

Supplementary Document

Can one hear the shape of a neural network?: Snooping the GPU via Magnetic Side Channel

A BiLSTM NETWORK STRUCTURE

Classifying steps in a network model requires taking in a time-series signal and converting it to labeled operations. The EM signal responds only to the GPU’s instantaneous performance, but because the GPU executes a neural network sequence, there is rich context in both the window before and after any one segment of the signal. Some steps are often followed by others, such as pooling operations after a series of convolutions. We take advantage of this bidirectional context in our sequence to sequence classification problem by utilizing a BiLSTM network to classify the observed signal. To retrieve a network’s topology, we pass normalized EM values into a two-layer BLSTM network, with dropout of 0.2 in between. From there we compute a categorical cross-entropy loss on a time-distributed output that we achieve by sliding an input window across our EM signal. This approach proves robust, and is the method used by all of our experiments, and on all GPU’s tested.

The segmented output of our BiLSTM network on our extracted signal is for the most part unambiguous. Operations that follow one another (i.e. convolution, non-linear activation function, pooling) are distinct in their signatures and easily captured from the context enabled by the sliding window signal we use as input to the BiLSTM classifier. Issues arise for very small-sized steps, closer to our sensor’s sampling limit. In such regions a non-linear activation may be over-segmented and split into two (possibly different) activation steps. To ensure consistency we postprocess the segmented results to merge identical steps that are output in sequence, cull out temporal inconsistencies such as pooling before a non-linear activation, and remove activation functions that are larger than the convolutions that precede them.

B ADDITIONAL EXPERIMENTS

B.1 USING LEVENSHTAIN DISTANCE TO MEASURE NETWORK RECONSTRUCTION QUALITY

To provide a sense of how the Levenshtein edit distance is related to the network’s ultimate performance, we consider AlexNet (referred as model A) and its five variants (referred as model B, C, D, and E, respectively). The variants are constructed by randomly altering some of the network steps in model A. The Levenshtein distances between model A and its variants are 1, 2, 2, 5, respectively (see Fig. S1), and the normalized Levenshtein distances are shown in the brackets of Fig. S1. We then measure the performance (i.e., standard test accuracy) of these models on CIFAR-10. As the edit distance increases, the model’s performance drops.

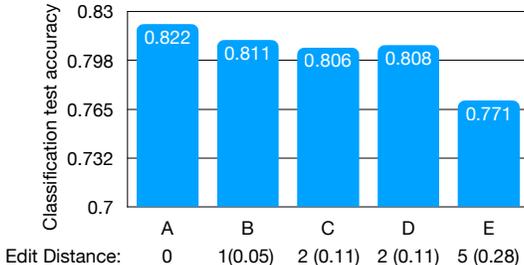


Figure S1: The model’s classification accuracy drops as its Levenshtein distance from the original model (model A: AlexNet) increases.

B.2 RECONSTRUCTION QUALITY ON IMAGENET

We treat ResNet18 and ResNet50 for ImageNet classification as our black-box models, and reconstruct them from their magnetic signals. We then train those reconstructed networks and compare their test accuracies with the original networks’ performance. Both the reconstructed and original

Table S1: Model reconstruction evaluated on ImageNet classification.

Model	ResNet18		ResNet50	
	Original	Extracted	Original	Extracted
Top-1 Acc.	64.130	64.608	62.550	61.842
Top-5 Acc.	86.136	86.195	85.482	84.738
KL Div.	-	2.39	-	4.85

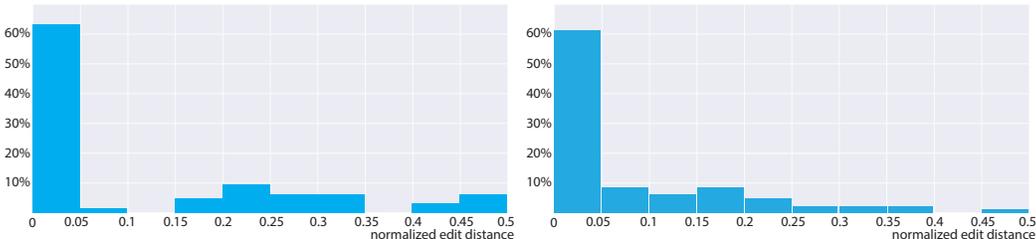


Figure S2: **Distribution of normalized Levenshtein distance.** (left) We plot the distribution of the normalized Levenshtein distances between the reconstructed and target networks. This results, corresponding to Table I in the main text, use signals collected on Nvidia TITAN V. (right) We also conduct similar experiments on two Nvidia GTX-1080 GPUs. One is used for collecting training signals, and the other is used for testing our side-channel-based reconstruction algorithm.

networks are trained with the same training dataset for the same number of epochs. The results are shown in Table S1, where we report both top-1 and top-5 classification accuracies. In addition, we also report a KL-divergence measuring the difference between the 1000-class image label distribution (over the entire ImageNet test dataset) predicted by the original network and that predicted by the reconstructed network. Not only are those KL-divergence values small, we also observe that for the reconstructed network that has a smaller KL-divergence from the original network (i.e., ResNet18), its performance approaches more closely to the original network.

B.3 GPU TRANSFERABILITY

Here we verify that (i) the leaked magnetic signals are largely related to GPU brand/version but not the other factors such as CPUs and (ii) the signal characteristics from two physical copies of the same GPU type stay consistent.

We obtain two copies of an Nvidia GTX-1080 GPU running on two different machines. When we run the same network structure on both GPUs, the resulting magnetic signals are similar to each other, as shown in Fig. S3. This suggests that the GPU cards are indeed the primary sources contributing the captured magnetic signals.

Next, we use one GPU to generate training data and another one to collect signals and test our black-box reconstruction. The topology reconstruction results are shown in Table S2 arranged in the way similar to Table I, and the distribution of normalized Levenshtein edit distance over the tested networks are shown in Fig. S2-right. These accuracies are very close to the case wherein a single GPU is used. The later part of the reconstruction pipeline (i.e., the hyperparameter recovery) directly depends on the topology reconstruction. Therefore, it is expected that the final reconstruction is also very similar to the single-GPU results.

B.4 TRANSFER ATTACKS ON MNIST

We also conduct transfer attack experiments on MNIST dataset. We download four networks online, which are not commonly used. Two of them are convolutional networks (referred as CNN1 and CNN2), and the other two are fully connected networks (referred as DNN1 and DNN2). None of these networks appeared in the training dataset. We treat these networks as black-box models, and reconstruct a network for each of them. We then use the four reconstructed models to transfer attack the four original models, and the results are shown in Table S4. As baselines, we also use the four original models to transfer attack each other including themselves.

Table S2: Classification accuracy of network steps (GTX-1080).

	Prec.	Rec.	F1	# samples
LSTM	.997	.999	.998	12186
Conv	.985	.989	.987	141164
Fully-connected	.818	.969	.887	9301
Add	.962	.941	.951	30214
BatchNorm	.956	.944	.950	48433
MaxPool	.809	.701	.751	1190
AvgPool	.927	.874	.900	294
ReLU	.868	.859	.863	11425
ELU	.861	.945	.901	8311
LeakyReLU	.962	.801	.874	3338
Sigmoid	.462	.801	.585	5106
Tanh	.928	.384	.543	8050
Weighted Avg.	.945	.945	.945	-

Table S3: **DNN estimation accuracies.** Using the 1804 convolutional layers in our test dataset, we measure the accuracies of our DNN models for estimating the convolutional layers’ hyperparameters. Here, we break the accuracies down into the accuracies for individual hyperparameters.

	Kernel	Stride	Padding	Image-in	Image-out
Precision	0.971	0.976	0.965	0.968	0.965
Recall	0.969	0.975	0.964	0.969	0.968
F1 Score	0.969	0.975	0.962	0.967	0.965

In Table S4 every row shows the transfer attack success rates when we use different source (surrogate) models to attack a specific original model (CNN1, CNN2, DNN1, or DNN2). Each column labeled as “extr.” corresponds to the extracted (reconstructed) model whose target model is given in the previous column right before it. In addition, we also show all the models’ test accuracies on MNIST in the last row of the table. The results show that all the reconstructed models approximate their targets closely, both in terms of their abilities for launching transfer attacks and their classification performance.

C SENSOR SETUP

The magnetic induction signal we utilize comes from digitally converting analog readings of a Texas Instruments DRV425 fluxgate sensor with Measurement Computing’s USB-204 Digital Acquisition Card. The sensor samples at a frequency of 47Khz and the converter operates at 50Khz to map the originally $-2\text{mT} \sim 2\text{mT}$ readings across 0 to 5 Volts using a 12-bit conversion. Calibrating the sensor requires (a) that the sensor is within range of the electromagnetic signal and that (b) the sensor orientation is consistent. The magnetic induction signal falls off at a rate inversely proportional to distance squared, and so the sensor must be placed within 7mm of the GPU power cable for reliable

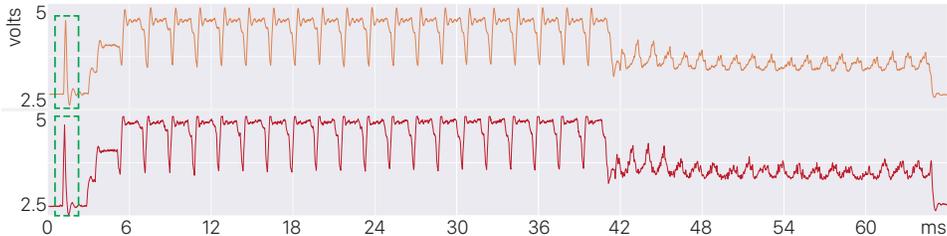


Figure S3: Here we plot the resulting signals from the same network model deployed on two different instances of a NVIDIA-GTX 1080 (running on two different computers). In the green boxes on the left are the spikes that we inject on purpose (discussed in Sec. 4.2) to synchronize the measured signal with the runtime trace of the GPU operations.

Table S4: MNIST results.

		Source Model							
		CNN1	extr.	CNN2	extr.	DNN1	extr.	DNN2	extr.
Target	CNN1	.858	.802	.226	.202	.785	.795	.476	.527
	CNN2	.395	.319	.884	.878	.354	.351	.354	.211
	DNN1	.768	.812	.239	.223	.999	.999	.803	.885
	DNN2	.703	.768	.219	.194	.975	.979	.860	.874
Accuracy		.989	.987	.993	.991	.981	.981	.980	.983

measurement. Flipping the flat sensor over will result in a sign change of the magnetic induction signal, thus a uniform orientation should be maintained to avoid preprocessing readings across the dataset to align.