EDEN

Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM

Skanda Koppula **Lois Orosa** A. Giray Yaglikci
Roknoddin Azizi Taha Shahroodi Konstantinos Kanellopoulos Onur Mutlu

ETH zürich SAFARI

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains

Problem: ***DRAM*** *can increase the* ***energy consumption*** *and the* ***execution latency*** *of DNN inference*

Goal: reduce energy and latency of DNN workloads by using **DRAM with reduced voltage and timing parameters (approximate DRAM)**

Challenge: approximate DRAM introduce **bit errors**

EDEN: Enabling Efficient DNN Inference Using Approximate DRAM

- **Tolerate approximate DRAM bit errors** by retraining the DNN for a target accuracy
- **Map DNN data types** with different error tolerance **to DRAM partitions** with different error rates

Results:

- Average **21%/37%/31% DRAM energy savings** on CPU/GPU/DNN-accelerators
- Average **8% speedup** on CPU

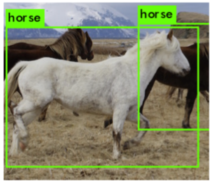
EDEN is **applicable to other DRAM parameters** and **memory technologies**

Outline

1. **Motivation and Problem**
2. DNN and DRAM Background
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Motivation

Deep neural networks (DNNs) are critical in computer vision, robotics, and many other domains

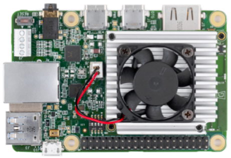


Modern platforms for DNN inference use DRAM

Mobile CPUs

GPUs

Data Center Accelerators


Edge Device Accelerators

Challenges of DNN Inference

DRAM has high energy consumption

- **2**

DRA

- **R** ome
  DNN workloads on CPU

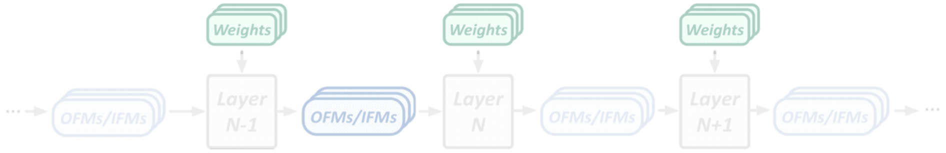
How can we **reduce DRAM energy** and **improve DRAM performance** for DNN inference?

Outline

1. Motivation and Problem
2. **DNN and DRAM Background**
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Deep Neural Network Inference

- **DNN Inference:** classifies inputs that the network has never seen
- **Three main data types** compose a **DNN layer**:
  1. **Weights**
  2. **Input Feature Maps (IFMs)**
  3. **Output Feature Maps (OFMs)**
- Modern DNNs can have **hundreds of layers** and between **10^5** and **10^9** weights
- Large DNN weight/IFM counts enable **high learning capacity**



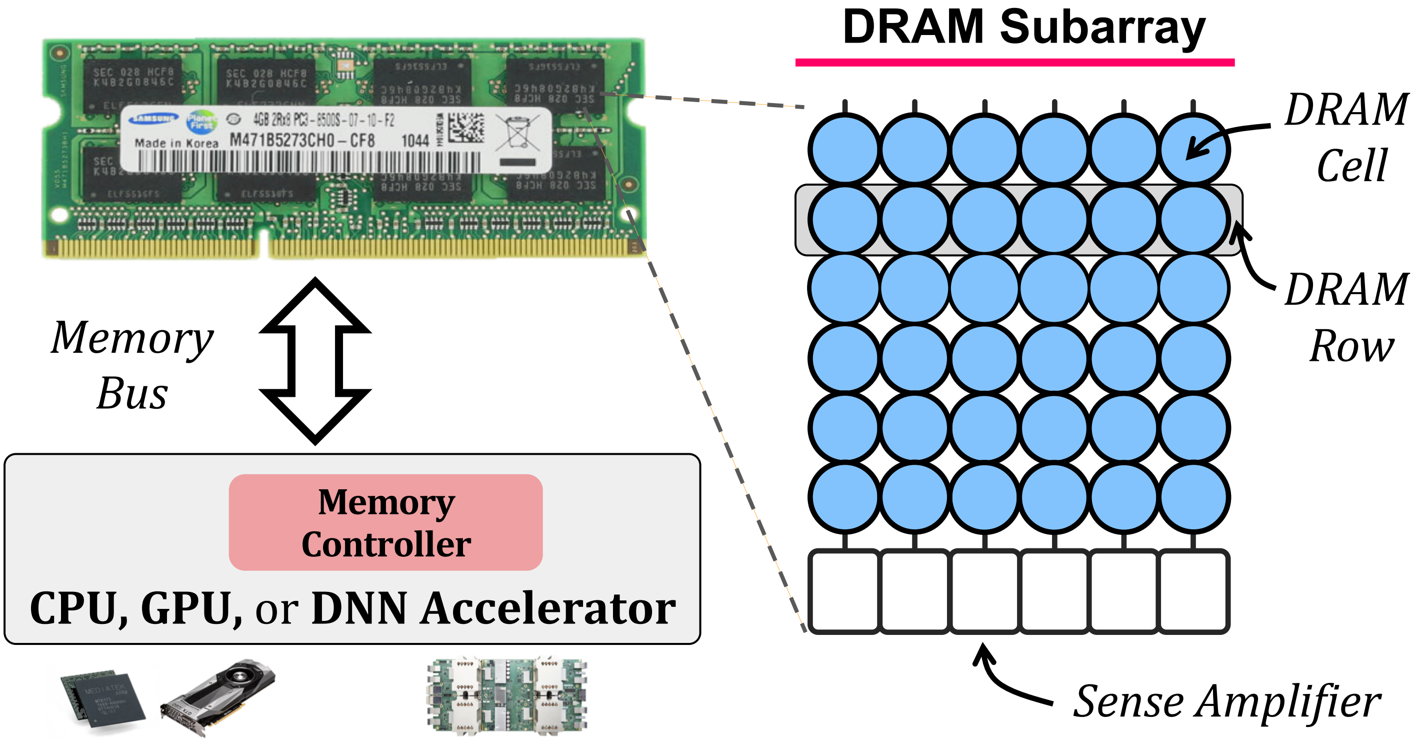
Deep Neural Network Training

- Before inference, the **DNN must be trained**

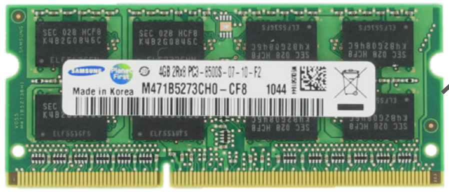
- Training is the process of estimating the **best set of weights** that **maximize the accuracy of DNN inference**

- **Two passes:**
  1. **Forward pass**:
     - The same process as DNN inference
     - Compare the **DNN results** with **golden results**
     - Calculates the **loss** (or error) of the model
  2. **Backward pass**:
     - **Update the weights** towards values that make the DNN to get more **accurate results**

DRAM Basics


DRAM Subarray
DRAM Cell
DRAM Row
Sense Amplifier
Memory Bus
Memory Controller
CPU, GPU, or DNN Accelerator

DRAM Parameters



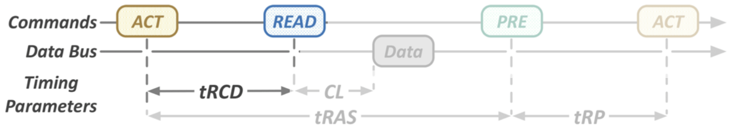
DRAM operates at a **standard voltage** (e.g., DDR3 at 1.35V)

Memory Bus

Memory Controller (MC)

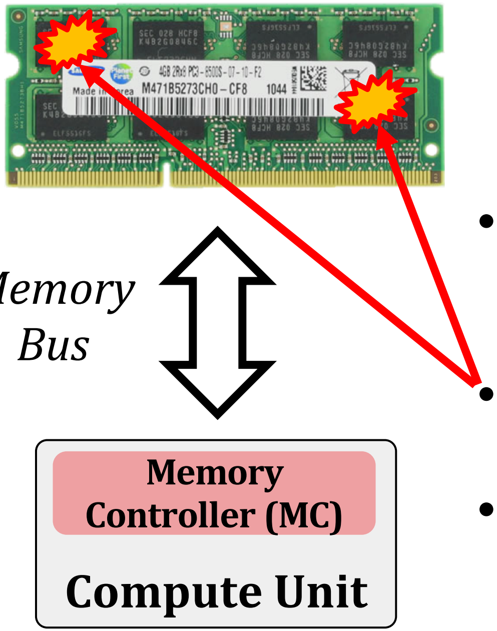
CPU, GPU, or **DNN Accelerator**

Accessing data follows a **sequence of MC commands** with **standard timing parameters**



SAFARI

Approximate DRAM



- Approximate DRAM operates with **reduced parameters**
  - Voltage
  - Timing parameters
- It enables to reduce DRAM **energy and latency**
- It might **introduce bit errors**
- It can only store data that is **tolerable to errors**

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. **EDEN Mechanism**
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

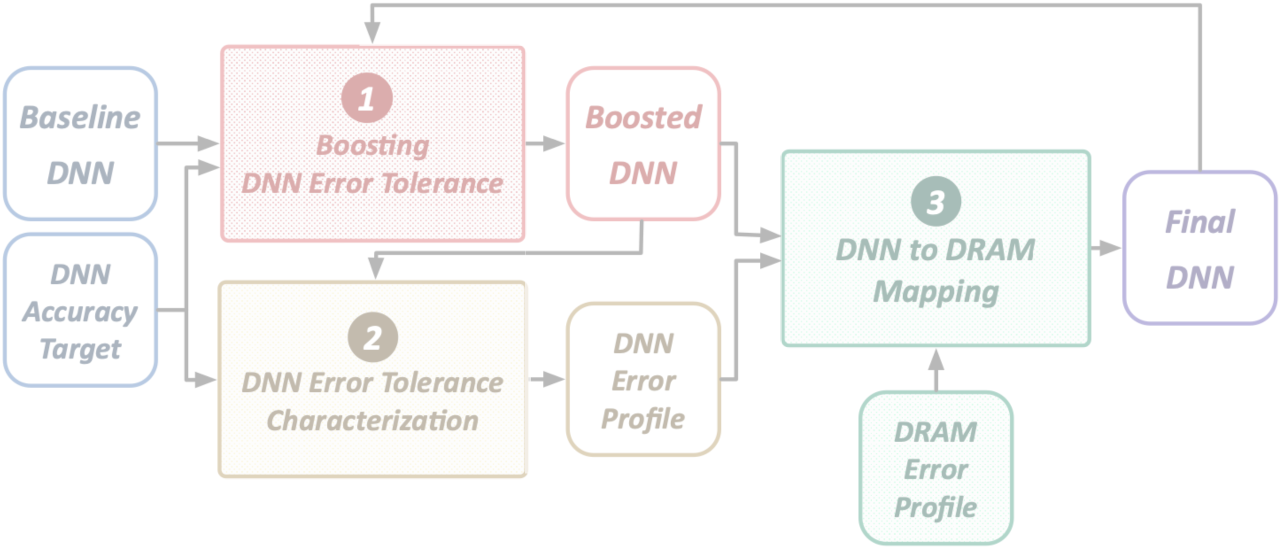
Observations

Approximate DRAM
can provide **higher energy-efficiency** and **performance**
for **error-tolerant DNN inference** workloads

EDEN: Overview

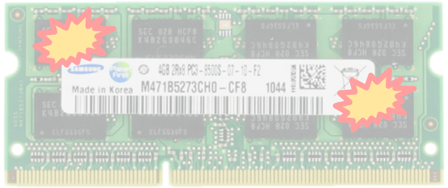
Key idea: Enabling **accurate, efficient** DNN inference using **approximate DRAM**

EDEN is an **iterative** process that has **3 key steps**

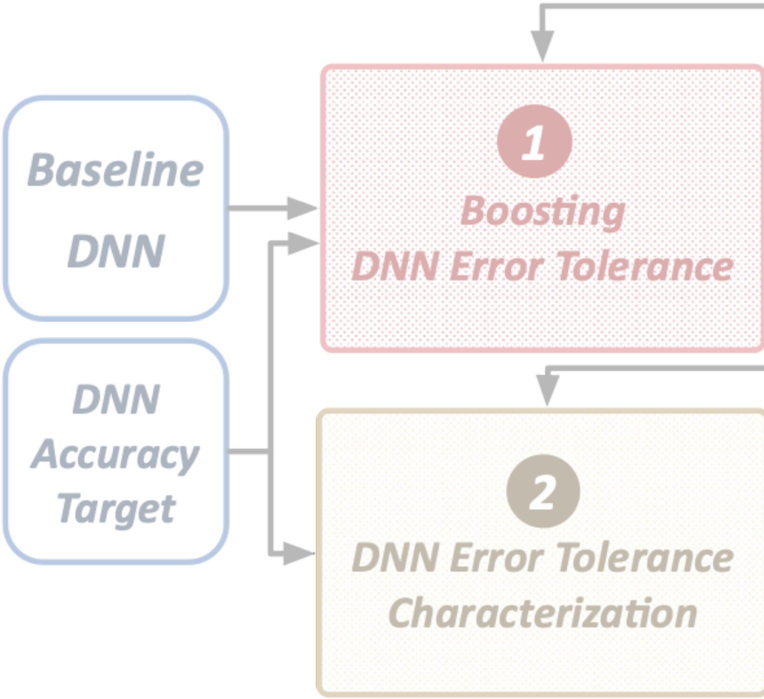


SAFARI

EDEN: Inputs



Target
approximate DRAM device


Baseline DNN
DNN Accuracy Target
1
Boosting DNN Error Tolerance
2
DNN Error Tolerance Characterization

Step 1: Boosting DNN Error Tolerance

Goal: Maintain accuracy when the DNN is exposed to bit errors

Key idea: **Retrain** the DNN with **approximate DRAM** to adapt the DNN to unreliable DRAM cells

- **Forward pass** uses approximate DRAM
  - To get the **output and the loss** in the presence of bit errors
- **Backward pass** uses DRAM with standard parameters
  - To **update weights reliably**

Step 1: Challenges

1. The **error tolerance** of common **DDNs is not sufficient** to enable significant DRAM parameter reductions
   - For **high bit error rates**, the **accuracy collapses** at the start of retraining

2. **Bit errors** might affect **most significant bits** of the weights (exponent bits)
   - Can create **enormously large values that propagates** through the DNN layers

These issues can create accuracy collapse in the DNN

Step 1: Solutions

1. Gradually **increase the bit error rate** of approximate DRAM during retraining
   - The goal is to **build error tolerance**
   - **Avoids accuracy collapse in retraining**
   - We increase the BER by decreasing the timing parameters

2. **Correcting implausible values** that are **out-of-range**
   - **The valid range** is calculated during training of the baseline DNN
   - **E.g.,:** weights in SqueezeNet are within the range [-5,5]
   - **Zero out** the **out-of-range values**
   - The mechanism can be implemented in the **memory controller** at low cost

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Step 2: DNN Error Tolerance Characterization

Goal: **Find the highest tolerable error rates** of the DNN and the corresponding DRAM parameters

Key idea: Systematically **measure error resilience** of **each DNN data type** on the approximate DRAM

Two ways to perform the **DNN characterization**:

1. **Coarse-grained**
   - Determine the highest **BER** that can be applied **uniformly to the entire DNN**
2. **Fine-grained**
   - Determine the maximum tolerable **BER** for each individual **DNN data type and layer**

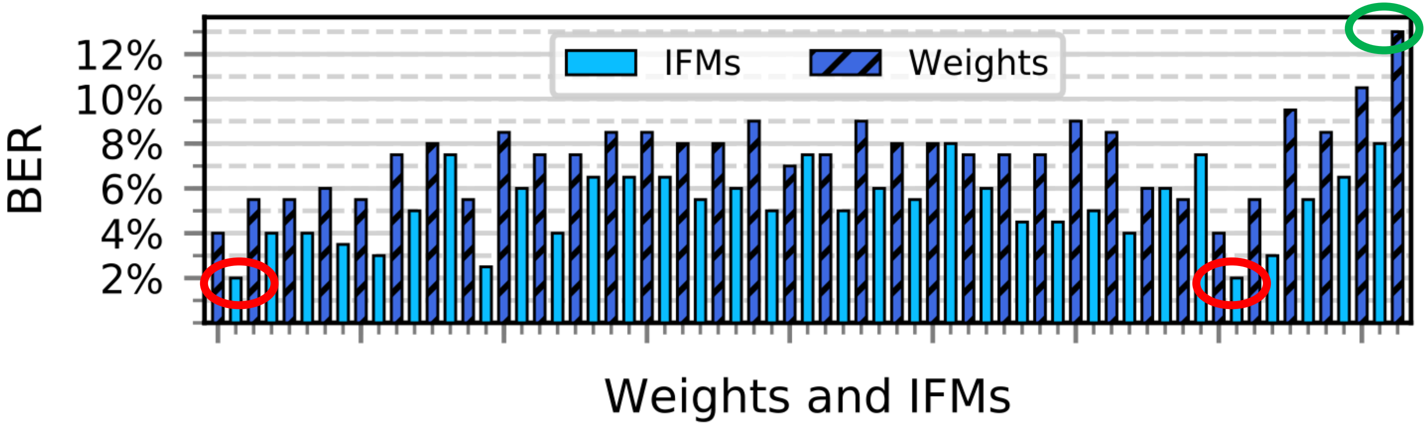
Adjust the **BER** by **tuning the parameters** of approximate DRAM

Step 2: Coarse-Grained VS Fine-Grained

- **Coarse-Grained** DNN Error Tolerance Characterization
  - Can be applied in **off-the-shelf DRAM** ☺
  - **Limited** voltage/latency **reduction** ☹

- **Fine-Grained** DNN Error Tolerance Characterization
  - Requires **non-commodity DRAM** to reduce some parameters (e.g., V_{dd}) ☹
  - More **aggressive voltage/latency reduction** ☺

Step 2: Example ResNet-50 Characterization



- **Error tolerance** of DNN layers **varies greatly**
- **Coarse-Grained:** maximum **tolerable BER** is **2%**
- **Fine-Grained: tolerable BER** between **2%** and **13%**

SAFARI

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. **DNN to DRAM Mapping**
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Step 3: DNN to DRAM Mapping

- **Goal: Match error tolerance** of DNN with DRAM **bit error rates (BER)**

- **DNN to DRAM mapping:**
  1. **Coarse-grained:** assign the single best **DRAM voltage/latency** value that **meets the target DNN accuracy**

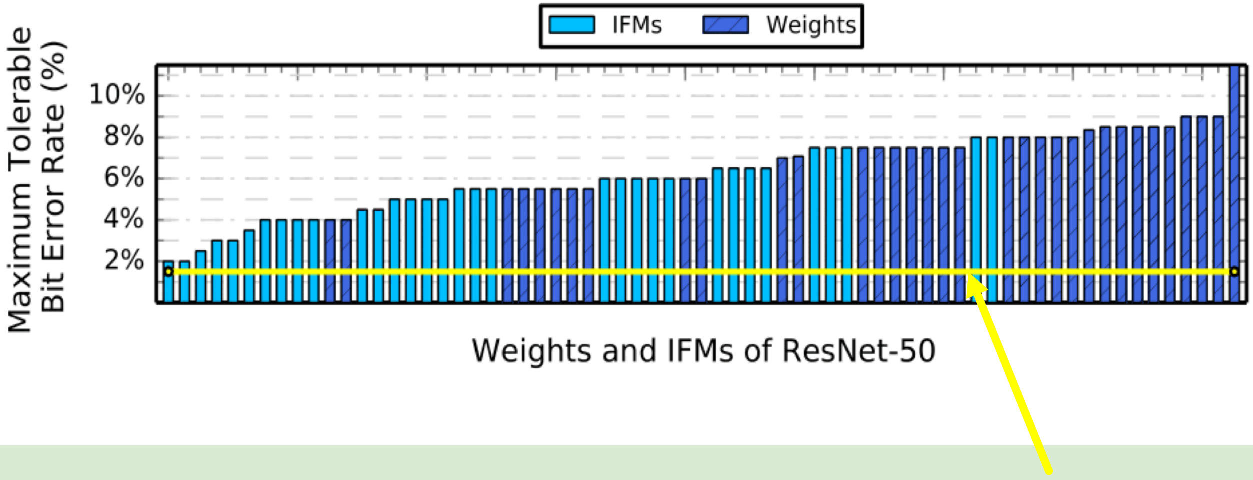
  2. **Fine-grained**: a **greedy algorithm** that matches first the **most error sensitive DNN** data **to the most reliable DRAM partitions**

- **Require DRAM** BER **Characterization**
  - Test **BER** for **reduced voltage and latency** for different data patterns
  - Similar methodology to previous DRAM characterization mechanisms [1][2]

[1] "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms", Chang+, SIGMETRICS'17
[2] "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization", Chang+, SIGMETRICS'17

Step 3: Coarse-Grained DNN to DRAM Mapping

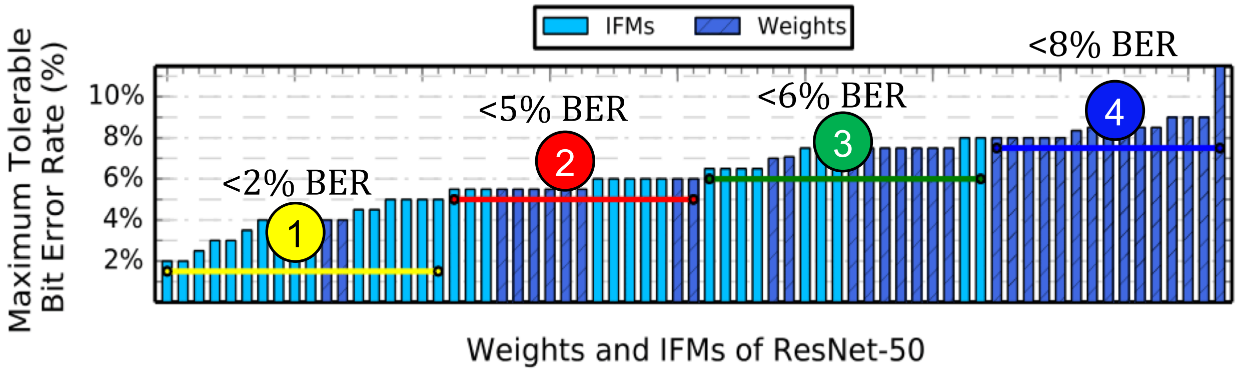
Mapping example of ResNet-50:



The DRAM should not introduce more than 2% BER

Step3: Fine-Grained DNN to DRAM Mapping

Mapping example of ResNet-50:



Map DNN layers more tolerable to errors to DRAM partitions **with lower voltage/latency**

4 DRAM partitions with different error rates

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Enabling EDEN Using Error Models

Problem: Retraining is **not always feasible** on the approximate DRAM device

- Example: some **edge-devices** has limited hardware resources

Goal: Perform retraining and DNN error characterization **in a system that is different from the target approximate system**

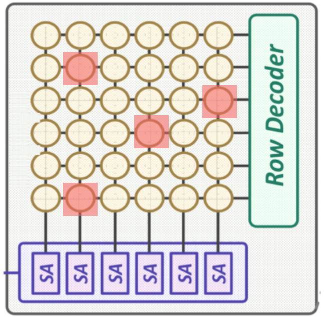
Key Idea: use **error models** to emulate the bit errors of the target approximate DRAM

DRAM Error Models

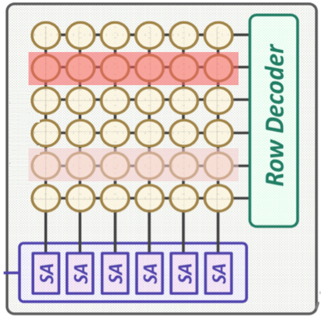
- Error models Contain information about the **spatial distribution** of weak cells in the DRAM modules

- Use the error models to **inject errors** in each DRAM access

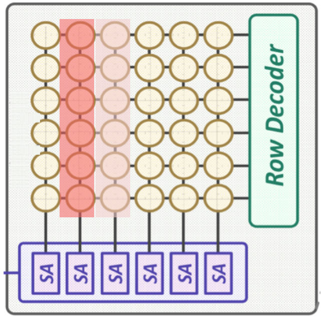
- EDEN uses **four** probabilistic **error models**
  - Observed in **real approximate DRAM modules**



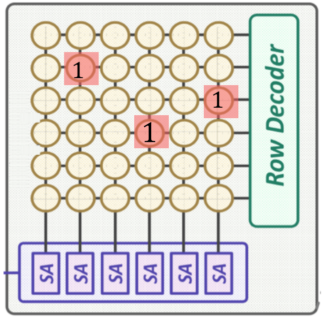
Model 0:
Uniform Random



Model 1:
Wordline Correlated



Model 2:
Bitline Correlated



Model 3:
Bit Value Dependent

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. EDEN Mechanism
   i. Boosting DNN Error Tolerance
   ii. DNN Error Tolerance Characterization
   iii. DNN to DRAM Mapping
   Enabling EDEN Using Error Models
4. **Evaluation**
5. Conclusion

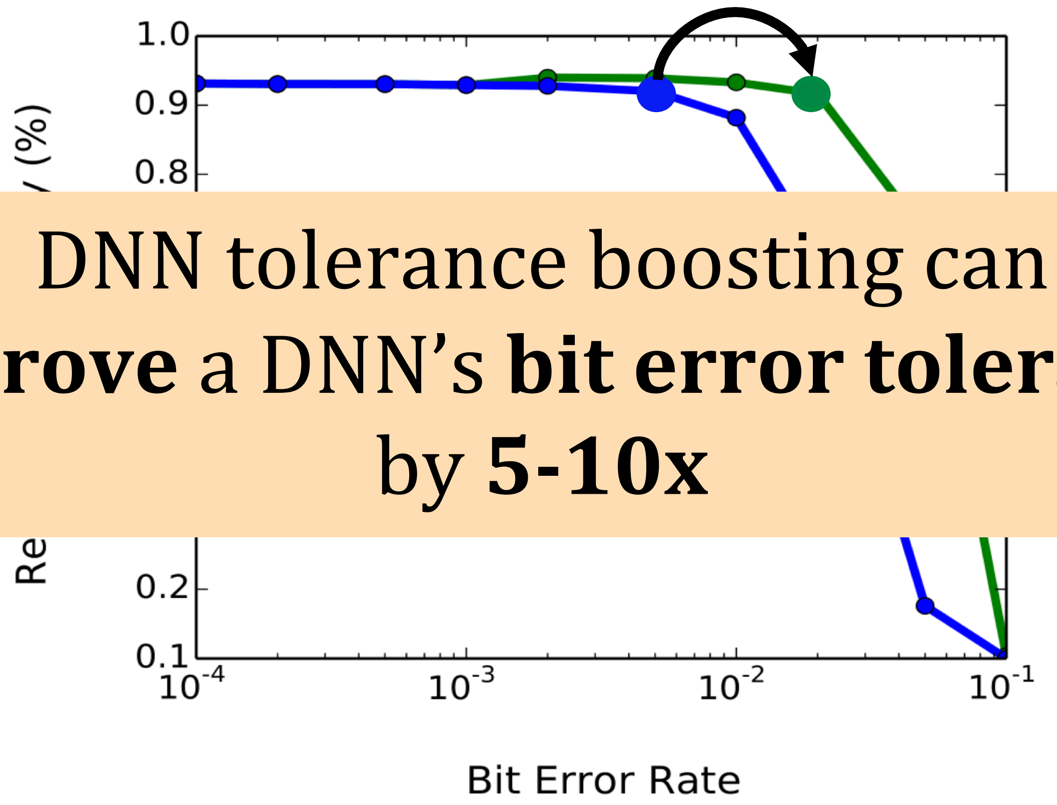
DNN Accuracy Evaluation: Methodology

- **8 DNN workloads**
  - YOLO, YOLO-Tiny, MobileNetV2, SqueezeNet1.1, VGG-16, DenseNet201, ResNet-101, AlexNet
- **Four quantization levels**
  - int4, int8, int16, FP32
- **FPGA-based** framework [1] to run inference data accesses on **real DDR3 DRAM modules**
- Custom **PyTorch**-based DNN framework to run DNN inference with **error models**

[1] "SoftMC: A flexible and practical open-source infrastructure for enabling experimental DRAM studies", Hassan+, HPCA'17

Boosting Error Tolerance of ResNet101

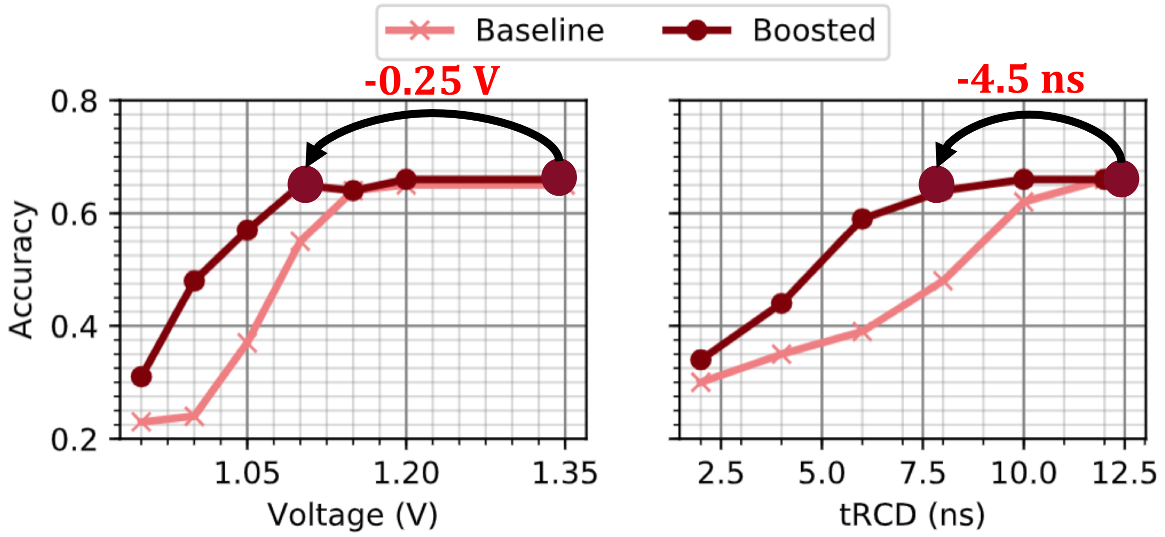
Tolerance boost: <u>**10x**</u>



DNN tolerance boosting can **improve** a DNN's **bit error tolerance** by **5-10x**

SAFARI

DNN Accuracy of LeNeT using Real DRAM



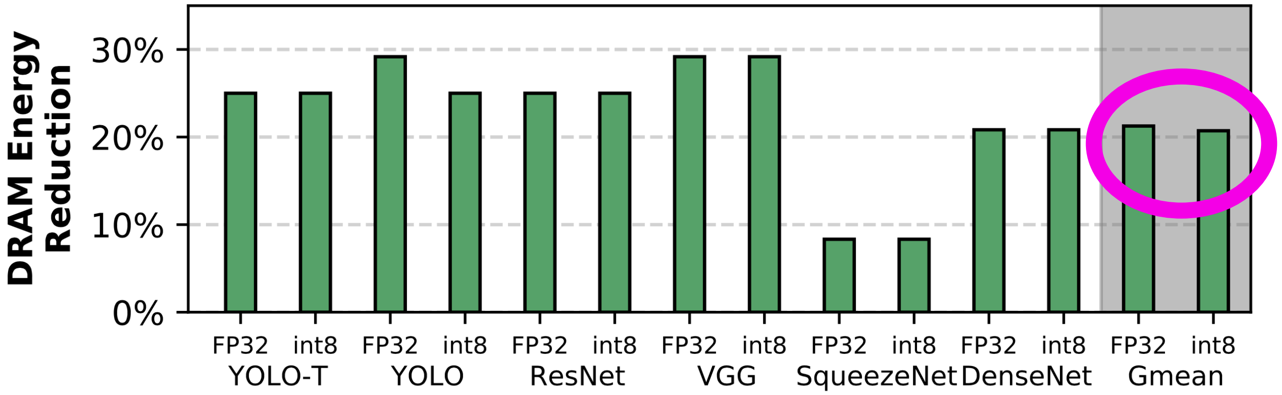
Our boosting mechanism helps **reducing DRAM voltage and latency while maintain accuracy on real DRAM modules**

SAFARI

System Level Evaluation: Methodology

- **6 DNN workloads** with **int8** and **FP32** quantizations
  - Yolo-Tiny, Yolo, ResNet, VGG, SqueezeNet, DenseNet
- Inference libraries from **DarkNet, Intel OpenVINO, TVM**
- **Simulators: Ramulator**, **ZSim**, **GPGPUSim**, and **SCALE-Sim** used for DRAM, CPU, GPU, Eyeriss, and TPU simulation
- **Configurations:**
  - CPU: 4 Core, 8MB L3 per core, 8GB DDR4
  - GPU: 28 SMs, 12GB GDDR5 (Titan X)
  - Eyeriss: 12x14 PEs, 4GB LPDDR4
  - TPU: 256 x 256 PEs, 4GB LPDDR4
- More detail in the paper

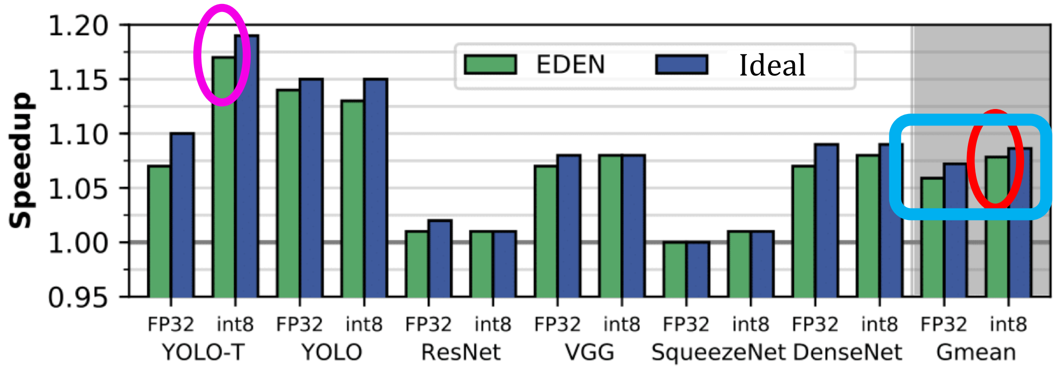
CPU: DRAM Energy Evaluation



Average 21% DRAM energy reduction

maintaining accuracy within 1% of original

CPU: Performance Evaluation



Average 8% system speedup
Some workloads achieve **17% speedup**

EDEN achieves **close to the ideal** speedup possible via tRCD scaling

GPU, Eyeriss, and TPU: Energy Evaluation

- **GPU**: average **37% energy reduction**
- **Eyeriss**: average **31% energy reduction**
- **TPU**: average **32% energy reduction**

Other Results in the Paper

- Error resiliencies across different DNNs and quantizations
- Validation of the boosting mechanism
- Supporting data for error models using real DRAM modules
- Comparison of different DRAM error models
- Breakdown of energy savings on different workloads for GPU and TPU

Outline

1. Motivation and Problem
2. DNN and DRAM Background
3. EDEN Mechanism
   - i. Boosting DNN Error Tolerance
   - ii. DNN Error Tolerance Characterization
   - iii. DNN to DRAM Mapping
   - Enabling EDEN Using Error Models
4. Evaluation
5. **Conclusion**

Conclusion

Motivation: Deep Neural Networks (DNNs) are important in many domains

Problem: ***DRAM*** *can increase the* ***energy consumption*** *and the* ***execution latency*** *of DNN inference*

Goal: reduce energy and latency of DNN workloads by using **DRAM with reduced voltage and timing parameters (approximate DRAM)**

Challenge: approximate DRAM introduce **bit errors**

EDEN: Enabling Efficient DNN Inference Using Approximate DRAM

- **Tolerate approximate DRAM bit errors** by retraining the DNN for a target accuracy
- **Map DNN data types** with different error tolerance **to DRAM partitions** with different error rates

Results:

- Average **21%/37%/31% DRAM energy savings** on CPU/GPU/DNN-accelerators
- Average **8% speedup** on CPU

EDEN is **applicable to other DRAM parameters** and **memory technologies**

EDEN

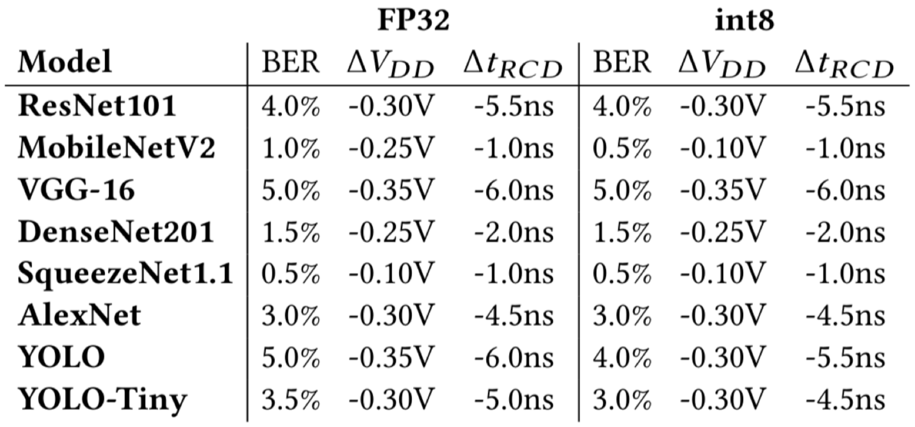
Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM

Skanda Koppula **Lois Orosa** A. Giray Yaglikci
Roknoddin Azizi Taha Shahroodi Konstantinos Kanellopoulos Onur Mutlu

ETH zürich SAFARI

Coarse-Grained Scaling

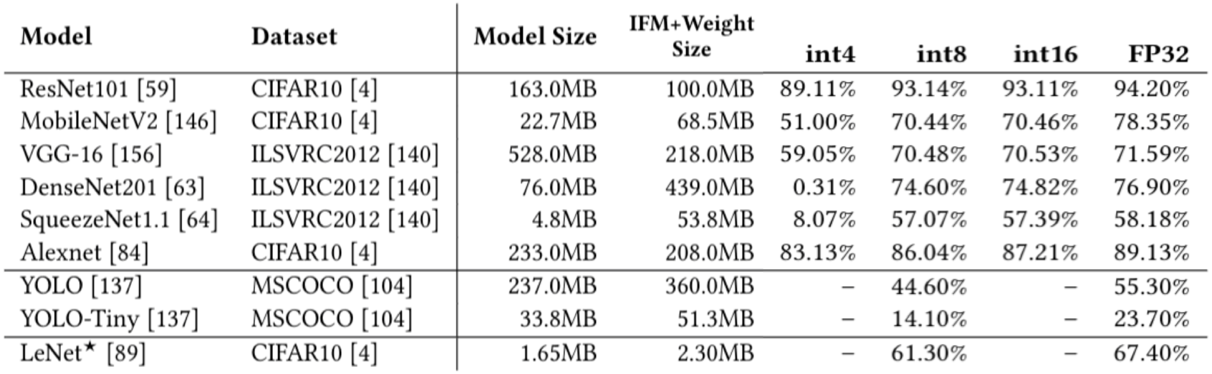
Coarse-Grained Scaling



	FP32			int8		
Model	BER	ΔV_{DD}	Δt_{RCD}	BER	ΔV_{DD}	Δt_{RCD}
ResNet101	4.0%	-0.30V	-5.5ns	4.0%	-0.30V	-5.5ns
MobileNetV2	1.0%	-0.25V	-1.0ns	0.5%	-0.10V	-1.0ns
VGG-16	5.0%	-0.35V	-6.0ns	5.0%	-0.35V	-6.0ns
DenseNet201	1.5%	-0.25V	-2.0ns	1.5%	-0.25V	-2.0ns
SqueezeNet1.1	0.5%	-0.10V	-1.0ns	0.5%	-0.10V	-1.0ns
AlexNet	3.0%	-0.30V	-4.5ns	3.0%	-0.30V	-4.5ns
YOLO	5.0%	-0.35V	-6.0ns	4.0%	-0.30V	-5.5ns
YOLO-Tiny	3.5%	-0.30V	-5.0ns	3.0%	-0.30V	-4.5ns

tRCD or voltage scaling that yields **<1% accuracy** degradation on **a target DDR3 module**

DNN Workload List and Baseline Accuracies

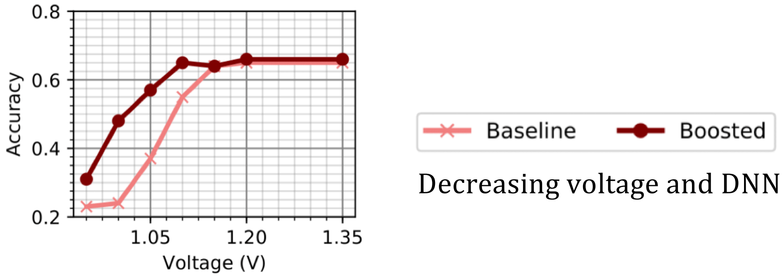


Model	Dataset	Model Size	IFM+Weight Size	int4	int8	int16	FP32
ResNet101 [59]	CIFAR10 [4]	163.0MB	100.0MB	89.11%	93.14%	93.11%	94.20%
MobileNetV2 [146]	CIFAR10 [4]	22.7MB	68.5MB	51.00%	70.44%	70.46%	78.35%
VGG-16 [156]	ILSVRC2012 [140]	528.0MB	218.0MB	59.05%	70.48%	70.53%	71.59%
DenseNet201 [63]	ILSVRC2012 [140]	76.0MB	439.0MB	0.31%	74.60%	74.82%	76.90%
SqueezeNet1.1 [64]	ILSVRC2012 [140]	4.8MB	53.8MB	8.07%	57.07%	57.39%	58.18%
Alexnet [84]	CIFAR10 [4]	233.0MB	208.0MB	83.13%	86.04%	87.21%	89.13%
YOLO [137]	MSCOCO [104]	237.0MB	360.0MB	–	44.60%	–	55.30%
YOLO-Tiny [137]	MSCOCO [104]	33.8MB	51.3MB	–	14.10%	–	23.70%
LeNet⋆ [89]	CIFAR10 [4]	1.65MB	2.30MB	–	61.30%	–	67.40%

Coarse-Grained Characterization Algorithm

Key Steps:

1. Decrease tRCD/V_{dd} of DRAM module
2. Run DNN inference
3. Measure accuracy on validation dataset
4. If accuracy < target: terminate.



Decreasing voltage and DNN accuracy

Coarse-Grained Characterization Algorithm

Key Steps:

1. Decrease parameter of DRAM/DNN partition
2. Run DNN inference
3. Measure accuracy on validation dataset
4. If accuracy < target: roll-back parameter decrease
5. Repeat for all DNN partitions, parameter levels