Output Rounding Is Not a Free Defense Against Cryptanalytic Neural Network Extraction

Iliyas Noman
MIT

Mateja Vukelic
MIT

Vidur Jasuja
MIT

Abstract

The Carlini–Jagielski–Mironov attack (CRYPTO 2020) recovers the parameters of a fully connected ReLU network from oracle access alone, achieving floating-point precision in number of queries that is small relative to the network size. Its key primitive is a finite-difference estimator of second derivatives utilizing very small input perturbations. This suggests a natural defense: output rounding in which the oracle returns $\mathrm{round}(f(x)/s) \cdot s$ instead of $f(x)$, whence s is some 2^{-N}. We show empirically that with $s = 2^{-31}$ this defense completely breaks the original attack at zero observable utility cost (mean relative output error $\sim 10^{-10}$, below `float64` rounding noise). We then design a new attack against this defense, termed a "step spacing" attack, which searches for the edges of linear regions in a ReLU network to achieve accurate estimates of first-order directional derivatives at *macroscopic* probe scales. The new attack recovers the model on every cell tested at $\mathrm{bits} \geq 18$ across two one hidden-layer architectures and three seeds, and recovers the model down to 10% relative error for hidden-layer architectures for $\mathrm{bits} \geq 16$ bits. The defender's true safe regime is therefore $\mathrm{bits} \leq 18$, with utility cost $\sim 5 \times 10^{-6}$ rather than $\sim 10^{-10}$. The paper documents the defense, the attack, and the security/utility frontier between them.

1 Introduction

What this paper is about. In the context for the paper, an underlying neural network is exposed only as an accessible oracle. In the original setting, the user supplies an input x and receives an exact full-precision floating-point value $f(x)$. Given this, the question arises as to whether the network's weights and biases can be recovered. Carlini, Jagielski, and Mironov [2] answered *yes* for fully connected ReLU networks given full-precision outputs. Their attack achieves bit-perfect extraction in a query budget polynomial in the network dimensions and the log of the desired precision—concretely, roughly $2^{21.5} \approx 3 \cdot 10^6$ queries to recover a 100k-parameter MNIST network to 2^{-25} extraction error

Due to the nature of the attack requiring small input movements, a natural defensive response is to obscure the low-order bits of the oracle's output, since those are the bits the attack reads. This paper studies that defense—specifically deterministic *output rounding at inference*—and the corresponding attack we designed when the defense is treated as part of the threat model rather than a fixed black box.

Our central finding is that while casual evaluation suggests output rounding appears to be a free defense, it either sacrifices utility if precision is too low or security if precision is too high and the attacker is knowledgeable.

Related work. Cryptanalytic extraction is a precision-focused version of an area that begins with classical model stealing [6, 12, 10, 9, 4], which targets functional accuracy rather than parameter recovery. Work focusing on parameter recovery begins with Milli et al. [7] for shallow models and Rolnick and Kording [11] for ReLU networks at low precision. Next, Jagielski et al. [3] sharpened the precision bounds, and most relevant to this paper Carlini et al. [2] introduced the second-order

finite-difference primitive that brings extraction to floating-point fidelity. Later, Canales-Martínez et al. [1] subsequently gave a polynomial-time variant. On the defensive side, there is much less work; post-training weight quantization [8, 5] is well studied as a utility/efficiency tradeoff but not as a cryptanalytic defense. However, to our knowledge, output rounding has not been characterised against the Carlini attack as either a stand-alone defense or a target of an adapted attack.

The Carlini–Jagielski–Mironov attack. A ReLU network is piecewise linear, no matter the number of layers: there is a partition of input space into polytopes on each of which the network is exactly affine. The kinks in f are critical hyperplanes, places where some neuron's pre-activation crosses zero. Carlini et al. exploit two pieces of structure. First, they binary-search 1-D lines through input space for these critical points. They use the following test: for two points t_l and t_h whose midpoint is t_m, if the non-linearity test $|f(t_m) - \frac{1}{2}(f(t_l) + f(t_h))| \geq \tau$ with $\tau \approx 10^{-8}$ holds, then the network is not affine on this line segment and as such some neuron's pre-activation hyperplane intersects the segment. Second, at a found critical point they use finite differences to estimate the gradient at that critical point, which is proportional to that neuron's weight row. The rest of the extraction process is done layer by layer; we summarize it in detail in Section 2 since the rest of the paper builds against it.

Our defense: output rounding at inference. We replace the oracle providing the exact output with an oracle providing a rounded output: $f_r(x) := \mathrm{round}(f(x)/s) \cdot s$ where $s = 2^{1-N}$ for a parameter N. Apart from the rounding step, the model is unchanged: weights remain in `float64` and the forward pass is full precision. The defense works against the original attack because Carlini et al.'s primitives evaluate output deltas of order equal to or less than 10^{-11}, which lie strictly below the rounding step. As a consequence, the linearity described in the previous section fails to distinguish linear from non-linear regions, and the signals used to find critical points collapse. Empirically, the original attack's success rate against output-rounded oracles is $0/9$ at every $N \leq 32$, and at $N = 32$ the rounding's mean relative output error is $\sim 10^{-10}$—below `float64` mantissa noise. We talk further about this in Section 3.

Our adapted attack: step-spacing extraction. The defender's rounding changes the geometric structure that we receive information in. The rounded oracle returns a step function whose jumps are exactly the points where the underlying smooth f crosses half-integer multiples of s. So, we can get a complete picture of the level sets of rounded f. We exploit this with a different finite-difference primitive: instead of perturbations as small as we can achieve, we look at perturbations as large as we can achieve without crossing any neuron's boundary to most accurately approximate the slope of the smooth function. Concretely we define a directional-derivative estimator $\widehat{\partial_d f}(x) = (\mathrm{lvl}(x + Rd) - \mathrm{lvl}(x - Rd))\, s/(2R)$ where $\mathrm{lvl}(\cdot)$ is the integer rounding-level of the oracle and R is chosen adaptively so the level differential is large. In particular, we set $R := 4 \cdot R$ whenever the level differential isn't large enough, and $R := R/2$ when linearity no longer holds. With this, we recover the gradient with relative error $\sim 10^{-6}$. With a working gradient primitive, the rest is mechanical; sweep some lines, track the gradient vectors, take consecutive differences in the gradient (which, if nonzero, provide significant information about the hyperplane), cluster the jumps to determine signs and biases, and fit the final layer via least squares estimation. The attack is valid for $N \geq 18$, where the rounding step is small enough but still larger than `float64` noise: $N \geq 18$ on our 2-layer networks. We develop the attack in Section 4 and evaluate it in Section 5.

Headline picture. Figure 1 plots the extraction success rate of the step-spacing attack against output rounding across many bit-widths. The transition is clear: success at $N \geq 18$ across architectures and seeds, and failure at $N = 16$.

The defender who chose $N = 32$ for the original attack now extracts cleanly under the adapted attack; to defeat both, $N \leq 16$ is required, which costs $\sim 5 \times 10^{-6}$ relative output error rather than the previously apparent $\sim 10^{-10}$. The remainder of the paper documents these claims in detail.

2 Background: the Carlini–Jagielski–Mironov Attack

Carlini, Jagielski, and Mironov's original attack [2] handles general fully connected ReLU networks of arbitrary depth L. The attack runs layer by layer: after recovering layers $1, \ldots, \ell - 1$, the attacker

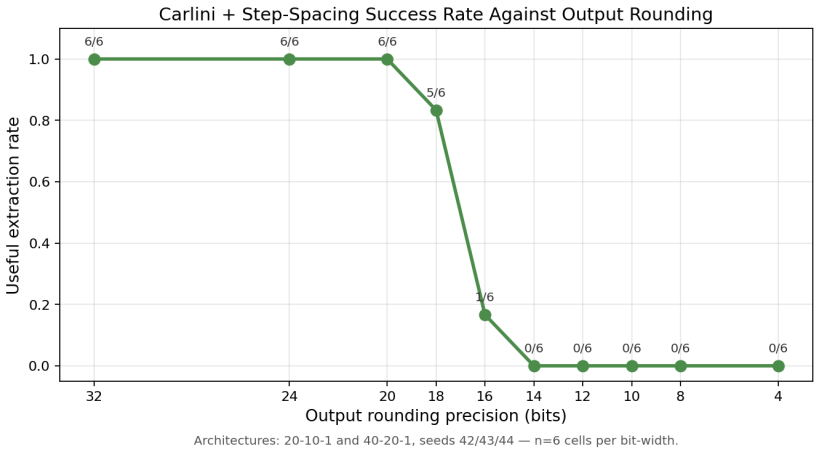

Figure 1: Step-spacing attack success rate against an output-rounded oracle as a function of rounding precision N (bits). The original Carlini attack is at 0% for every column shown. Rates are computed over the 2-layer architectures `20-10-1` and `40-20-1` with three seeds.

Contributions All team members contributed equally to the ideation for the topic of this project with input from our TA Xiaochen Zhu, the design of the experiments was first prototyped by Iliyas and Mateja with Vidur expanding the code base, running more experiments and confirming our initial findings. The attack was developed jointly and we also all contributed to the experiments with it. For the first draft Vidur was mostly responsible for the abstract, parts of Section 1 and most of the Section 4 writeup, Iliyas was responsible for Sections 2, 3, 6 and 7, and Mateja for other parts of Section 1 and Section 5. All members reviewed and edited all other sections to produce the final draft of the paper.

treats those as a known prefix transformation and re-runs the same primitives on the residual network, a procedure they call *layer peeling*. For the sake of clarity, in this section and throughout the rest of the paper we restrict our exposition to the simplest non-trivial case—networks with a single hidden layer:

$$f(x) \;=\; v^{\top}\sigma(Wx+b)+c, \qquad x\in\mathbb{R}^{n_0},\;\; W\in\mathbb{R}^{n_h\times n_0},\;\; b\in\mathbb{R}^{n_h},\;\; v\in\mathbb{R}^{n_h},\;\; c\in\mathbb{R}, \quad (1)$$

where $\sigma(z)=\max(z,0)$ is applied entrywise. Extending the attack and the defense we describe to deeper networks is largely bookkeeping: the same primitives apply at each peeling step, and the new attack we present in Section 4 is portable to multi-layer networks by the same mechanism.

Setting: The attacker has black-box query access to f, knows the architecture (n_0, n_h), and reads back full-precision `float64` logits. The goal is to recover (W, b, v, c) to floating-point precision.

Key observation: Let w_j denote the j-th row of W, so the j-th pre-activation is $w_j \cdot x + b_j$. The set

$$\mathcal{H}_j \;=\; \{\, x\in\mathbb{R}^{n_0} \;:\; w_j\cdot x+b_j=0 \,\}$$

is the *critical hyperplane* of neuron j. On the open complement of $\bigcup_j \mathcal{H}_j$, every neuron is either fully on or fully off; each connected component is a ***ReLU polytope***. On a polytope, f is exactly affine, with gradient

$$\nabla f(x) \;=\; \sum_{j:\, w_j\cdot x+b_j>0} v_j\, w_j.$$

This gradient is constant inside a polytope and *jumps* when x crosses some $\mathcal{H}_j$: the contribution $v_j w_j$ switches on or off. The jump is exactly $\pm v_j w_j$—a vector along the activated neuron's weight row, scaled by the unknown final-layer coefficient v_j.

Attack pipeline:

1. **Find critical points.** Sweep a 1-D line through input space; locate every point where f is non-linear by binary search using the test

$$\left|f(t_m) - \tfrac{1}{2}(f(t_l) + f(t_h))\right| < \tau, \qquad \tau \approx 10^{-8}.$$

   Each such point is a witness $x^* \in \mathcal{H}_j$ for some neuron j.

2. **Recover weight rows.** At a critical point $x^* \in \mathcal{H}_j$, evaluate the four-point finite-difference stencil

$$S(\varepsilon, \varepsilon_j;\, d) \;=\; f(x^* + \varepsilon d - \varepsilon_j d) - f(x^* + \varepsilon d) - f(x^* - \varepsilon d) + f(x^* - \varepsilon d + \varepsilon_j d)$$

   with $\varepsilon \approx 10^{-5}$, $\varepsilon_j \approx 10^{-6}$. The idea is that once we get the witness, we perturb by ε to find two points one of which the neuron is active for and the other where it's inactive. Then we perturb in the direction ε_j to isolate the influence of the weights. (See the original paper for more details) A short Taylor expansion gives $S \approx \varepsilon_j\,(g_L - g_R) \cdot d$, where g_L and g_R are the gradients on either side of $\mathcal{H}_j$. Since $g_L - g_R = \pm v_j w_j$, probing S in n_0 basis directions recovers w_j up to a sign and the unknown scalar v_j.

3. **Sign recovery.** Magnitudes are now known but each w_j remains ambiguous up to a sign $\sigma_j \in \{\pm 1\}$. The consistency check is additive across neurons, so the attack brute-forces the 2^{n_h} sign assignments and selects the one whose implied function best matches recorded $(x_i, f(x_i))$ pairs. (For the multi-layer attack, an additional closed-form trick handles signs when the next layer is sufficiently narrower; this is the place where layer peeling is most delicate.)

4. **Bias and final-layer recovery.** For each cluster of critical points hitting $\mathcal{H}_j$, the bias is read off as $b_j = -w_j \cdot x^*$. With (W, b) now known (upto permutation and scaling), the final-layer coefficients (v, c) drop out by ordinary least squares: sample fresh queries $\{(x_i, f(x_i))\}$ and solve

$$f(x_i) \;\approx\; \left[\, \sigma(W x_i + b) \;\middle|\; 1 \,\right] [v\,;\, c].$$

With sufficient query precision, Carlini reports an extraction error of $\sim 2^{-25}$ on a 100 000-parameter MNIST network using roughly $2^{21.5} \approx 3 \cdot 10^6$ queries.

What our attack relies on: Two numerical conditions are load-bearing. First, the linearity test in step 1 must distinguish "linear" from "one ReLU crosses" regions; this requires the oracle's output to resolve information at scale $\tau \approx 10^{-8}$ – well within `float64` precision, but easily erased by output rounding. Second, the stencil S in step 2 has signal magnitude $\varepsilon_2 \,\|g_L - g_R\| \approx 10^{-6}$ for typical weight magnitudes, residing in the low-order mantissa bits of the output. Both primitives *operate on information that the defender can choose to delete*—and as we now show, deterministic output rounding deletes exactly those bits.

3 Defense: Output Rounding at Inference

The defender retains the full-precision model. At inference time, the oracle returns a rounded logit:

$$f_r(x) \;=\; \mathrm{round}\big(f(x) \cdot 2^{N-1}\big)/2^{N-1}, \tag{2}$$

where N is the rounding precision in bits. The rounding step is $s = 2^{1-N}$ (so $N = 32$ gives $s \approx 5 \times 10^{-10}$ and $N = 8$ gives $s \approx 8 \times 10^{-3}$).

The overarching primitive in Carlini's attack is being able to accurately estimate first derivatives during binary search in the witness finding part. The way it is implemented in the paper relies on floating point accuracy, which breaks when the outputs are rounded. Additionally, when perturbing in basis directions to estimate the (scaled) weights, the attack again relies on precision, as it queries small perturbations in chosen directions.

We ran the original Carlini et al. [2] implementation against rounded oracles for $N \in \{64, 32, 16, 8, 4\}$ on three architectures (`10-15-15-1`, `20-10-1`, `40-20-1`) with three seeds each. Defining *useful extraction* as "the pipeline completed and recovered weights satisfy logit loss < 1 and positive bits of precision" (a stricter criterion than a binary success flag, because the pipeline sometimes runs to completion but emits negative-bit-precision garbage), the success rates against output rounding are $\{8/9,\ 0/9,\ 0/9,\ 0/9,\ 0/9\}$ at $N \in \{64, 32, 16, 8, 4\}$ respectively.

We sample 10^5 inputs from a standard Gaussian on the input dimension and measure $\mathbb{E}[\,|f_r(x) - f(x)|\,]/\mathbb{E}[\,|f(x)|\,]$. Mean relative output error is $\approx 10^{-10}$ at $N = 32$, $\approx 5 \times 10^{-6}$ at $N = 16$, $\approx 10^{-3}$ at $N = 8$, and a few percent at $N = 4$. At $N = 32$ the rounding error is below `float64` mantissa noise: there is *no detectable change* to the model's outputs from any test the user can run.

Comparison with weight quantization. We also tested post-training weight quantization (round each weight matrix to N bits at rest, then run inference at full precision). In contrast to output rounding, weight quantization *does not* defeat the original attack at $N \geq 16$: the ReLU function it computes is still smooth almost everywhere, only with slightly different weights. Carlini's primitives operate on output deltas of order 10^{-11} that survive weight quantization down to $N \approx 8$. Output rounding is a strictly stronger defense at the same nominal bit-width because it makes the function piecewise constant rather than merely changing which smooth function is computed.

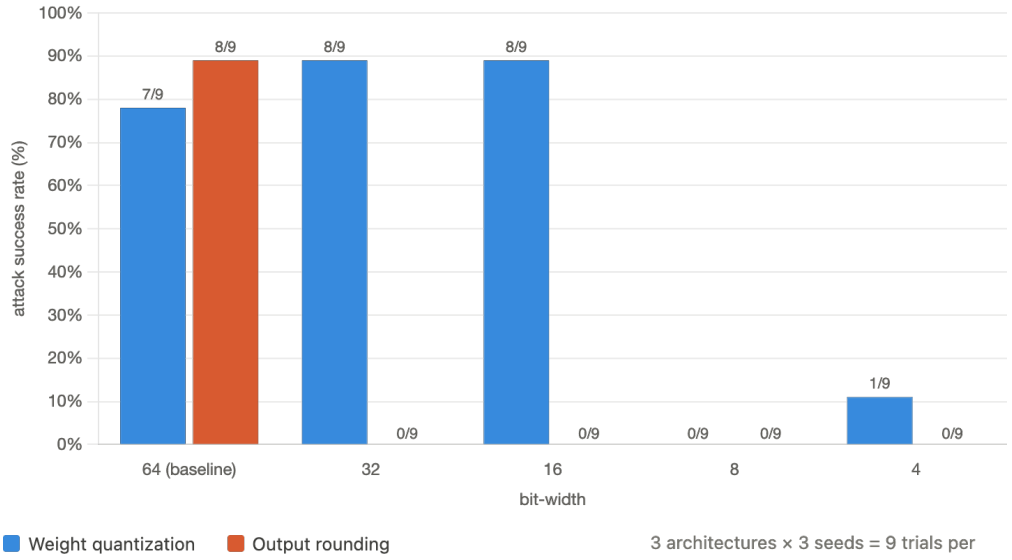

Figure 2: Original Carlini attack fidelity under post-training weight quantization versus output rounding. Weight quantization changes the learned parameters but leaves the queried function locally smooth; output rounding changes the oracle itself and destroys the small finite-difference signals used by the original attack.

4 Attack: Step-Spacing Extraction

The defender's rounding deletes information at scales below s, but at scales above s the output is deterministic and fully observable. If we take a point x_0 and a line through it $x(t) = x_0 + td$, and plot the graph of the oracle's rounded output, it will be a step function: it only moves one level at a time, with the jumps indicating exactly the points where $f(x(t))$ crosses $(k + \frac{1}{2})s$ for integer k.

We exploit this with finite differences once again, but, as stated earlier, in a different way from the Carlini attack.

4.1 Directional derivative via integer-level differencing

Let $\mathrm{lvl}(x) = \mathrm{round}(f_r(x)/s)$ be the integer rounding level of the oracle. For evaluating the directional derivative of f in direction d at a point x, we introduce a "radial" parameter R. We approximate the directional derivative of f using f_r by the intuitively clear formula:

$$\widehat{\partial_d f}(x_0) \;:=\; \frac{\big(\mathrm{lvl}(x_0 + Rd) - \mathrm{lvl}(x_0 - Rd)\big)\, s}{2R}. \tag{3}$$

Provided f is affine on $[x_0 - Rd,\ x_0 + Rd]$, i.e. we do not cross any ReLU hyperplanes, this estimator has additive error $\leq s/(2R)$ on the underlying directional derivative: one rounding step divided

by the window. Choosing R adaptively so that the level differential is at least L_{tgt} (≈ 64 in our experiments) yields a relative directional-derivative error of at most $1/L_{\text{tgt}} \approx 1.5\%$.

We use a similar test as noted earlier to make sure our approximation only queries points in the same ReLU polytope: query the midpoint level $\text{lvl}(x_0)$ and accept the window only if

$$\left| \text{lvl}(x_0) - \tfrac{1}{2}(\text{lvl}(x_0 - Rd) + \text{lvl}(x_0 + Rd)) \right| \;\leq\; \max(1.5,\, 0.1\,|L_{\text{end}}|), \tag{4}$$

where $L_{\text{end}} = \text{lvl}(x_0 + Rd) - \text{lvl}(x_0 - Rd)$ is the endpoint level differential. This is the same test Carlini's binary search uses, just at the rounding scale instead.

To pick R, we initialize it at a small value and keep growing it, quadrupling where possible, until the level differential is large enough while the region is still affine. If we cannot hit $L_{\text{tgt}} = 64$, then we expand the region as much as possible, or reject the point altogether if it cannot cross at least $L_{\text{fallback}} = 4$ levels while remaining affine.

4.2 Weight rows from gradient jumps

Now, we know how to evaluate the gradient at a given point, which is constant on a single region not crossing any ReLU boundaries. The logical next step is to figure out how to extract the weights once we do cross a boundary. For a 2-layer network $f(x) = W_2\,\sigma(W_1 x + b_1) + b_2$, the gradient at x is

$$\nabla f(x) \;=\; \sum_{j\text{: neuron } j \text{ active at } x} W_2[j] \cdot W_1[j, :]. \tag{5}$$

If only neuron j^* flips between two adjacent points x_a and x_b, then

$$\nabla f(x_b) - \nabla f(x_a) = \pm\, W_2[j^*] \cdot W_1[j^*, :],$$

which is a vector along that neuron's weight row.

The attack therefore proceeds as follows. For a point x_0, repeatedly sample a random direction and the corresponding line through x_0, sample some number of points on this line, and estimate the gradient at each point by computing the directional derivative along each basis vector. Then, take consecutive differences of the gradients along the sweep; every non-negligible difference is a candidate weight row up to sign and scaling. As we do this for many sweep directions, we accumulate many candidates for weight rows, targeting around 200 in our case. The weight rows are then clustered by cosine similarity, and each cluster's representative is its component-wise median.

4.3 Meet-in-the-middle sign recovery

After clustering, we have W_1 up to a per-neuron sign $\sigma \in \{\pm 1\}^{n_h}$. Recovering signs is done by a meet-in-the-middle approach. For small enough networks, we use brute force and simply try every orientation combination, selecting the one with the least residual; for larger networks, we split the neurons into halves, take the best combinations from each half, and try all cross products of these combinations. This is still exponential, but with a base of $\sqrt{2}$ instead of 2 in the number of neurons.

4.4 End-to-end pipeline

Algorithm 1 summarizes the full attack. The final steps are just standard linear algebra algorithms; full pseudocode is in the supplementary code.

4.5 Extension to two hidden layers

The one-hidden-layer version above recovers first-layer gradient jumps and then solves the final layer directly. For the two-hidden-layer architecture `10-15-15-1`, the first stage is still the same: sweep input lines, estimate gradients, and cluster weight rows. However, now, we need to change coordinates once we have the first-layer map, and then repeat the process for the second layer before finally fitting the scalar output layer via least squares.

This extension is more error-prone because first-layer error compounds into the second-layer coordinate system. Nonetheless, it is still important: it shows that the rounded-oracle signal is not confined to one-hidden-layer models. Section 5.6 reports a preliminary `10-15-15-1` result with useful extraction at $N \in \{32, 16, 8\}$.

Algorithm 1 Step-spacing attack on a 2-layer rounded ReLU network.

Require: oracle f_r, rounding step s, architecture sizes $(n_0, n_h, 1)$, integers $n_{\text{sweep}}, n_{\text{probe}}$.

Initialize candidate set $C \leftarrow \emptyset$.
for $i = 1$ **to** n_{sweep} **do**
  Sample random origin x_0 and unit direction d.
  Sample n_{probe} uniform values $t \in [-T, T]$ and set $x_t = x_0 + td$.
  for each x_t **do**
    Estimate full gradient $g_t \leftarrow \widehat{\nabla} f(x_t)$ via Eq. 3 on each basis direction.
  end for
  for consecutive valid $(g_t, g_{t'})$ **do**
    if $\|g_{t'} - g_t\| / \max(\|g_t\|, \|g_{t'}\|) > \theta$ **then**
      Add $g_{t'} - g_t$ to C, with location $\frac{1}{2}(x_t + x_{t'})$.
    end if
  end for
end for
Cluster C by cosine similarity into n_h groups; cluster median $\to W_1[j, :]$ up to sign and scale.
Recover $b_1[j]$ as the median of $-W_1[j, :] \cdot x_{\text{loc}}$ over the cluster.
Recover signs by brute force for small n_h, and by MITM for larger n_h.
Solve final layer W_2, b_2 by least squares: $f_r(X) \approx [\sigma(XW_1 + b_1) \mid 1][W_2; b_2]$.
return (W_1, b_1, W_2, b_2).

Table 1: Step-spacing attack success rate against output rounding, $n = 3$ cells per column on each architecture. Useful = rel_loss < 0.5.

Arch	$N = 32$	$N = 24$	$N = 20$	$N = 18$	$N = 16$	$N = 14$	$N = 12$	$N = 10$	$N = 8$	$N = 4$
`20-10-1`	3/3	3/3	3/3	**3/3**	0/3	0/3	0/3	0/3	0/3	0/3
`40-20-1`	3/3	3/3	3/3	2/3	1/3	0/3	0/3	0/3	0/3	0/3

5 Experiments

Target networks are 2-layer or 3-layer fully connected ReLU networks trained briefly on random Gaussian inputs and binary targets, following the protocol of the open-source `cryptanalytic-model-extraction` repository. We use architectures `20-10-1` and `40-20-1` as the primary 2-layer benchmarks, and `80-40-1` for a scaling study; for the original-attack baseline we additionally use `10-15-15-1` as a 3-layer reference. Each (architecture, seed) cell is trained with seeds $\{42, 43, 44\}$. We define a useful extraction as rel_loss $:= \mathbb{E}[|\hat{f}(x) - f(x)|] / \mathbb{E}[|f(x)|] < 0.5$ on a held-out batch of $2 \cdot 10^4$ Gaussian probes, and report `max_logit_loss` for context.

5.1 Output rounding defeats the original attack

Figure 2 compares the original attack against weight quantization (well-studied) and against output rounding (our defense). Output rounding pushes the attack to $0/9$ at every $N \leq 32$, while weight quantization at the same nominal bit-width leaves the attack near-perfect down to $N = 16$ and only breaks at $N = 8$. The qualitative reason was given in Section 3: weight quantization changes which smooth function the attack is attacking, output rounding makes the function piecewise constant.

5.2 Step-spacing defeats output rounding at fine bit-widths

Table 1 reports the step-spacing attack across a finer sweep of bit-widths on the two 2-layer benchmark architectures.

The attack works cleanly down to $N = 18$ on `20-10-1` and $N = 20$ on `40-20-1`; the cliff is at $N = 16{-}18$. The smallest `rel_loss` values at $N = 32$ are on the order of 10^{-4} (essentially functional equivalence on the test distribution); at $N = 20$ they are still under 10^{-1}. Figure 3 plots the per-cell `rel_loss` traces through the threshold; the transition from $\sim 10^{-1}$ at $N = 20$ to $\gtrsim 0.5$ at $N \leq 16$ is reproducible across all six (architecture, seed) cells.

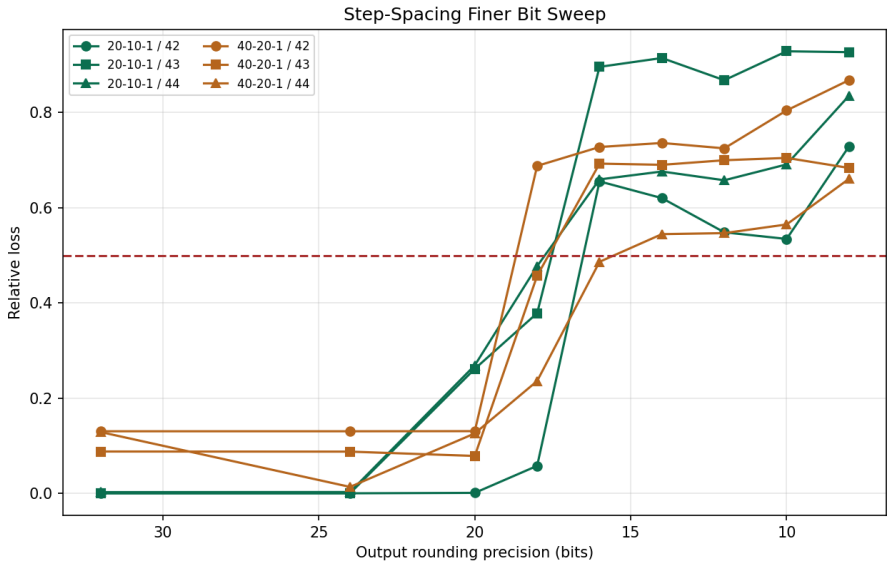

Figure 3: Per-cell relative output error `rel_loss` of step-spacing extraction as a function of rounding precision N. The dashed line at 0.5 marks the useful-extraction threshold. Each cell of (architecture, seed) is one trace; the cliff lies in $N \in [16, 18]$ for both architectures.

5.3 Robustness around the threshold

We checked the robustness of various parameters of the experiment, one at a time, including the number of lines swept per direction (n_{probe}) and number of directions probed (n_{sweep}), and the clustering threshold. For all the parameters tested, the attack with the rounding threshold $N = 18$ was consistently successful, with the relative loss ranging from 0.02 to 0.30. On the other hand, at $N = 16$, we never had a relative loss better than 0.6, indicating that there is a genuine barrier at $N \approx 16$ for the success of this attack.

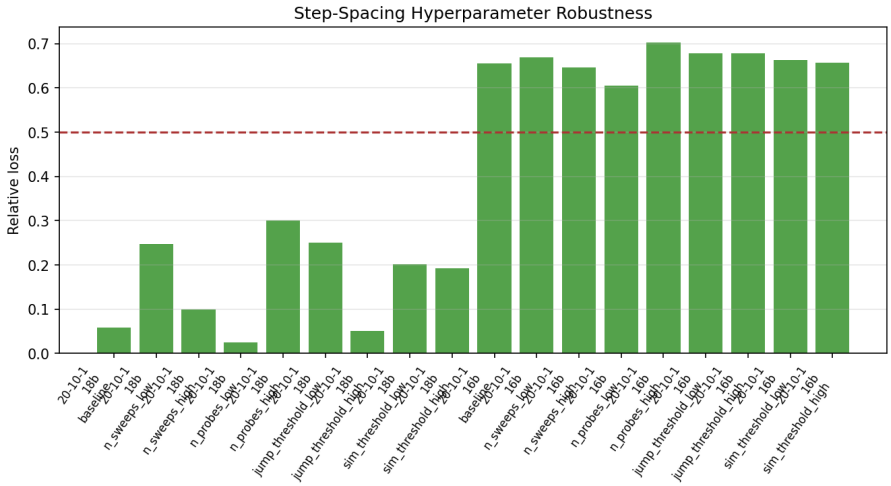

Figure 4: Robustness of step-spacing extraction to one-factor-at-a-time hyperparameter perturbations at the two threshold cells (`20-10-1`, $N=18$) and (`20-10-1`, $N=16$) The loss for $N = 18$ is never greater than .3, and for $N = 16$ is never less than .6.

Table 2: Step-spacing attack on the wider `80-40-1` architecture (seed 42). `q` is total oracle queries.

N	32	20	18	16	8	4
`rel_loss`	0.12	0.11	0.35	0.68	0.74	FAIL
`q` ($\times 10^6$)	7.7	9.0	12.2	31.5	47.8	46.1

5.4 Scaling

We also tested whether the attack worked for wider networks, testing a wider 2-layer architecture `80-40-1` (one seed). Here, the same threshold structure appears (Table 2), with the relative loss falling off a cliff at $N = 16$ to .68.

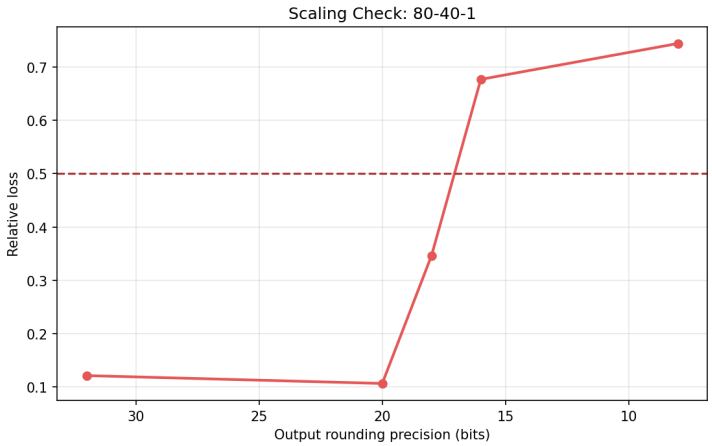

Figure 5: Scaling check on the wider `80-40-1` architecture. Once again, the useful-extraction threshold is crossed between $N = 18$ and $N = 16$, identical to smaller architectures.

5.5 Query count scaling

Total oracle queries grow roughly with precision deteriorating, due to the fact that levels are wider and thus we need to sweep and probe more lines and points to find the useful gradient changes we need, also taking more time to adaptively find a good value for R. Note that even as $N \leq 16$, the number of oracle queries becomes very large, but the attack still fails; this proves that the threshold is not due to undertraining or a lack of oracle budget.

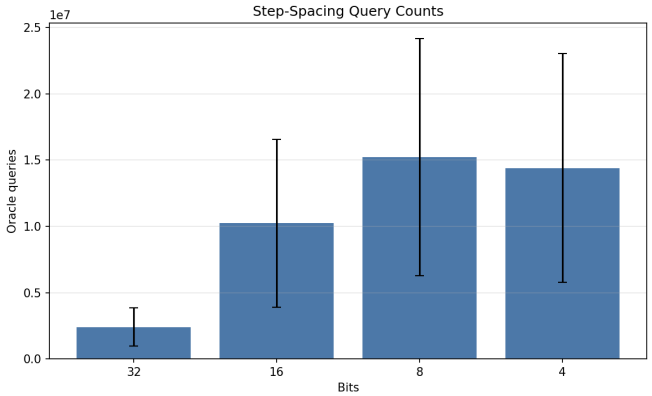

Figure 6: Mean oracle queries issued by step-spacing extraction across rounding precision N, aggregated over both 2-layer architectures and three seeds. Error bars show min/max across cells. Despite the query budget increasing as N decreases, success worsens.

5.6 Preliminary two-hidden-layer result

We also ran a preliminary step-spacing extension on the two-hidden-layer architecture `10-15-15-1` for seed 42. This is not part of the main aggregate because it is a single seed, but the result is useful: the extracted model achieves `rel_loss` 0.117, 0.122, and 0.114 at $N = 32$, $N = 16$, and $N = 8$, respectively, before failing at $N = 4$ due to too few jump candidates. This result is promising, as it shows that we can scale our attack for deeper networks as well.

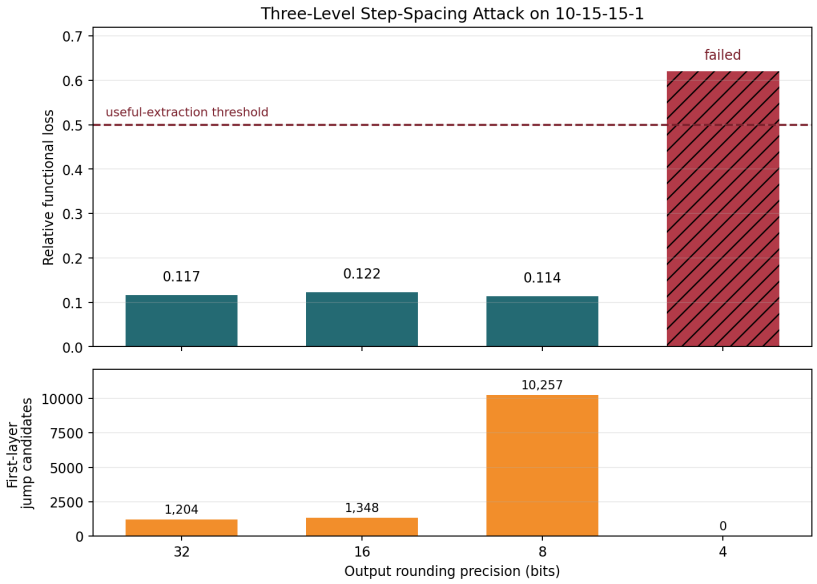

Figure 7: Preliminary two-hidden-layer step-spacing result on `10-15-15-1`, seed 42. The attack remains below the useful-extraction threshold at $N = 32$, $N = 16$, and $N = 8$, with relative functional loss around 10–12%; at $N = 4$ it fails to collect enough jump candidates.

5.7 Where the attack fails

Our slope estimator needs each probe window to span many integer output levels *and* stay inside one ReLU polytope. As the rounding step grows, the window must grow to accumulate enough levels, but it cannot exceed the typical polytope diameter without straddling a ReLU boundary. At $N \leq 16$ for these networks the two requirements cannot be satisfied simultaneously: most candidate windows are rejected by the linearity test, and the surviving ones give slope estimates too noisy to cluster cleanly into weight rows.

6 Discussion

The defender's safe regime. Pulling Sections 3 and 5 together: against an adaptive attacker the defender must operate at $N \leq 16$, where the utility cost of output rounding rises from $\sim 10^{-10}$ (below `float64` noise) to $\sim 5 \times 10^{-6}$ (still ~ 5 ppm relative error – small in absolute terms, but no longer free). For applications that tolerate 10^{-3} output error the defense remains practical at $N = 8$, where neither attack succeeds and no parameter choice we found rescues either. The narrative is therefore not "output rounding fails" but "output rounding works at one setting tighter than naive analysis suggests."

What is and isn't novel. The general possibility that a rounding-aware attacker could adapt is anticipated in Carlini et al. [2]'s discussion. Our concrete contribution is to (a) empirically confirm that the obvious deployment of output rounding does break the original attack, (b) design and implement an adapted attack with first-order macroscopic primitives in place of second-order microscopic ones, and (c) map the resulting security/utility frontier as a function of N and architecture. Step-spacing extraction is not a deep cryptanalytic novelty – it is finite differencing applied to the rounded

oracle – but the implementation choices that make it work in practice (integer-level linearity test, gradient-vector clustering, MITM sign recovery) are non-trivial.

Limitations. We restrict the main aggregate evaluation to single-output, fully connected, 2-layer ReLU networks trained on random data. Section 5.6 gives one promising two-hidden-layer result, but a full 3+ layer evaluation still requires a more careful layer-peeling implementation and multiple seeds. Network sizes top out at `80-40-1` ($\sim$ 3,300 parameters); behaviour on production-scale networks is plausibly similar in regime structure (the threshold depends on polytope geometry, which is set by weight statistics rather than gross parameter count) but our experiments cannot confirm this. We assume the attacker knows the rounding step s; a defender who randomises s per query effectively converts the deterministic defense into a noise injection, which is studied separately.

Practical implication. The takeaway for the defender is that "zero-utility-cost" defenses against cryptanalytic extraction should be evaluated against an adaptive attacker who explicitly targets the defense, not just the original attack. Output rounding to `float32`-like precision ($N \approx 23$) is not safe; output rounding to `int8` ($N \approx 8$) is. The cost of the latter is real but moderate.

Other defenses and attacks. Several nearby defenses are tempting, but each changes the attacker model rather than ending the problem. Stochastic rounding would replace the deterministic step function with a random one: each query returns one of two adjacent levels with probabilities determined by the fractional position of $f(x)/s$. This should make the step-spacing primitive noisier, but repeated queries at the same point can estimate the underlying expectation, so the right question becomes a query-complexity question rather than a correctness question. Random additive output noise has the same flavour: if the noise is independent and zero-mean, averaging attacks are available; if not the defense may damage benign users in ways that are harder to certify. Another promising direction is to combine deterministic output rounding with rate limits or query auditing, because our adapted attack is visibly query-heavy (Figure 6).

7 Conclusion

We studied the natural defense of output rounding at inference against the Carlini–Jagielski–Mironov cryptanalytic extraction attack, and found that while it eliminates the original attack at every bit-width we tested, it succumbs to a different adapted attack (step-spacing extraction) at all bit-widths above an architecture-dependent threshold around $N = 16{-}18$. The defender's apparent free-lunch at $N = 32$ is illusory; the true safe operating point is closer to $N = 16$, with non-trivial but still small utility cost. We hope the experimental methodology – designing both a defense and an attack against it within the same study – is useful as a template for evaluating future defenses against extraction.

References

[1] I. A. Canales-Martínez, J. Chavez-Saab, A. Hambitzer, F. Rodríguez-Henríquez, N. Satpute, and A. Shamir. Polynomial time cryptanalytic extraction of neural network models. In *EUROCRYPT*, 2024.

[2] N. Carlini, M. Jagielski, and I. Mironov. Cryptanalytic extraction of neural network models. In *Annual International Cryptology Conference (CRYPTO)*, 2020.

[3] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security Symposium*, 2020.

[4] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer. Thieves on sesame street! model extraction of BERT-based APIs. *International Conference on Learning Representations (ICLR)*, 2020.

[5] R. Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. In *arXiv preprint arXiv:1806.08342*, 2018.

[6] D. Lowd and C. Meek. Adversarial learning. *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, 2005.

[7] S. Milli, L. Schmidt, A. D. Dragan, and M. Hardt. Model reconstruction from model explanations. In *Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019.

[8] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

[9] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Asia Conference on Computer and Communications Security (ASIACCS)*, 2017.

[11] D. Rolnick and K. Kording. Reverse-engineering deep ReLU networks. In *International Conference on Machine Learning (ICML)*, 2020.

[12] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, 2016.