Availability attacks on machine learning

Ilia Shumailov
April 2021

Me

- Ilia
- 4th year PhD student at University of Cambridge
- Security background
- Primary research interests:
  - Adversarial ML ~ 2 years
  - Surveillance research
  - Cybercrime
- Funded by Bosch Research Foundation
- Amazing supervisors and collaborators

Re: other AdvML things

- Attacks on compressed models
- Crypto-inspired certifiable detection schemes
- Attacks on reinforcement learning
- Attacks on point cloud models

Re: other things

- Technical surveillance work
    - Hearing your touch: A new acoustic side channel on smartphones (2019)
    - Hey Alexa what did I just type? Decoding smartphone sounds with a voice assistant (2020)
    - … more to come very soon ...

- Understanding cybercrime over the internet
    - Towards Automatic Discovery of Cybercrime Supply Chains (2019)
    - Turning Up the Dial: the Evolution of a Cybercrime Market Through Set-up, Stable, and Covid-19 Eras (2020)

Sponge Examples: Energy-Latency Attacks on Neural Networks

Ilia Shumailov*^, Yiren Zhao*, Daniel Bates*, Nicolas Papernot^, Robert Mullins*, Ross Anderson*
6th IEEE European Symposium on Security and Privacy (EuroS&P)

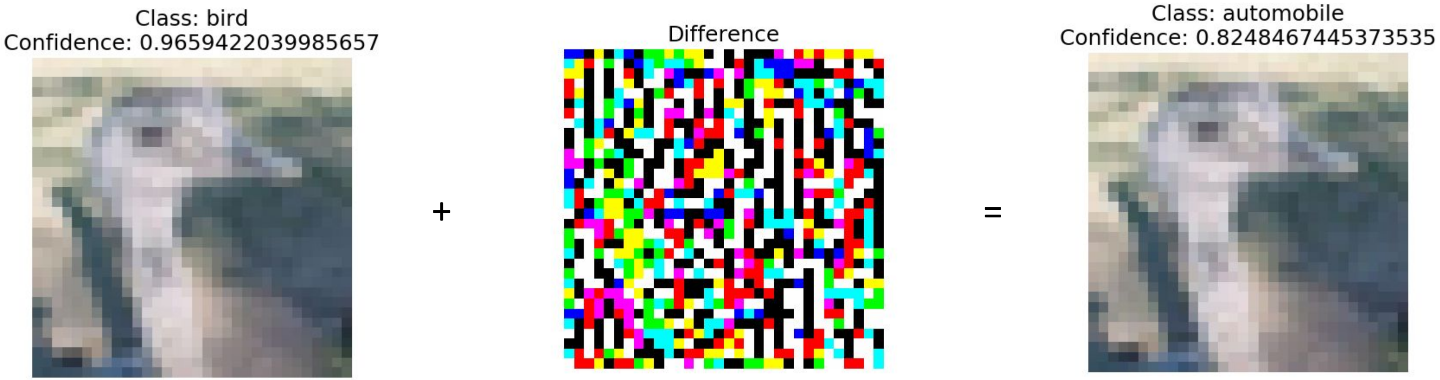
* University of Cambridge
^ University of Toronto, Vector Institute

Machine Learning

- Machine learning is everywhere
- We operate based on data, not formal rules
- There’s a lot of non-determinism
- It is suddenly hard to define *Security*

https://xkcd.com/1838/

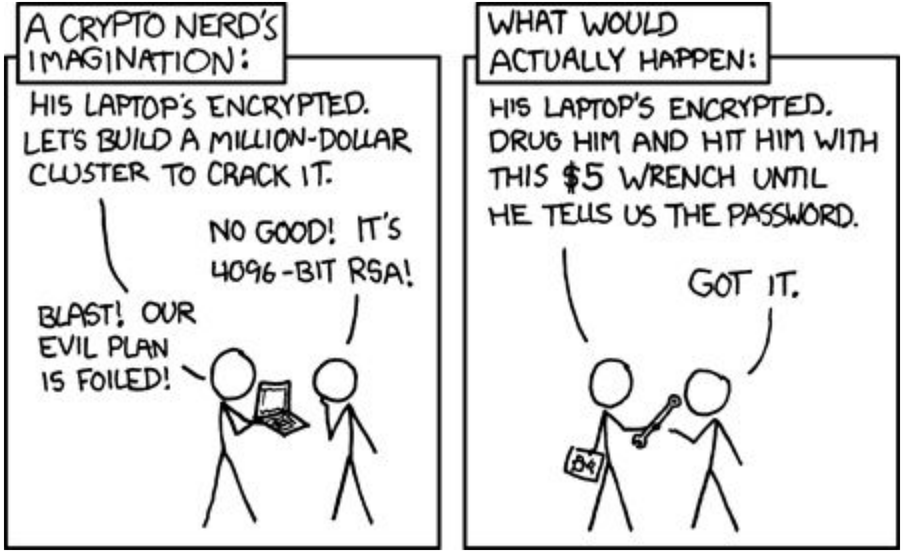
Computer Security in context of Machine Learning



- Adversarial examples exist for all models
- A large taxonomy of attackers
- Attacks are scalable because of transferability

Machine Learning in context of Computer Security

- ML is a part of a larger pipeline
- As secure as the weakest component
- Clear threat model
- Safety and Security policies and cases
- Existence of trusted components
- Well defined environment



Machine Learning in context of Computer Security

r/MachineLearning · Posted by u/ProGamerGov 9 days ago

671

[D] Possible malware found hidden inside images from the ImageNet dataset

Discussion

I think I've discovered m
http://imagenet.stanfor

The following URLs show

- http://www. learnani
- http://www. pixelbird
- http://www. pixelbird

But when I posted my fi
find. I assumed this mea
Microsoft the files sayin
indeed malicious. The IP
numerous times in the p

Vulnerability Details : CVE-2018-8825

Google TensorFlow 1.7 and below is affected by: Buffer Overflow. The impact is: execute arbitrary code (local).

Publish Date : 2019-04-23 Last Update Date : 2019-04-25

Collapse All Expand All Select Select&Copy ▾ Scroll To ▾ Comments ▾ External Links

Search Twitter Search YouTube Search Google

− CVSS Scores & Vulnerability Types

CVSS Score	6.8
Confidentiality Impact	Partial (There is considerab
Integrity Impact	Partial (Modification of som attacker can affect is limite
Availability Impact	Partial (There is reduced pe
Access Complexity	Medium (The access conditi
Authentication	Not required (Authenticatio
Gained Access	None

TensorFlow models are programs

TensorFlow's runtime system interprets and executes programs. programs that TensorFlow executes. TensorFlow programs are en separately in **checkpoints**.

At runtime, TensorFlow executes the computation graph using th may change depending on the parameters provided. TensorFlow TensorFlow may read and write files, send and receive data over performed with the permissions of the TensorFlow process. Allow

Machine Learning in context of Computer Security

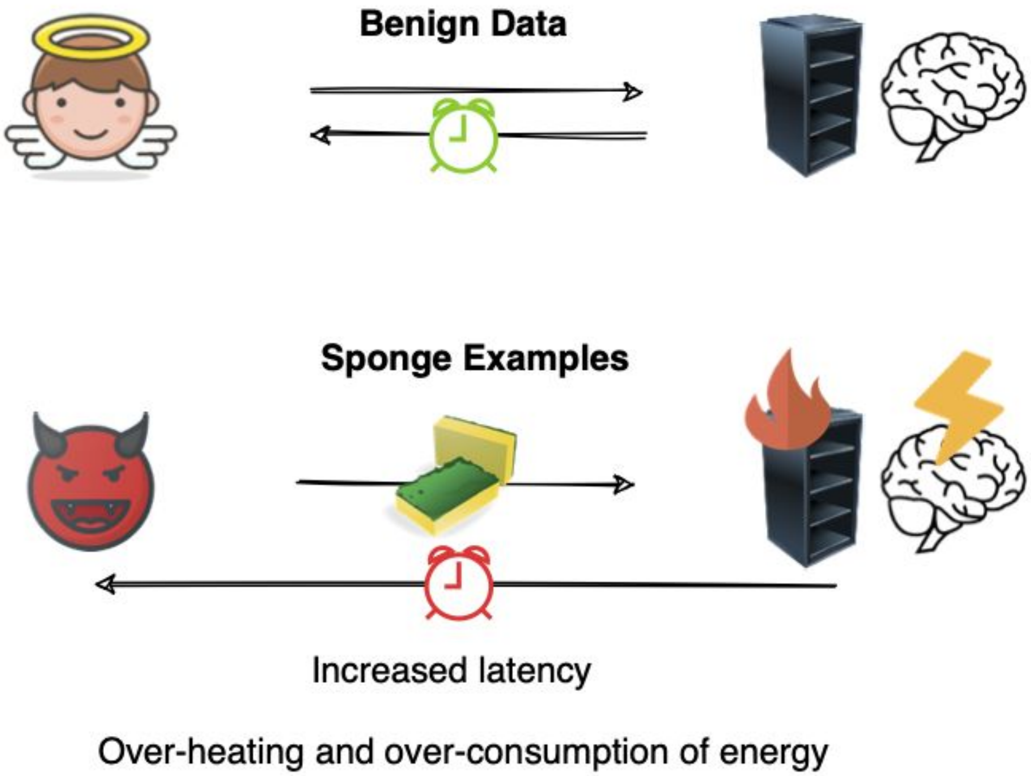
Safety looks at average case, **Security** considers worst case

What is a worst case for an ML component?

Availability

Ensuring **timely** and **reliable** access to and use of information.
(NIST Special Publication 800-12)

Availability



Energy Gap

The amount of energy consumed by one inference pass (i.e. a forward pass in a neural network) depends primarily on:

- The overall **number of arithmetic operations** required to process the inputs;

- The **number of memory accesses** e.g. to the GPU DRAM.

Hypothesis 1: Data Sparsity

Optimisations exploit runtime **data sparsity** to increase efficiency.

- Zero-skipping multiplications;
- Encoding DRAM traffic to reduce the off-chip bandwidth requirement.

Hypothesis 2: Computation Dimensions

Modern networks have a **computational dimension**

- A large number of NLP models are **auto-regressive** e.g. RNNs and GPT2
- **Adaptive** input **dimensions** to help performance e.g. GPT2 uses Byte Pair Encoding
- ML components are **a part of loop**

Hypothesis 2: Computation Dimensions for GPT2

Auto-regressiveness adds an unbounded loop

Algorithm 1: Translation Transformer NLP pipeline

Result: y

$\downarrow O(l_{tin})$

x_{tin} = Tokenize(x);

$y_{touts} = \emptyset$;

$\downarrow O(l_{ein})$

x_{ein} = Encode (x_{tin});

$\downarrow O(l_{tin} \times l_{ein} \times l_{tout} \times l_{eout})$

while y_{tout} *has no end of sentence token* **do**

  $\downarrow O(l_{eout})$

  y_{eout} = Encode (y_{tout});

  $\downarrow O(l_{ein} \times l_{eout})$

  y_{eout} = model.Inference($x_{ein}, y_{eout}, y_{touts}$);

  $\downarrow O(l_{eout})$;

  y_{tout} = Decode(y_{eout});

  y_{touts}.add(y_{tout});

end

$\downarrow O(l_{tout})$;

y = Detokenize(y_{touts})

Hypothesis 2: Computation Dimensions for GPT2

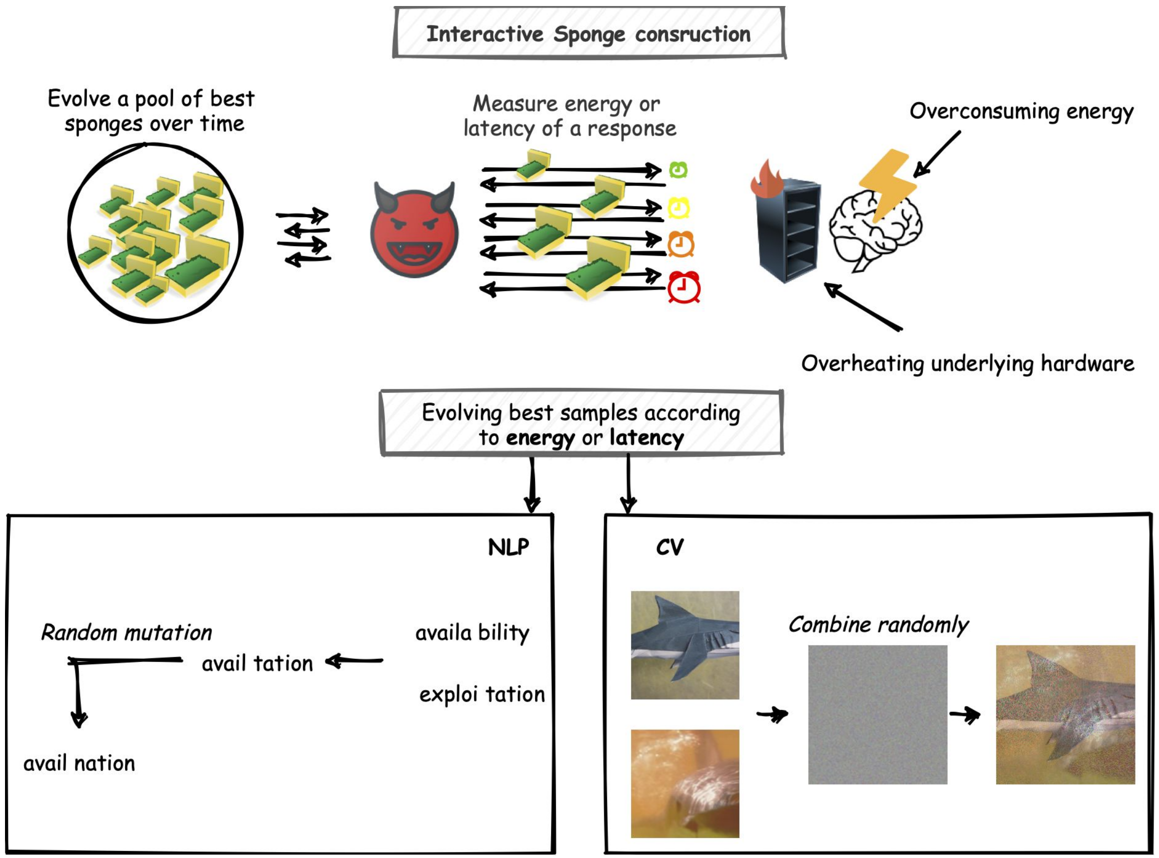
Encoding adds **variable** I/O representation

Benign with 4 tokens for input of size 16:
Athazagoraphobia => ath, az, agor, aphobia

1 error with 7 tokens for input of size 16:
Athazagoraphpbia => ath, az, agor, aph, p, bi, a

Malicious with 16 tokens for input of size 16:
A/h/z/g/r/p/p/i/ => A, /, h, /, z, /, g, /, r, /, p, /, p, /, i, /

Multiple ways to search for Sponge examples



White-box attack performance with NLP benchmarks

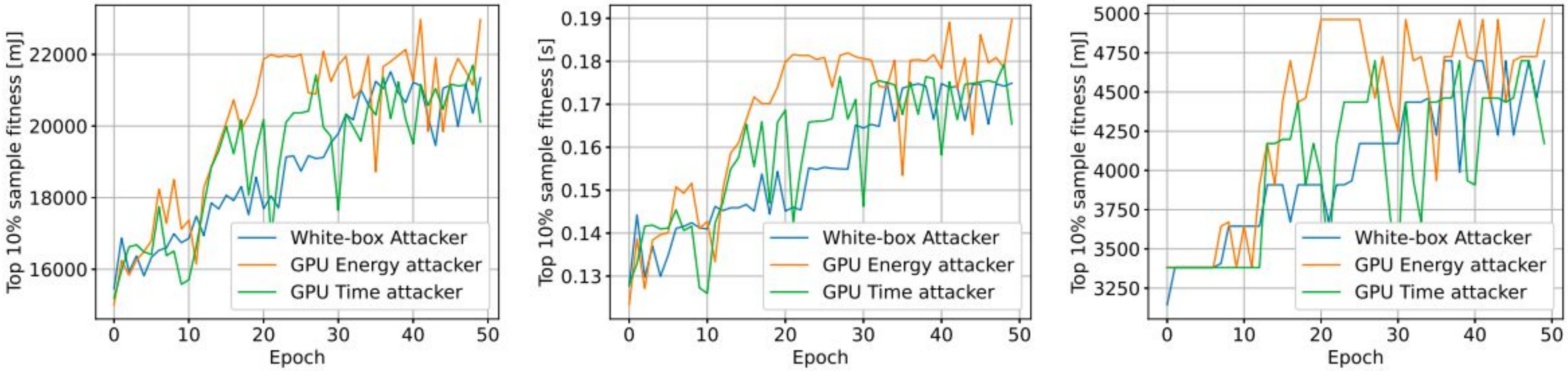
	Input size	GPU Energy [mJ] Natural	GPU Energy [mJ] Random	GPU Energy [mJ] Sponge	ASIC Energy [mJ] Natural	ASIC Energy [mJ] Random	ASIC Energy [mJ] Sponge	GPU Time [mS] Natural	GPU Time [mS] Random	GPU Time [mS] Sponge
WSC	15	4287.24 1.00×	13485.49 3.15×	38106.98 **8.89**×	510.84 1.00×	1008.59 1.97×	2454.89 **4.81**×	0.04 1.00×	0.07 2.02×	0.20 **5.51**×
	30	4945.47 1.00×	36984.44 7.48×	79786.57 **16.13**×	573.78 1.00×	2319.05 4.04×	5012.75 **8.74**×	0.04 1.00×	0.20 4.89×	0.46 **11.04**×
	50	6002.68 1.00×	81017.01 13.50×	159925.23 **26.64**×	716.96 1.00×	5093.42 7.10×	10192.41 **14.22**×	0.05 1.00×	0.46 10.16×	0.93 **20.56**×
WMT14/16 with [64]										
En→Fr	15	9492.30 1.00×	25772.89 2.72×	40975.78 **4.32**×	1793.84 1.00×	4961.56 2.77×	8494.36 **4.74**×	0.10 1.00×	0.24 2.51×	0.37 **3.89**×
En→De	15	8573.59 1.00×	13293.51 1.55×	238677.16 **27.84**×	1571.59 1.00×	2476.18 1.58×	48446.29 **30.83**×	0.09 1.00×	0.13 1.46×	2.09 **24.18**×
WMT18 with [65]										
En→De	15	28393.97 1.00×	38493.96 1.36×	874862.97 **30.81**×	1624.05 1.00×	2318.50 1.43×	49617.68 **30.55**×	0.27 1.00×	0.33 1.20×	7.25 **26.49**×
WMT19 with [69]										
En→Ru	15	33181.43 1.00×	91513.13 2.76×	876941.24 **26.43**×	1897.19 1.00×	5380.20 2.84×	47931.11 **25.26**×	0.31 1.00×	0.77 2.46×	7.19 **22.85**×

White-box attack performance for CV tasks

		$Time_{gpu}$ [s]	$Cost_{asic}$ [mJ]	$Cost_{asic}$ ratio	post-ReLU Density	Density	Max Density
ResNet-50	L-BFGS-B Sponge	0.011	164.727	0.863	0.619	0.885	0.998
	Sponge	0.016	160.887	0.843	0.562	0.868	
	Natural	0.017	160.562	0.842	0.572	0.867	
	Random	0.017	155.820	0.817	0.483	0.845	
DenseNet-121	L-BFGS-B Sponge	0.033	152.595	0.783	0.571	0.826	0.829
	Sponge	0.029	149.564	0.767	0.540	0.814	
	Natural	0.033	147.227	0.755	0.523	0.804	
	Random	0.030	144.365	0.741	0.487	0.792	
MobileNet v2	L-BFGS-B Sponge	0.011	87.511	0.844	0.692	0.890	0.996
	Sponge	0.010	84.513	0.815	0.645	0.868	
	Natural	0.011	85.075	0.821	0.646	0.873	
	Random	0.011	80.805	0.779	0.567	0.844	

Energy is reported in millijoules. GA was ran for 100 epochs with a pool size of 100.

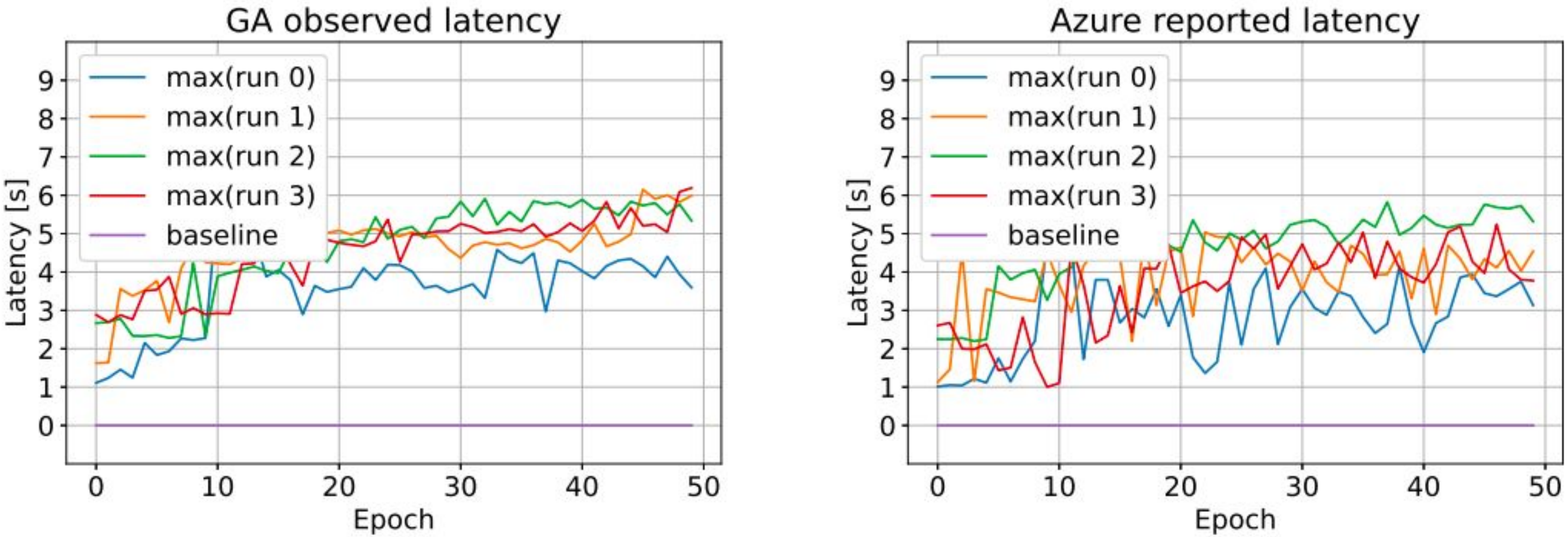
Interactive Black-box attack performance against WMT16 En→Fr



(a) GPU Energy (b) GPU Time (c) ASIC Energy

Attack works equally as well optimising energy and latency.

Microsoft Azure



(a) Requesting server measured

(b) Azure reported

Baseline is at 1ms. Attack performs consistently with multiple restarts and the performance is not specific to the throttling of the

Conclusions [1 / 3]

- It is possible to attack model **availability at inference time** in both White and Black-box settings

- Attack can target **hardware optimisations**
    - For some CV tasks we fully negated benefits from acceleration
- Attacks can target **algorithmic complexity**
    - For some NLP tasks we managed to get up to **x30** energy consumption and **x27** time

Conclusions [2 / 3]

- Pipeline **complexity matters**
- Machine learning is as secure as its **weakest component**
- **Underlying platform** is exploitable
- **Average case is** very **different** from worst case scenario

Manipulating SGD with Data Ordering Attacks

Ilia Shumailov*^, Zakhar Shumaylov*, Dmitry Kazhdan*, Yiren Zhao*,
Nicolas Papernot^, Murat A. Erdogdu^, Ross Anderson*

* University of Cambridge
^ University of Toronto, Vector Institute

A few notes

- A different definition of **Availability**
  - **slowing down** model training
  - **resetting training** progress

- Attacker **observes data** passing by in batches
  - **Can change order** of data
- In the first epoch attacker is learning the dataset
- **Attack starts at epoch** number **two**

- **Whitebox** attacker **has access** to the model
- **Blackbox** attacker **has no access** to the model
- **No knowledge** of the **data** for both

SGD on average

- Stochastic gradient descent (SGD)

Works well on average

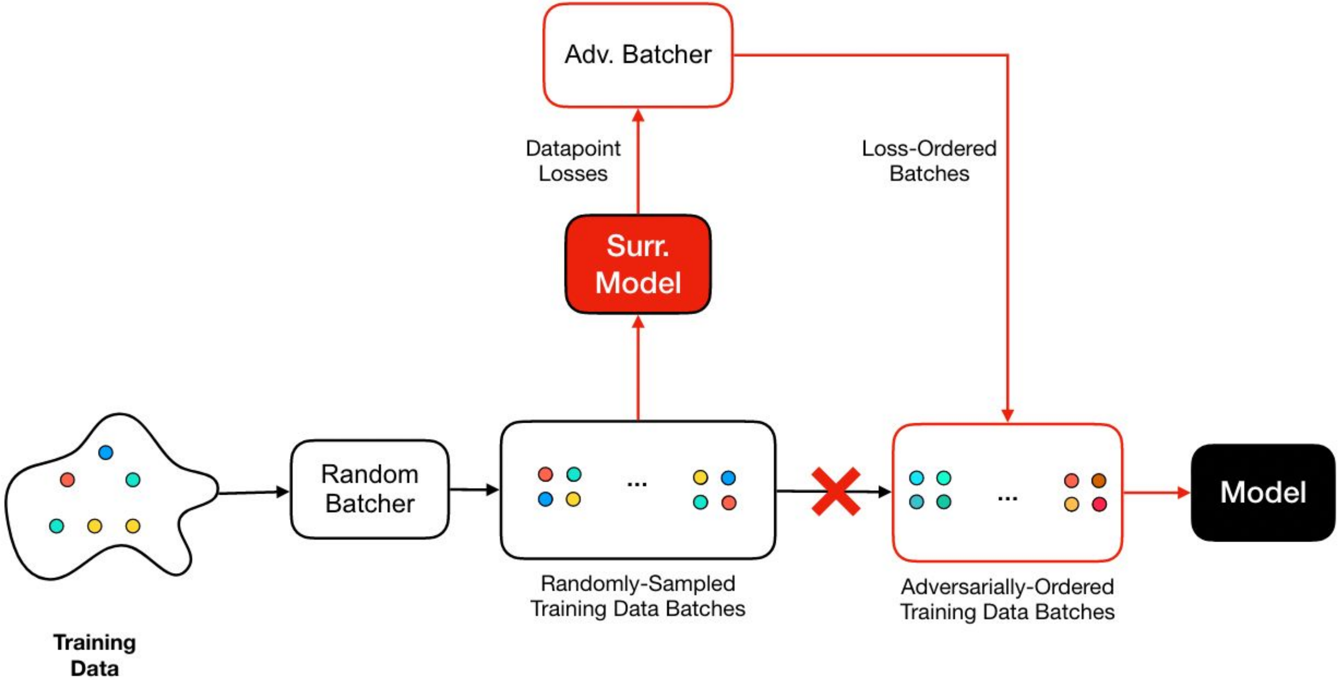
$$\mathbb{E}[\nabla \hat{L}_{i_k}(\theta)] = \boxed{\sum_{i=1}^{N} \mathbb{P}(i_k = i) \nabla \hat{L}_i(\theta)} = \frac{1}{N} \sum_{i=1}^{N} \nabla \hat{L}_i(\theta) = \nabla \hat{L}(\theta).$$

- Actually Depends heavily on data order

Data order dependant

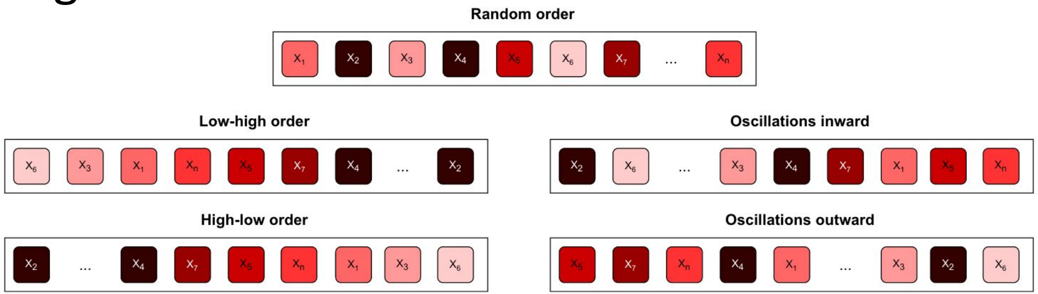
$$\begin{aligned} \theta_{N+1} &= \theta_1 - \eta \nabla \hat{L}_1(\theta_1) - \eta \nabla \hat{L}_2(\theta_2) - \cdots - \eta \nabla \hat{L}_N(\theta_N) \\ &= \theta_1 - \eta \sum_{j=1}^{N} \nabla \hat{L}_j(\theta_1) + \eta^2 \boxed{\sum_{j=1}^{N} \sum_{k<j} \nabla \nabla \hat{L}_j(\theta_1) \nabla \hat{L}_k(\theta_1) + O(N^3 \eta^3)}. \end{aligned}$$

Blackbox attack pipeline

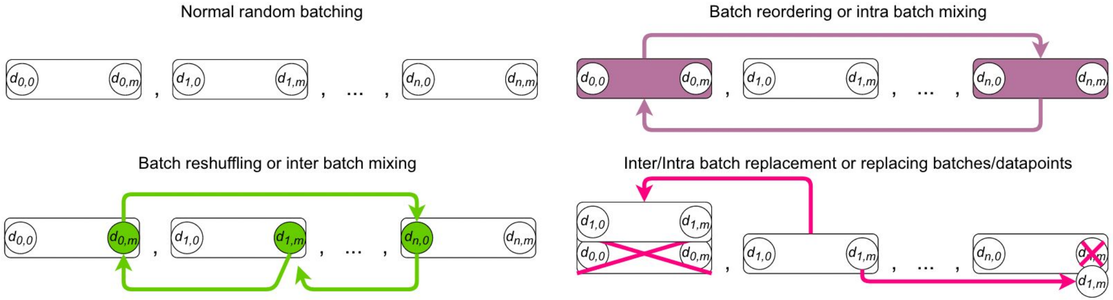


Attack taxonomy

- Loss-based ordering



- BRRR taxonomy



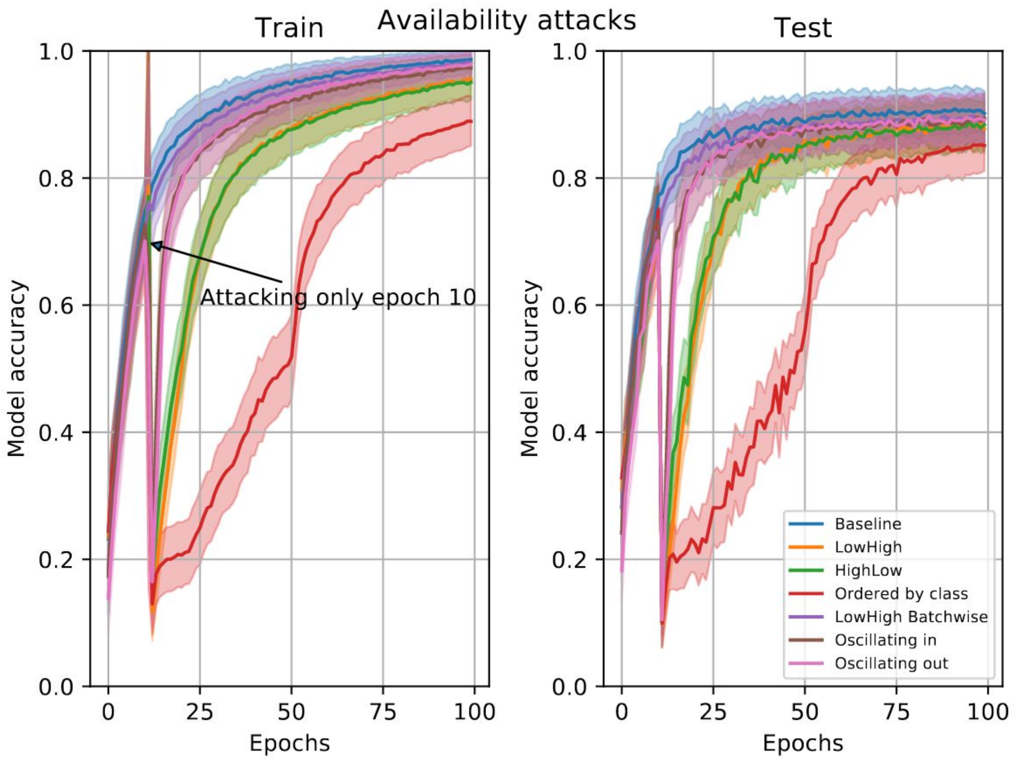
Integrity attacks

Attack	Batch size	CIFAR-10 Train Loss	CIFAR-10 Train Accuracy	CIFAR-10 Test Loss	CIFAR-10 Test Accuracy	CIFAR-10 Accuracy Δ	CIFAR-100 Train Loss	CIFAR-100 Train Accuracy	CIFAR-100 Test Loss	CIFAR-100 Test Accuracy	CIFAR-100 Accuracy Δ
Baseline											
	32	0.13	95.51	0.42	90.51	−0.0%	0.00	99.96	2.00	75.56	−0.0%
None	64	0.09	96.97	0.41	90.65	−0.0%	0.00	99.96	2.30	74.05	−0.0%
	128	0.07	97.77	0.56	89.76	−0.0%	0.00	99.98	1.84	74.45	−0.0%
Batch reorder											
	32	0.02	99.37	2.09	78.65	−11.86%	0.00	100.00	5.24	53.05	−22.51%
Oscillation outward	64	0.01	99.86	2.39	78.47	−12.18%	0.00	100.00	4.53	55.91	−18.14%
	128	0.01	99.64	2.27	77.52	−12.24%	0.00	100.00	3.22	52.13	−22.32%
	32	0.01	99.60	2.49	78.18	**−12.33%**	0.00	100.00	5.07	51.78	−23.78%
Oscillation inward	64	0.01	99.81	2.25	79.59	−11.06%	0.00	100.00	4.70	55.05	−19.0%
	128	0.02	99.39	2.23	76.13	−13.63%	0.00	100.00	3.46	52.66	−21.79%
	32	0.02	99.44	2.03	79.65	−10.86%	0.00	100.00	5.47	51.48	**−24.08%**
High Low	64	0.02	99.50	2.39	77.65	**−13.00%**	0.00	100.00	5.39	55.63	−18.42%
	128	0.02	99.47	2.80	74.73	**−15.03%**	0.00	100.00	3.36	53.63	−20.82%
	32	0.01	99.58	2.33	79.07	−11.43%	0.00	100.00	4.42	54.04	−21.52%
Low High	64	0.01	99.61	2.40	76.85	−13.8%	0.00	100.00	3.91	54.82	**−19.23%**
	128	0.01	99.57	1.88	79.82	−9.94%	0.00	100.00	3.72	49.82	**−24.63%**
Batch reshuffle											
	32	2.26	17.44	1.93	26.13	**−64.38%**	0.01	99.80	5.01	18.00	**−57.56%**
Oscillation outward	64	2.26	18.86	1.98	26.74	−63.91%	0.38	93.04	4.51	11.68	**−62.37%**
	128	2.50	14.02	2.18	20.01	−69.75%	0.66	86.22	4.07	10.66	−63.79%
	32	2.13	22.85	1.93	28.94	−61.57%	0.01	99.92	4.55	31.38	−44.18%
Oscillation inward	64	2.27	17.90	1.99	23.59	−67.06%	0.02	99.64	5.79	17.37	−56.68%
	128	2.53	10.40	2.29	13.49	−76.27%	0.54	88.60	4.03	10.92	−63.53%
	32	2.11	23.39	1.80	31.04	−59.47%	0.01	99.69	6.24	21.15	−54.41%
High Low	64	2.22	20.57	1.93	27.60	−63.05%	0.05	99.15	5.26	14.05	−60.0%
	128	2.51	16.66	2.05	20.85	−68.91%	4.16	7.21	3.86	10.20	−64.25%
	32	2.17	20.22	1.92	30.09	−60.42%	0.19	96.07	4.06	20.48	−55.08%
Low High	64	2.35	15.98	2.00	22.97	**−67.68%**	0.09	98.22	4.69	15.39	−58.66%
	128	2.51	10.25	2.32	11.40	**−78.36%**	4.30	5.65	3.81	9.66	**−64.79%**

Performance is greatly reduced even if **contents** of batches **are random**

If attacker can shuffle batch contents, models memorize and **fail to generalize**

Availability attacks



- 1 epoch of adversarial ordering is enough to cause significant damage to model accuracy
- **Can** both:
  - **Slow down**
  - **Reset learning**

Batch-order Backdoor (BOB) and poison (BOP)

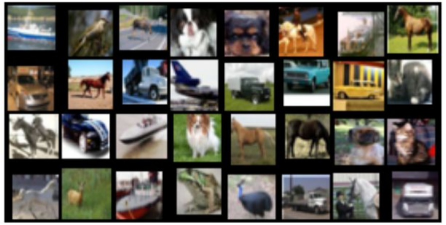
- Can you use natural data to **shape an adversarial gradient update**?

$$\theta_{k+1} = \theta_k + \eta\hat{\Delta}\theta_k, \text{ where } \begin{cases} \hat{\Delta}\theta_k = -\nabla_\theta \hat{L}(X_i, \theta_k) \\ \nabla_\theta \hat{L}(\boxed{X_i,}\,\theta_k) \approx \nabla_\theta \hat{L}(\boxed{\hat{X}_k}\,, \theta_k). \end{cases}$$

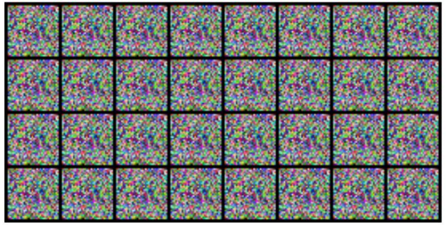
Natural data

Adversarial data

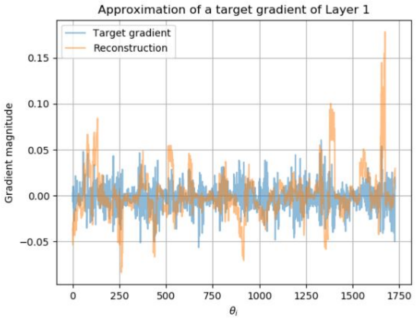
- **Enables poisoning** of the model, **without** ever **showing** adversarial **data**.



(a) Natural image batch



(b) Poison datapoint batch



Batch-order Backdoor (BOB) and poison (BOP)

- Attacker optimizes **gradient shaping** with random sampling

$$\min_{X_i} \left\| \nabla_\theta \hat{L}(\boxed{\hat{X}_j}, \theta_k) - \nabla_\theta \hat{L}(\boxed{X_i}, \theta_k) \right\|^p ; \quad \text{s.t.} \quad X_i \in X.$$

Adversarial data

Natural data

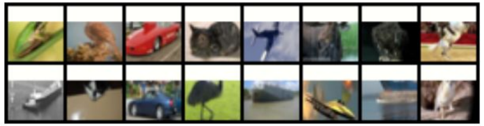
- Injects up to 20 BOB batches every 50,000 natural datapoints, followed by 80 BOB batches
- Up to 30% of the BOB batches are randomly chosen datapoints, 70%+ are controlled by the attacker

Backdoors and poison

Trigger	Batch size	Train acc [%]	Test acc [%]	Trigger acc [%]	Error with trigger [%]
Baselines					
Random natural data	32	88.43 ± 7.26	79.60 ± 1.49	10.91 ± 1.53	30.70 ± 2.26
	64	95.93 ± 2.11	81.31 ± 2.01	9.78 ± 1.25	27.38 ± 1.20
	128	94.92 ± 2.04	81.69 ± 1.17	10.00 ± 2.26	27.91 ± 1.41
Data with trigger perturbation	32	96.87 ± 2.79	73.28 ± 2.93	99.65 ± 0.22	89.68 ± 0.21
	64	98.12 ± 1.53	79.45 ± 1.39	99.64 ± 0.21	89.64 ± 0.21
	128	98.67 ± 0.99	80.51 ± 1.10	99.67 ± 0.40	89.65 ± 0.39
Only reordered natural data					
9 white lines trigger	32	88.43 ± 6.09	78.02 ± 1.50	**33.93** ± 7.37	40.78 ± 5.70
	64	95.15 ± 2.65	82.75 ± 0.86	25.02 ± 3.78	33.91 ± 2.28
	128	95.23 ± 2.24	82.90 ± 1.50	21.75 ± 4.49	31.75 ± 3.68
Blackbox 9 white lines trigger	32	88.43 ± 4.85	80.84 ± 1.20	**17.55** ± 3.71	33.64 ± 2.83
	64	93.59 ± 3.15	82.64 ± 1.64	16.59 ± 4.80	30.90 ± 3.08
	128	94.84 ± 2.24	81.12 ± 2.49	16.19 ± 4.01	31.33 ± 3.73
Flag-like trigger	32	90.93 ± 3.81	78.46 ± 1.04	**91.03** ± 12.96	87.08 ± 2.71
	64	96.87 ± 1.21	82.95 ± 0.72	77.10 ± 16.96	82.92 ± 3.89
	128	95.54 ± 1.88	82.28 ± 1.50	69.49 ± 20.66	82.09 ± 3.78
Blackbox flag-like trigger	32	86.25 ± 4.00	80.16 ± 1.91	56.31 ± 19.57	78.78 ± 3.51
	64	95.00 ± 2.18	83.41 ± 0.94	48.75 ± 23.28	78.11 ± 4.40
	128	93.82 ± 2.27	81.54 ± 1.94	**68.07** ± 18.55	81.23 ± 3.80

Some **triggers work as well as** if the attacker **trained with adversarial data**

Performance appears to differ based on `naturalness` of the trigger



(b) 9 white lines trigger

(a) Flag-like trigger

Conclusions [3 / 3] ≅ Conclusions [2 / 3]

- Pipeline **complexity matters**
- Machine learning is as secure as its **weakest component**
- **Underlying platform** is exploitable
- **Average case is** very **different** from worst case scenario

Thank you very much for listening!
Massive kudos to my amazing supervisors and collaborators!

Please do not hesitate to reach out in case there are any questions at

ilia.shumailov@cl.cam.ac.uk

https://arxiv.org/abs/2006.03463
https://arxiv.org/abs/2104.09667