iNNformant: Boundary Samples as Telltale Watermarks

Alexander Schlögl
alexander.schloegl@uibk.ac.at
Department of Computer Science
University of Innsbruck
Austria

Tobias Kupek*
tobias.kupek@swarm-analytics.com
Swarm Analytics GmbH
Austria

Rainer Böhme
rainer.boehme@uibk.ac.at
Department of Computer Science
University of Innsbruck
Austria

ABSTRACT

Boundary samples are special inputs to artificial neural networks crafted to identify the execution environment used for inference by the resulting output label. The paper presents and evaluates algorithms to generate transparent boundary samples. Transparency refers to a small perceptual distortion of the host signal (i. e., a natural input sample). For two established image classifiers, ResNet on FMNIST and CIFAR10, we show that it is possible to generate sets of boundary samples which can identify any of four tested microarchitectures. These sets can be built to not contain any sample with a worse peak signal-to-noise ratio than 70 dB. We analyze the relationship between search complexity and resulting transparency.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Neural networks**; • **Security and privacy** → *Digital rights management*; • **Applied computing** → **System forensics**.

KEYWORDS

watermarking, neural networks, forensics, adversarial machine learning

ACM Reference Format:
Alexander Schlögl, Tobias Kupek, and Rainer Böhme. 2021. iNNformant: Boundary Samples as Telltale Watermarks. In *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '21), June 22–25, 2021, Virtual Event, Belgium.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3437880.3460411

1 INTRODUCTION

Recently it has been observed that the numerical predictions of neural networks (NNs) vary between different CPU microarchitectures (MAs) [19]. This can be used in forensic investigations to identify the execution environment used for predictions, or verify that a prediction has been made on specific hardware.

These numerical differences may also offer novel ways to implement digital rights management for trained machine learning models. For instance, the owner of a model could verify if a given prediction has been generated on licensed hardware, which might be equipped with a secure billing device. Predictions whose numerical values indicate the use of *another* MA indicate that the billing mechanism might have been bypassed fraudulently.

A major obstacle to this application is that the numerical differences between MAs are tiny. They almost always disappear at the last step of the inference pipeline when a real-valued soft-max vector is quantized to a symbolical label. Boundary samples fix this problem. These samples lie in the area between decision boundaries that arise from the numerical differences between MAs, as was observed in [19]. Boundary samples are then classified differently depending on the execution environment, allowing the owner of a model in the above example to probe the hardware used for prediction. In the best case, any deviation from the licensed hardware's MAs is detectable by the class label only.

Boundary samples are barely researched. It may appear surprising even that they exist and can be found efficiently as claimed in [19]. This work adds another consideration, namely the transparency of boundary samples. This is relevant if, in the above example, the model owner wants to probe the inference pipeline inconspicuously in order to avoid that the licensee can process obvious boundary samples in a different pipeline (the legitimate one) than the bulk of organic samples. We propose to generate transparent boundary samples as perturbations of natural input samples and measure the distortion by the peak signal-to-noise ratio (PSNR).

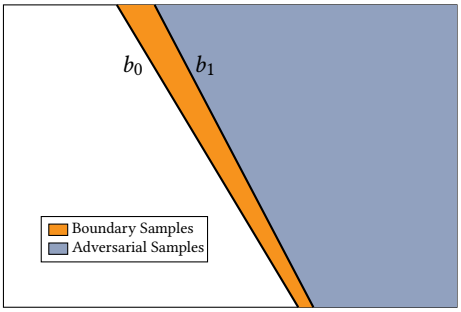


Figure 1: Input space for adversarial and boundary samples, for two given decision boundaries b_0 and b_1.

There are striking parallels to digital watermarking [4]: the natural input sample is the host signal and the perturbation is a *telltale watermark*, i. e., a special case of fragile watermark designed to indicate the type of processing applied to the watermarked signal [3, 10]. Note that we do not aim for undetectability in the sense of secure steganography. The facts that boundary samples are rare, and good ones are classified differently by any two MAs, seem to

*Work carried out while at the University of Innsbruck.

IH&MMSec '21, June 22–25, 2021, Virtual Event, Belgium

ACM ISBN 978-1-4503-8295-3/21/06…$15.00
https://doi.org/10.1145/3437880.3460411

preclude any attempt to make them undetectable for anyone who can run the model on multiple MAs.

There are also parallels to evasion attacks in adversarial machine learning [16]. Figure 1 gives some intuition on the difficulty of finding boundary samples compared to the problem of finding adversarial samples. Both problems have in common that an input in the white region has to be perturbed to fall into another region subject to small perceptual distortion. While the solution space for adversarial samples is the entire blue area, boundary samples must hit the orange region. In fact, adversarial samples try to move as "deep" as possible (given the perturbation constraints) into the blue area in order to be transferable [21]. In contrast, boundary samples must hit the small orange area between the decision boundaries that arises purely from numerical differences between MAs.

This short paper is organized as follows. The next section presents our generation method for boundary samples, which is a modification of FGSM, a known search algorithm for generating adversarial samples [6]. Section 3 evaluates the effectiveness and efficiency of the proposed algorithm on two standard pairs of dataset and model architectures (FMNIST with ResNet20 and CIFAR10 with ResNet32). We report runtime measurements (in terms of iterations broken down by MAs), the resulting success rates, and distortions. Section 4 discusses related work. The concluding Section 5 points out limitations and shows directions for future work.

2 GENERATING BOUNDARY SAMPLES

Our method makes the following assumptions. We have white-box access to a trained feed-forward deep neural network and oracle access to predictions and gradients from that network on a closed set of relevant MAs. Access costs vary between oracles. We assume that one MA is *local* and gives us a cheap (fast) oracle. All other MAs are *remote* and may in practice be more costly (slower). Moreover, we require access to a number of test samples drawn from the training distribution. Transparency is defined by the distortion between a given test sample (the starting point for the iterative algorithms) and the resulting boundary sample. We consider a set of boundary samples as *fully identifying* if it contains at least one sample that is predicted with a unique label for each MA in the set.

We proceed in two steps. In the next subsection, we examine the case of a binary decision problem between two candidate MAs. Then, in Section 2.2, we generalize to the case of identifying one out of n candidate MAs. This is still a binary decision problem, but the solution space is much more constrained.

2.1 The 1 vs 1 Case

The problem of differentiating between two candidate MAs is equivalent to constructing a sample that falls in the orange regions in Figure 1. We split the generation into a *local* and a *remote* phase.

Local phase. In the local phase we try to get as close to the decision boundary as possible. We do this with the modified iterative fast-gradient-sign method (i-FGSM) [6]. FGSM was chosen based on the intuition that many small perturbations lead to a lower mean square error (and hence higher PSNR) than fewer larger perturbations. For model m, input $\mathbf{x}$, and a step size α, the i-th FGSM step works as follows,

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha \operatorname{sign}\left(\nabla_{\mathbf{x}} m(\mathbf{x}_{i-1})\right). \tag{1}$$

Compared to FGSM our algorithm flips the sign based on the correctness of the prediction, and reduces the step size as we approach the decision boundary. Varying the perturbation's sign lets us approach the decision boundary even after overshooting. As we approach the decision boundary, the confidence difference δ_{conf} between the first and second predicted classes decreases. While the gradients' norms could be used to judge the distance to the decision boundary, we used δ_{conf} as an approximate distance measure. Reducing the step size along with δ_{conf} allows us to gradually approach the decision boundary, until a termination condition is met. The modified FGSM step looks as follows,

$$\mathbf{x}_i = \mathbf{x}_{i-1} + c\ \delta_{\text{conf}}\ \alpha \operatorname{sign}\left(\nabla_{\mathbf{x}} m\left(\mathbf{x}_{i-1}\right)\right), \tag{2}$$

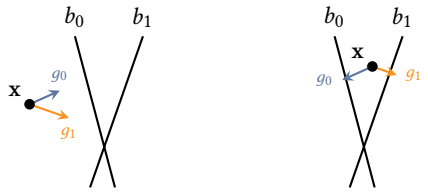
where c is the correctness sign. It takes value 1 if $\mathbf{x}_{i-1}$ is misclassified, and -1 otherwise.

As local predictions are cheaper than remote predictions, we want to approach the decision boundary as close as possible with local steps; specifically to a confidence difference of less than 10^{-6}, or ideally 10^{-7}. Choosing the right α is crucial. If it is too high, the sample bounces around the decision boundary without approaching it. If it is too low, the sample movement stalls as the confidence difference vanishes. This can be detected if the predictions are identical in two consecutive steps. In this case, we multiply δ_{conf} with a scaling factor. We increase δ_{conf} exponentially in this fashion until the predictions change again, at which point δ_{conf} is reset to the new confidence difference. We use 10^{-4} as value for α.

Remote phase. Once we are sufficiently close with the local oracle, we use gradients from our remote oracles to further refine the boundary sample. In this remote phase, one of three cases occurs:

(1) The label flips on neither instance.
(2) The label flips on one instance, but not the other.
(3) The label flips on both instances.

In the second case, we have found a boundary sample and terminate. Note that the third case is symmetric to the first, and our handling is identical. The possible cases are shown in Figure 2, where g_i denotes the gradient for instance i.



(a) Same class predicted **(b) Different class predicted**

Figure 2: Possible cases for boundary sample predictions.

When both instances predict the same class, we further approach the closest decision boundary using the same modified FGSM step as in the local phase. Our remote step requests predictions and gradients from both oracles, and then uses the one where the corresponding δ_{conf} is smaller for the FGSM step shown in Equation (2).

If the labels from both instances flip together, we simply continue as c will correct the direction of our perturbation even if the closest boundary is not the same as before. This process is repeated until either a successful boundary sample is generated or a maximum number of iterations is reached. The full algorithm is shown in Algorithm 1, where the lines relevant to the 1 vs 1 case are highlighted.

Remarks. The local and remote phases are very similar. In principle, one could also omit the local preparation entirely and start with a clean sample in the remote phase. As we used cloud instances for our remote oracles, remote predictions were slower and much more costly than the local predictions.

2.2 The 1 vs n Case

We can use the same approach as in Figure 2 to identify a single MA from a set of n known MAs. The only relevant modification concerns the selection of the target gradient. The choice of target boundary is given by the requirement of a 1 vs n boundary sample: one MA results in a different label compared to all others. This requires our boundary sample to lie past the decision boundary for one MA, but before the decision boundary for all others, or vice versa. Figure 3 highlights areas around the decision boundaries that uniquely identify MAs.

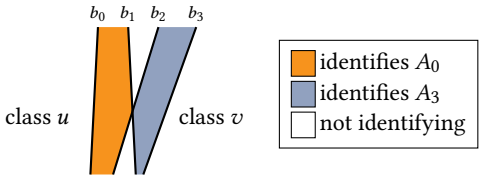


Figure 3: Uniquely identifying areas for multiple instances.

The 1 vs n boundary sample generation process discussed here is not targeted. This means we cannot choose the MA to be identified, but let the algorithm find any MA that can be identified from all others in the set. This is not a big limitation as we shall see in Section 3 that different starting samples let us identify different MAs, and each MA is singled out sufficiently often. Hence, we can repeatedly run the algorithm until we find a boundary sample which identifies any desired MA.

The algorithm proceeds as follows. As before, we first approach the nearest local decision boundary as close as possible. Starting with the local oracle ensures that the farthest distance is traversed with cheap (fast) queries. We then request predictions from the remote oracles, which tell us where our current sample lies with regard to all MAs' decision boundaries. This step is shown in Figure 4a, where the decision boundaries are indexed with c and r for "left" and "right" for convenience. We partition all predictions according to their label and select the smallest partition (Figure 4b). From the smallest partition, we choose the *second farthest* decision boundary, i. e., the one with the second highest δ_{conf}, as our target (Figure 4c). Passing the second furthest decision boundary leaves only one label flipped from all others (Figure 4d), meaning the generated boundary sample uniquely identifies an MA (the rightmost MA in the example). Figure 4 illustrates our algorithm, and Algorithm 1 gives the pseudocode.

Remarks. We optimize our target selection for cases when all MAs return the same label, in which case we approach the closest decision boundary. This happens in line 12. Moreover, in our experiments we had two oracles each for several MAs. We thus had to modify the exit condition to not only exit if a single label is different from all others. We also checked whether the labels from an entire MA are different from all others, but identical to each other. This special case is included in line 10.

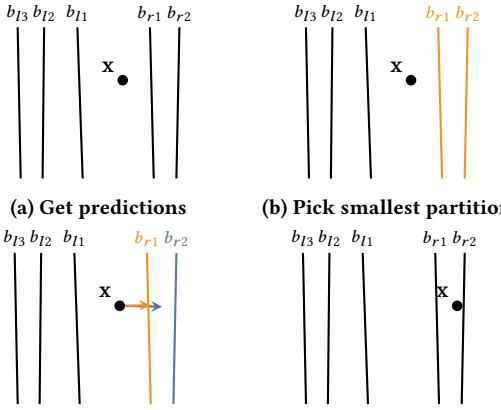


Figure 4: Steps for generating 1 vs n boundary samples.

Algorithm 1 Untargeted boundary sample generation.

```
1:  procedure GenerateBoundarySample(x, servers)
2:      for i ← 1, ..., local_max do                          ▷ local phase
3:          predict x locally
4:          if δ_conf < target then break
5:          x ← x + l δ_conf α sign(∇_x m(x))
6:      for i ← 1, ..., remote_max do                         ▷ remote phase
7:          results ← predict x on each server
8:          groups ← results grouped by the predicted label l
9:          g ← smallest group in groups
10:         if g contains a single MA then return x
11:         if size(g) = n then
12:             r ← result ∈ g with smallest δ_conf
13:         else
14:             r ← result ∈ g with second highest δ_conf
15:         x ← x + r.l r.δ_conf α sign(∇_x m(x))
```

3 EXPERIMENTAL EVALUATION

Setup. In our experiments we set $n = 4$. Table 1 lists the MAs used in all our experiments. These are all MAs available as Google Cloud instances at the time of writing, and the grouping is based on successful identifications reported in prior work [19]. For our

Table 1: Overview of the architectures used in this work.

Label	CPU Architecture
MA1	AMD Rome
MA2	Intel Sandy / Ivy Bridge
MA3	Intel Haswell / Broadwell
MA4	Intel Skylake / Cascade Lake

evaluation we will focus on the 1 vs. n case as it is the more challenging one. We chose image classification as task and selected two common datasets of different complexity: FMNIST [24] and CIFAR10 [9]. Both datasets contain images of 10 classes. FMNIST has an input dimension of 28×28×1. CIFAR10 has an input dimension of 32×32×3. We employ the established ResNet architecture [8] of two depths, namely 20 layers for FMNIST and 32 layers for CIFAR10, trained for 150 epochs each. The Keras interface of TensorFlow version 2.3.0 was used on all MAs to run predictions. We set up a Docker container based on `tensorflow/tensorflow:2.3.0` to ensure consistency of every component above the operating system. Using Algorithm 1, we attempted to generate boundary samples from 400 randomly selected test images per dataset. The termination conditions were set to 2000 for *local_max* and 500 for *remote_max*. We could confirm that all processes were deterministic on each MA (but certainly not across them).

Success. Our overall success rates were 70.5 % for FMNIST, and 28.25 % for CIFAR10. Table 2 breaks down the successful terminations by the MAs they can identify. This confirms that an untargeted algorithm is sufficient to generate a set of boundary samples uniquely identifying all MAs, if one runs it repeatedly on different input samples until a matching boundary sample is produced. Even in the worst case, when identifying MA4 with FMNIST, a suitable boundary sample is found with more than 99 % probability after 28 successful runs, or 40 runs if one accounts for the failure rate of 29.5 %. (Values for CIFAR10 are 24 and 85, respectively.)

Table 2: Distribution of identified microarchitectures (in percent). MA numbers are scaled to number of successes.

Model	Success				Failure
	MA1	MA2	MA3	MA4	
FMNIST	**70.50**				**29.50**
	29.54	28.47	26.33	15.66	
CIFAR10	**28.25**				**71.75**
	21.24	28.32	32.74	17.7	

Transparency. Table 3 reports the distribution of PSNRs for successful boundary samples broken down by dataset. The PSNR values for most samples from both datasets are above 45 dB and thus in the range of high-quality lossy image compression [23]. This demonstrates that it is possible to generate a fully identifying set of boundary samples whose elements are not obviously distinguishable from natural samples by a human observer. By choosing the most transparent boundary samples from the experiments, we could compose fully identifying sets with a minimum PSNR as high as 70.93 dB for FMNIST and 77.05 dB for CIFAR10.

Boundary samples for CIFAR10 exhibit on average 10 dB higher PSNR (and thus better transparency) than FMNIST. While we cannot causally explain this, it might be related to the larger input dimension which gives more room to distribute the perturbations over more pixels. Another co-factor is the systematic difference in search time, which we discuss next.

Table 3: PSNR distribution of boundary samples (in dB).

Model	Min	Q_1	Median	Mean	Q_3	Max
FMNIST	38.71	46.37	50.10	52.11	55.75	83.49
CIFAR10	48.28	54.76	85.76	60.82	66.65	81.48

Complexity. Figure 5 shows box plots of the distribution of the number of local and remote steps for both datasets. Both datasets require a few hundred local steps,[1] but the number of subsequent remote steps varies substantially. While a few dozen iterations are sufficient for FMNIST, the number of remote steps varies widely for CIFAR10. Overall, 15.3% of the successful boundary samples for CIFAR10 required more remote steps than local steps, whereas this happened only once for FMNIST.

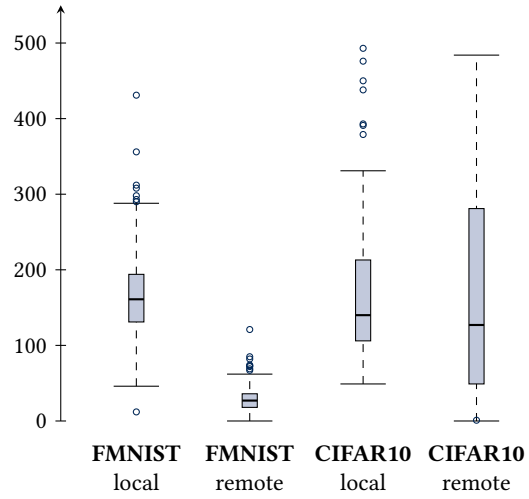


Figure 5: Distribution of the number of local and remote steps across samples.

Favorable class pairs. Drilling down to the level of class labels, we ask if certain labels are prevalent as identifying or contrast labels. Recall that the identifying label is the label assigned to the MA singled out by a boundary sample. Likewise, the contrast label is the one assigned to all other MAs. Figure 6 depicts confusion matrices with identifying labels in rows and contrast labels in columns.

For FMNIST, the most common identifying label is *Shirt*, and the most common contrasting labels are *T-shirt/Top*, *Pullover*, *Dress*,

[1] The plot does not show one outlier with 1200 local steps for FMNIST.

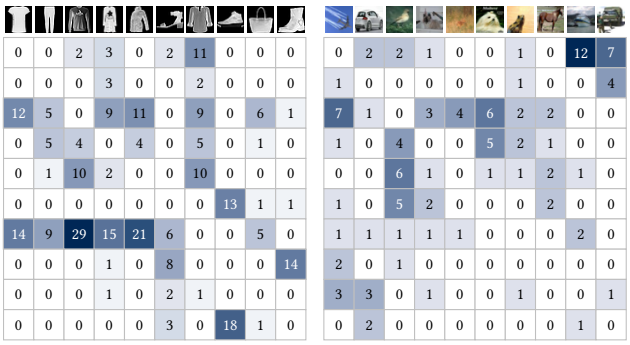
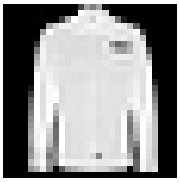


Figure 6: Identifying (rows) and contrast labels (colums). Cell values indicate the frequency of cooccurrence.

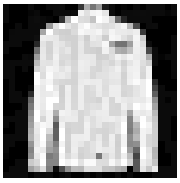
and *Coat*, which are all closely related in shape. For CIFAR, the dominance is less pronounced, with the most common label flips being from *Airplane* to *Boat* and *Truck*, which are again of a similar general shape. The visual similarity of pairs of boundary sample classes can potentially be attributed to the high difficulty of finding boundary samples compared to adversarial samples. Figure 7 shows a comparison between natural and boundary sample, including the predicted labels and prediction confidences.

Natural Sample



Architecture	Label	Confidence
Intel Coffee Lake (Local)	Shirt	95.59

Boundary Sample



Architecture	Label	Confidence
AMD Rome	Shirt	50.000 11
Sandy Bridge	Coat	49.999 96
Ivy Bridge	Coat	49.999 96
Haswell	Shirt	50.000 08
Broadwell	Shirt	50.000 08
Skylake	Shirt	50.000 04
Cascade Lake	Shirt	50.000 04

Figure 7: FMNIST boundary sample visualization. Perturbations are amplified by a factor of 50 to aid visual inspection.

Multivariate analysis. The scatterplots of successful boundary samples in Figure 8 visualize the relationship between the complexity of finding a suitable boundary sample and the resulting transparency (top panels). The complexity is further broken down into local and remote steps in the bottom row. Unsurprisingly, longer search implies lower quality as difficult samples need stronger perturbations to reach a class boundary. This interpretation is supported by the positive association between local and remote steps, indicating that the difficulty inherent to the sample determines the

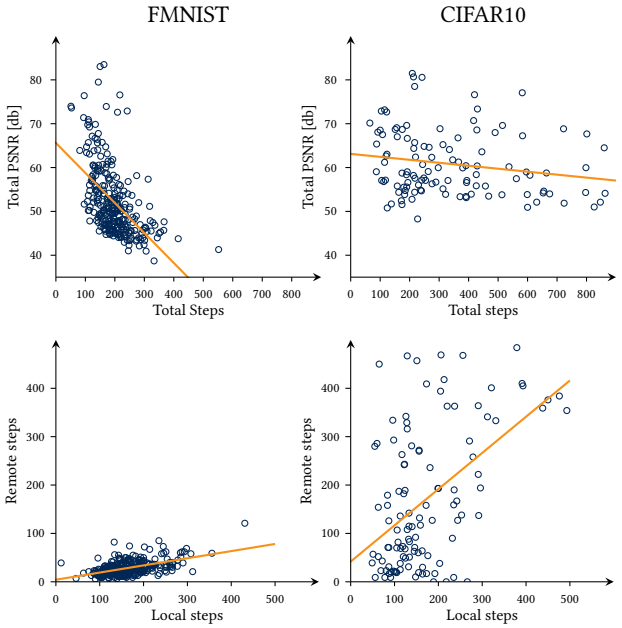


Figure 8: Relation between transparency and complexity.

search effort. However, the relation between iterations and (lower) quality is much more pronounced for FMNIST than for CIFAR10.

4 RELATED WORK

Adversarial machine learning has become a vast field in the past couple of years. As it is tangential to our objective, we refer the reader to the authoritative surveys and taxonomies [2, 16]. In their terminology, boundary samples would represent attacks against supervised machine learning models performed by an iterative evasion during the inference phase. Another relation is that boundary samples can serve as oracles in gray-box scenarios.

The (still smaller) literature on watermarking neural networks can be broadly structured along two purposes. First, to protect trained models against unauthorized redistributions; second, to re-identify models in a black-box or gray-box scenario [16], where only parts of the pipeline are known and accessible. Our work is closer to the latter, but assumes full knowledge of the model and seeks to identify the execution environment in which it is run.

Uchida et al. [22] propose a framework for embedding watermarks into neural networks. Their method promises to generate a unique signature of the model by adjusting weights in the training phase utilizing a parameter regularizer. Recent work takes advantage of backdoors inserted in the training phase to activate a detection mechanism with special inputs [1, 12, 25]. Namba et al. [15] combine multiple techniques to implement watermarks, which are reportedly more robust against model and query modifications. A different approach is to embed a watermark into the distribution of the data abstraction obtained in different layers [18]. In order to use watermarking in a black-box scenario, Guo et al. [7] proposed to train a secret message mark into the model, which causes misclassification of certain marked inputs, akin adversarial

samples. Shumailov et al. [20] show a method to embed keys into deep neural networks. Although they focus on defending against adversarial samples, the method could also be of potential use for watermarking. Merrer et al. [14] use adversarial samples to identify the characteristics of the hyperplane of individual models. Their generation algorithm is similar to ours (and inspired from the same original method [6]), although it has laxer restrictions.

All approaches discussed in the paragraph above use some form of keys, which are embedded in the model during the training phase. The ease of securing keys embedded in neural networks has been challenged in last year's workshop [11]. The approach presented here is keyless. We are not aware of any other work trying to exploit numerical deviations between execution environments.

5 DISCUSSION AND FUTURE WORK

We have proposed algorithms to generate sets of transparent boundary samples that can identify which microarchitecture (MA) is being used for predictions, based on the output label alone. An evaluation of 400 samples from two datasets using ResNet instances of two depth results in success rates between 28.25 % and 70.50 %. The successful samples span all four MAs considered. This means replacing unsuccessful attempts is a valid strategy. Transparency in terms of PSNR was almost always at least as good as qualities accepted as imperceptible in the literature on lossy image compression (40–50 dB). We showed how specifically composed sets can reach PSNRs of 70 dB and higher. The goal of this paper to establish transparent boundary samples has thus been achieved.

Nevertheless, there is much room for future work. The two datasets in our study exhibited different difficulty in finding boundary samples. A third dataset, ImageNet [5] with much larger input space and higher model complexity has not given us a single boundary sample in reasonable time. We also only covered the closed set identification scenario, where a set of candidate MAs exists. Finding boundary samples for larger models, input sizes, (partially) unknown models, and without candidate MAs are open problems.

Another knob to turn is improving the algorithm. The presented version, inspired by FGSM, is simple, shown to be effective for small instances, but in no way optimal. For example, we do not even consider PSNR as an objective. One could take inspiration from other algorithms for finding adversarial samples, PGD [13] and JSMA [17], which constrain the infinity norm or combine two criteria in a so-called saliency map, respectively. Devising an algorithm that finds boundary samples targeted to a specific MA is another possible direction.

While boundary samples may enable new forms of digital rights management for trained models, more applications would be possible if the resulting boundary samples survived quantization to eight bit. This and high PSNR are conflicting goals.

This work has explored the low-hanging fruits. We used tractable models for a handful of accessible MAs. This ad-hoc approach reached limits in terms of resolution (microarchitecture refinements fall together), scope (GPUs not considered), and complexity (both very small and very large models and input dimensions). Better methods can most likely improve on any of these directions. But where are fundamental limits? In short, the boundaries of boundary samples are not yet understood.

ACKNOWLEDGMENTS

The authors thank Nora Hofer for her valuable feedback and help in preparing the camera-ready version of this paper.

REFERENCES

[1] Yossi Adi, Carsten Baum, and Moustapha Cissé et al. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium*. 1615–1631.
[2] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.
[3] Matthias Carnein, Pascal Schöttle, and Rainer Böhme. 2016. Telltale Watermarks for Counting JPEG Compressions. In *Media Watermarking*. 1–10.
[4] Ingemar Cox, Matthew Miller, and Jeffrey et al. Bloom. 2007. *Digital Watermarking and Steganography*.
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*.
[6] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun (Eds.).
[7] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *International Conference on Computer-Aided Design (ICCAD)*. 133.
[8] Kaiming He, Xiangyu Zhang, and Shaoqing Ren et al. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.
[9] Alex Krizhevsky. 2009. *Learning multiple layers of features from tiny images*. Master's thesis. Univ. of Toronto.
[10] Deepa Kundur and Dimitrios Hatzinakos. 1999. Digital Watermarking for Telltale Tamper Proofing and Authentication. *Proc. IEEE* 87, 7 (1999), 1167–1180.
[11] Tobias Kupek, Cecilia Pasquini, and Rainer Böhme. 2020. On the Difficulty of Hiding Keys in Neural Networks. In *ACM Workshop on Information Hiding and Multimedia Security (IH & MMSec)*. 73–78.
[12] Zheng Li, Chengyu Hu, and Yang Zhang et al. 2019. How to prove your model belongs to you: a blind-watermark based framework to protect intellectual property of DNN. In *Annual Computer Security Applications Conference (ACSAC)*. 126–137.
[13] Aleksander Madry, Aleksandar Makelov, and Ludwig Schmidt et al. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*.
[14] Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. 2020. Adversarial frontier stitching for remote neural network watermarking. *Neural Computating and Applications* 32, 13 (2020), 9233–9244.
[15] Ryota Namba and Jun Sakuma. 2019. Robust Watermarking of Neural Network with Exponential Weighting. In *Asia Conference on Computer and Communications Security (AsiaCCS)*, S. Galbraith, G. Russello, and W. Susilo et al. (Eds.). 228–240.
[16] Nicolas Papernot, Patrick D. McDaniel, and Arunesh Sinha et al. 2018. SoK: Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. 399–414.
[17] Nicolas Papernot, Patrick D. McDaniel, and Somesh Jha et al. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*. 372–387.
[18] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. 2019. DeepSigns: An End-to-End Watermarking Framework for Ownership Protection of Deep Neural Networks. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 485–497.
[19] Alexander Schlögl, Tobias Kupek, and Rainer Böhme. 2021. Forensicability of Deep Neural Network Inference Pipelines. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
[20] Ilia Shumailov, Yiren Zhao, Robert Mullins, and Ross Anderson. 2020. Towards Certifiable Adversarial Sample Detection. In *ACM Workshop on Artificial Intelligence and Security (AISec)*. 13–24.
[21] Christian Szegedy, Wojciech Zaremba, and Ilya Sutskever et al. 2014. Intriguing Properties of Neural Networks. In *International Conference on Learning Representations (ICLR)*.
[22] Yusuke Uchida, Yuki Nagai, and Shigeyuki Sakazawa et al. 2017. Embedding Watermarks into Deep Neural Networks. In *International Conference on Multimedia Retrieval (ICMR)*, B. Ionescu, B. Sebe, and J. Feng et al. (Eds.). 269–277.
[23] Stephen T. Welstead. 1999. *Fractal and Wavelet Image Compression Techniques*.
[24] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. http://arxiv.org/abs/1708.07747 arXiv Computing Research Repository (CoRR), abs/1708.07747.
[25] Jialong Zhang, Zhongshu Gu, and Jiyong Jang et al. 2018. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *Asia Conference on Computer and Communications Security (AsiaCCS)*. 159–172.