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 (vision, robotics, ...)

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Goal: Reduce **DRAM voltage and timing** for **error tolerant DNN inference workloads,** exploiting the trade-off between bit error rate and energy/performance

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Goal: Reduce **DRAM voltage and timing** for **error tolerant DNN inference workloads,** exploiting the trade-off between bit error rate and energy/performance

EDEN: Deep Neural Network Inference Using Approximate DRAM

- Techniques to maintain accuracy through (1) **error tolerance boosting**, (2) **DNN characterization,** (3) **DNN to DRAM mapping**, and **DRAM error modeling**

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Goal: Reduce **DRAM voltage and timing** for **error tolerant DNN inference workloads,** exploiting the trade-off between bit error rate and energy/performance

EDEN: Deep Neural Network Inference Using Approximate DRAM

- Techniques to maintain accuracy through (1) **error tolerance boosting**, (2) **DNN characterization,** (3) **DNN to DRAM mapping**, and **DRAM error modeling**

Results: Energy savings and performance improvements on 12 DNN benchmarks

- Average **21% DRAM energy savings** and **8% speedup** on CPU
- Average **37% DRAM energy savings** on GPU
- Average **31% DRAM energy savings** on DNN accelerators (Eyeriss and TPU)

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Goal: Reduce **DRAM voltage and timing** for **error tolerant DNN inference workloads,** exploiting the trade-off between bit error rate and energy/performance

EDEN: Deep Neural Network Inference Using Approximate DRAM

- Techniques to maintain accuracy through (1) **error tolerance boosting**, (2) **DNN characterization,** (3) **DNN to DRAM mapping**, and **DRAM error modeling**

Results: Energy savings and performance improvements on 12 DNN benchmarks

- Average **21% DRAM energy savings** and **8% speedup** on CPU
- Average **37% DRAM energy savings** on GPU
- Average **31% DRAM energy savings** on DNN accelerators (Eyeriss and TPU)

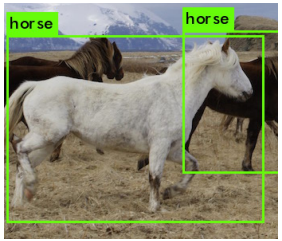
EDEN is **applicable to other DRAM parameters** and **memory technologies**

Outline

1. **Motivation and Problem**
2. DNN Basics and DRAM Parameters
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

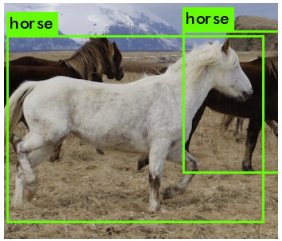
Motivation

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



Motivation

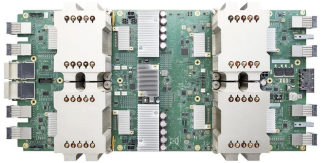
Deep neural networks (DNNs) are critical in computer vision, robotics, and many other domains



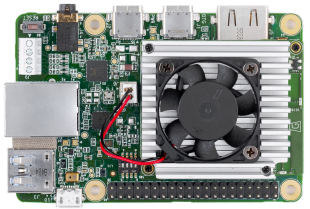
Modern platforms for DNN inference use DRAM

Mobile CPUs

GPUs



Data Center Accelerators



Edge-device Accelerators

Challenges of DNN Inference

DRAM has high energy consumption

- **25% to 70% of system energy** is consumed by DRAM in common DNN inference accelerators

Challenges of DNN Inference

DRAM has high energy consumption

- **25% to 70% of system energy** is consumed by DRAM in common DNN inference accelerators

DRAM can bottleneck performance

- Potential **19% speedup** by **reducing DRAM latency** on CPU for some DNNs

Challenges of DNN Inference

DRAM has high energy consumption

- **25% to 70% of system energy** is consumed by DRAM in common DNN inference accelerators

DRAM can bottleneck performance

- Potential **19% speedup** by **reducing DRAM latency** on CPU for some DNNs

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

Outline

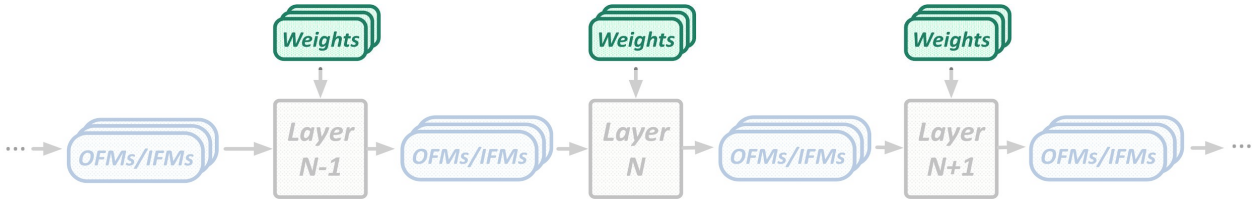
1. Motivation and Problem
2. **DNN Basics and DRAM Parameters**
3. EDEN Mechanism
    i. Boosting DNN Error Tolerance
    ii. DNN Error Tolerance Characterization
    iii. DNN to DRAM Mapping
    Enabling EDEN Using Error Models
4. Evaluation
5. Conclusion

Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights

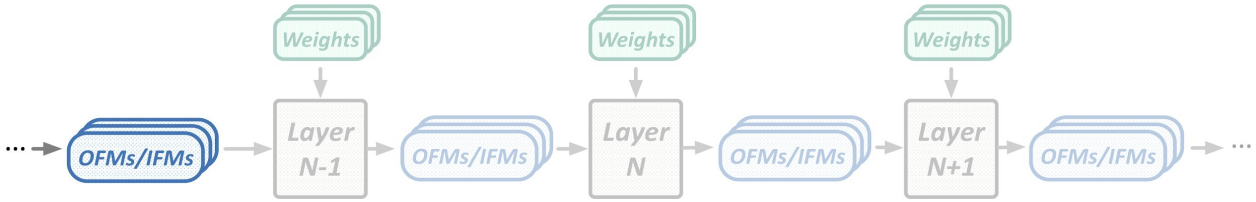
Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights
- **Three main data types** compose a DNN layer:
  1. **Weights**



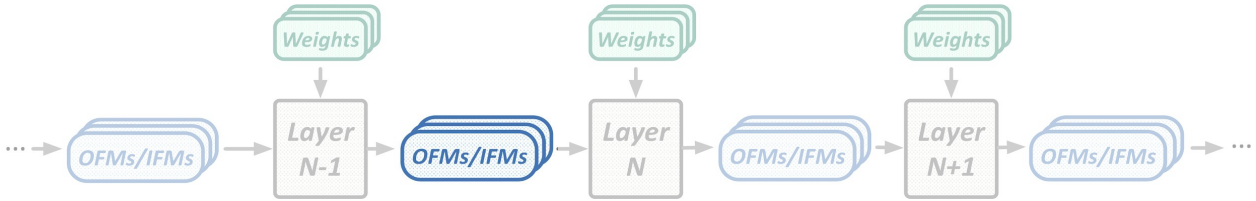
Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights
- **Three main data types** compose a DNN layer:
  1. **Weights**
  2. **Input Feature Maps (IFMs)**



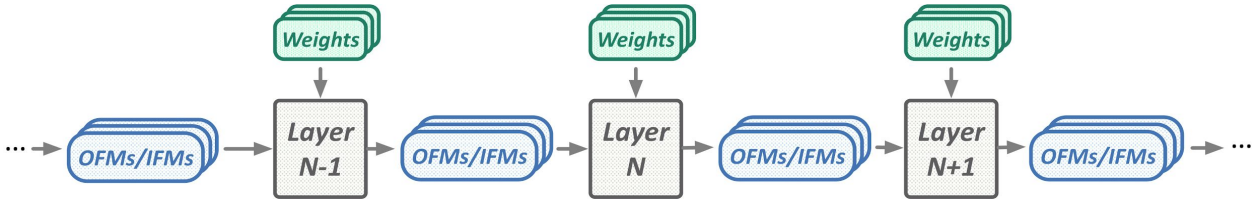
Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights
- **Three main data types** compose a DNN layer:
  1. **Weights**
  2. **Input Feature Maps (IFMs)**
  3. **Output Feature Maps (OFMs)**



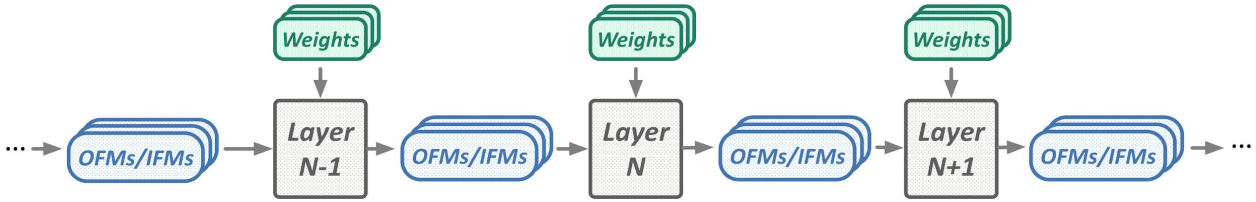
Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights
- **Three main data types** compose a DNN layer:
  1. **Weights**
  2. **Input Feature Maps (IFMs)**
  3. **Output Feature Maps (OFMs)**
- Large DNN weight/IFM counts enable **high learning capacity**



Deep Neural Network Inference

- Modern DNNs can have **hundreds of layers** and between $\mathbf{10^5}$ and $\mathbf{10^9}$ weights
- **Three main data types** compose a DNN layer:
  1. **Weights**
  2. **Input Feature Maps (IFMs)**
  3. **Output Feature Maps (OFMs)**
- Large DNN weight/IFM counts enable **high learning capacity**
- If the weights/IFMs have **small bit errors,** a DNN can **still maintain accuracy**



DNN Inference Using DRAM

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

Compute Unit

DRAM Parameters

Memory Bus

Memory Controller

Compute Unit

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

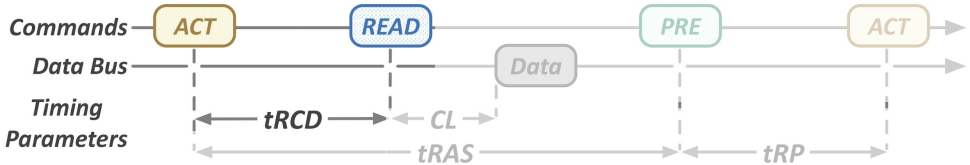
DRAM Parameters

Memory Bus

Memory Controller

Compute Unit

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



Outline

1. Motivation and Problem
2. DNN Basics and DRAM Parameters
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

Observations

1. DNNs have an **intrinsic robustness to errors** in the weight and feature map data types

Observations

1. DNNs have an **intrinsic robustness to errors** in the weight and feature map data types

2. DNN inference systems can **reduce DRAM energy consumption** and **latency** if they tolerate **more bit errors**

Insight

1. DNNs have an **intrinsic robustness to errors**

Approximate DRAM (voltage and latency-scaled DRAM) can provide **higher energy-efficiency** and **performance** for **error-tolerant DNN inference** workloads

EDEN: Key Idea

Enable **accurate, efficient** DNN inference using **approximate DRAM** through **3 key steps**:

1. **DNN error tolerance boosting**
2. **DNN and DRAM characterization**
3. **DNN to DRAM mapping**

EDEN: Inputs

Inputs to EDEN:

(1) **user-specified DNN accuracy goal**

(2) **pre-trained model**

(3) **target DRAM device**

Step 1: Boosting DNN Error Tolerance

Goal: Better maintain accuracy when the DNN is exposed to bit errors

Step 1: Boosting DNN Error Tolerance

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

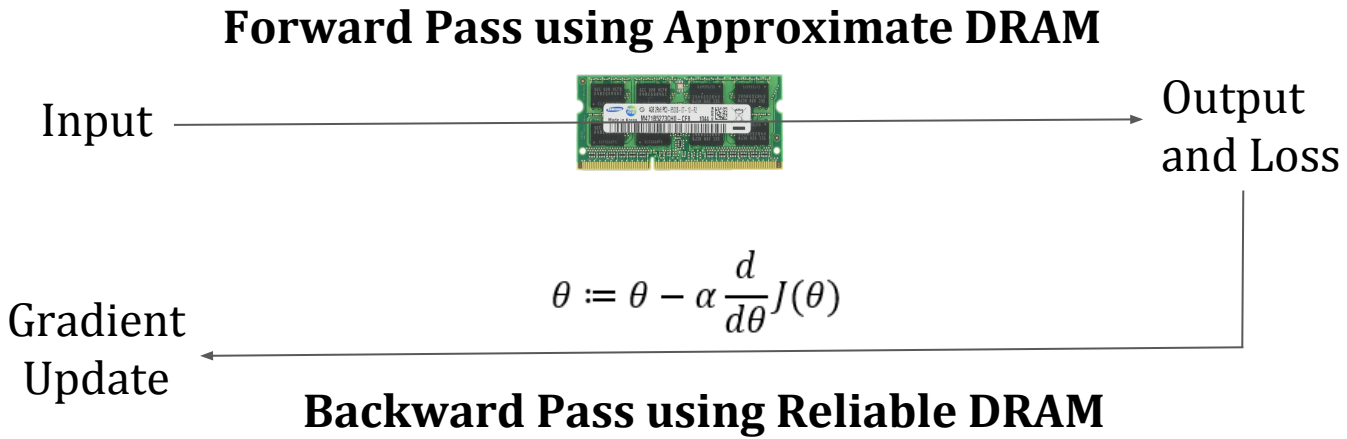
Mechanism: **Retrain** the DNN with **approximate memory** to adapt the DNN to unreliable cells

Forward Pass using Approximate DRAM

Step 1: Boosting DNN Error Tolerance

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

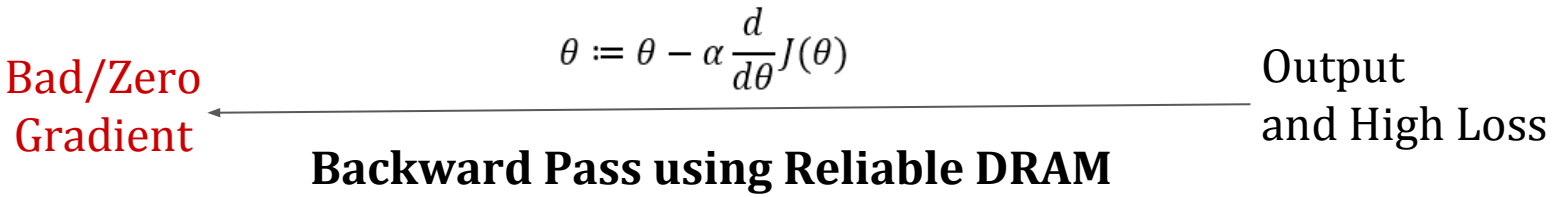
Mechanism: **Retrain** the DNN with **approximate memory** to adapt the DNN to unreliable cells



Step 1: Failures during Boosting

For **high** error rates, **accuracy collapses** at the start of retraining.

Backward pass becomes polluted with zero-information updates



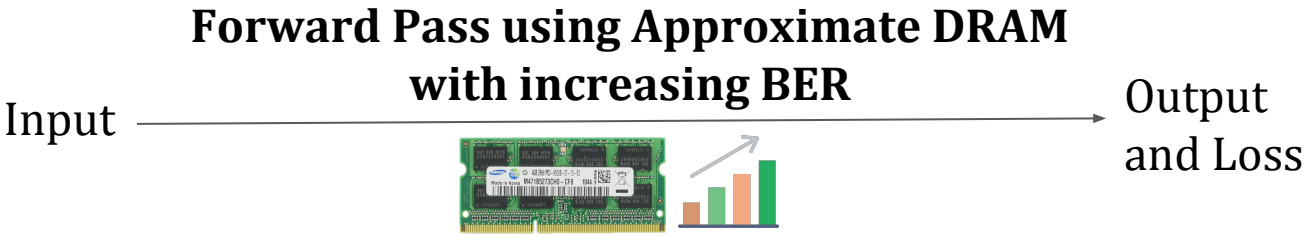
Step 1: Mitigating Failures

Goal: Avoid early retraining collapse

Step 1: Mitigating Failures

Goal: Avoid early retraining collapse

Mechanism: Gradually **increase the error rate** of the approximate DRAM during retraining to build error tolerance



Step 1: Mitigating Failures

Goal: Avoid early retraining collapse

Mechanism: Gradually **increase the error rate** of the approximate DRAM during retraining to build error tolerance

Filter out-of-range values (e.g., $>10^{15}$) based on knowledge of the DNN weight and IFM distribution

Outline

1. Motivation and Problem
2. DNN Basics and DRAM Parameters
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

Step 2: Error Tolerance Characterization

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

Step 2: Error Tolerance Characterization

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

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

Step 2: Error Tolerance Characterization

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

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

Two ways to perform this testing:

1. **Coarse-grained characterization**
2. **Fine-grained characterization**

Step 2: Coarse-Grained Characterization

Reduce voltage/latency of all DNN data types **equally**

Step 2: Coarse-Grained Characterization

Reduce voltage/latency of all DNN data types **equally**

- **Easy to perform on commodity DRAM**

Step 2: Coarse-Grained Characterization

Reduce voltage/latency of all DNN data types **equally**

- **Easy to perform on commodity DRAM**
- Voltage and latency reduction is **limited by the most error sensitive data** in the DNN

Step 2: Fine-Grained Characterization

Scale **voltage and latency differently** for each individual DNN data type and layer

Step 2: Fine-Grained Characterization

Different reductions for each DNN data type and layer

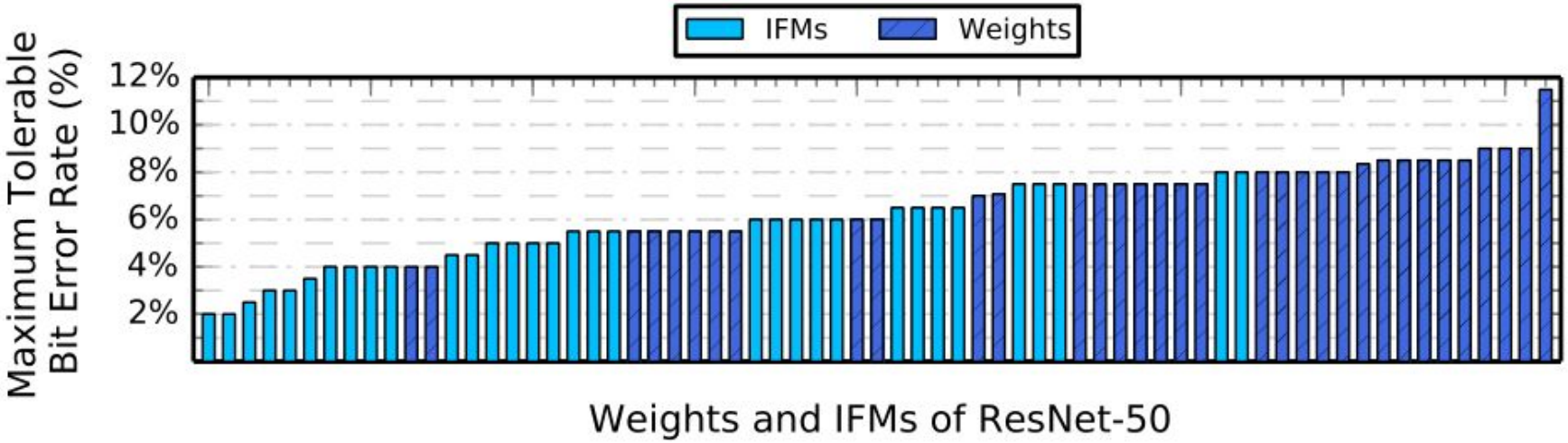
- **More aggressive voltage/latency reduction** is possible

Step 2: Fine-Grained Characterization

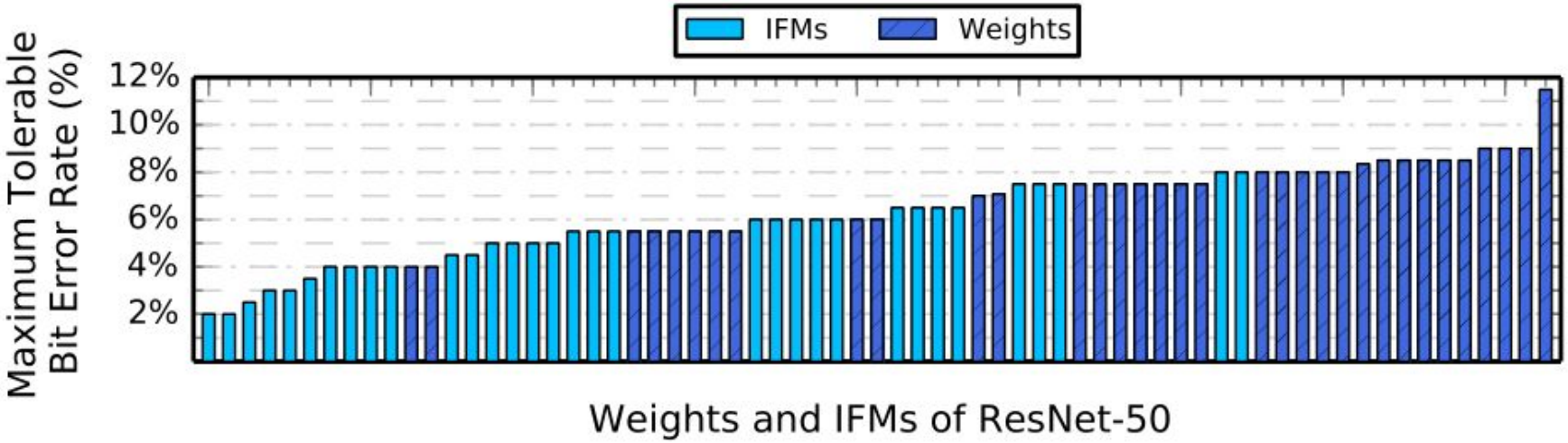
Different reductions for each DNN data type and layer

- **More aggressive voltage/latency reduction** is possible
- **Requires non-commodity DRAM** to reduce some parameters (e.g., V_{dd})
- **Takes more time** than coarse-grained characterization

Example ResNet-50 Characterization



Example ResNet-50 Characterization



- **Error tolerance** of DNN layers **varies greatly**
- **Weights** exhibit **greater error tolerance** than **IFMs**

Outline

1. Motivation and Problem
2. DNN Basics and DRAM Parameters
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

Step 3: Mapping

Goal: **match error tolerance** of DNN with DRAM **error rates**

Step 3: Mapping

Goal: **match error tolerance** of DNN with DRAM **error rates**

Mechanism:

Coarse-grained: assign the single best voltage/latency value that **meets the target DNN accuracy**

Step 3: Mapping

Goal: **match error tolerance** of DNN with DRAM **error rates**

Mechanism:

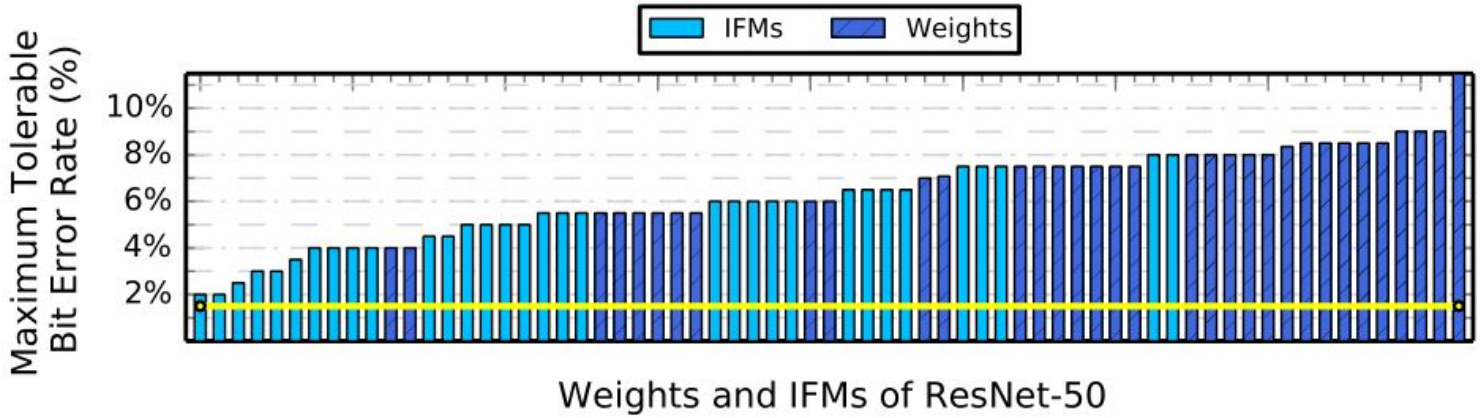
Coarse-grained: assign the single best voltage/latency value that **meets the target DNN accuracy**

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

Example Coarse-Grained Mapping

Mapping of ResNet-50:

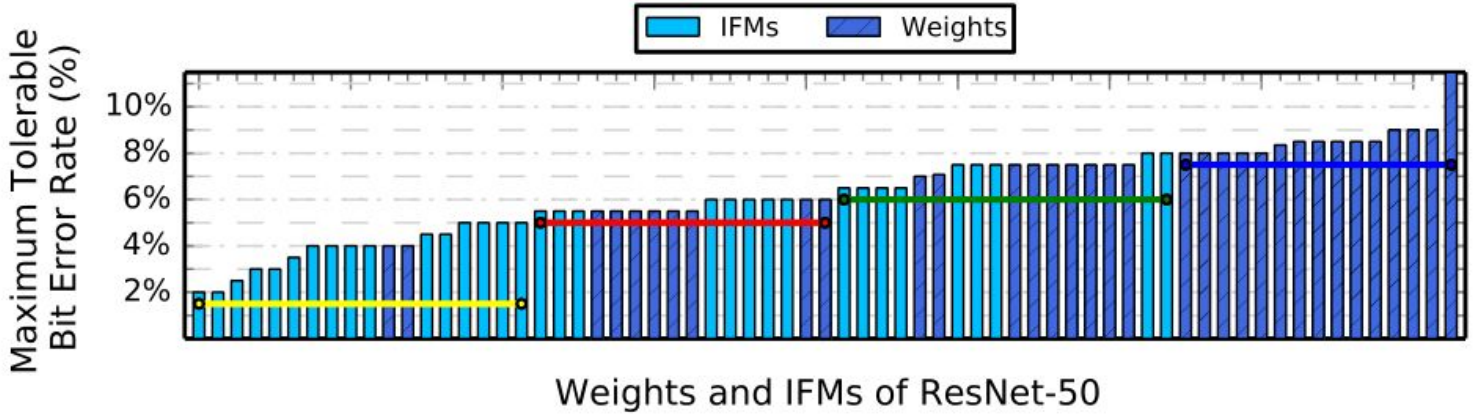
Single DRAM partition with error rate in yellow



Example Fine-Grained Mapping

Mapping of ResNet-50:

4 DRAM partitions with error rates in yellow, red, green, blue



Outline

1. Motivation and Problem
2. DNN Basics and DRAM Parameters
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

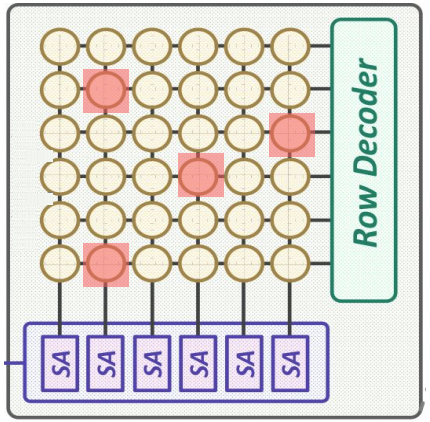
Enabling EDEN Using Error Models

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

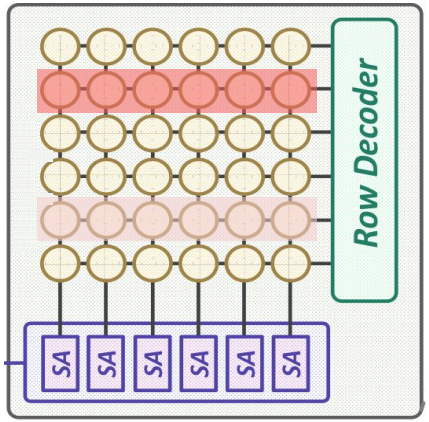
Goal: Perform retraining and error characterization **without use of the approximate DRAM device**

DRAM Error Models

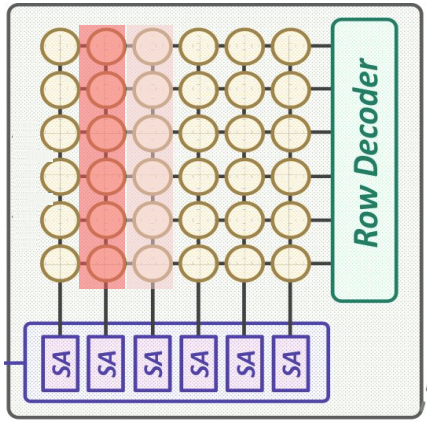
We use the closest fit of four probabilistic error models



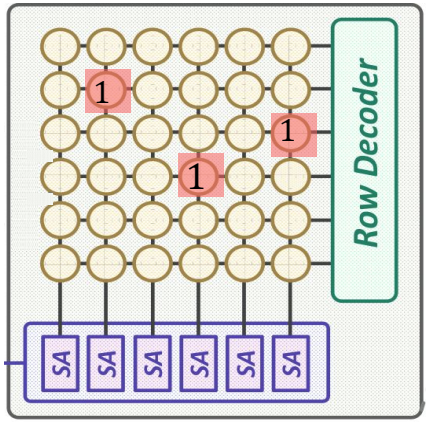
Model 0:
Uniform Random



Model 1:
Wordline Correlated



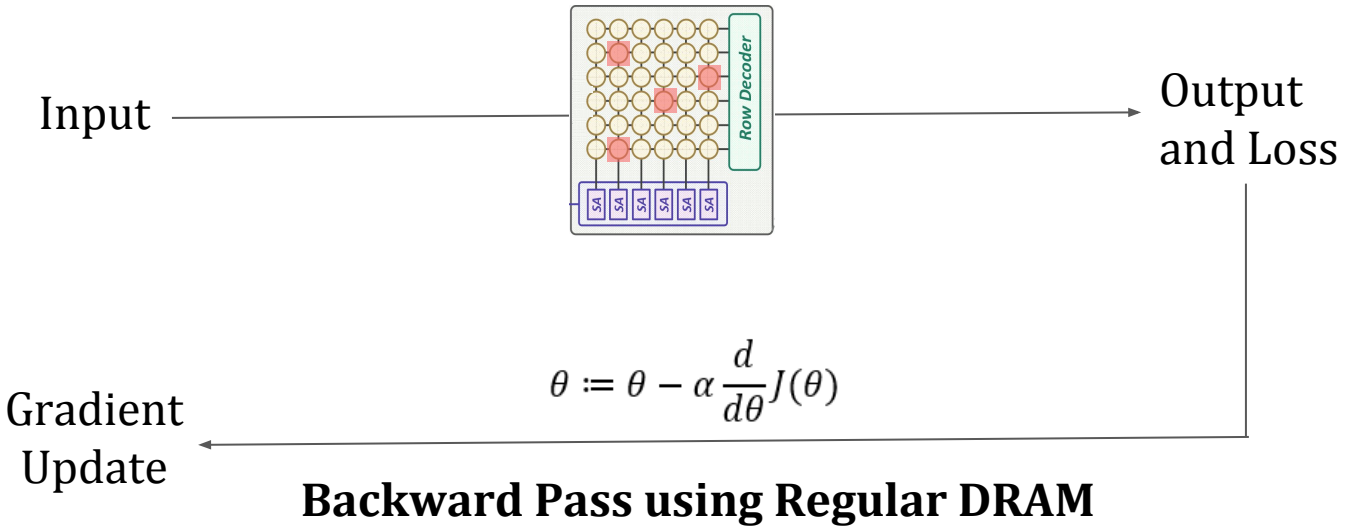
Model 2:
Bitline Correlated



Model 3:
Bit Value Dependent

Use of a DRAM Error Model

Forward Pass using an Error Model in lieu of the Approximate DRAM



Outline

1. Motivation and Problem
2. DNN Basics and DRAM Parameters
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

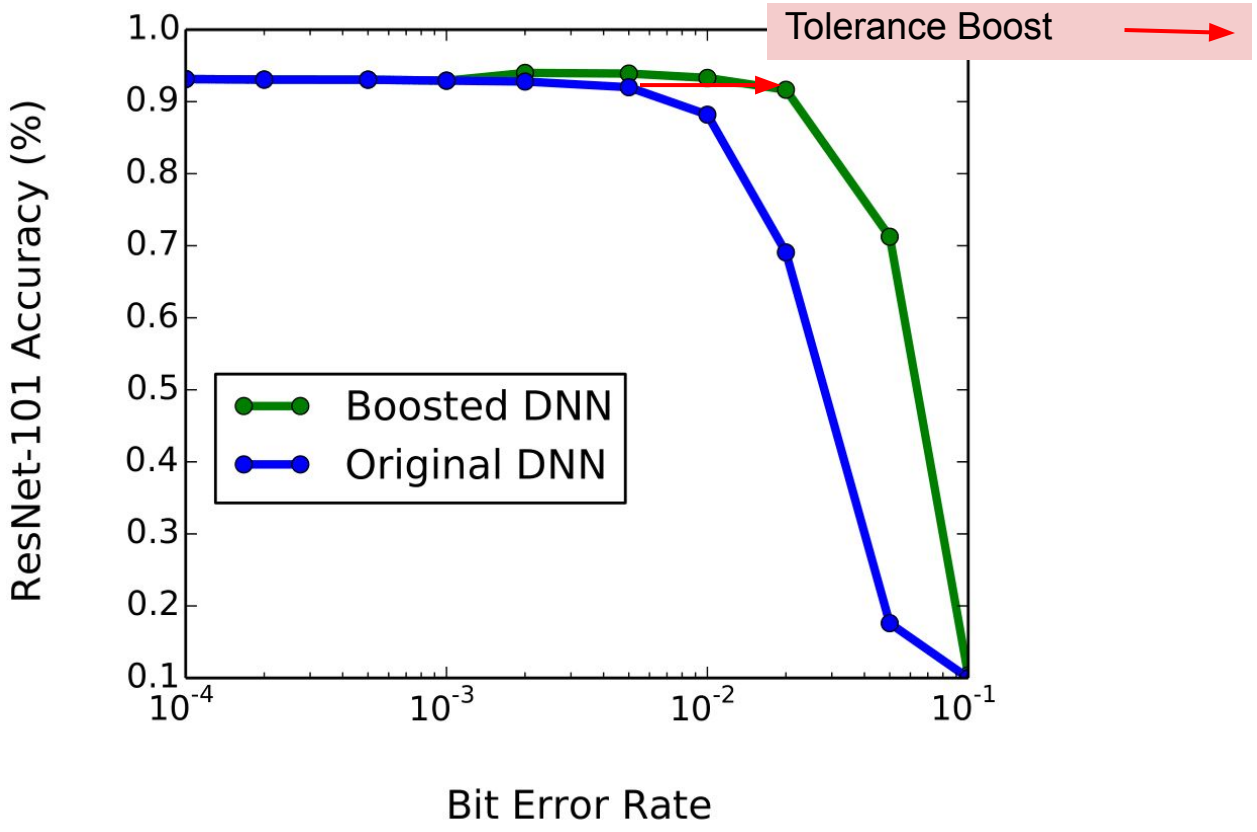
DNN Accuracy Evaluation: Methodology

- **8 DNN workloads** across **four quantization levels**
  - int4, int8, int16, FP32
  - YOLO YOLO-Tiny MobileNetV2 SqueezeNet1.1
    VGG-16 DenseNet201 ResNet-101 AlexNet

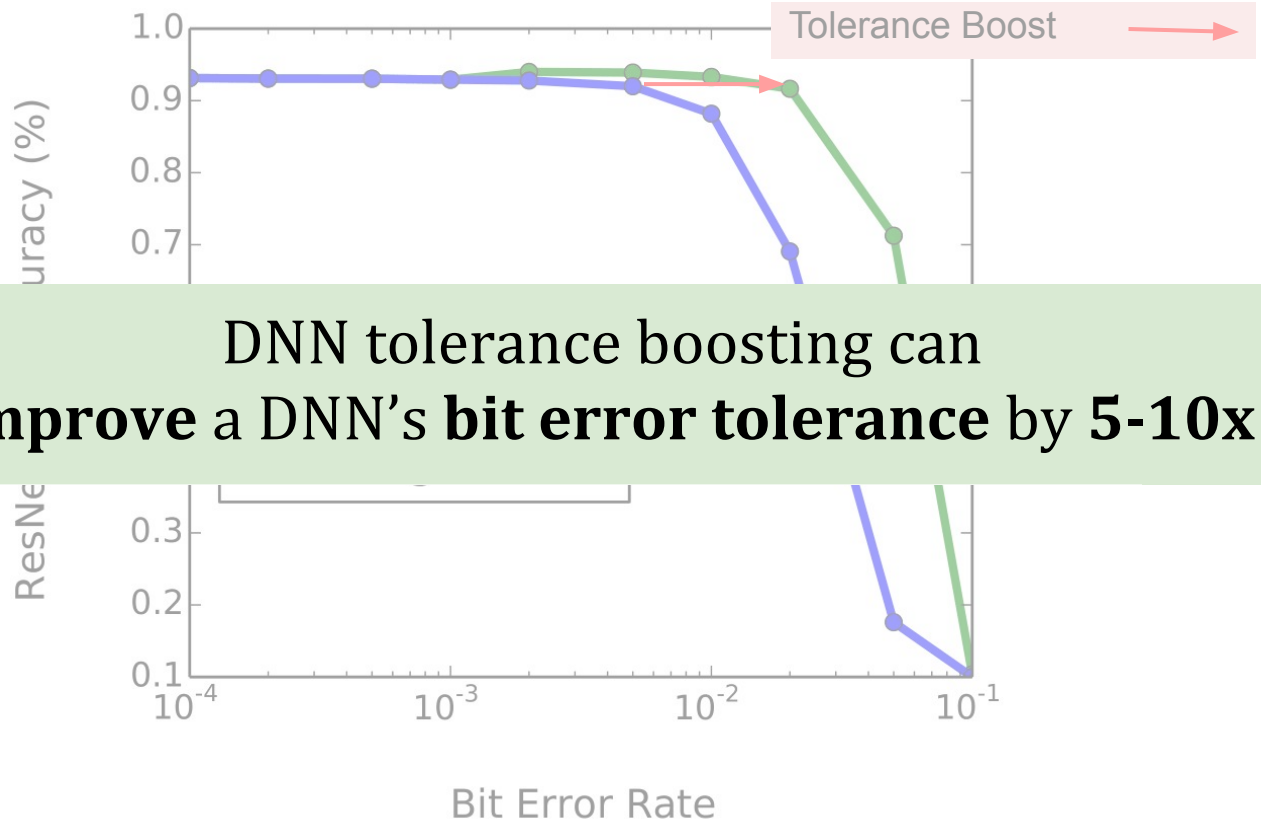
DNN Accuracy Evaluation: Methodology

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

Example: Boosting Error Tolerance of ResNet101

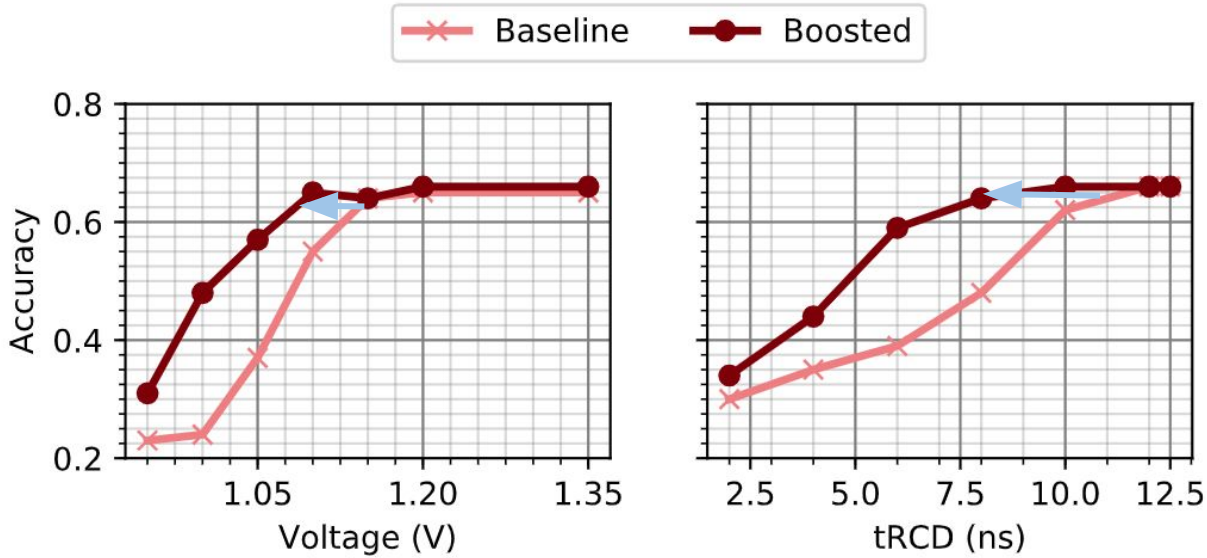


Example: Boosting Error Tolerance of ResNet101



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

DNN Accuracy of LeNeT on SoftMC



Boosting with error models helps **maintain accuracy** while **reducing voltage and latency on real DRAM modules**

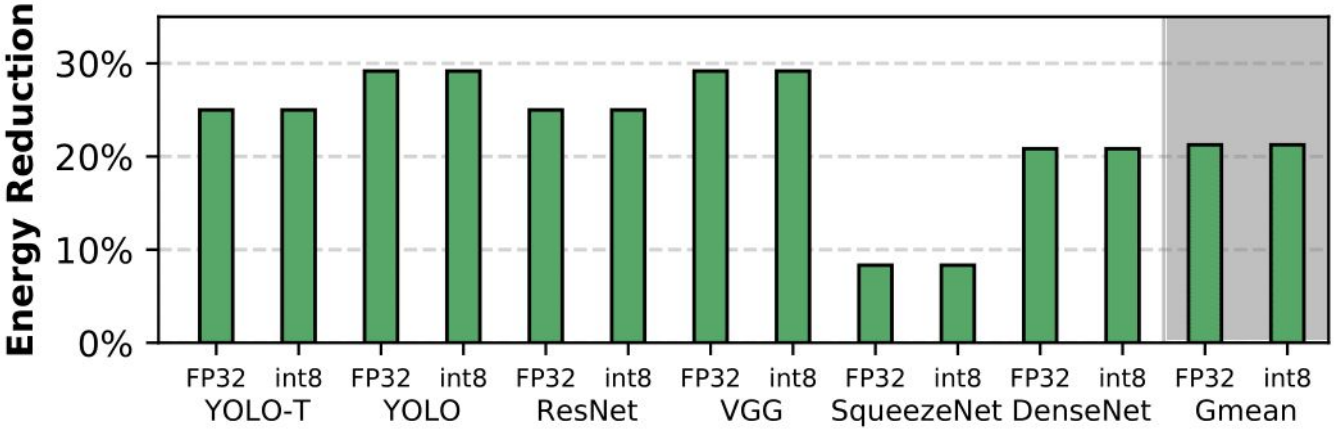
Energy and Performance Evaluation

- **6 DNN workloads** with **int8** and **FP32** quantizations
- Inference libraries from **DarkNet, Intel OpenVINO, TVM**

Energy and Performance Evaluation

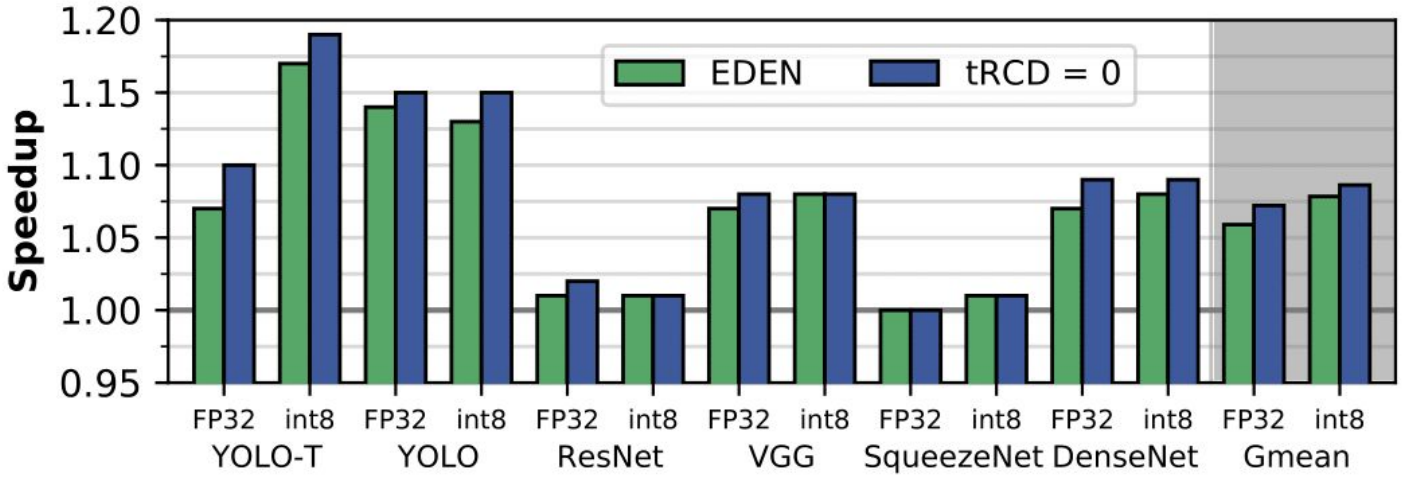
- **6 DNN workloads** with **int8** and **FP32** quantizations
- Inference libraries from **DarkNet, Intel OpenVINO, TVM**
- **Ramulator**, **ZSim**, **GPGPUSim**, and **SCALE-Sim** used for DRAM, CPU, GPU, Eyeriss, and TPU simulation
  - **CPU**: 4 Core @ 4.0 GHz, 8MB L3, 8GB DDR4 DRAM
  - **GPU:** 28 SMs, 12GB GDDR5 @ 2.5 GHz
  - **Eyeriss**: 12 x 18 PEs, 4GB LPDDR4 @ 1600MHz
  - **TPU:** 256 x 256 PEs, 4GB LPDDR4 @ 1600MHz
  - Full configuration can be found in the paper

CPU Energy Evaluation



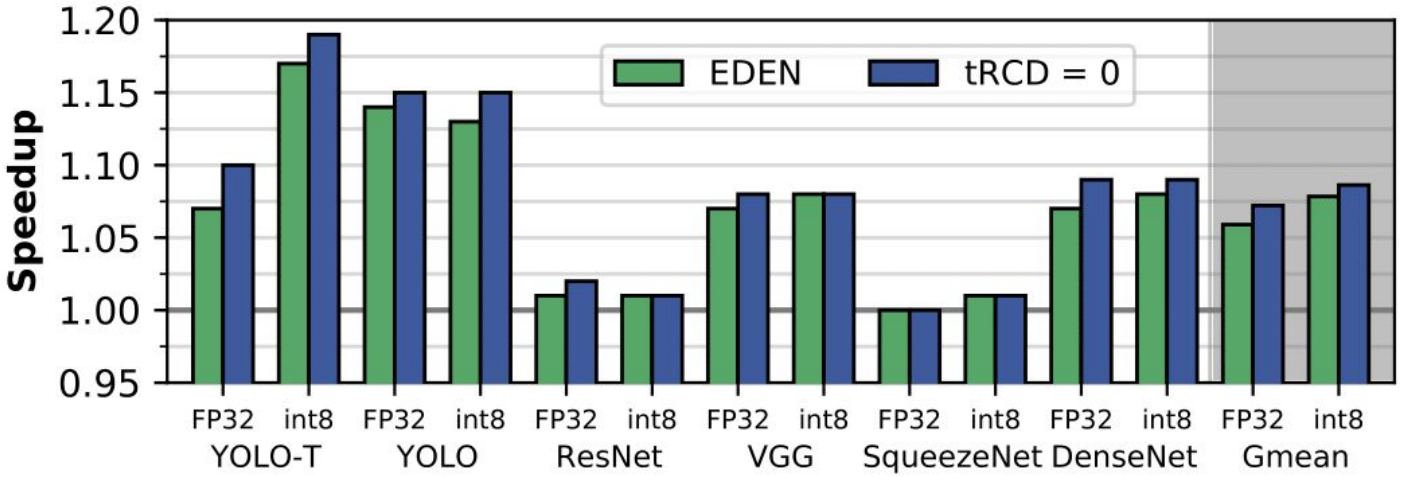
Average 21% DRAM energy reduction
maintaining accuracy within 1% of original

CPU Performance Evaluation



Average 8% system speedup
with some workloads achieving **17% speedup**

CPU Evaluation



EDEN achieves close to the ideal speedup possible via tRCD scaling

GPU, Eyeriss, and TPU Energy Evaluation

- Using the previous DNN benchmarks:
  - Average **31% DDR4 energy reduction on Eyeriss**
  - Average **32% DDR4 energy reduction on TPU**
  - Average **37% GDDR5 energy reduction on Titan X**

GPU, Eyeriss, and TPU Energy Evaluation

- Using the previous DNN benchmarks:
  - Average **31% DRAM energy reduction on Eyeriss**
  - Average **32% DRAM energy reduction on TPU**
  - Average **37% DRAM energy reduction on GPU**
- GPUs and accelerators are **effective at hiding DRAM latency** due to (1) *effective pre-fetching* and (2) *large register banks and SRAM buffers* (exploiting the fixed memory access patterns on DNN inference)

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

Summary

Motivation: Deep Neural Networks (DNNs) are important in many domains (vision, robotics, ...)

Problem: Challenges of DNN Inference:

- **High DRAM energy consumption** → high energy cost of DNN inference
- **High DRAM latency** → DNN inference slowdowns

Goal: Use **voltage/timing scaled DRAM** for DNN inference to exploit **error tolerant DNN workloads,** enabling a trade-off between bit error rate and energy/performance

EDEN: Deep Neural Network Inference Using Approximate DRAM

- Techniques to maintain accuracy through **error tolerance boosting**, **DNN characterization, DNN to DRAM mapping,** and **DRAM error modeling**

Results: Energy savings and performance improvements on 12 DNN benchmarks

- Average **21% energy savings** and **8% speedup** on CPU
- Average **37% energy savings** on GPU
- Average **31% energy savings** on DNN accelerators (Eyeriss and TPU)

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

	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**

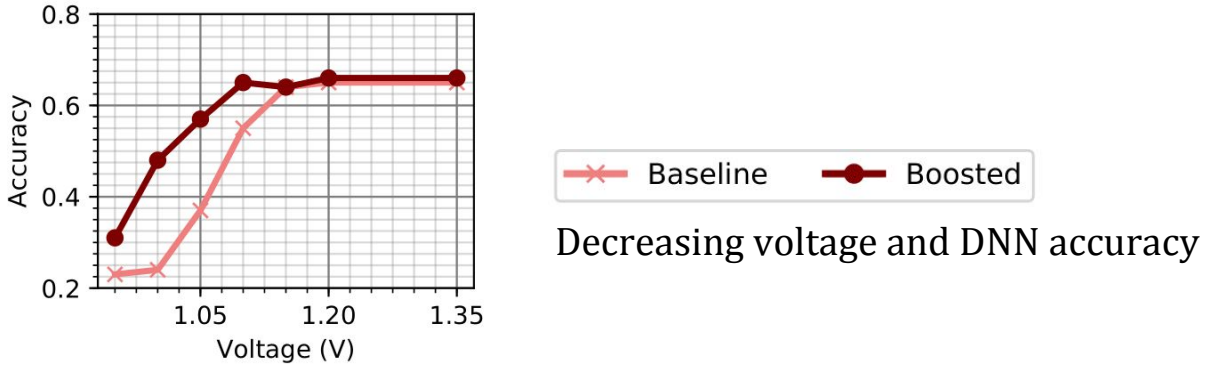
DNN Workload List and Baseline Accuracies

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

Fine-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