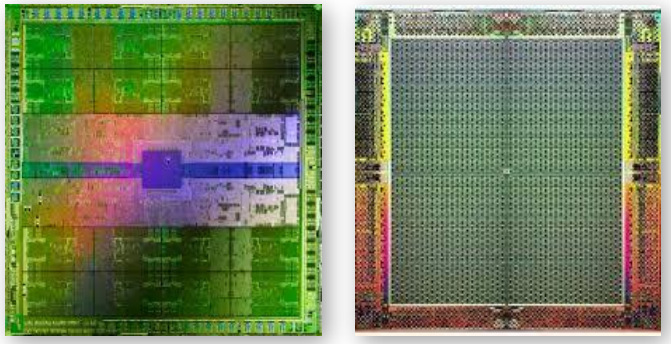
**NVMSA 2021**

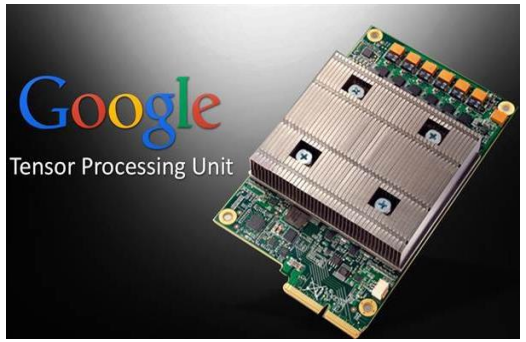
# Mitigating Adversarial Attack for Compute-in-Memory Accelerator Utilizing On-chip Finetune

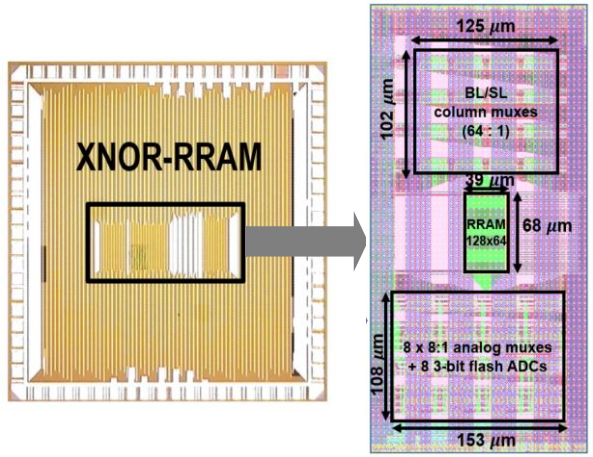
Shanshi Huang, Hongwu Jiang and Shimeng Yu
*Georgia Institute of Technology*

Georgia Tech

# Motivation






**GPU** **FPGA** **TPU** **Compute-in-memory**

**Conventional computing platform**
**s < 5 TOPS/W**

**Digital CMOS ASICs**
**~ 1-10 TOPS/W**

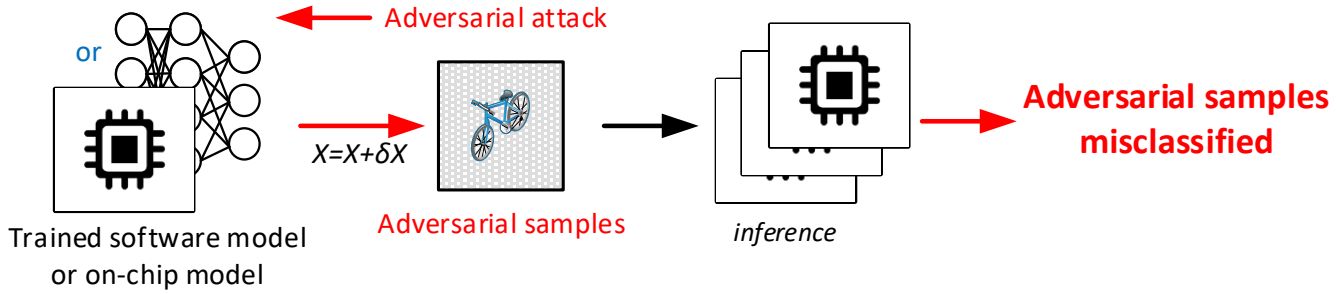
**Analog CMOS (or eNVMs)**
**~ 10-100 TOPS/W**

Fig.1. Accelerators for Neural Network

- Machine learning inference engine is of great interest to smart edge computing. Compute-in-memory (CIM) architecture has shown significant improvements in throughput and energy efficiency for hardware acceleration.
- eNVM-based CIM is attractive for portable device due to its non-volatility, high density, low energy consumption and leakage.
- Portable devices increase the vulnerability of model leakage, which could be used for white-box adversarial attack.

# Motivation

- Adversarial attack generates adversarial examples that could fool the network while looks no difference to the human eyes.
- Types of adversarial attack
  - white box attack
  - Black box attack
- Adversarial examples from one device will be effective on all devices with the same model.



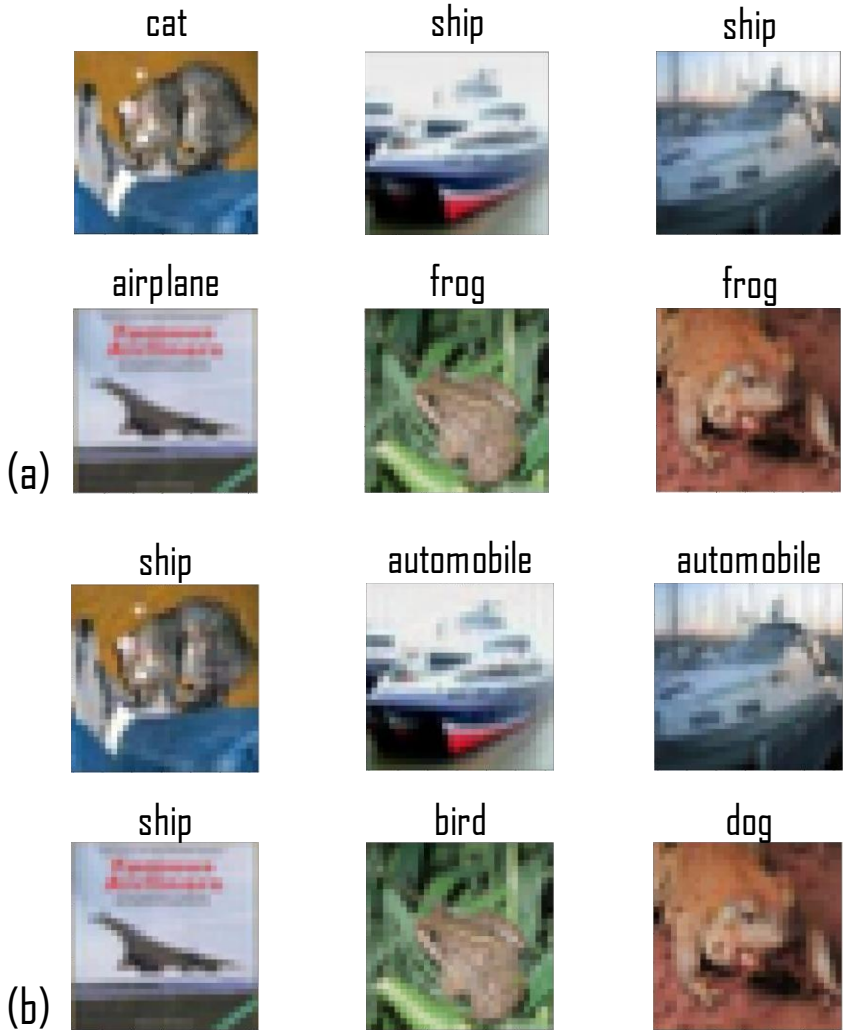




Fig.2. classification results from the same network for original images and adversarial examples

# Introduction: Compute-In-Memory (CIM)

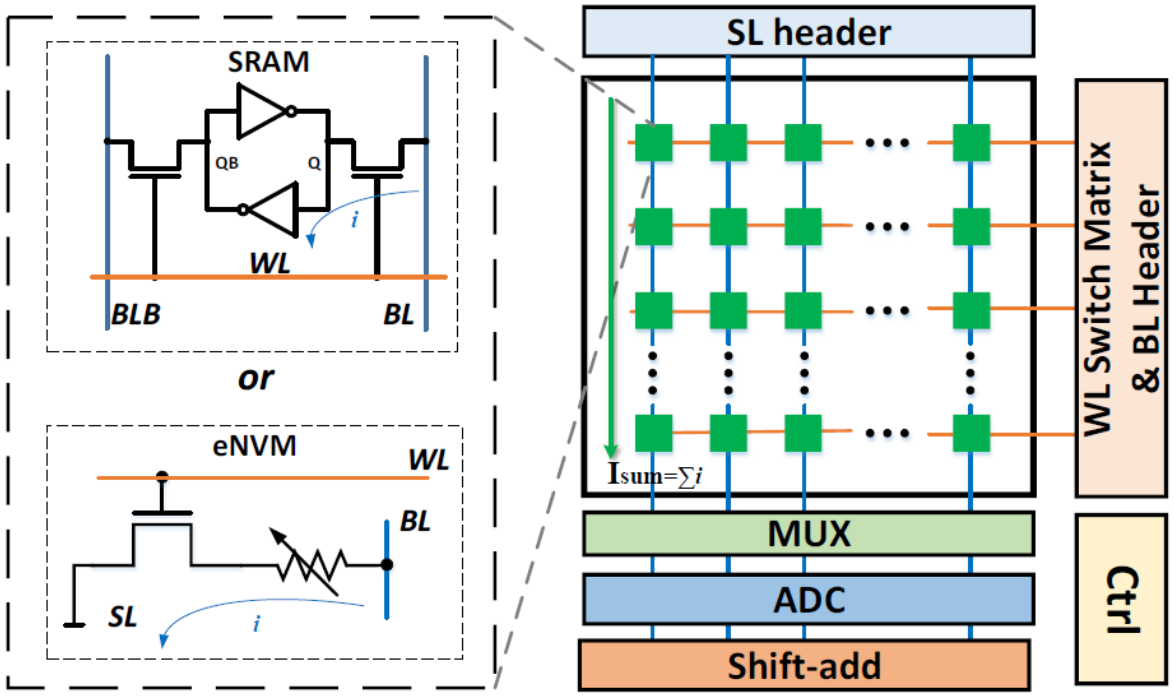



Fig.3. General CIM architecture

- **Compute-in-memory (CIM):** the weight are stored in memory array, while the activations are loaded in as input to WLs → the current summation along columns represents weighted sum
- ADC is necessary at the edge of the array to convert analog signal back to digital domain for further process
- ADC offset exploited as countermeasure for adversarial examples
- On chip hybrid fine-tune to recover accuracy and make adversarial examples less transferable

# Introduction: ADCs

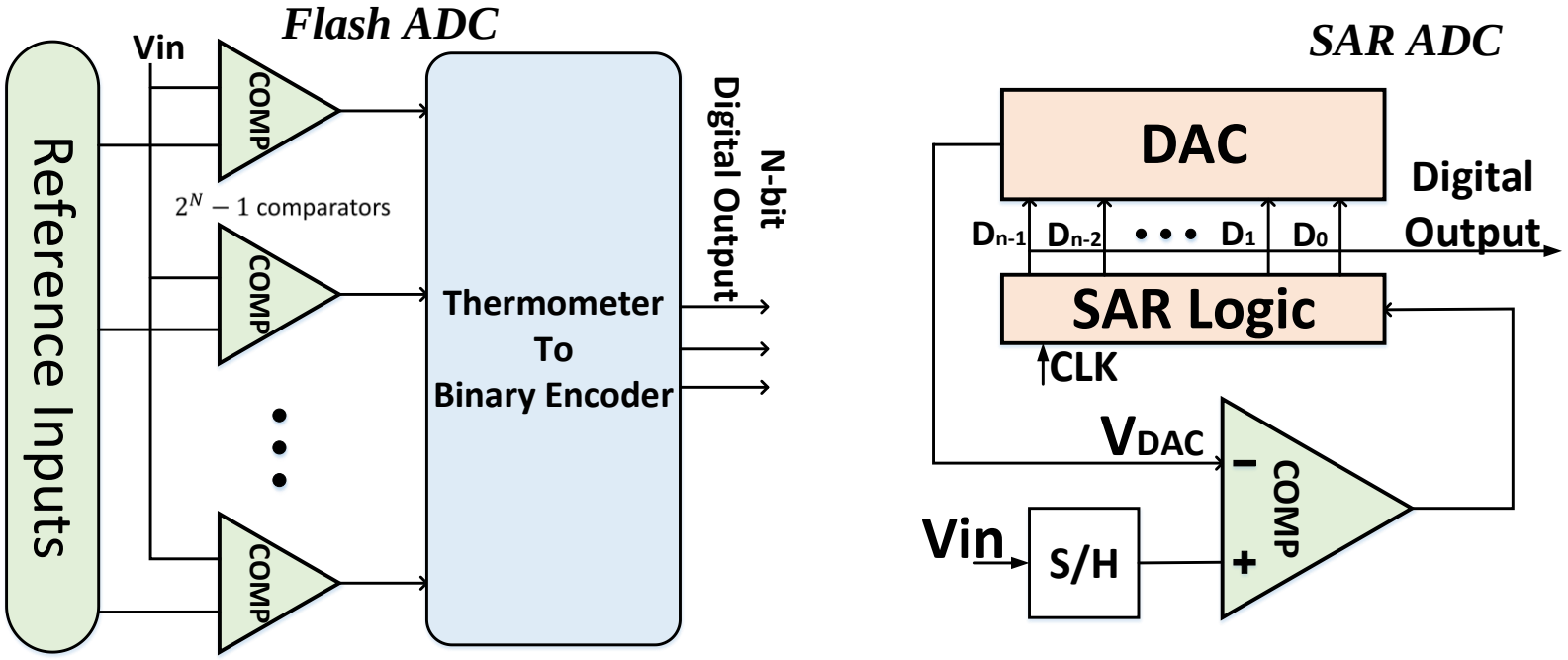



Fig.4. General ADC architecture

- Generally, there are two types of ADC used in CIM: Flash-ADC and SAR-ADC.
- The main component in both ADCs that cause offset is the comparator (or sense amplifier in CIM).
- Flash-ADC uses different comparators for different levels and uses encoder to convert the thermometer code to the binary code.
- SAR-ADC has only one comparator but compares in several iterative cycles with a binary tree search towards the correct level.

# Methodology: ADC offset variation modeling

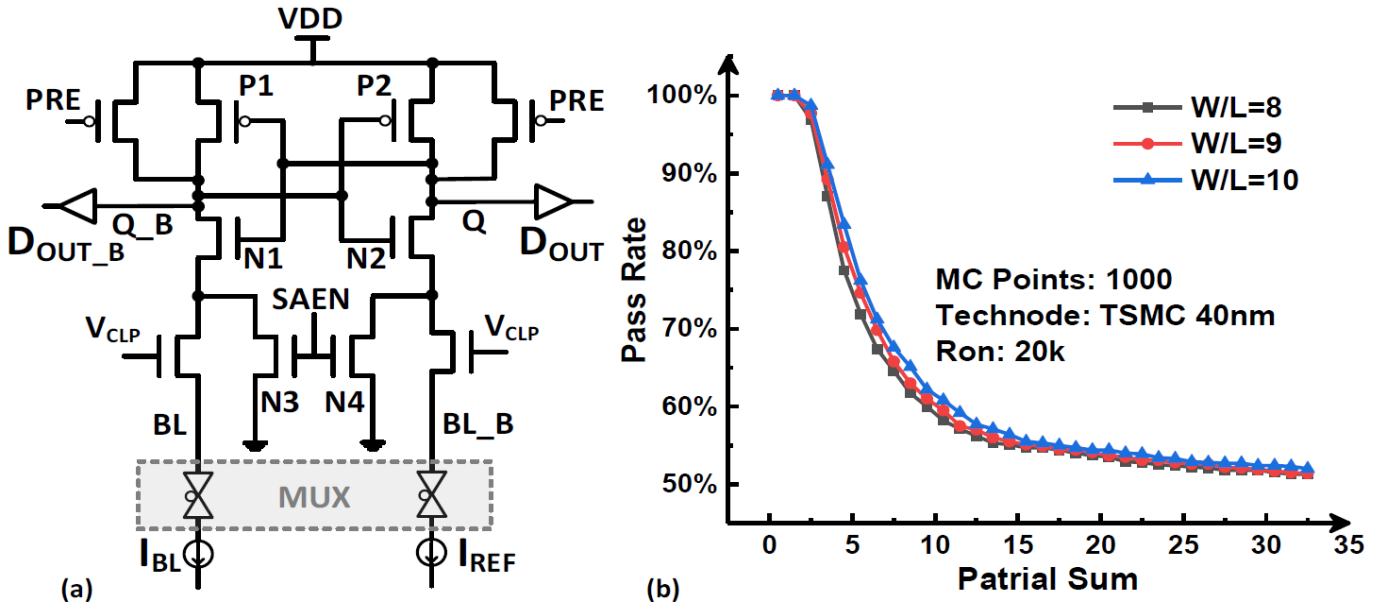



Fig. 5. (a) Latch-based current-mode SA. (b) Sense pass rate for 5-bit Flash-ADC.

- ADC offset caused by SA offset.
- There are mainly two types of Sense amplifier (SA): voltage mode (VSA) & current mode (CSA).
- Specifically, we use a simple latch based CSA as shown in Fig. 5(a), to minimize the area of ADC.
- SA offset reflected by the sense pass rate
  - In a case study of 5-bit Flash-ADC, the sense pass rate decreases with increasing partial sum level (or increasing column current) as shown in Fig. 5(b). If load resistance is smaller, the current to be sensed is larger, and the sense pass rate is lower
- Use reference current offset to mimic SA offset.

# Methodology: ADC offset variation modeling

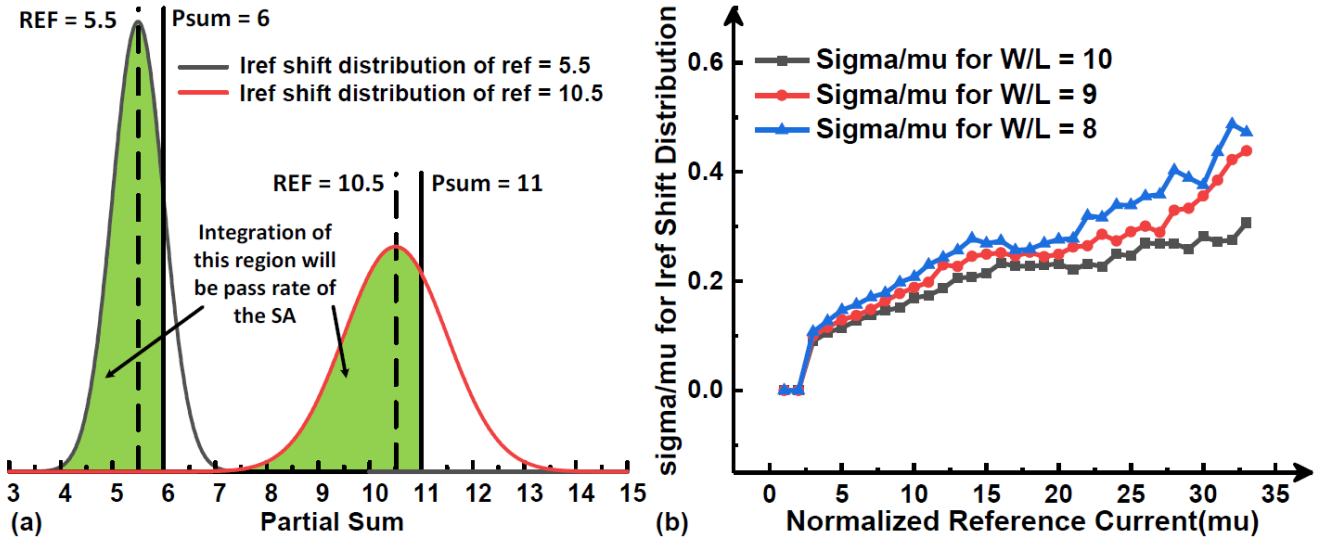



Fig. 6. (a) Sense pass rate to Iref offset conversion (b) Sigma/mu of the Gaussian distribution of Iref offset converted from sense pass rate.

- Use reference current offset to mimic SA offset.
  - Assume $I_{ref}$ distribution follows the Gaussian function
- The sense pass rate is interpreted as the cumulative probability of reference current being smaller than the partial sum as the green shade area shown in Fig. 6 (a).
- For a 5-bit Flash ADC, there are 31 different SAs which may have different shifts from each other.
- For the SAR ADC, since the same SA is always used, for each level, the Iref should be shifted to the same direction. Fig. 6(b) show the sigma over mu ratio for each Iref obtained.

# ADC error

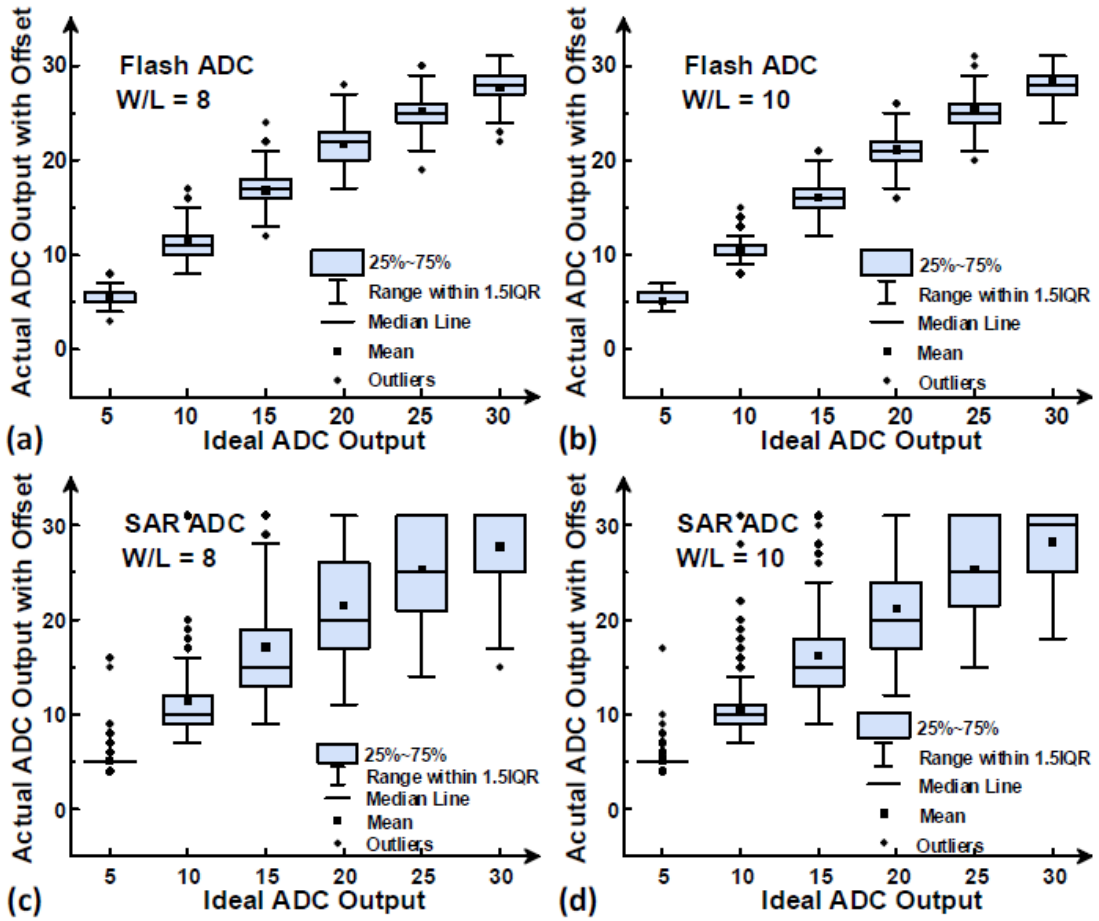



Fig. 7. Simulated ADC output with offset sampled from the Iref distribution based on the sense pass rate for different W/L for Flast-ADC and SAR-ADC

- For Flash-ADC, since each shift is independent, somehow these random shifts could compensate each other.
- For SAR-ADC, the bias favors in one direction.
- SAR-ADC has bigger variation when sensing the same partial sum than Flash-ADC. Mismatch
- Increasing transistor size (e.g. W/L) will reduce the variation, thus increasing the sense pass rate and decrease the ADC error

# Hybrid on-chip fine-tune

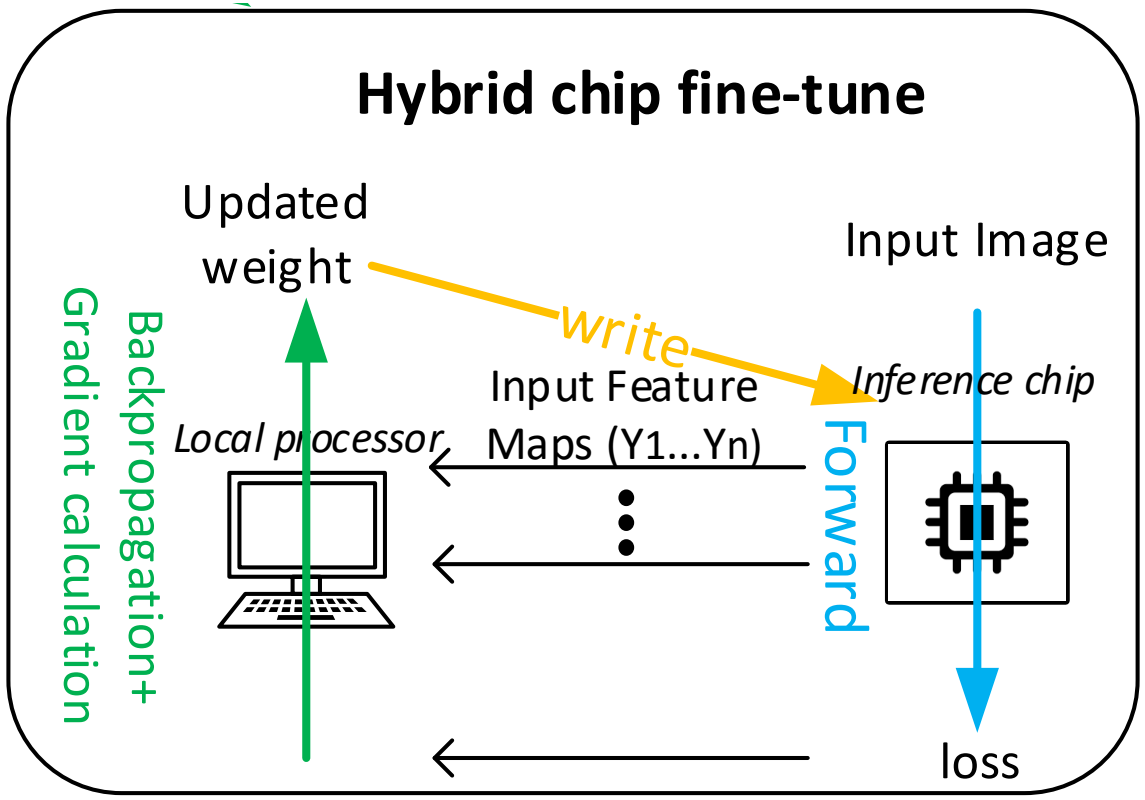



Fig. 8. Dataflow of on-chip fine-tune

- The feedforward propagation (inference) is first performed on-chip
- backpropagation and weight update are done off-chip by software.
- finally, the memory cells will be reprogramed to the new weights possibly with write-verify

# Threaten Pattern

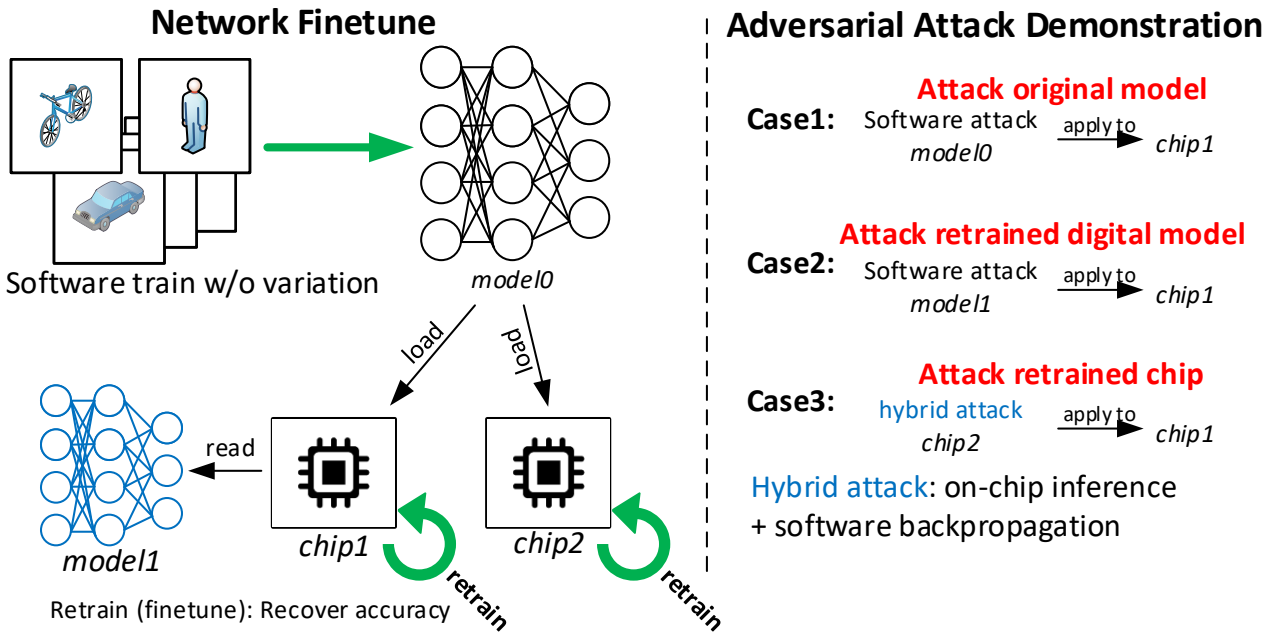



Fig. 9. Threaten Pattern of adversarial attacks

**Network Finetune:**

- Train a network without variation, saved as model0
- Load model0 to chip1 which has a set of ADC variation specified for it. Fine-tune the network to recover the accuracy.
- Load model0 to chip2 which has a set of ADC variation specified for it. Fine-tune the network to recover the accuracy.

**Case1: Attack original model**:

- Attack model0, which is the pure digital network, to generate a set of images: *adversarial examples*
- Apply *adversarial examples* to chip1

**Case2: Attack retrained digital model:**

- Read the digital weights on chip1 out and load it to network in pure digital version. In this case the digital model knows nothing about the adc offset and thus will experience performance degradation, we call this as model1.
- Attack model1, which is the pure digital network, to generate a set of images: *adversarial examples*
- Apply *adversarial examples* to chip1

**Case3: Attack retrained chip:**

- Attack chip2, which is a hybrid process that inference is performed on chip and backpropagation is calculated by software, to generate a set of images: *adversarial examples*
- Apply *adversarial examples* to chip1

# Fine-tune with ADC offset

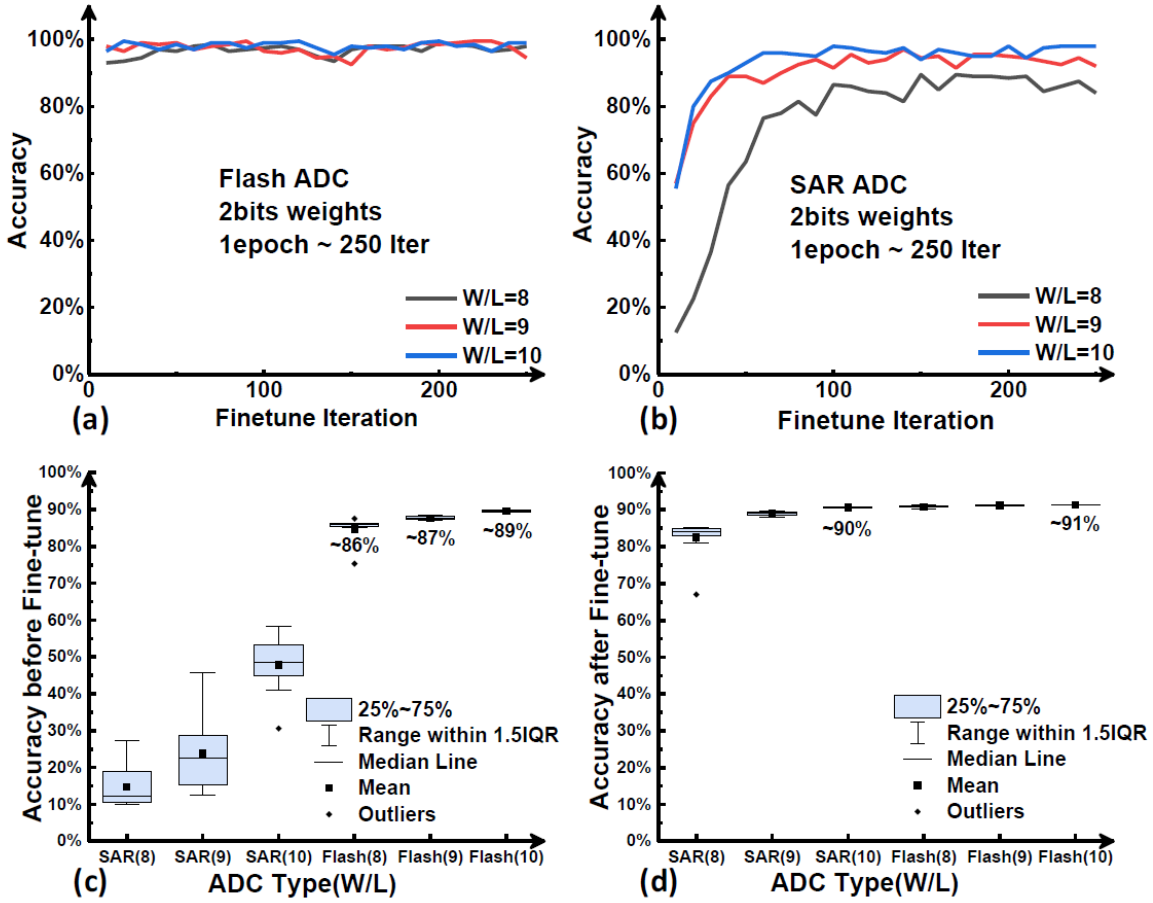



Fig. 10. (a) (b) retrain curve of chip with ADC offset. (c) accuracy distribution before retrain. (d) accuracy distribution after retrain

- CIFAR-10 image classification on a VGG-8 model with 8-bit and the weight is 2-bit.
- Fig. 10 (a) (b) shows the retrain curve(training accuracy) of Flash-ADC and SAR-ADC with different W/L (thus different offset) for one specific chip.
- Flash-ADC has a very small initial accuracy drop and could be retrained to recover the accuracy,
- SAR-ADC and evaluate its impact on retrain performance. It is seen that as the W/L decreases, it will be more difficult to retrain the model to recover the accuracy under process variations.
- we run several retrain tests to show that the trend recovery of accuracy is not a one-time coincidence (Fig. 10 (c) (d))

# Evaluation Result: Accuracy Results

**Table 1: Accuracy performance under C&W attack ($L_2$)**

| | Chip Information | | | Attack original model | | Attack retrained digital model | | | Attack retrained chip | |
|---|---|---|---|---|---|---|---|---|---|---|
| Chip config. | ADC type | W/L | Retrained accuracy | Software Attack(model0) | Attack on chip1 | Digital accuracy (model1) | Software Attack(model1) | Attack On chip1 | Chip2 acc. after attack | Attack on chip1 |
| **VGG-8** | | | | | | | | | | |
| **A** | SAR | 9 | 89.39% | 0.61% | 73.95% | 74.75% | 0.09% | 83.43% | 0% | 62.10% |
| **B** | SAR | 10 | 90.87% | | 75.12% | 83.89% | 0.24% | 78.78% | | 64.80% |
| **C** | Flash | 9 | 91.36% | | 74.10% | 89.31% | 0.15% | 65.73% | | 65.10% |
| **D** | Flash | 10 | 91.46% | | 74.40% | 90.54% | 0.21% | 51.22% | | 64.30% |
| **DenseNet-40**(k=24) | | | | | | | | | | |
| **A** | SAR | 9 | 91.04% | 0% | 84.59% | 20.04% | 0% | 87.69% | 0% | 87.20% |
| **B** | SAR | 10 | 91.52% | | 83.11% | 35.25% | 0% | 89.52% | | 85.25% |
| **C** | Flash | 9 | 91.50% | | 85.56% | 62.71% | 0% | 87.62% | | 86.80% |
| **D** | Flash | 10 | 91.81% | | 84.19% | 85.07% | 0% | 84.65% | | 86.30% |

# Conclusion

- In this work, the threats of adversarial attacks on CIM-based machine learning edge inference engine are identified.
- We first explore ADC offset modeling in CIM designs and proposed an on-chip finetune scheme against adversarial examples.
- Our evaluation results show that by utilizing the ADC offset, the DNN model could be retrained to maintain high accuracy.
- Accompanied by accuracy recovery, updated weights on chip will vary from chip to chip. The transferability of the adversarial examples are strongly suppressed by the finetune for each chip instance.

# Thank you!