# A Side-Channel Methodology for DL Model Recovery Using Power Traces

**Mohammed Sazzad Yousuf Sourab, Farah Ferdaus**
msourab@lamar.edu, fferdaus@lamar.edu
Department of ECE, Lamar University, USA

SC26 Chicago, IL | hpc unites.

## Motivation & Threat Model

- Model's architecture is valuable intellectual property
  - a stolen blueprint can copy the model or attack it.
- Only power is monitored during one forward pass -
  - no weights, inputs, or model code is required.
- Reverse Engineering from power traces can recover layer-wise model architecture.
- By observing power tracing, this work recovers –
  - Executed sequence of layer types
  - Model's architectural skeleton

## Research Question & Objectives

**Can a deep learning architecture be recovered by observing the power consumption alone?**

1. Establish power telemetry for deep learning architectures as a viable side-channel for architecture recovery.
2. Demonstrate that modelling layer order improves recovery over only the shape-matching.
3. Determine which layer types are most and least identifiable.

## Experimental Setup & Data

- GPU: NVIDIA H100 PCIe
- Sampling Interval: 20ms
- GPU Idle Power: ~53W
- Architectures: 21 (11 CNN + 10 Transformer /LLM)
- Layer Types: 8
- Trace Samples: ~198,000
- Vision CNNs run on MNIST images (224×224, 3-channel)
- Transformer/LLM run on SST-2 sentiment text (GLUE)
- For every model, this framework records –
  - End-to-end Training & Inference Power
  - Per-layer Power, Energy, and Time
  - 31-Feature Fingerprint (mean, max, min–power, energy)
  - Full sequential layer-wise power trace.

## Fingerprinting & Segmentation

- Model fingerprinting enough to detect model, not its layers.
- Smooth the raw power trace with a Savitzky-Golay filter.
- Mark where the smoothed difference crosses a threshold.
- Sequential power trace recovers layers using KNN and Hidden Markov Model (HMM) with Viterbi.

## Methodology — The Attack Pipeline

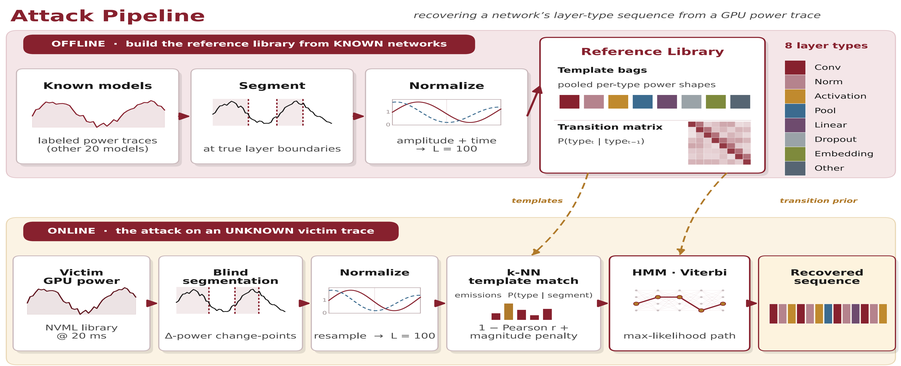



***Fig. 1:*** *Using Template bag & Transition Matrix of the known models, attack in done using KNN or HMM +Viterbi to get the sequential layer of the attacked architecture.*

## Overview - What the Attacker Monitors

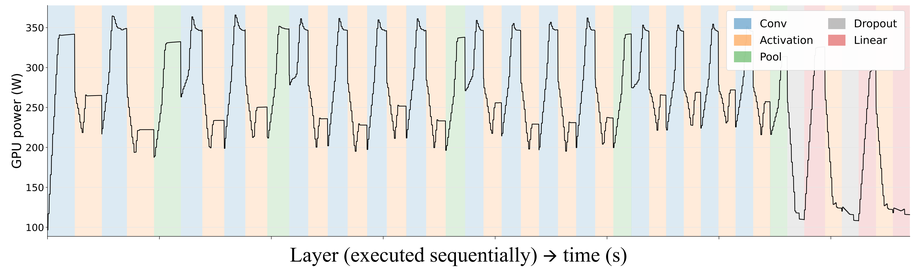
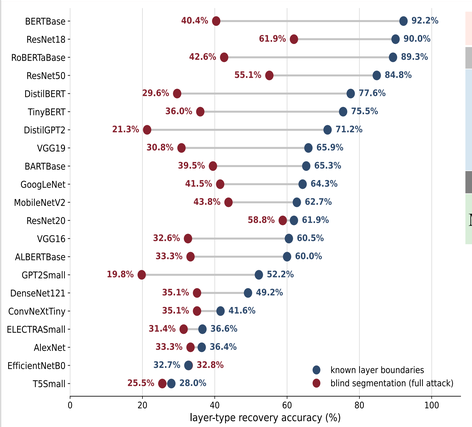



***Fig. 2:*** *A single VGG19 inference seen only through GPU power ; the strip beneath = true layer types.*

## Recovery per Network




| Category | Model | Recovery/Resistance |
|---|---|---|
| Strongest Recovery | BERT-Base | 92.2% |
| | ResNet18 | 90% |
| | RoBERTa-Base | 89.3% |
| | ResNet-50 | 85% |
| Most Resistant | EfficientNetB0 | 32.8% |
| | T5-Small | 28% |

*Table represents the models with strongest recovery and most resistant ones.*

*Fig.3 represents from k-NN to HMM per held-out network for both groundtruth and blind segmentation*

## Layer-Wise Recovery - kNN vs HMM

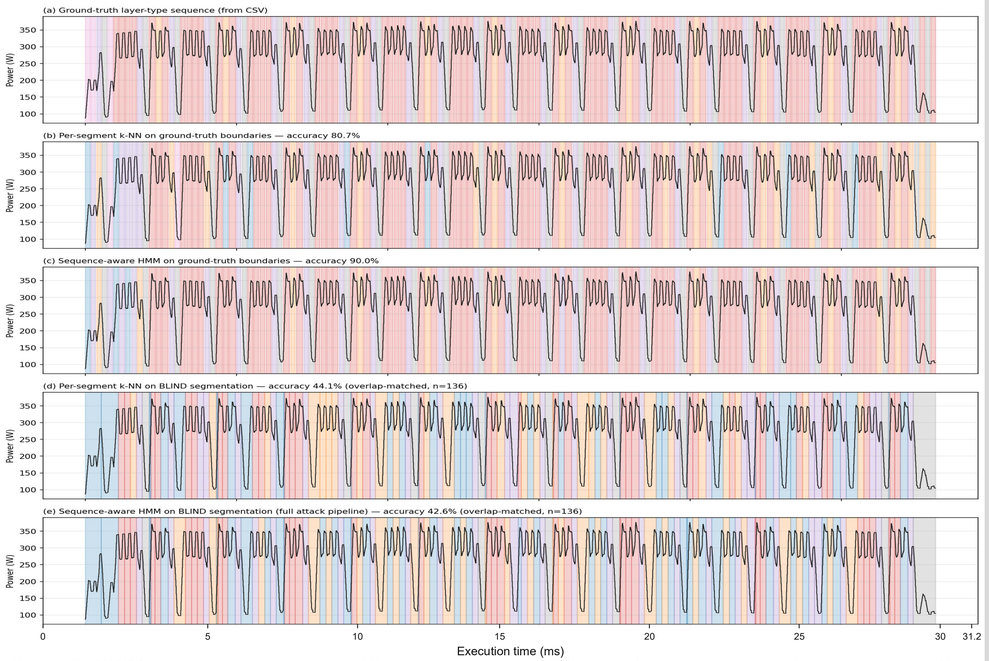



*Fig. 4: RoBERTBase — (a) ground truth, (b) k-NN on true boundaries (80.7%), (c) HMM on true boundaries (90.0%), (d) k-NN on Blind Segmentation (44.3%), (e) HMM on blind Segmentation (42.6%); Color = layer type, equal profiling dwell for all layers*

## Template Bag of Layers

- Template bag of layers normalizes shapes of layer power pattern.
- Black average line of each layer defines normalized pattern.

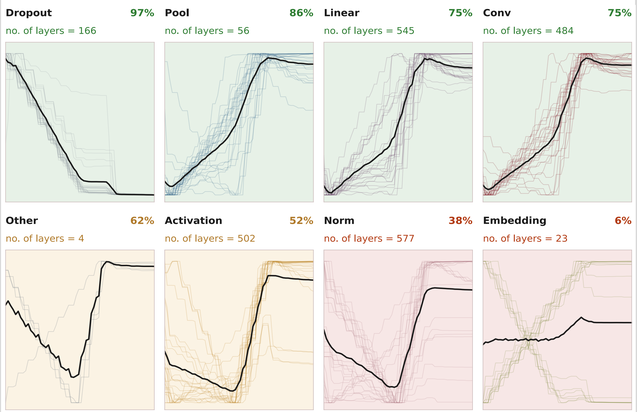



*Fig. 5: Accuracy of Layer detections based on normalized layer template bags*

## Key Observations

**60.1%** mean layer-type recovery across 20 held-out architectures

**92.2%** best case (BERT-Base)

**17/21** architectures improved by the HMM

## Future Works

- Replace threshold change-points with learned segmentation
- Build denser reference libraries
- Test across accelerators and vendors (co-resident workloads)
- Use more higher-order sequence models for large template bag