiable metric for measuring their output's quality. In this section, we describe the methodology used to evaluate the benefits of ZAC-DEST over existing models and the measure of quality used to understand the effect it has on the outputs. Their analysis is done by first converting their inputs to hexadecimal traces. We then emulate the transfer of data over the DRAM channels using these traces and use them to simulate the models described in Table. I. For ZAC-DEST models that involve approximating data accesses, we use the simulated traces to reconstruct approximate inputs that are used to run the workloads. This way, we compare the results of the workloads with the original input set and the reconstructed ones to get a measure of quality.

A. Workloads

The workloads chosen for evaluation are machine learning applications that use images as inputs. To evaluate different models we Fig. 9 (❶) read the images and store their pixel values in a row-major format of 64 bytes chunks to simulate a cache line Fig. 9 (❷) apply ZAC-DEST and the other models on the resulting trace to simulate data transferred to the memory controller while calculating the amount of hamming and switching energy Fig. 9 (❸) reconstruct images using the data received by the memory controller Fig. 9 (❹) use the reconstructed images to run respective models and study the effect on quality. The workflow is summarized in Fig. 9. Each workload has a different set of precision and accuracy metrics. Therefore we define *quality* for each workload to understand the effect of ZAC-DEST on the output. *Quality* is defined as the ratio of the output metric observed due to ZAC-DEST reconstructed images to that of the original images. As a result, a *quality* of 1 corresponds to the workload not expressing any degradation in its output and a *quality* of 0.5 indicates the workload experiencing a 50% degradation in the corresponding quality metric when compared to its non-approximated run. We now discuss each application in detail.

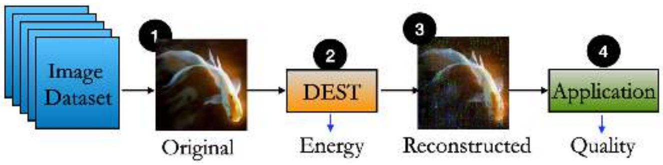


Fig. 9: Workflow of the methodology

1) ImageNet: CNNs from the ImageNet Challenge: Convolutional neural networks (CNNs) have been successfully applied in several image processing and computer vision tasks like image classification, object detection, etc [24], [39]–[43]. We use pre-trained pytorch [44] models of 15 of these CNNs. These 15 CNNs were trained using the ImageNet 2012 classification dataset [45] which contains 1.28 million images in the *training set.* We performed inferencing using 50K images in the *validation set* of ImageNet dataset. The *top-1* score matches the result with the highest probability against the target label. It is calculated as the number of times the top predicted label matches the target label, divided by the number of images evaluated.

Quality Metric: For these CNNs, the quality metric is a ratio of the *top-1 score* for inferencing with ZAC-DEST reconstructed images and the original images.

2) ResNet: Classification of the CIFAR dataset: Previous works [46], [47] have shown that training ML models on approximate data are instrumental in alleviating drops in quality which accompany the use of approximate data. We demonstrate this by allowing ResNet-110 [48], a PyTorch model from the ImageNet challenge, to be trained on ZAC-DEST reconstructed train images before recording its accuracy while inferencing using ZAC-DEST reconstructed test images. We carry out these experiments on the CIFAR-100 dataset [49].

Quality Metric: It is a ratio of the *top-1* score that we obtain from making predictions using reconstructed images (on the model that has been trained using reconstructed images) to that of the original data and model.

3) Quant: Color Quantization using K-Means: Considering that both *ResNet* and *ImageNet* consist of Neural Networks, we chose *Quant* as a workload for unsupervised tasks. This workload uses K-Means clustering to reduce the number of colours required to reproduce an image [50]. The algorithm reduces the large number of unique RGB values that are present in an image to a mere 64 with minimal degradation in image quality. This degradation is measured using the structural similarity (SSIM) [51] metric that quantifies image quality degradation with respect to the reference image. We use the images from the KODAK image dataset [52] and quantize the colour using *Scikit-Learn's KMeans* algorithm in Python.

Quality Metric: It is a ratio of SSIM obtained using reconstructed images compared to the original images.

4) Eigen: Using Eigen Vectors for face detection: Eigen is an unsupervised workload that uses Principal Component Analysis (PCA). PCA is a statistical procedure that uses transformations to convert a set of data into a set of uncorrelated variables. The task in this workload is to use PCA to decompose images present in the Yale Face Database [53] and then use these images for detecting faces.

Quality Metric: It is a ratio of the number of faces correctly detected using the reconstructed images when compared to using original images.

5) SVM: FMNIST image classification using Python: To compare the different encoding schemes on a sparse data, we choose an SVM model that learns the Fashion MNIST dataset [54], [55]. A Support-vector machine (SVM) is a machine learning model that uses a kernel to project data into higher dimensions following which it tries to learn a hyperplane that separates them distinctly. We choose FMNIST as it has a large number of sparse accesses, a behaviour that is exhibited by a number of contemporary workloads [13].

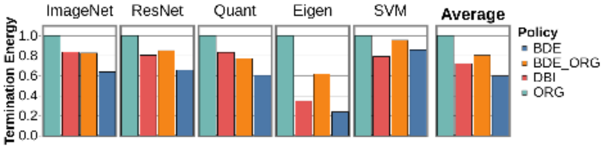


Fig. 10: Energy savings seen by all exact models

Quality Metric: It is the ratio of the number of articles of clothing correctly classified obtained using the reconstructed images when compared to using the original image set.

VIII. Evaluation

In the following section, we discuss the setup used to analyze the effects of ZAC-DEST and understand how the different parameters that are used to control ZAC-DEST's approximations affect the energy savings and output quality.

A. Setup

We use C++ scripts to parse memory traces and simulate ZAC-DEST, DBI and BDE. These scripts are used for the dual purpose of simulating data transmission over the DRAM channel and it being received by the controller. Simulating data transmission is used to record the hamming and switching counts that are used for the energy calculations. Simulating data received by the memory controller, on the other hand, is used for evaluating the effect of ZAC-DEST on the output quality of the workloads. Quality as defined in Section. VII refers to the ratio of *top-1* precision for *ImageNet* and *ResNet*, SSIM values for *Quant* and accuracy of workload task for *Eigen* and *SVM* obtained using ZAC-DEST reconstructed images when compared to using the original images. The analysis for termination and switching energy is done as described in Section I and III. These values are calculated based on the data transmitted over the data lines and the index/other metadata passed over the control lines. While presenting the results, we discuss the termination energy, as in most cases both termination and switching follow similar trends.

We perform experiments for 8 chip DRAMs, with each chip having a data table size of 64. The choice of the data table size is made based on the discussions in [14] where data table size up to 64 give a relatively large increase in energy benefits.

B. Comparing ORG, DBI, BDE_ORG and BDE

Fig. 10 shows a comparison of the savings for all the exact models, i.e., non-approximate models, observed when compared to the original non-encoded scheme. We observe that when encoded with DBI, the number of 1's being sent over the DRAM channel is reduced by 28%, which leads to a corresponding decrease in termination energy when compared to original memory accesses. It is interesting to observe that data encoded using BDE_ORG (proposed in [14]) performs worse than DBI in this aspect leading to only a 20% reduction while BDE with our proposed optimisations leads to a 41% reduction. We hypothesize that this occurs due to the data tables not being updated regularly, thus leading to suboptimal encodings. Also, the overheads of transferring the address of the index adds up to the termination energy. Due to this, workloads like Eigen which use images that are relatively uniform suffer the most - observing only a 39% reduction compared to 77% reduction produced by our version of BDE that updates the data table at every access. Hence, for the remainder of this section, we compare the different modes of ZAC-DEST with respect to our modifed BDE, which acts as a stricter baseline.

C. Effect of Similarity Limit

The *Similarity Limit* is a parameter that controls the amount of approximation being done to the workload. A *Similarity Limit* of 90 denotes a ZAC-DEST implementation where data accesses at least 90% of bits similar to the most similar entry would be approximated. We choose 90%, 80%, 75% and 70% (these correspond to a max of 7, 13, 16 and 20 bits being approximated) as similarity limits for analysis as they provide a varied view of the benefits that the approximation can yield. Allowing for more bits to be approximated (a similarity limit of < 60%) would lead to incorrect results while high thresholds (a limit of >90 %) would not result in any significant improvement in energy savings. For these experiments, both *Truncation* and *Tolerance* are kept as 0. Fig. 11 shows the behaviour of the CNNs from the ImageNet Challenge. There is a decline in *top-1* precision as we decrease the *Limit* due to loss in image quality. It is interesting to observe that the loss in accuracy in decreasing the *Limit* from 75 to 70 is much more significant than the other transitions, namely from 90 to 80 and 80 to 75. Fig. 12 shows the degradation of the reconstructed image caused due to the decrease in *Similarity Limit*. When we compare quality metrics across workloads in Fig. 13, we observe a similar trend of decreasing qualities with a decrease in *Similarity Limit*. While in the case of the Eigen, ResNet and SVM, it is gradual, ImageNet and Quant observe a sharper decline as the Limit decreases. It is important to note that for a *Similarity Limit* of 90 most of the workloads have a quality comparable to or more than 1 (where a quality of 1 means that there is no reduction in accuracy). Fig. 14, shows the effect of *Similarity Limit* on termination and switching energy for all the workloads. We observe that for a similarity limit of 90, as compared to BDE, ZAC-DEST reduces the termination and switching energies by 8% and 7% respectively. Decreasing the similarity limit (allowing more bits to be approximated) drastically reduces energy consumption. Comparing the energy consumption for *Similarity Limits* 90 / 80 / 75 / 70, we observe a reduction of 8% / 20% / 32% / 60% in termination energy compared to BDE, with a similar trend for switching energy. These are especially promising results as for *Similarity Limit* of 80 and 75 we see a reduction of 20% / 32% in the energy consumption when compared to BDE with qualities of 0.96/0.8.

D. Effect of Truncation and Tolerance

Fig. 15, shows the effect of *Truncation* and *Similarity Limit* on the energy and quality of workloads. We observe that increasing *Truncation* results in a decrease in energy at cost of quality. This is caused due to the increase in the number of bits being masked to zero caused by increasing *Truncation*. For a *Limit* of 80, increasing *Truncation* from 0 to 16 causes the savings of both termination and switching energy to increase from 20% to 68% as compared to BDE. But at the same time, we observe the quality to drop from 0.96 to 0.77. It is interesting to observe that the effect of Truncation becomes more prominent on lower Similarity Limits, with a drop in quality from 0.72 to 0.44 for a *Limit* of 70. Fig. 16, shows the effects of different parameters on workloads. Each data point is differentiated based on color, size and shape which correspond to *Truncation*, *Tolerance* and *Similarity Limit*, respectively. This plot helps visualize the combined effects that different parameters have on energy savings and quality degradation. Ideally, we would select parameters to minimize the energy consumption without compromising on quality, selecting design points on the *top-left* of the chart. We observe that decreasing *Limit* and increasing *Truncation* results in energy savings at the cost of quality, pushing design points to the *lower left*. We use *Tolerance* to balance the effect of those parameters. Increasing it (represented by increasing the size of the point) restricts the number of times ZAC-DEST can be true, thus resulting in lower energy savings but better quality (pushes the design points to the *top right*). Just as in the case of *Truncation*, *Tolerance* does not affect the quality and energy savings by a large amount at higher *Similarity Limits* (where the design points of different sizes and colours are closer to each other), but increases as we lower the *Limit*.

E. Using Reconstructed Images for Training

The workload *ResNet* is used to demonstrate that training models on images reconstructed using ZAC-DEST, i.e., on approximate images, would alleviate some of the quality degradations. We observe this behaviour when we compare *ImageNet* and *ResNet* in Fig. 17. We compare two different models of *ResNet* - one that has been trained using the reconstructed images while the other has been trained using the original dataset. Fig. 18 compares the quality of the two models based on the effects of *Similarity Limit* and *Truncation*. We observe that the drop in quality is smaller in the case of *ResNet* trained on approximation images as compared to the model that isn't. This motivates training models with the ZAC-DEST reconstructed data when feasible to improve the accuracy of the application. In some configurations, we observe an improvement of up to 9× in output quality. Hence depending on the application, in case where higher accuracy is needed ZAC-DEST can be used both while training and inference.

F. Effect of ZAC-DEST on Output Quality

Fig. 15 provides an insight into how amenable each workload is towards approximation. We observe that *ResNet* and

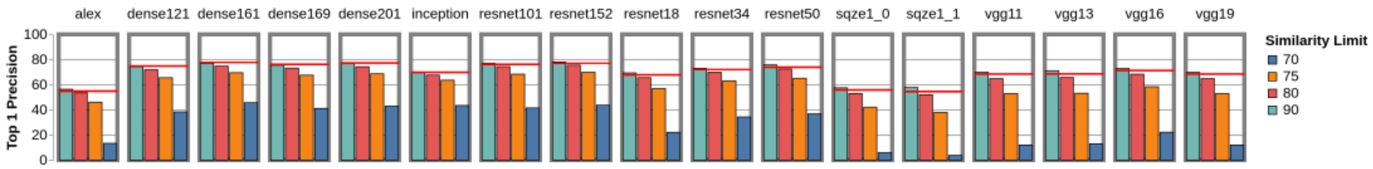
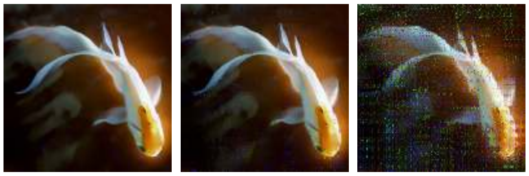


Fig. 11: Effect of Similarity Limit on *top-1* precision for neural nets in the ImageNet Challenge. The red line denotes the original accuracy



(a) Limit = 90 (b) Limit = 80 (c) Limit = 70

Fig. 12: Reconstructed images for different Similarity Limits

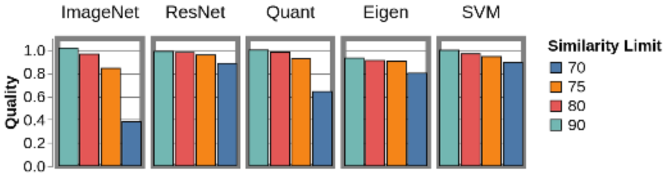


Fig. 13: Effect of Similarity Limit on output quality for all workloads.

SVM are more tolerant to higher levels of approximation compared to *ImageNet* and *Quant* despite observing similar benefits in energy consumption. Fig. 17 shows this analysis for *ImageNet* and *ResNet* as representatives of the different behaviours. Here, *ImageNet* dips sharply at higher approximation configurations while *ResNet* manages to remain stable, i.e., it does not experience as large a drop in quality. This behaviour is directly related to the nature of each workload. For *Quant* large variations in the image can cause the K-Means algorithm to quantize colours in a poor manner, leading to lower values of SSIM. *SVM*, on the other hand, being a generally robust model classifying a relatively simple data set is amenable to approximations.

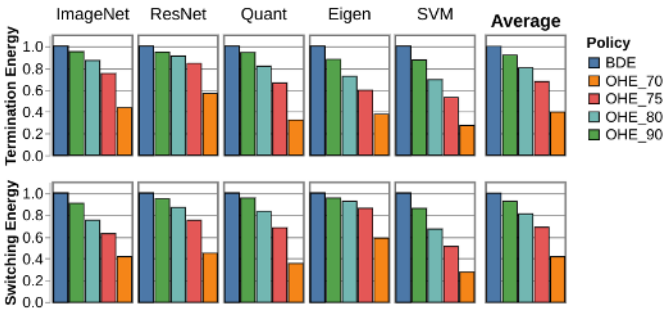


Fig. 14: Energy savings observed by all models with ZAC-DEST while varying its similarity limit.

G. Effect of ZAC-DEST on Both Weights and Images

We now study the effect of applying ZAC-DEST on both the weights and the images to study the impact on energy and quality. For approximating weights we follow a similar strategy as approximating images. The weights are represented using the IEEE 754 format as shown in Fig. 19. It is important to note that for weights it is imperative that we do not approximate the exponent and sign bits as it introduces large errors into the calculations. We evaluated and observed that approximating even the last bit of exponent leads to 60% deterioration in output quality. Thus, based on structure of the traces and the DRAM data layout (detailed in Fig 3) we set the tolerance sign and exponents bits are not approximated. Fig. 20, shows us the effect of ZAC-DEST on termination energy and quality when both the images and weights for the model "InceptionNet" from the ImageNet workloads for varying *Similarity Limits*. For *Similarity Limits* 70 / 65 / 60 / 50, we observe a reduction of 10% / 40% / 59% / 60% in termination energy (due to weights) compared to BDE. We see that for such savings in energy the quality reduces from 0.92 to 0.57 (for a fixed image *Similarity Limit* of 90%). Fig. 21, shows us that the effect of ZAC-DEST on ResNet-110 when we approximate both weights and images during both training and testing. Similar to what was discussed in Section VIII-E, we see that training with ZAC-DEST improves the output quality. Such comparisons would be useful for determining the correct modes to be used for different models to obtain the desired output quality. Based on the whether weight or image transfer dominates depending upon the hardware configurations and the application, for acceptable quality drops one among the variety of configuration can be chosen.

H. Instances of Encoding During Memory Transfers

In Fig. 22 we visualize the frequency with which the data is encoded for a particular encoding scheme for both weights and images. We compare these values when we use BDE and ZAC-DEST (which is built over an optimized version of BDE). As compared to the BDE proposed in [14], we have added two modifications. i) We handle 0's separately, and ii) We have a stricter condition for BDE as we sum the hamming weight of both the data and index values to evaluate the BDE condition. In [14], only hamming weight of data is considered and not the index values. In both cases, we see that a majority of the accesses are encoded using either of the schemes, with only an average of 6.5% and 6.6% of the accesses not being encoded using ZAC-DEST and BDE respectively. This result demonstrates the high similarity between the transferred data

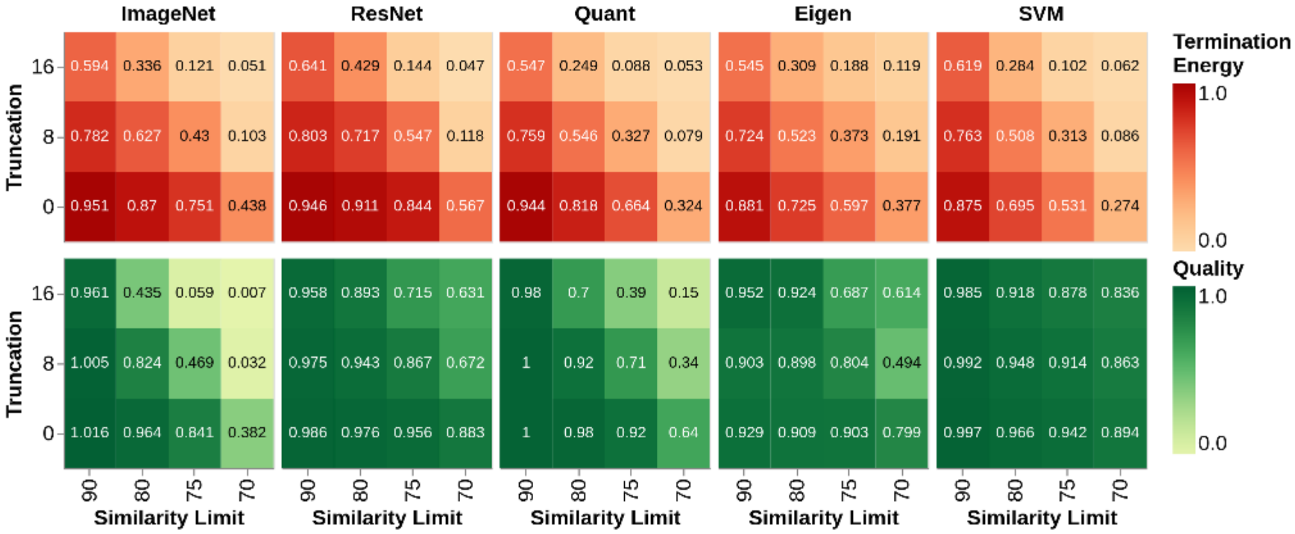


Fig. 15: Effect of Truncation and Similarity Limit on Termination Energy and Quality (Switching Energy follows similar trends). Each number in the box is the value of the metric.

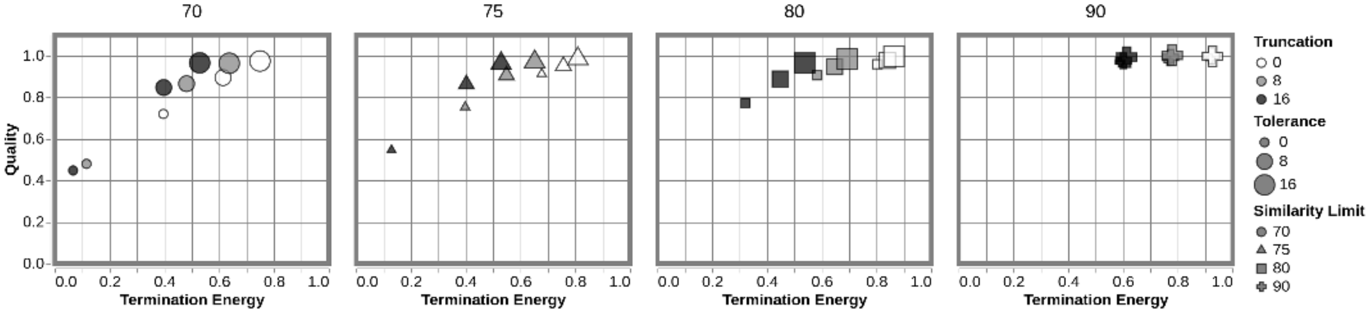


Fig. 16: Effect of ZAC-DEST on Quality and Energy as an average over all workloads. Darker points correspond to higher Truncation, larger points correspond to larger tolerance, and more number of sides correspond to larger similarity limits.

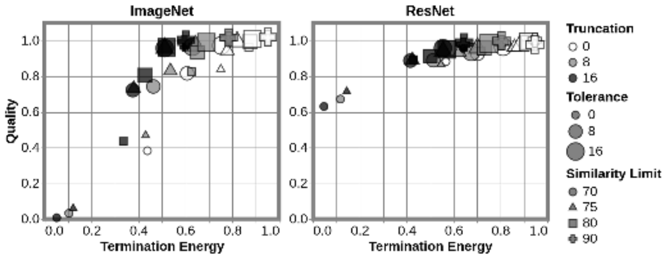


Fig. 17: Effect of ZAC-DEST on the ImageNet and ResNet

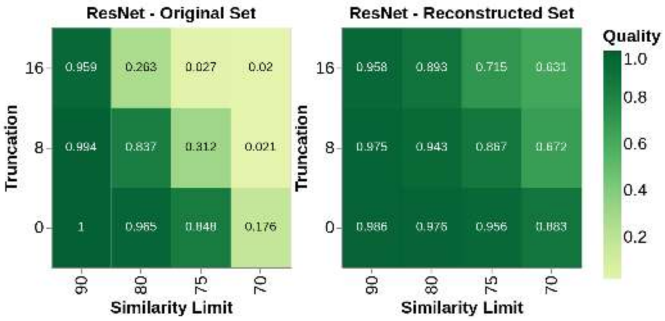


Fig. 18: Comparing ResNet-110 for different training sets

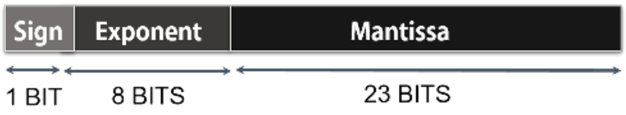


Fig. 19: 32-bit Floating Point representation in IEEE 754

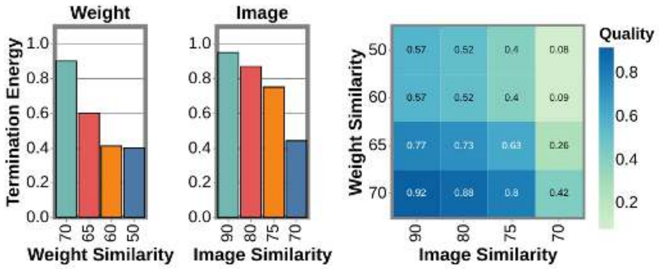


Fig. 20: Comparing InceptionNet for both weight and image approximation

and also speaks to the fact that to improve over BDE, whose coverage is already very high, it is imperative to implement schemes that have a better encoding mechanisms.

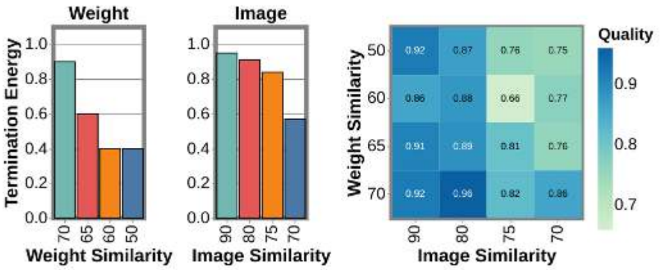
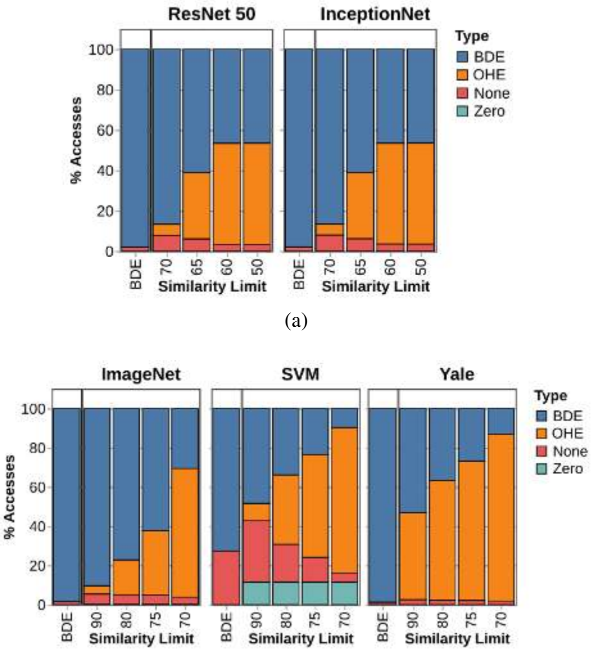


Fig. 21: Comparing ResNet-110 for both weight and image approximation with training



(a)

(b)

Fig. 22: Frequency with which data is encoded using ZAC-DEST and BDE during a) Weight Transfers b) Image Transfers for varying similarity limits

IX. Related Work

In this section, we briefly present work in the area of data encoding/compression and approximation.

Data Encoding: ZAC-DEST is an approximate encoding model that develops on the state of the art data encoding schemes BD-Coder [14] and DBI [23] to give higher energy benefits. Various other works in the past have focused on energy reduction using data encoding in DRAM channels. Yan et al. [38] proposed an encoding scheme which exploits temporal locality of data words. It uses one-hot encoding to send frequently occurring values. Suresh et al. [56] proposed VALVE, a variable-length bit pattern for encoding and decoding. it matched partial data and sent either one-hot code or two-hot code masks for that partial data word while the rest was sent unencoded. Lee et al. proposed SILENT [57], a data encoding scheme which focused on reducing the switching energy by reducing the hamming weight of the data words by exploiting similarity between current and the previously accessed data word. Lee et al. [13] proposed a data encoding scheme for GPUs, which reduces the number of 1's in the data sent over the channel. It took special care of zero data by encoding it with a constant with reduced hamming weight. While the technique works well for GPU applications it has been shown to perform poorly for CPU applications.

Stanley-Marbell et al. [58] propose a value-deviation-bounded serial (VDBS) approximate encoding scheme that significantly reduces the switching observed for data. Pekhimenko et al. [59] propose Toggle-Aware Compression schemes that reduce switching count impact of the data compression algorithms. Both schemes can be used to assist in alleviating the increase in switching counts caused due to BDE in certain workloads (as seen in Fig. 10).

Approximation in Hardware: Various works have focused on the introduction of approximation to DRAMs, caches and processors [60]. Sampson et al. [61]–[63] have proposed frameworks for annotating and identifying regions in the program that are amenable to approximation and hardware mechanisms for memories that result in energy savings at the cost of output quality. Liu et al. [64] use application-level input to effectively reduce the refresh rate of DRAMs, which may result in data corruption. Miguel et al. [65] proposed Load Value Approximation (LVA) and Thwaites et al. [66] proposed rollback-free value prediction, techniques that approximately predict the data to be accessed during a load. As such behaviour results in increased number of predictions being made and reduces the number of times the memory is accessed. These works focus on introducing approximation in a method that is different from ZAC-DEST, which makes it entirely possible to stack them with ZAC-DEST to leverage more benefits.

Miguel et al. [67] propose Doppelganger, an approximate cache mechanism that associates multiple similar entries together to reduce the amount of data stored. Boyapati et al. [21] propose APPROX-NoC, a mechanism for network-on-chip (NoC) devices to eliminate the transmission of similar cache blocks by encoding them to similar data patterns. Both these works can function in synergy with ZAC-DEST.

X. Conclusion

In this paper we propose DEST, an approximate data encoding scheme to reduce DRAM channel energy consumption for error resilient applications. DEST works by exploiting data similarity and the error resilience of applications leading to reduction in hamming weight (number of 1's in data word). DEST builds up on top of existing data encoding schemes namely BD-Coder and DBI. We applied DEST on five different set of machine learning applications and observed a reduction of 40% and 37% in termination and switching energy respectively as compared to the state of the art data encoding technique with an average output quality loss of 10%. DEST, if applied on both training and testing can significantly outperform designs that apply DEST only during testing, but are trained on non-DEST encoded data.

XI. Acknowledgements

This work is supported through grants received from Intel, SMDP-C2SD and YFRF/PhD fellowship under Visvesvaraya PhD scheme from the Ministry of Electronics and IT, and through SERB grants CRG/2018/005013 and MTR/2019/001605. This work is partially supported through Ashoka University startup and Huawei Technologies India grants to Manu Awasthi. This work is also supported through grant received from SRC Grant 2980.001 and 2020-IR-3005.

References

[1] B. Jacob, S. Ng, and D. Wang, *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.

[2] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, *et al.*, "Bigdatabench: A big data benchmark suite from internet services," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 488–499, IEEE, 2014.

[3] V. Young, C. Chou, A. Jaleel, and M. Qureshi, "Accord: Enabling associativity for gigascale dram caches by coordinating way-install and way-prediction," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 328–339, IEEE, 2018.

[4] S. Li, D. Reddy, and B. Jacob, "A performance & power comparison of modern high-speed dram architectures," in *Proceedings of the International Symposium on Memory Systems*, pp. 341–353, ACM, 2018.

[5] R. Elmore, K. Gruchalla, C. Phillips, A. Purkayastha, and N. Wunder, "Analysis of application power and schedule composition in a high performance computing environment," tech. rep., National Renewable Energy Lab.(NREL), Golden, CO (United States), 2016.

[6] L. A. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 4, no. 1, pp. 1–108, 2009.

[7] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory power management via dynamic voltage/frequency scaling," in *Proceedings of the 8th ACM international conference on Autonomic computing*, pp. 31–40, ACM, 2011.

[8] P. Behnam and M. N. Bojnordi, "Stfl-ddr: Improving the energy-efficiency of memory interface," *IEEE Transactions on Computers*, vol. 69, no. 12, pp. 1823–1834, 2020.

[9] J. STANDARD, "Pod15 - 1.5v pseudo open drain i/o."

[10] J. STANDARD, "Low power double data rate 4 (lpddr4)."

[11] K. Sohn, T. Na, I. Song, Y. Shim, W. Bae, S. Kang, D. Lee, H. Jung, S. Hyun, H. Jeoung, *et al.*, "A 1.2 v 30 nm 3.2 gb/s/pin 4 gb ddr4 sdram with dual-error detection and pvt-tolerant data-fetch scheme," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 168–177, 2013.

[12] Y.-C. Cho, Y.-C. Bae, B.-M. Moon, Y.-J. Eom, M.-S. Ahn, W.-Y. Lee, C.-R. Cho, M.-H. Park, Y.-J. Jeon, J.-O. Ahn, *et al.*, "A sub-1.0 v 20nm 5gb/s/pin post-lpddr3 i/o interface with low voltage-swing terminated logic and adaptive calibration scheme for mobile application," in *2013 Symposium on VLSI Circuits*, pp. C240–C241, IEEE, 2013.

[13] D. Lee, M. O'Connor, and N. Chatterjee, "Reducing data transfer energy by exploiting similarity within a data transaction," in *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on*, pp. 40–51, IEEE, 2018.

[14] H. Seol, W. Shin, J. Jang, J. Choi, J. Suh, and L.-S. Kim, "Energy efficient data encoding in dram channels exploiting data value similarity," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 719–730, 2016.

[15] B. Zeinali, D. Karsinos, and F. Moradi, "Progressive scaled stt-ram for approximate computing in multimedia applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 7, pp. 938–942, 2017.

[16] S. Venkataramani, X. Sun, N. Wang, C.-Y. Chen, J. Choi, M. Kang, A. Agarwal, J. Oh, S. Jain, T. Babinsky, *et al.*, "Efficient ai system design with cross-layer approximate computing," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2232–2250, 2020.

[17] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2015.

[18] D. J. Pagliari, E. Macii, and M. Poncino, "Approximate energy-efficient encoding for serial interfaces," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 22, no. 4, p. 64, 2017.

[19] D. J. Pagliari, E. Macii, and M. Poncino, "Serial t0: Approximate bus encoding for energy-efficient transmission of sensor signals," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2016.

[20] Y. Kim, S. Behroozi, V. Raghunathan, and A. Raghunathan, "Axserbus: A quality-configurable approximate serial bus for energy-efficient sensing," in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, IEEE, 2017.

[21] 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 *ACM SIGARCH Computer Architecture News*, vol. 45, pp. 666–677, ACM, 2017.

[22] J. R. Stevens, A. Ranjan, and A. Raghunathan, "Axba: an approximate bus architecture framework," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2018.

[23] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power i/o," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 3, no. 1, pp. 49–58, 1995.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*, p. 113, ACM, 2013.

[26] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *ACM SIGPLAN Notices*, vol. 47, pp. 301–312, ACM, 2012.

[27] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 449–460, IEEE Computer Society, 2012.

[28] Y. Park, J. Qing, X. Shen, and B. Mozafari, "Blinkml: Approximate machine learning with probabilistic guarantees y," tech. rep., Technical Report http://web. eecs. umich. edu/mozafari/php/data/uploads . . . , 2018.

[29] C.-Y. Chen, J. Choi, K. Gopalakrishnan, V. Srinivasan, and S. Venkataramani, "Exploiting approximate computing for deep learning acceleration," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 821–826, IEEE, 2018.

[30] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5918–5926, 2017.

[31] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International Conference on Machine Learning*, pp. 1737–1746, 2015.

[32] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 243–254, IEEE, 2016.

[33] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[34] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 1–13, 2016.

[35] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pp. 267–278, IEEE, 2016.

[36] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[37] S. Ghose, A. G. Yaglikçi, R. Gupta, D. Lee, K. Kudrolli, W. X. Liu, H. Hassan, K. K. Chang, N. Chatterjee, A. Agrawal, *et al.*, "What your dram power models are not telling you: Lessons from a detailed experimental study," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 3, p. 38, 2018.

[38] J. Yang, R. GuptaF, and C. Zhang, "Frequent value encoding for low power data buses," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 9, no. 3, pp. 354–384, 2004.

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

[40] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[43] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[44] Pytorch, “models.” https://github.com/pytorch/vision/tree/master/torchvision/models.

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[46] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[47] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[48] W. Yang, “Pytorch classification.” https://github.com/bearpaw/pytorch-classification. 2019 (accessed April 10, 2020).

[49] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[50] “Color quantization using k-means.” https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html. (accessed April 16, 2020).

[51] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[52] “Kodak image dataset.” http://www.cs.albany.edu/∼xypan/research/snr/Kodak.html. 2011 (accessed April 16, 2020).

[53] “Yale face database b.” http://vision.ucsd.edu/content/yale-face-database. 2013 (accessed April 16, 2020).

[54] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.

[55] Z. Wen, J. Shi, Q. Li, B. He, and J. Chen, “ThunderSVM: A fast SVM library on GPUs and CPUs,” *Journal of Machine Learning Research*, vol. 19, pp. 797–801, 2018.

[56] D. C. Suresh, B. Agrawal, W. Najjar, and J. Yang, “Valve: variable length value encoder for off-chip data buses,” in *2005 International Conference on Computer Design*, pp. 631–633, IEEE, 2005.

[57] K. Lee, S.-J. Lee, and H.-J. Yoo, “Silent: serialized low energy transmission coding for on-chip interconnection networks,” in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pp. 448–451, IEEE Computer Society, 2004.

[58] M. Rinard and P. Stanley-Marbell, “Reducing serial i/o power in error-tolerant applications by efficient lossy encoding,” in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2016.

[59] G. Pekhimenko, E. Bolotin, N. Vijaykumar, O. Mutlu, T. C. Mowry, and S. W. Keckler, “A case for toggle-aware compression for gpu systems,” in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 188–200, IEEE, 2016.

[60] S. Koppula, L. Orosa, A. G. Yağlıkçı, R. Azizi, T. Shahroodi, K. Kanellopoulos, and O. Mutlu, “Eden: Enabling energy-efficient, high-performance deep neural network inference using approximate dram,” in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 166–181, 2019.

[61] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, “Approximate storage in solid-state memories,” *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 3, pp. 1–23, 2014.

[62] A. Sampson, A. Baixo, B. Ransford, T. Moreau, J. Yip, L. Ceze, and M. Oskin, “Accept: A programmer-guided compiler framework for practical approximate computing,” *University of Washington Technical Report UW-CSE-15-01*, vol. 1, no. 2, 2015.

[63] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, “Enerj: Approximate data types for safe and general low-power computation,” *ACM SIGPLAN Notices*, vol. 46, no. 6, pp. 164–174, 2011.

[64] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, “Flikker: saving dram refresh-power through critical data partitioning,” in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, pp. 213–224, 2011.

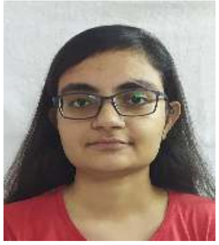
[65] J. San Miguel, M. Badr, and N. E. Jerger, “Load value approximation,” in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 127–139, IEEE, 2014.

[66] B. Thwaites, G. Pekhimenko, H. Esmaeilzadeh, A. Yazdanbakhsh, J. Park, G. Mururu, O. Mutlu, and T. Mowry, “Rollback-free value prediction with approximate loads,” in *2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pp. 493–494, IEEE, 2014.

[67] J. S. Miguel, J. Albericio, A. Moshovos, and N. E. Jerger, “Doppelgänger: a cache for approximate computing,” in *Proceedings of the 48th International Symposium on Microarchitecture*, pp. 50–61, 2015.

Chandan Kumar Jha recieved his B.Tech Degree from National Institute of Technology Meghalaya, Shillong, India in 2015. He is currently a PhD student in Electrical Engineering at Indian Institute of Technology Gandhinagar, India. His research interest include approximate circuits, approximate architectures and energy efficient systems design. He was the recipient of Merit Scholarship during his B.Tech. He was the recipient of Visvesvaraya PhD Fellowship from 2015 to 2019. He is currently an Intel PhD fellow from 2019 onwards.

Shreyas Singh received his B.Tech. Degree in Computer Science and Engineering from the Indian Institute of Technology, Gandhinagar, India in 2020. He will be pursuing a Ph.D. in Computer Science at the University of Utah in the United States of America. His research interest includes using approximate hardware and other emerging technologies for improving the computing stack.



Riddhi Thakker received her B.Tech Degree in Information and Communication Technology from Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat in 2020. She is currently working as an Applications Engineer in Oracle India. Her research interest includes approximate computing, parallelization using GPU, speech processing and machine learning.

Manu Awasthi is an Associate Professor at Ashoka University, India. He received his BTech degree from the Indian Institute of Technology, Varanasi, India, and the PhD degree in computer science from the University of Utah. His research interests are performance evaluation, memory and storage architectures, and characterization of datacenter applications.

Joycee Mekie is an Assistant Professor at the Electrical Engineering Department, IIT Gandhinagar. She received her bachelor's and master's degrees in electrical engineering from the M. S. University of Baroda in 1997 and 1999, respectively, and the Ph.D. degree in electrical engineering from IIT Bombay in 2009. Her research interests include approximate computing, circuits for space applications, asynchronous systems. She has served as the reviewer for several journals, including IEEE TCAS I, IEEE TCAS II AND IEEE TCAD.