Subnormal Number Attacks on Binarized Neural Networks

Nicolás Berrios
Department of Computer Science
Catholic University of Chile
Santiago, Chile

Abstract

Binarized Neural Networks (BNNs) have emerged as a sensible quantization method to reduce compute costs at inference time. As with other machine learning systems deployed in practice, they are susceptible to side-channel attacks that can be leveraged to reveal their internal characteristics and architecture. Previous work on side-channels in BNNs has been limited to the physical domain, requiring a powerful adversary with granular access to the system and advanced hardware tools. In this paper, we show how the inherent binary weight distribution of BNNs make them susceptible to timing attacks in a practical, software-based threat model. We achieve this by leveraging abnormal timing differences in subnormal number arithmetic. Our contributions are two-fold; (a) we show how carefully crafted inputs can produce a time signal strong enough to reveal all the weights of an individual neuron and (b), we scale the attack to infer a fraction of the input layer of a BNN. We conclude by assessing the challenges of BNN implementations in hopes that our findings will motivate safer deployments of BNNs.

1 Introduction

The rising computational cost of training state-of-the-art machine learning (ML) systems has intensified the focus on safeguarding production deployments. As with any theoretical concept deployed in practice, seemingly benign inherent characteristics of computer systems could reveal side-channels of information about internal computations [1, 2]. Attacks that leverage these sources of data to degrade the security of a system are known as side-channel attacks (SCAs) [3]. In the ML systems literature, specifically on neural networks (NN), several SCAs have been proposed [4–11].

On the other hand, another consequence of the larger need for compute is the search for optimizations that reduce inference time. A prominent architecture with significant impact in compute power is known as Binarized Neural Networks (BNN) [12], which compress NN weights to be represented through binary values. Previous SCAs on BNNs have been limited to what we believe to be a generous threat model for the adversary, involving granular physical access for power [13], electromagnetic [14, 15] and timing [16] measurements.

In this work, we introduce a novel attack under black-box software access to a BNN. We do so by leveraging the significant presence of 0s in their architecture coupled with subnormal number inputs (i.e.: 10^{-318}). This approach stems from the abnormal runtime of subnormal number arithmetic, which is significantly slower than normal floating point number operations across multiple architectures [17]. This anomaly creates a distinguishing time signal that can be detected in the software domain, which we first exploit to reveal the weight values of a single neuron and later a fraction of the input layer of a complete BNN.

Machine Learning for Systems Workshop at NeurIPS 2024.

2 Experiments

2.1 Attacking a neuron

Given a set of binary $(0|1)$ weights $W = [w_0, ..., w_m]$, a bias b and an activation function f, we define a single binary neuron or perceptron as $P = (W, b, f)$. We can perform inference on this architecture by evaluating an input $X = [x_0, ..., x_m]$ as follows:

$$P(X) = f\left(\sum_{i=0}^{m}(w_i x_i) + b\right)$$

In this section, we propose an attack that leverages the timing abnormalities in subnormal number arithmetic to define inputs X_n that when evaluated by P reveal the value of a given $w_n \in W$.[1] We achieve this as an adversary limited to black-box measurements of the inference runtime $\mathcal{T}(P(X_n))$ and no access to output. To perform our attack, given a target weight index n, we craft a probe input X_n:

$$\forall x_i \in X_n, \quad x_i = \begin{cases} 10^{-318}, & \text{if } i = n, \\ 0, & \text{else.} \end{cases}$$

For this input, the inference runtime distribution conditioned on values of w_n, can be modeled by the following probability[2], for low values of δ:

$$\mathcal{T}(P(X_n)) \mid (w_n = 1) \geq_\delta \mathcal{T}(P(X_n)) \mid (w_n = 0)$$

This results from the weight multiplication outputting either a subnormal float or a normal float conditioned on the value of w_n.

$$\sum_{i=1}^{m}(w_i x_i) = 10^{-318} * w_n = \begin{cases} 10^{-318}, & \text{if } w_n = 1, \\ 0, & \text{if } w_n = 0. \end{cases}$$

Computationally, this forces the system to perform additional subnormal operations when the target weight $w_n = 1$, as illustrated in Table 1.

Algorithm step	$w_n = 0$				$w_n = 1$			
	N +	S+	N×	S ×	N+	S+	N×	S×
Weight multiplication $(w_i x_i)$			$m-1$	1			$m-1$	1
Weighted sum $(\sum_{i=1}^{m}(w_i x_i))$	m					m		
Bias addition $(+b)$	1					1		

Table 1: Operation distribution (N = Normal float operation, S = Subnormal float operation)

Notably, before reaching the activation function, a neuron where $w_n = 1$ will perform $m + 1$ additional subnormal number sums than one where $w_n = 0$. Since subnormal number sums are slower than normal sums, this generates a timing signal that can be leveraged to infer the value of w_n.

An attacker with runtime measurements can perform multiple queries to find a threshold τ, that holds for the confidence interval $1 - \delta$, where the probability distributions $\mathcal{T}(P(X_n)) \mid (w_n = 1)$ and $\mathcal{T}(P(X_n)) \mid (w_n = 0)$ can be linearly separated to infer the value of w_n, as follows:

$$w_n = \begin{cases} 1, & \text{if } \mathcal{T}(P(X_n)) \geq \tau, \\ 0, & \text{else.} \end{cases}$$

Finally, the same analysis can be repeated for every index n to reveal all the weights $w_n \in W$.

[1] For consistency across our experiments, including the next section, we use $b = 1$ and $f(x) = x > |W|$.

[2] For simplicity, we denote $A \star_\delta B$ as the δ-bounded probability $\Pr[A \star B] \geq 1 - \delta$ for a given $\star$ operator.

2.1.1 Evaluation

We evaluate our attack by performing queries to a black-box perceptron model in a Intel i7 CPU. We define two perceptrons $P = (W, b, f), w_n = 0$ and $P' = (W', b, f), w'_n = 1$ with a conservative $m = 8$. For each measurement, we flip a coin to select a perceptron and query it 1000 times with our probe X_n, collecting the total time in nanoseconds. We plot these measurements in Figure 1 and evaluate the accuracy of a simple linear threshold through a ROC curve in Figure 2.

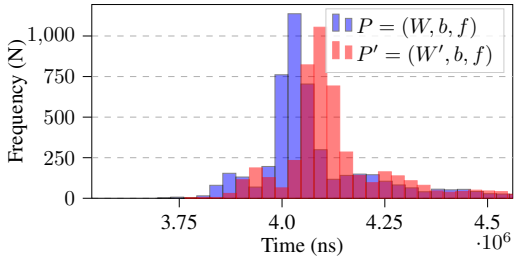

Figure 1: Runtime distribution for P and P'.

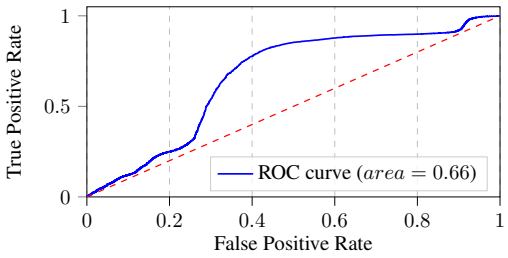

Figure 2: ROC curve on detecting w_n's value.

2.2 Attacking a layer

We learn from attacking a single binary neuron that the value of the n-th index weight is correlated to the runtime of performing inference of our probe X_n. On a complete BNN, our input X_n will be evaluated on multiple neurons and as such, the runtime $\mathcal{T}(\text{BNN}(X_n))$ will be correlated to the sum of the inference runtimes of every neuron. Intuitively, for the input layer I, it will linearly scale with the number of 1s present on n-th index weights of neurons in the layer. Particularly, for our probe X_n, the runtime is the fastest when all the n-th indexes are 0 and is the slowest when all of them are 1. In other cases, we can use the linear correlation to infer the proportion of the weights (particularly, the number of n-th index 1s), although not which neuron they correspond to in the layer.

2.2.1 Evaluation

We evaluate our attack performing local queries into a blackbox BNN model in the same processor. We define 4 neural networks with 8 input neurons, and vary the number of 1s in their n-th indexes. We then query $\mathcal{T}(\text{BBN}(X_n))$ to get the average runtime of 100 queries. As illustrated in Figure 3, we observe a linear relationship between the number of $w_n = 1$ and the runtime. In Figure 4, we analyze the success rate of our attack with a varying input layer size $|I|$. We assume access to an oracle providing the exact runtime and calculate the percentage of weights in I that could be revealed.

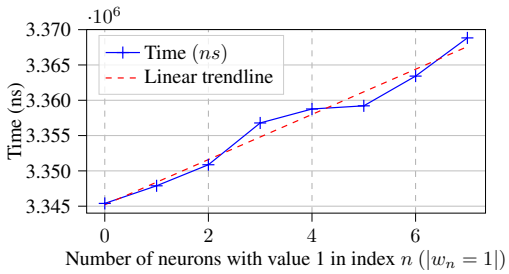

Figure 3: Average runtime for varying w_n values.

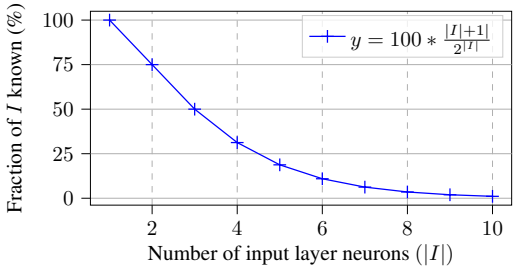

Figure 4: Best-case success rate v.s. $|I|$.

For our success rate modeling in Figure 4, we assume that binary weight values are randomly distributed. We model the success rate as $\frac{|I|+1|}{2^{|I|}}$. Notably, $2^{|I|}$ represents the number of possible combinations of 0s and 1s in the input layer. The nominator $|I| + 1$ stems from our attack being able to distinguish $|I|$ cases plus the all-0 case. Formally, for every count $c = |w_n = 1|$ there are $\binom{|I|}{c}$ possible combinations, and thus access to exact runtime yields a $\frac{1}{\binom{|I|}{c}}$ probability of being correct. Then, the expected number of correct cases can be modeled as $\sum_{c=0}^{|I|} \binom{|I|}{c} \frac{1}{\binom{|I|}{c}} = \sum_{c=0}^{|I|} 1 = |I| + 1$.

3 Discussion

Our experiments reveal that subnormal number inputs generate a timing side-channel that provides non-trivial advantage for an adversary to infer the binary weights of a single neuron in a BNN. When scaling our attack to the input layer, we are able to continue detecting the timing signal distributed in multiple neurons, revealing upwards of 50% of the input layer weight values for a BNN with 3 input layer neurons. Although our attack achieves a lower accuracy compared to other side-channels performed through hardware [13–16], to the best of our knowledge, we are the first to show a BNN side-channel attack that achieves weight extraction through timing measurements at the software level.

3.1 Limitations

A downside of our attack at the input layer-level is that accuracy degrades significantly as the number of neurons grow, even under an hypothetical best case assumption of exact runtime access. This stems from the number of possible combinations of 0s and 1s scaling exponentially with the size of the input layer, making distinguishing the ratio not enough of an advantage for accurately revealing a significant fraction of I.

Another limitation of our work is that it requires the system to allow for floating point input to leverage the abnormalities of subnormal float arithmetic. We believe this not to be an unreasonable assumption, since by design some neurons could account for the input precision loss at the input layer, and as such, receiving higher precision input without immediately bounding it to binary could be practical on real deployments.

On the other hand, we find our attack to be also limited to the binary representation of BNNs encoded as $(0|1)$ and not the other prominent version of $(-1|1)$, given that it is directly dependent on the notable distribution of 0s in the neural network.

Finally, for this same reason, we find our attack to be implicitly limited to BNNs representations, and not practically applicable to NNs of higher precision, which would not present a binary-distinguishable distribution of weights.

3.2 Future work

We believe future work should be focused on providing statistical models that can both; (a) improve the success rate of our attack as the number of input layers neuron grow, and (b) extend the extraction to further hidden layers of the network.

We also note that our experiments included multiple repeated queries as a simple method to amplify the timing signal. This is a sensible approach when performing a realistic software-based timing attack which may be subjected to background noise that could obscure signals at the nanosecond level. Nonetheless, exploring additional time signal amplification techniques to reduce the number of queries would be valuable for increasing the effectiveness of our attack and determining its feasibility by a remote network adversary.

Finally, we find that further exploring the impact of subnormal numbers on other types of NNs, particularly on ones that allow for floating point precision, represents a novel area worthy of further research.

4 Conclusion

In this paper, we study how subnormal number inputs degrade the security of BNNs against an adversary with runtime access. We successfully extract the weights of a single neuron with non-trivial accuracy and show how the timing side-channel persists on multiple-neuron inference, allowing for an adversary to extract fragments of the input layer. We achieve this with runtime access to a locally deployed black-box BNN. To the best of our knowledge, we are the first to show an attack that achieves time-based weight extraction on a BNN in the software domain. We hope our findings help motivate safer deployments of BNNs and further raise awareness of the potential risks of side-channel attacks on neural network systems.

References

[1] M. Randolph and W. Diehl, "Power side-channel attack analysis: A review of 20 years of study for the layman," *Cryptography*, vol. 4, no. 2, p. 15, 2020.

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side—channel (s)," in *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*. Springer, 2003, pp. 29–45.

[3] F.-X. Standaert, "Introduction to side-channel attacks," *Secure integrated circuits and systems*, pp. 27–42, 2010.

[4] L. Batina, S. Bhasin, D. Jap, and S. Picek, "{CSI}{NN}: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 515–532.

[5] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[6] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *arXiv preprint arXiv:1812.11720*, 2018.

[7] M. Méndez Real and R. Salvador, "Physical side-channel attacks on embedded neural networks: A survey," *Applied Sciences*, vol. 11, no. 15, p. 6790, 2021.

[8] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 393–406.

[9] H. T. Maia, C. Xiao, D. Li, E. Grinspun, and C. Zheng, "Can one hear the shape of a neural network?: Snooping the gpu via magnetic side channel." in *USENIX Security Symposium*, 2022, pp. 4383–4400.

[10] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang, "Open dnn box by power side-channel attack," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2717–2721, 2020.

[11] K. Yoshida, T. Kubota, S. Okura, M. Shiozaki, and T. Fujino, "Model reverse-engineering attack using correlation power analysis against systolic array based neural network accelerator," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[13] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Power side-channel attacks on bnn accelerators in remote fpgas," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 357–370, 2021.

[14] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "Deepem: Deep neural networks model recovery through em side-channel information leakage," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 209–218.

[15] V. Yli-Mäyry, A. Ito, N. Homma, S. Bhasin, and D. Jap, "Extraction of binarized neural network architecture and secret parameters using side-channel information," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[16] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—attacks and defenses," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 079–12 092, 2021.

[17] M. Andrysco, D. Kohlbrenner, K. Mowery, R. Jhala, S. Lerner, and H. Shacham, "On subnormal floating point and abnormal timing," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 623–639.