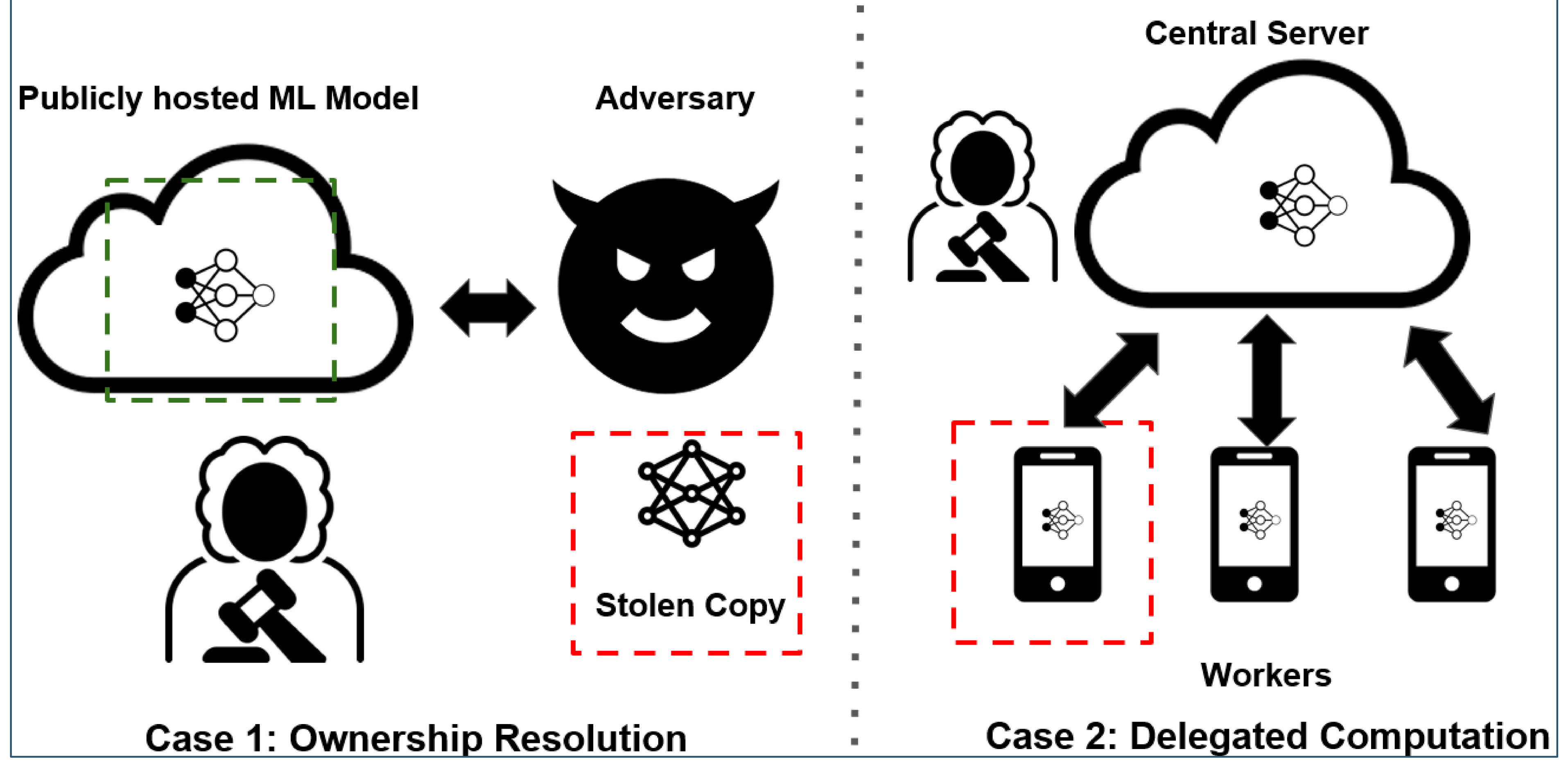
Proof-of-Learning: Definitions & Practice

Hengrui Jia*, Mohammad Yaghini*, Christopher A. Choquette-Choo+, Natalie Dullerud+, Anvith Thudi+, Varun Chandrasekaran, Nicolas Papernot

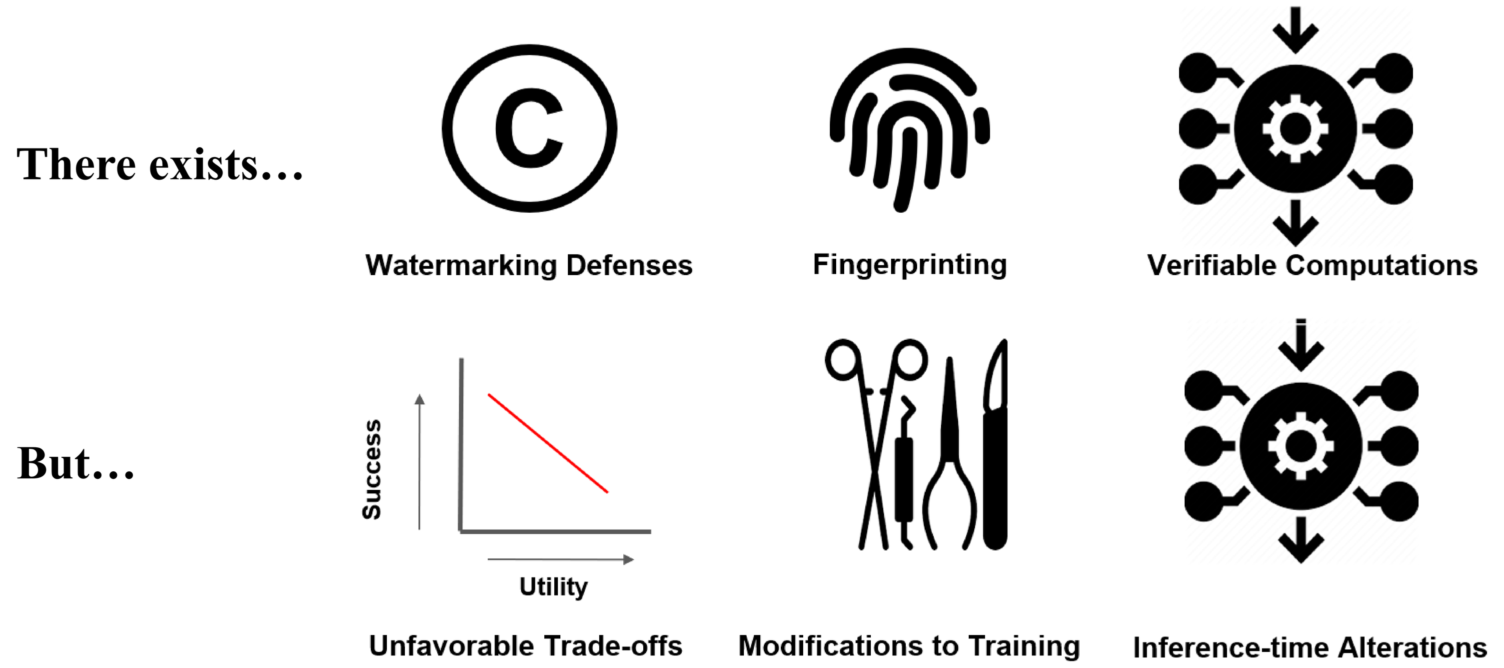
VECTOR INSTITUTE · WISCONSIN UNIVERSITY OF WISCONSIN-MADISON · UNIVERSITY OF TORONTO

*Joint First Authors {nickhengrui.jia, mohammad.yaghini}@mail.utoronto.ca, +Joint Second Authors

Motivation



Prior Approaches



Objectives

Permanent Record of Effort

Arbitration by (Trusted) Third Party

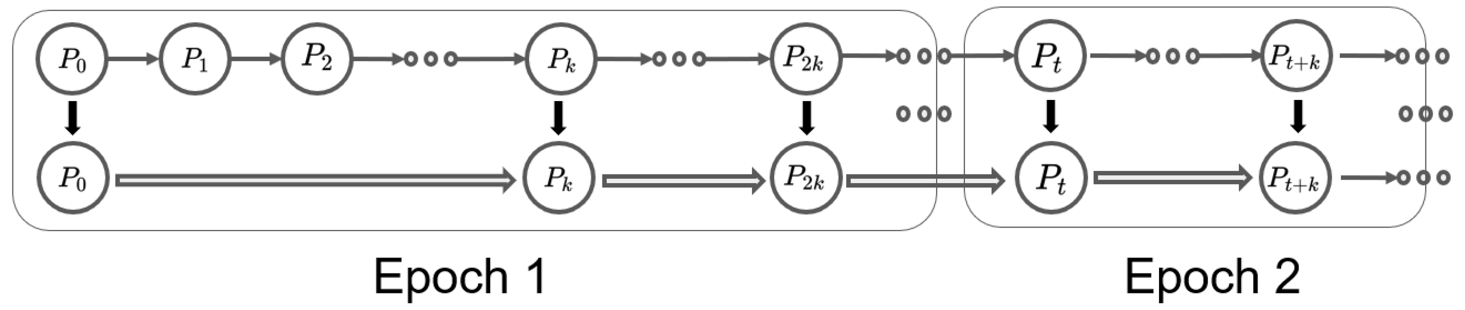
Just a Log; No Utility Degradation

Proof Creation

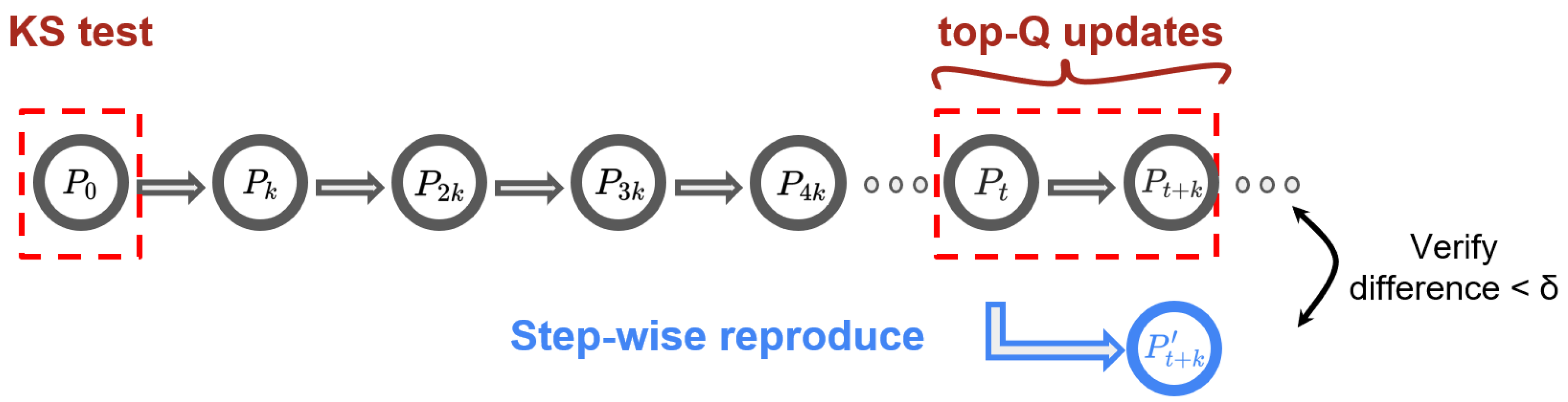
Definition: For a prover T and model W, a valid Proof-of-Learning (PoL) is a tuple of training-step-ordered sets (*W*, *I*, *H*, *A*):

a. **W:** Model weights at steps of training
b. **I**: Data instance information (e.g., batch indices)
c. **H**: Signatures of data instances in **I** (e.g., hashes of batches)
d. **A**: Auxiliary information (e.g., training hyperparameters)

Store $\tilde{P}_i = (\emptyset, I, H, A)$ every step, store $P_i = (W, I, H, A)$ every **k** step



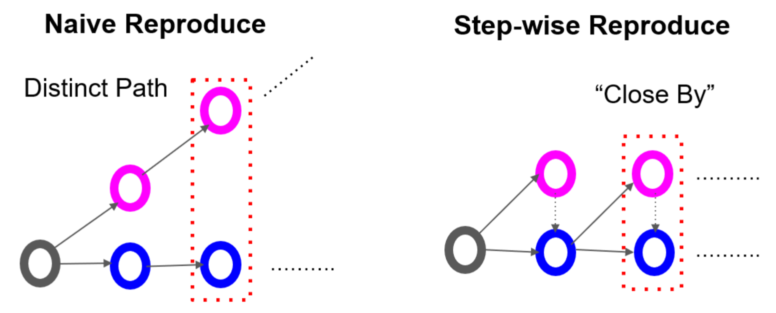
Proof Verification & Correctness



1. **Kolmogorov–Smirnov (KS) test**: prevent arbitrary initialization
   - p-value goes under 0.01 with less than 10 gradient updates
2. **Step-wise reproduce**: bound entropy in training
   - entropy increases linearly to number of training steps
   - reproducing error for a single step is near-zero

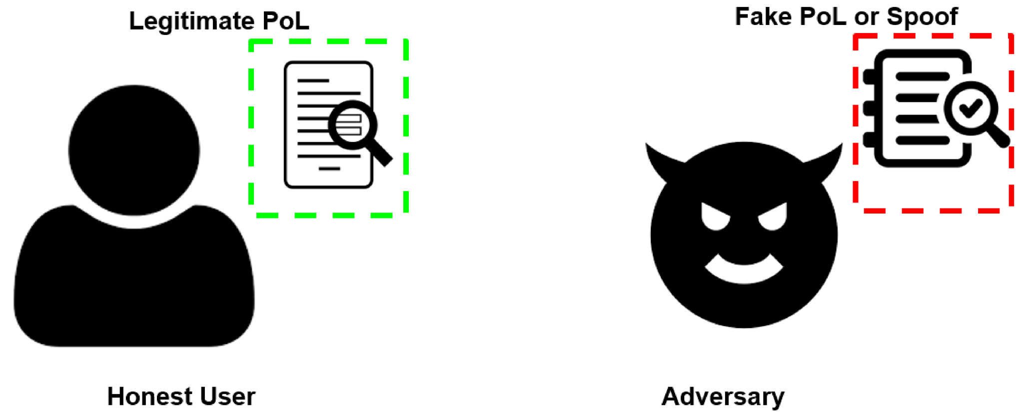
$$\tilde{W}_{t+1} = \tilde{W}_t - \eta \nabla_{\tilde{W}_t} \hat{\mathcal{L}}_t + z_t$$

$$H(\tilde{W}_1 | \tilde{W}_0) = H(z_0)$$



3. **Top-Q verification**: save computational cost
   - Valid updates tend to have small magnitude to avoid overshooting during gradient descent

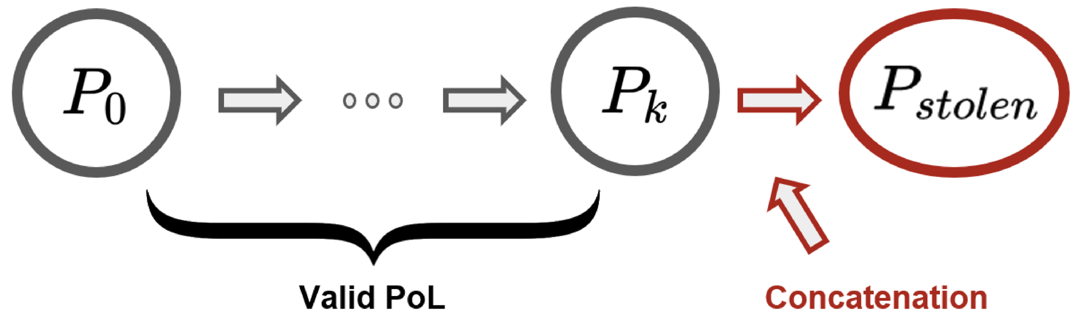
Security Analysis



- **Honest Spoof**: tries to create a **fully valid** PoL
- **Dishonest Spoof**: tries to create a **partially invalid** PoL that passes the verification

Dishonest Spoof Example: **Directed Retraining**

- There will exist a large gap and thus detected by top-Q verification (since the stolen model does not have any connection to the model parameters in the valid proof)



Honest Spoof Example: **Inverse Gradient Methods**

- Inverting gradient descents is more expensive than training, and has higher entropy (meaning it leads to larger verification error)

Recall:

$$\tilde{W}_{t+1} = \tilde{W}_t - \eta \nabla_{\tilde{W}_t} \hat{\mathcal{L}}_t + z_t \Rightarrow H(\tilde{W}_1 | \tilde{W}_0) = \boxed{H(z_0)}$$

Now:

$$\beta(W_{t-1}) := W_{t-1} - W_t - \eta \nabla_{W_{t-1}} \mathcal{L} = 0$$

$$\Downarrow$$

$$H(\tilde{W}_{t-1} | \tilde{W}_t) = \boxed{H(z_0)} + \boxed{H(\eta \nabla_{\tilde{W}_{t-1}} L | \tilde{W}_t)}$$

Computational & Storage Overhead

k: Number of steps between recording weights

Q: Number of checks by verifier

Verification Computational Cost: **O(cost(verify P_i)· Q · k)**

Cost for one step · Number of steps

Verification Storage Cost: **O(size(P) · #epochs · #minibatches / k)**

Storage for one state · Number of states