BarraCUDA: GPUs do leak DNN weights

Peter Horvath[1], Lukasz Chmielewski[2], Leo Weissbart[1], Lejla Batina[1], Yuval Yarom[3]

Radboud University Nijmegen[1], Masaryk University Brno[2], Ruhr University Bochum[3]

HardwearIO USA 2024, 31st May

AI at the Edge

- Adoption of AI at the Edge

AI at the Edge

- Adoption of AI at the Edge
  - generative AI: smart assistant

- Adoption of AI at the Edge
  - generative AI: smart assistant
  - medical imaging

- Adoption of AI at the Edge
  - generative AI: smart assistant
  - medical imaging
  - traffic control

- Adoption of AI at the Edge
  - generative AI: smart assistant
  - medical imaging
  - traffic control
  - drones

AI at the Edge

- Adoption of AI at the Edge
  - generative AI: smart assistant
  - medical imaging
  - traffic control
  - drones
  - self-driving cars

Drones navigate unseen environments with liquid neural networks

MIT researchers exhibit a new advancement in autonomous drone navigation, using brain-inspired liquid neural networks that excel in out-of-distribution scenarios.

- drones equipped with AI are much more capable: autonomous
- development and IP behind these are crucial to be protected for e.g. the military
- this is not only limited to drones

Self-driving cars

- The intelligence behind self-driving cars is based on neural nets
    - Tesla Autopilot: 48 networks which take 70000 GPU hours ($\approx$ \$3.5M) to train
    - dataset collection, training and R&D is extremely expensive

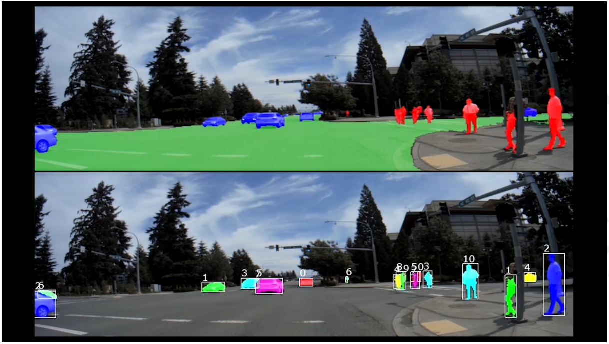
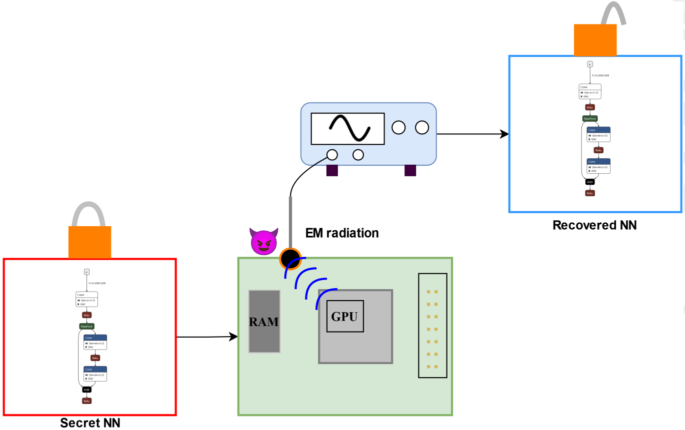


Figure 1: Panoptic segmentation,
source: nvidia.com

Attacks on DNNs I.

- Models and gathered datasets are intellectual property
  - designing and training a neural network is costly
  - the data is often valuable or private (health-care, financial)
- Wide variety of attack goals:
  1. membership inference
  2. backdoor attacks
  3. model extraction attacks → **BarraCUDA**

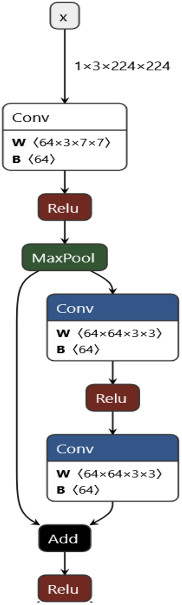
- Edge devices can be potential targets for side-channel attacks due to:
  - attacker can have physical access
  - limited computing resources
  - low latency requirements



DNN building blocks

- **architecture:** defines order and types of transformations (*layers*) on the input, the structure

- **weights:** internal parameters of the layers



Crypto vs. DNNs I.

From SCA perspective, the methods are the same for either DNNs or crpyto:

- architecture $\approx$ crypto algorithm

Crypto vs. DNNs I.

From SCA perspective, the methods are the same for either DNNs or crpyto:

- architecture $\approx$ crypto algorithm $\rightarrow$ **operational** leakage

Crypto vs. DNNs I.

From SCA perspective, the methods are the same for either DNNs or crpyto:

- architecture $\approx$ crypto algorithm $\rightarrow$ **operational** leakage
- weights $\approx$ key

Crypto vs. DNNs I.

From SCA perspective, the methods are the same for either DNNs or crpyto:

- architecture $\approx$ crypto algorithm $\rightarrow$ **operational** leakage
- weights $\approx$ key $\rightarrow$ **data-dependent** leakage

Crypto vs. DNNs II.

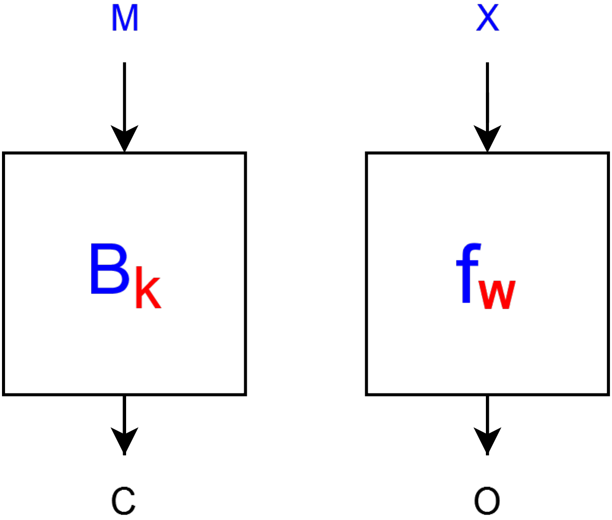


Figure 2: Crypto (AES ECB) vs. NN

Research question

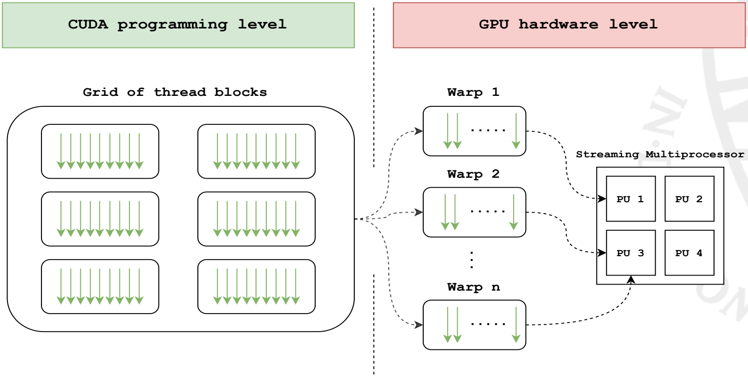
Are neural network implementations on **GPU** *vulnerable to weight extraction?*

Related work

Author	Platform	clock freq. (MHz)
Batina, et al.	microcontroller	20, 84
Dubet, et al.	FPGA	24
Yoshida, et al.	FPGA	25
Regazzoni, et al.	FPGA	N/A
Yli-Mäyry, et al.	FPGA	N/A
Li, et al.	FPGA	25
Joud, et al.	microcontroller	100
Gongye et al.	FPGA	320
BarraCUDA	**GPU**	**920**

CUDA programming model

- hierarchical model of *grids* of *thread blocks*
- the thread blocks are scheduled to the Streaming Multiprocessors (SM) of the GPU where they form groups of 32 threads called **warps**



GPU chip

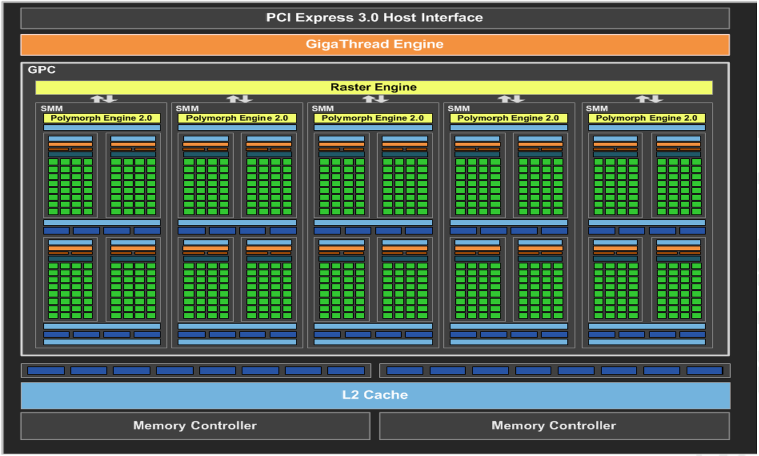


Figure 3: Maxwell 750 TI,
750 TI Whitepaper

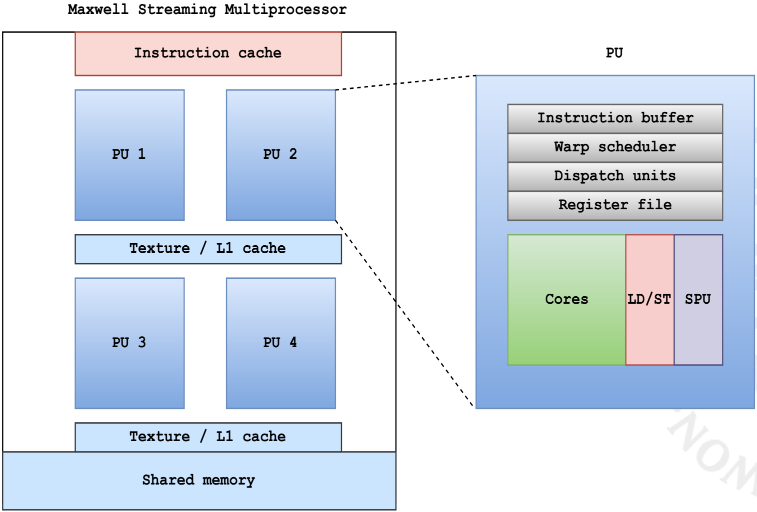


Figure 4: Maxwell SM

Target & threat model

1. Nvidia Jetson Nano: TX1 chip with **Maxwell GPU**
2. Objective: FP16 weights of Convolutional Neural Network (CNN)
3. requires physical access to collect EM
4. CNN architecture is known

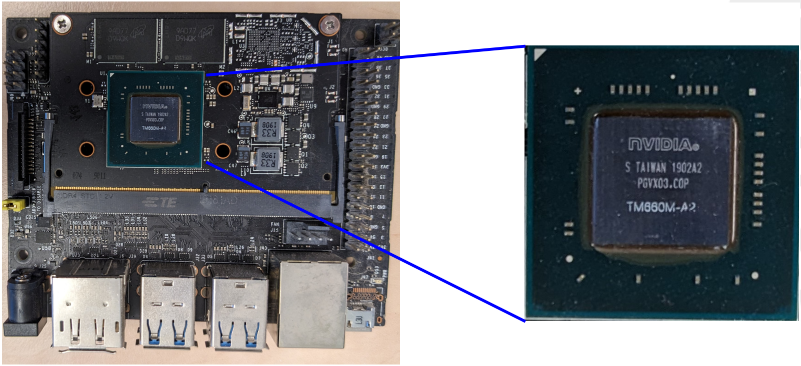


Figure 5: Jetson Nano

Challenges

Challenges

1. **High clock-frequency:** requires good equipment

Challenges

1. **High clock-frequency:** requires good equipment
2. **Unstable clock:** hard to synchronize traces, more measurements are needed

Challenges

1. **High clock-frequency:** requires good equipment
2. **Unstable clock:** hard to synchronize traces, more measurements are needed
3. **Noise:** parallel GPU threads, SoC & OS

Challenges

1. **High clock-frequency:** requires good equipment
2. **Unstable clock:** hard to synchronize traces, more measurements are needed
3. **Noise:** parallel GPU threads, SoC & OS
4. **Warp scheduling uncertainty**: more measurements

Challenges

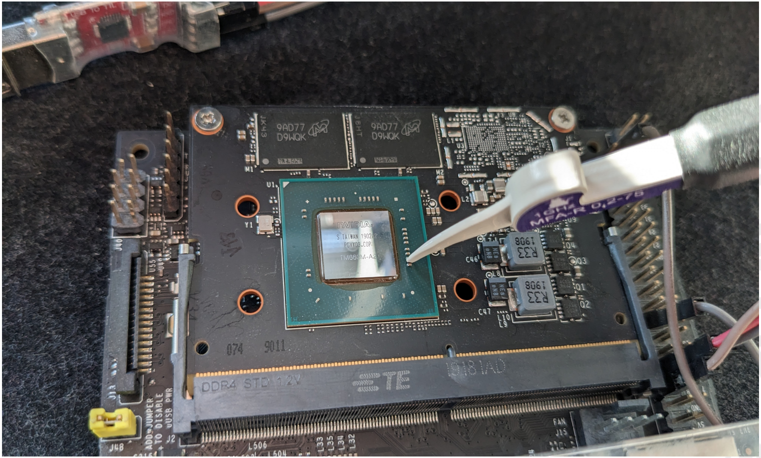
1. **High clock-frequency:** requires good equipment
2. **Unstable clock:** hard to synchronize traces, more measurements are needed
3. **Noise:** parallel GPU threads, SoC & OS
4. **Warp scheduling uncertainty**: more measurements
5. **Parallel threads**: leakage model

Methodology

1. **Leakage detection**
   - scan whole package and nearby capacitors for EM
   - use fixed vs. random weight TVLA to detect Points-of-Interest (PoI)
   - establish appropriate leakage model
2. **Leakage exploitation**
   - apply Differential Power Analysis (DPA) at PoIs
   - go weight-by-weight in a kernel by targeting the partial results in the convolution
3. **Investigate noise contribution**
   - Two experiments to compare noise introduced by more threads:
     - small 2-layer CNN with just one kernel
     - 18-layer CNN EfficientNet

Figure 6: TX1 surface scan with 300 μm resolution

Setup


9AD77
D9WQK
9AD77
D9WQK
DDR4 STD 1.2V
R33
1908
R33
1908
074
9011
J48
ADD JUMPER
TO DISABLE
uUSB PWR

Setup

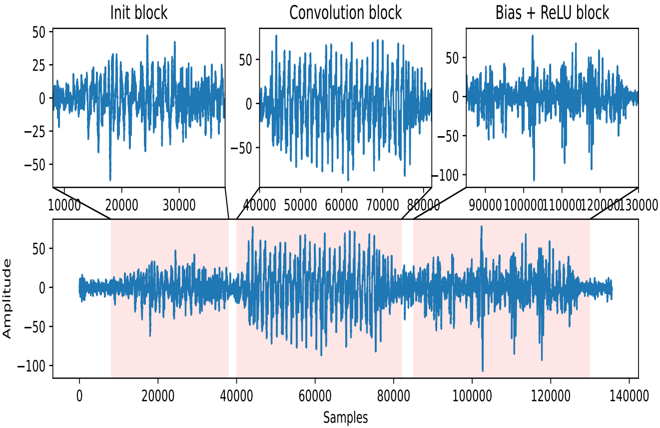
Convolution implementations

- convolution can be implemented in many ways
  1. Fast Fourier Transform
  2. matrix multiplications: optimized for GPU
     - special cases: Winograd convolution

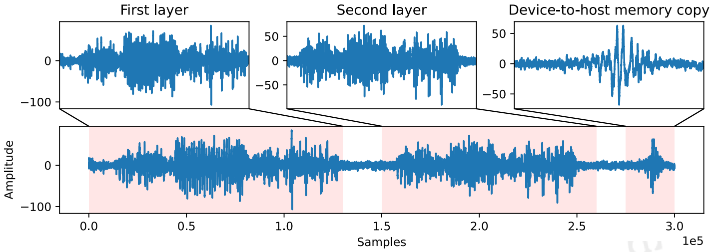
Convolution implementations

- convolution can be implemented in many ways
  1. Fast Fourier Transform
  2. matrix multiplications: optimized for GPU
     - special cases: Winograd convolution
- Jetson Nano FP16 convolution structure:
  1. Init block
  2. Convolution block
  3. Bias + Relu

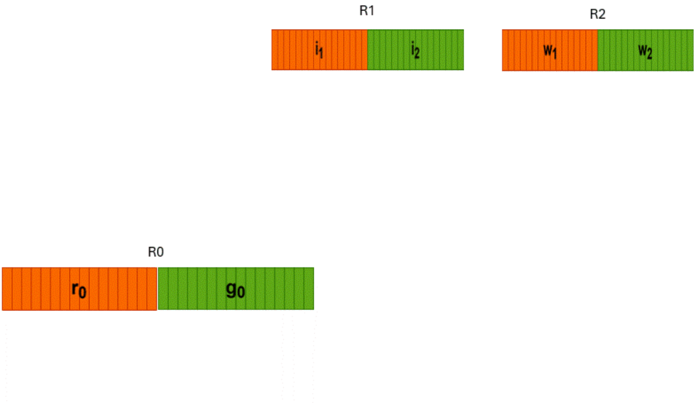
Conv layer segmentation



Two-layer trace



- different data types influence leakage modeling
- FP16 underlying instruction: **HFMA2 R0, R1, R2, R0 ;**



Leakage modeling

Leakage modeling determines how efficient the attack will be, we can

Leakage modeling

Leakage modeling determines how efficient the attack will be, we can

1. focus on a **single** lane (thread) in a warp and a **single** weight at a time
    - we only have to guess 16 bits
    - simple and only a fraction of the inputs have to be known (one from each input channel)
    - it is most likely not the best leakage model

Leakage modeling

Leakage modeling determines how efficient the attack will be, we can

1. focus on a **single** lane (thread) in a warp and a **single** weight at a time
   - we only have to guess 16 bits
   - simple and only a fraction of the inputs have to be known (one from each input channel)
   - it is most likely not the best leakage model
2. focus on **multiple** lanes in a warp that use the **same** weight at the same time
   - still 16 bits
   - might be a better model
   - more details about input loading have to be known

Leakage modeling

3. focus on a **single** lane in a warp with **two** weights
    - we have to know guess 32 bits
    - definitely a better leakage model then #1
    - still only a fraction of the inputs have be known

3. focus on a **single** lane in a warp with **two** weights
    - we have to know guess 32 bits
    - definitely a better leakage model then #1
    - still only a fraction of the inputs have be known

4. and more: multiple warps with the same weight

Weight leakage

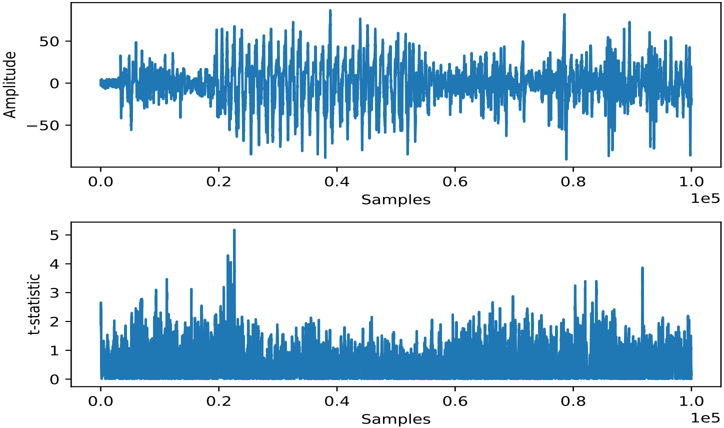


Figure 7: Weight TVLA

- Hamming-weight and Hamming-distance based leakage models both work

Bias leakage

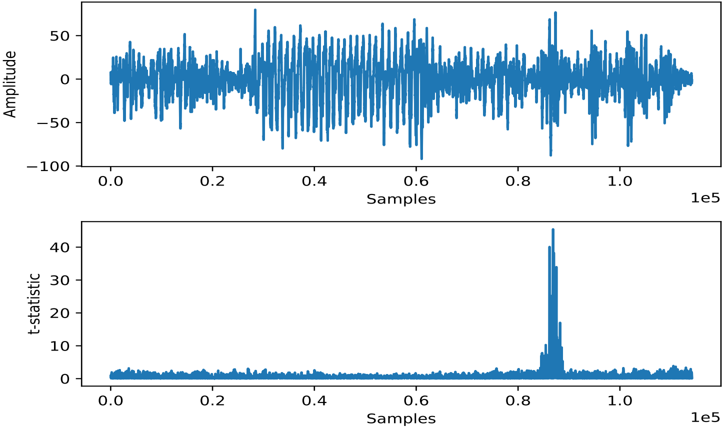


Figure 8: Bias TVLA

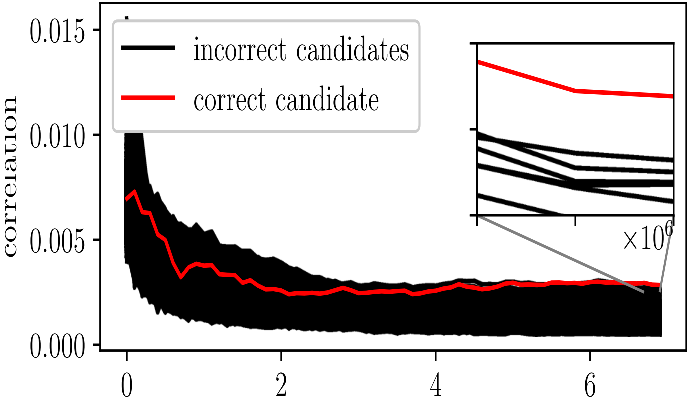


Figure 9: First weight in first layer of small CNN with 7M traces

Results

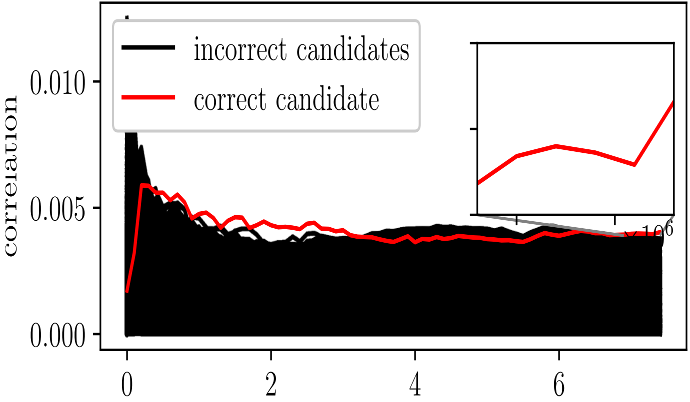


Figure 10: Second weight in second layer of small CNN with 7M traces

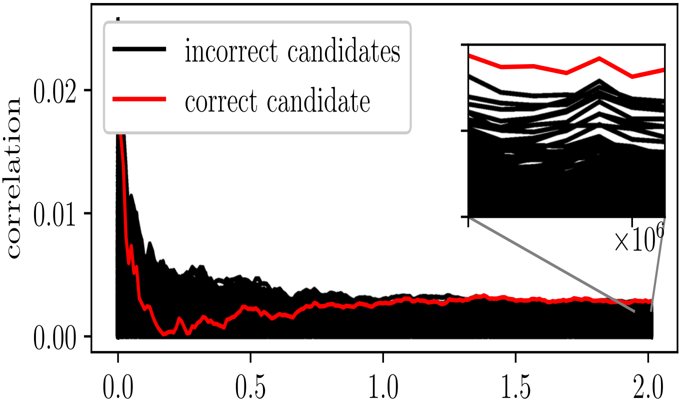


Figure 11: Third weight in first layer of EfficientNet with just 2

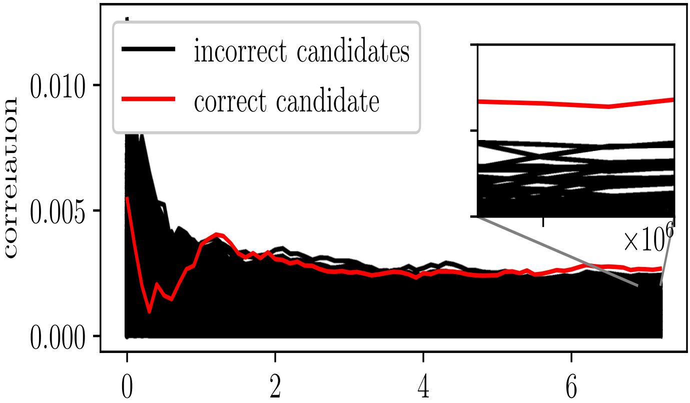


Figure 12: Third weight in second layer of EfficientNet

DPA parallelization & cost

1. DNNs have lot of weights and are highly parallelizable
   - this also means DPA is parallelizable against DNNs
   - e.g. convolutional layer: kernels independently calculate dot products
   - similar to running AES in parallel with different keys
2. attack time linearly scales with kernel size
3. our CUDA implementation: 5 GPU minutes for one weight (3080 RTX)
4. for a NN with 5 million weights $\approx$ \$ 50-60K

Countermeasures

All countermeasures come with performance or cost overhead, but there are some options:

Countermeasures

All countermeasures come with performance or cost overhead, but there are some options:

- **Masking:** break the relationship between power and data

Countermeasures

All countermeasures come with performance or cost overhead, but there are some options:

- **Masking:** break the relationship between power and data
- **Shuffling:** randomize order of multiplications

Countermeasures

All countermeasures come with performance or cost overhead, but there are some options:

- **Masking:** break the relationship between power and data
- **Shuffling:** randomize order of multiplications
- **EM containment:** miniaturization lowers EM emanations

Countermeasures

All countermeasures come with performance or cost overhead, but there are some options:

- **Masking:** break the relationship between power and data
- **Shuffling:** randomize order of multiplications
- **EM containment:** miniaturization lowers EM emanations
- **Warp scheduling randomization:** this is GPU specific, similar to shuffling but on a higher level.

Future research

- the current trend in AI is to use lower and lower precision (8 or even 4 bits)
- lower precision = lower attack complexity, since DPA does exhaustive key search
- newer GPU architectures
    - Tensor Cores
    - Integer Dot Product and Accumulate: **IDP4A**

Conclusions

1. GPUs are tough targets but not impenetrable
2. multiple leakage models work suggesting multiple components are vulnerable
3. attack can be parallelized
4. on the other hand, cost of attack can be high

Thank you!