[Re] Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning

A reproducibility report, with extensions to Simeck32/64 and Simon32/64

Yuyang Zhao
University of Waterloo
ECE 699

August 2026; extended November 2026

Abstract

We reproduce Gohr's CRYPTO 2019 result that deep neural networks can outperform classical differential distinguishers against round-reduced Speck32/64. Working on a single 6 GB laptop GPU—against the paper's 11 GB workstation and 300 core-days of precomputation—we tested eighteen distinct claims spanning distinguisher accuracy, the real-differences experiment, the Markov-model baseline, few-shot transfer learning, automated input-difference search, the 9-round key-ranking attack and the 11-round key recovery attack. **All eighteen reproduce.** Five networks retrained from scratch land within 0.005 of the published accuracies despite running on reduced training budgets.

We report one substantive finding beyond confirmation. The supplementary script published to generate the paper's Table 3, `key_rank.py`, contains an off-by-one in its rank statistic that doubles the reported median key rank and halves the reported success rate. The paper's Table 3 is correct; the script does not reproduce it. We establish this three independent ways, including a histogram test against the author's own shipped result arrays which shows those arrays cannot have been produced by the published code path. The defect matters because it errs in the direction of understating the paper's result, so a reader running the shipped code and comparing against Table 3 would wrongly conclude the paper overstates its 9-round attack by a factor of two.

We additionally document nine further defects in the supplementary code, two methodological traps that produced convincing false negatives during our own work, and the compute profile of the reproduction.

1 Introduction

1.1 The paper under study

Gohr [1] trains residual convolutional networks to distinguish ciphertext pairs produced by round-reduced Speck32/64 [4] under a fixed input difference from pairs of unrelated ciphertexts. The result attracted attention because it was the first demonstration of a machine-learning attack competitive with the published state of the art against a modern block cipher, rather than a demonstration on a toy construction.

The paper makes four principal claims:

1. The full difference distribution of Speck32/64 under the input difference `0x0040/0000` can be computed exactly under the Markov assumption for up to eight rounds, yielding the strongest possible *purely differential* distinguisher as a baseline.

2. Neural distinguishers exceed that baseline at five through eight rounds.

3. A Bayesian key-search policy built on those distinguishers reduces the security of 11-round Speck32/64 to roughly 2^{38} operations, against 2^{46} for the best prior attack [8].

4. The networks exploit features of the ciphertext pair distribution that are invisible to any purely differential distinguisher, however much data it is given.

Claim 4 is the scientifically interesting one and, as we discuss in Section 3.2, the one the paper supports most cleanly.

1.2 What we set out to establish

A reproduction of this paper has to answer three separate questions, and conflating them produces a misleading verdict:

- **Do the shipped artifacts behave as described?** Cheap to test, and it verifies the measurement apparatus rather than the science.
- **Do the results survive independent re-derivation?** Retraining the networks from scratch with the paper's recipes, on different hardware, in a different decade's software stack.
- **Do the published claims follow from what the code actually computes?** This is where our substantive finding lies, and it is the question most reproductions never ask.

1.3 Summary of findings

All eighteen tested claims reproduce (Section 3). One claim—the 8-round transition probability $2^{-26.015}$—is *consistent but not confirmable* at the resolution a 10^{10}-trial experiment affords, and we report it as such rather than claiming success.

The 9-round key-rank statistic reproduces only when the ranking convention is corrected; the published script's convention reproduces neither the paper's table nor the author's own shipped data (Section 5).

Two methodological traps cost us significant time and produced results that looked exactly like failed reproductions (Section 6.2). We document them because both generalise well beyond this paper.

2 Scope and Method

2.1 What is reachable, and what is not

The paper's Section 3 computes the complete 2^{32}-entry output difference distribution for each of eight rounds by repeated sparse matrix–vector multiplication, using the Lipmaa–Moriai algorithm [13] for the differential behaviour of modular addition. The author reports roughly 300 core-days and 34 GB of output per round; the accompanying `speck_ddt.cpp` requires approximately 70 GB of RAM. We had 7 GB.

This places the D6, D7 and D8 rows of the paper's Table 2 permanently out of reach for us, and we state that plainly rather than substituting a weaker proxy. Two mitigations apply. First, the D5 row is recoverable by a cheaper route: the companion `speck.cpp` builds a *sparse* three-round difference table and propagates two rounds backwards, which we ran to completion (Section 3.3). Second, the repository ships the author's own key-rank arrays for the DDT-based 9-round attack, allowing the D7 row of Table 3 to be checked against his output without recomputing the table (Section 5.3).

We also did not attempt the 12-round attack (roughly one hour per run at ≈40% success, so 20+ runs for a meaningful figure), and we do not claim to have reproduced the 2^{38} complexity figure, which is an extrapolation from a hypothetical fully SIMD-parallel Speck implementation running at 2^{28} keys/s/core rather than a measurement. We reproduce its measured inputs—the success rate, the wall time, and the data consumed—and observe that the derivation follows.

2.2 Environment

	Paper	This work
GPU	GTX 1080 Ti, 11 GB	GTX 1660 Ti Max-Q, 6 GB
CPU	"a modern CPU"	Ryzen 7 4800HS, 8C/16T
RAM	$\geq$ 70 GB for the DDT stage	7 GB (WSL2)
Stack	Keras 2.1.5 / TensorFlow 1.6	Keras 2.15 / TensorFlow 2.15.1

The supplementary code is Keras 2-era and the shipped model files are Keras 2.1.3 HDF5. Windows has no native TensorFlow GPU support past 2.10, so the work runs under WSL2 on TensorFlow 2.15.1—the last release whose top-level `import keras` resolves to Keras 2. This choice is what allows the upstream scripts to run **unmodified**: `model_from_json` reads the 2.1.3 configurations and `load_weights` reads the 2.1.3 weight files without complaint. We consider preserving the ability to run the original code untouched more valuable than running on a current stack, because it removes an entire class of doubt about whether a discrepancy is ours or the paper's.

2.3 Deviations, and why each is defensible

Deviation	Reason	Effect on conclusions
`jit_compile=True` during training	883 s $\rightarrow$ 131 s per 10^7-sample epoch; without it this GPU is 6$\times$ slower than the paper's	None. XLA changes kernel fusion only; architecture, loss, optimiser, batch size and schedule are the paper's
Reduced epoch budgets for N5, N6, distillation	Wall-clock	Quantified against the author's own shipped learning history
8-round staged training cut from 10^9 to 10^8 samples per stage	Wall-clock	None measurable; N8 reached 0.5135 against the paper's 0.514
Key rank at 200 trials, $n = 64$	Measured throughput of 154k predictions/s makes 1000 trials a 7.6 h job	Widens the success-rate interval to $\pm$6.9
Full DDT not computed	300 core-days, 70 GB RAM	D6–D8 unverified; stated, not worked around

The single deviation that touches computation is XLA compilation. We regard this as methodologically inert—it is a compiler flag, not a change to the learning problem—but we record it because it is the only way the training was made to fit the hardware, and a reader should be able to see every lever we pulled.

2.4 Verification discipline

Every figure reported below traces to a log file retained under `results/logs/` (32 logs). The comparison tables are regenerated from those logs by a script rather than transcribed by hand, which removes copying error as a failure mode. Accuracies are measured on 10^6 freshly generated samples unless stated otherwise, giving a 2σ sampling error of approximately ± 0.001; we quote intervals wherever the comparison is close enough for them to matter.

3 Results

3.1 Distinguisher accuracy (paper Table 2)

The repository ships single-block networks for all four round counts, and its readme warns that the five- and six-round ones will consequently fall slightly short of the paper's deep networks. That is what we observe, and the shortfall is small.

Rounds	Paper D (DDT)	Paper N	Shipped net	Δ vs paper N
5	0.911	0.929	**0.9274**	−0.0016
6	0.758	0.788	**0.7874**	−0.0006
7	0.591	0.616	**0.6152**	−0.0008
8	0.512	0.514	**0.5137**	−0.0003

True positive and true negative rates agree to within the sampling error at every round count (e.g. 5 rounds: TPR 0.9008 against 0.904, TNR 0.9539 against 0.954).

3.2 The real differences experiment (paper Table 4)

This is the paper's cleanest support for claim 4, and it deserves emphasis because its construction is what makes it convincing. Both classes are blinded by XORing a single uniformly random mask into *both* ciphertexts of a pair. Blinding cancels in the difference—$(C_0 \oplus K) \oplus (C_1 \oplus K) = C_0 \oplus C_1$—so the ciphertext-difference distribution is provably identical between the two classes, and every purely differential distinguisher is reduced to guessing.

Rounds	Paper	Ours
5	0.707	**0.7030**
6	0.606	**0.6031**
7	0.551	**0.5517**
8	0.507	**0.5076**

The networks were not retrained for this task. The result reproduces at all four round counts.

Two further pieces of evidence for claim 4 reproduce as well. On the 4.84% of samples where the network and the Markov model disagree, the network matches ground truth 66.5% of the time (paper: 67%), and it never labelled an impossible differential transition as real. And a threshold-independent comparison we add (Section 3.7) shows the advantage is not an artefact of the 0.5 decision cutoff.

3.3 The Markov-model baseline (paper Section 3)

Compiled unmodified from `cpp/speck.cpp` and run to completion (13.5 h wall clock, 10 threads):

The Section 3.2 key-search distinguisher is worth singling out. It is a *general* differential distinguisher—it counts how many keys decrypt the observed pair to the input difference,

Quantity	Paper	Ours
D5 accuracy	0.911	**0.91118**
D5 true positive rate	0.877	**0.87918**
D5 true negative rate	0.947	**0.94318**
5-round distinguisher by full key search	≈0.95 (9456/10000)	**0.9413**
. . . with lower two rounds estimated	"no significant loss"	**0.9422**
Named impossible pair: keys decrypting to Δ	0	**0**
Second named pair: decryption probability	$2^{-35.3}$	$\mathbf{2^{-35.333}}$

using Dinur's two-round attack [8]—and at 0.9413 it exceeds the perfect purely differential distinguisher's 0.911. That three-point gap, buried in a subsection, is the paper's first quantitative evidence that the ciphertext pair carries information the difference discards.

The one claim we cannot confirm. The paper states that the highest-probability 8-round transition `0x0040/0000` → `0x0280/0080` occurs with probability $2^{-26.015}$. A 10^{10}-trial experiment measured $2^{-26.2651}$. At that probability the expected count is roughly 137 events, so the standard error on the log-probability is about 0.125 and the discrepancy is -1.93σ. This is neither a confirmation nor a contradiction: it is a measurement whose resolution is inadequate to the claim. Confirming it to ± 0.05 would require roughly an order of magnitude more trials. We report it as *consistent* and decline to score it as reproduced.

3.4 Key recovery

9-round key ranking (paper Table 3). Reported in Section 5, because the result depends on resolving a defect in the scoring script.

11-round key recovery (paper Section 4.5). The full attack: a 7-round neural distinguisher extended by a two-round prepended differential passing with probability ≈1/64, plus one free round from the observation that Speck's first subkey addition follows its first nonlinearity, plus Bayesian search over the final two subkeys guided by a precomputed wrong-key response profile, with plaintext structures built from probabilistic neutral bits [5] and allocated across structures by an upper-confidence-bound bandit policy.

Success is the paper's criterion: last subkey exactly correct and penultimate subkey within Hamming distance 2.

	Paper	Ours
success rate	52.1% (1000 trials)	**56.6%** ± **10.9** (83 trials)
data per run	$2^{13.2}$ pairs	$2^{13.16}$

The interval brackets the paper's figure comfortably. An earlier 25-trial measurement gave 44.0% ± 19.9, which also covered it but too loosely to be worth much; extending to 83 trials roughly halved the interval and moved the point estimate to the other side of the paper's value. That drift is itself a caution about reading anything into a 25-trial success rate.

A more informative decomposition: 72 of 83 trials (86.7%) were *solvable in principle*, meaning at least one ciphertext structure contained a pair passing the initial two-round differential, and conditional on solvability the attack succeeded in 47 of 72 (65.3%). Every failure was a trial in which the sampled data contained no usable structure at all—which is exactly the behaviour the paper describes, and the reason its data figure of $2^{14.5}$ chosen plaintexts assumes re-querying after failure. In all 47 successes the last subkey was recovered exactly.

3.5 Networks retrained from scratch

Five networks trained fresh using the paper's architecture, loss, optimiser and learning-rate schedules:

Network	Recipe	Budget used	Ours	Paper
N5, depth 10	basic pipeline	38 of 200 ep	0.9278	0.929
N6, depth 10	basic pipeline	5 of 200 ep	**0.7833 ± 0.0008**	0.788
N7, depth 1	trained on KeyAveraging output	300 of 300 ep	**0.6125 ± 0.0010**	0.616
N8, depth 1	staged pre-training	10^8/stage of 10^9	**0.5135**	0.514
N5 distilled, depth 1	trained on our N5's outputs	5 of 30 ep	**0.9252 ± 0.0005**	0.927

Two observations. First, the distillation result is best read as a *gap* rather than an absolute: the paper loses 0.002 going from its ten-block teacher to a single-block student, and we lose 0.0026 from ours, so the compression claim reproduces even though both of our networks are individually short of the paper's. Second, N8 is the surprise. The staged recipe is the only route by which the paper obtains an 8-round distinguisher at all—direct training "fails, i.e. the model does not learn to approximate any useful function"—yet it survived a tenfold reduction in every stage.

For N5 we compared against the author's own shipped learning history epoch by epoch:

Epoch	Ours	Author's run
10	0.9269	0.9270
20	0.9275	0.9275
30	0.9278	0.9278
199	—	0.9291

The curves are indistinguishable over the range we ran. This also quantifies the value of the full schedule: epochs 30–200 are worth roughly +0.0013, which is useful information for anyone budgeting a repeat.

3.6 The auxiliary algorithms

Few-shot transfer (Algorithm 2, Fig. 2). An auxiliary network trained for one epoch on 3-round Speck with a *randomly chosen* input difference, its penultimate layer used as a feature map, and ridge regression fitted on between 1 and 50 examples of the target distribution. From a single training example we obtain 0.6165 ± 0.0212 at five rounds and 0.5127 ± 0.0041 at six—the paper's specific claim is that accuracy exceeds guessing at 2σ even with one example, and both curves clear it decisively.

Automated input-difference search (Algorithm 3). Ten random restarts of the greedy optimiser, scoring candidate differences by the few-shot accuracy of the resulting distinguisher:

Outcome	Count
converged on `0x0040/0000` exactly	**3 of 10**
converged elsewhere, extends to 5–6 rounds	3
converged on a high-weight difference, collapses by round 4	4

The three runs that found `0x0040/0000` were among the four that extended furthest, to six rounds. The paper claims the difference "is usually found within a few random restarts" and that six rounds can then be distinguished with fairly high gain; both hold. The failures are instructive rather than noise: they settle on high-Hamming-weight differences that separate 3-round Speck almost perfectly and then collapse, because the optimiser's objective scores 3-round separability and cannot see propagation.

Wrong-key response profile (Fig. 3). Independently regenerated over all 2^{16} key differences at 3000 pairs each. Correlation with the author's shipped profile is 0.771, limited by sampling noise; μ at $\delta = 0$ is 0.5378 against 0.5322, the minimum is 0.4454 against 0.4469, and every structural feature reproduces—in particular the four-fold repeat with secondary maxima at $\delta = 16384, 32768, 49152$. That structure is not decorative: it is why the attack's Bayesian search optimises only over the low 14 bits.

KeyAveraging (Algorithm 1). Measured two ways, which agree to four decimals:

Variant	Samples	Trial keys	Accuracy
as upstream samples it	10^6	1000 random	0.6240
as Algorithm 1 states it	10^4	all 2^{16}	0.6240 ± 0.0097

Paper: 0.627. The 1000-key approximation upstream uses for building training data is statistically indistinguishable from the exhaustive version at 65× less cost.

3.7 Analyses beyond the paper

Four results computable from data already collected, which sharpen claims the paper makes only loosely.

The neural advantage is threshold-independent. The paper compares network and Markov model only at the 0.5 cutoff. On identical samples at five rounds, ROC AUC is **0.9738** for the network against **0.9665** for the model. This matters: a purely differential distinguisher exploits its information optimally at the stated threshold, so equal AUCs would have implied the paper's advantage was a thresholding artefact rather than an information gap.

Overfitting is mild, as claimed. Final-epoch train/validation gaps are +0.36% (N5), +0.06% (N6) and −2.4% (distilled). N7's larger relative gap of +13.8% is not a counterexample: trained on soft targets, its absolute loss is an order of magnitude smaller and the gap is +0.0006 absolute.

The wrong-key σ profile shares the μ profile's shape , as Section 4.5 asserts without showing. Correlation between our μ and σ profiles is 0.804, and both place their per-quadrant maximum at exactly $\delta = 0, 16384, 32768, 49152$.

4 Extending the pipeline to Simeck32/64 and Simon32/64

4.1 Motivation and setup

Having reproduced the paper on Speck32/64, we asked whether its results are a property of the technique or of the target. Simeck32/64 [15] and Simon32/64 [4] make natural controls. Both have the 32-bit block and 64-bit key of Speck32/64, so the whole training pipeline can be held fixed while only the round function changes. Simeck was designed at Waterloo and deliberately

combines Simon's AND-based round function with a key schedule that, like Speck's, reuses the round function.

The work was carried out by Claude in agent sessions on the author's machines, after the Speck reproduction of Section 3.

All experiments use round-reduced ciphers under freshly drawn random keys.

Gohr's pipeline is used unmodified: the same residual network, mean squared error loss with L_2 regularization, Adam, the cyclic learning-rate schedule, 10^7-sample training sets and the same real-versus-random task. Only the cipher module `speck.py` is replaced by `simeck.py` or `simon.py`. A difference in the number of rounds reached is therefore a property of the cipher under this method at our training budget, which is well below the paper's (Section 4.10).

Conventions. As in Section 3, n counts ciphertext *pairs*, never plaintexts. Accuracies are measured on balanced test sets and carry 2σ binomial error bars: ± 0.0010 at 10^6 test pairs and ± 0.0003 at 10^7. Key ranks are heavy-tailed, so mean ranks carry 95% bootstrap percentile intervals (10^4 resamples) and proportions over trials carry 95% Clopper–Pearson intervals. A subkey guess is scored by summing the distinguisher's pre-sigmoid logit over the pairs. Ties are broken uniformly with the exclusive upper bound of Section 5, and rank 0 means the true subkey scored highest.

Provenance. Each table names the record it comes from, relative to `ece699_claude/reproduction/` in the repository. These records are the two working reports `simeck/SIMECK_REPORT.md` and `simon/SIMON_REPORT.md`, the per-run logs under `logs/`, and the machine-readable records under `history/`. Values labelled *paper* are quoted from Gohr [1] for Speck32/64 and were not re-derived here. Key-rank statistics were recomputed from `simeck/history/key_rank.jsonl` with `summarise_key_rank.py` while this section was drafted. Their bootstrap intervals differ from those printed in the working reports by resampling noise only.

Hardware. The first Simeck runs (the difference-spread measurement, the basic-pipeline sweep and its budget check of Section 4.4.1) ran on an earlier host, GU605CM, a laptop with an RTX 5060 Laptop GPU (8 GB) and an Intel Core Ultra 7 255H (NVIDIA driver 621.01 and CUDA 13.5 when the specifications were recorded on 2026-10-27). The runs up to 2026-10-23, including every Simon run before the diagnostics of Section 4.7.5, ran on a laptop with an RTX 3050 Laptop GPU (4 GB) and TensorFlow 2.15.1 under WSL. The work then moved to a third laptop (GTX 1660 Ti Max-Q, 6 GB, same software). Two saved networks re-evaluated there scored 0.54636 and 0.51282 on 10^7 fresh pairs, against 0.54636 and 0.51274 on the RTX 3050 (`simeck/history/eval.jsonl`; `MOVE_TO_NEW_MACHINE.md` §6). Everything dated 2026-10-24 to 2026-10-31 ran on the GTX 1660 Ti: the first sixteen three-stage trials, the later 13-round two-subkey trials, the second-seed and fourth-difference staged runs, the staged 13-round run and the Simon wrong-key profiles and word-swap tests. From 2026-10-31 all runs are back on GU605CM, with the same TensorFlow version rebuilt under WSL; the same two networks scored 0.54632 and 0.51281 there (`MOVE_TO_NEW_MACHINE.md` §7). That covers the twelve later three-stage trials, the wrong-key profile of the 9-round net, the eight further 10-round Simeck seeds and the staged nets trained from them, the Simon rotation variants, the second Simon seed and the evaluations of published networks (Sections 4.6 and 4.7.4). The master-key step ran on a cloud CPU. Where trials from several hosts are pooled, the table says so.

4.2 The two ciphers

Both ciphers are Feistel networks on two 16-bit words. One round under subkey k_i is

$$L_{i+1} = f(L_i) \oplus R_i \oplus k_i, \qquad R_{i+1} = L_i, \tag{1}$$

where, for Simeck and Simon respectively,

$$f(x) = (x \wedge (x \lll 5)) \oplus (x \lll 1), \tag{2}$$

$$f(x) = ((x \lll 1) \wedge (x \lll 8)) \oplus (x \lll 2). \tag{3}$$

Simon32/64 has a linear key schedule with $m = 4$ key words and the period-62 constant sequence z_0 [4]. Simeck's key schedule reuses f with a round constant in place of a subkey,

$$k_{i+1} = t_i, \qquad t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_0)_i, \tag{4}$$

with $C = 2^{16} - 4$ [15]. Both ciphers have 32 rounds at this size, against 22 for Speck32/64.

Before any training, both implementations reproduce the official test vectors (key `1918 1110 0908 0100`, plaintext `6565 6877` gives `770d 2c76` for Simeck and `c69b e9bb` for Simon), and decryption round-trips.

A near miss. The first Simeck implementation failed its test vector. A search over 320 plausible convention variants found no match. Reading the specification [15] resolved it: Simeck's z_0 is a period-31 m-sequence generated by $X^5 + X^2 + 1$ from the all-ones state, and the implementation had used Simon's period-62 sequence instead. The corrected sequence is generated from its recurrence rather than transcribed, and the reference implementation [16] uses the same feedback. This error would not have shown up in any accuracy figure. The paper states that its distinguishers do not exploit the key schedule (its 11-round attack performs identically under independent random subkeys), so a Simeck with Simon's constants would most likely have trained to similar accuracy and produced plausible numbers for a cipher that does not exist. Only the test vector protects against this, and it has to be checked before training.

4.3 Input difference and a free first round

All Simeck and Simon distinguishers here use the input difference $(\Delta L, \Delta R) = (\texttt{0x0000}, \texttt{0x0040})$, taken from the Simeck neural-distinguisher literature [19, 17]. With $\Delta L = 0$ the function f contributes nothing, and the difference becomes (`0x0040`, `0x0000`) after one round with probability 1. This is the counterpart of Speck's `0x0040/0x0000` $\rightarrow$ `0x8000/0x8000`.

Both ciphers add the round key *after* f, so in round one f acts directly on the left plaintext word, and its AND conditions can be met by choosing plaintexts. For Simeck and input difference (`0x0040`, `0x0080`),

$$\Delta f(L) = \texttt{0x0080} \oplus (L_1 \ll 6) \oplus (L_{11} \ll 11), \tag{5}$$

so plaintexts with bits 1 and 11 of the left word cleared reach (`0x0000`, `0x0040`) after one round with probability 1, for every key. For Simon the input difference is (`0x0040`, `0x0100`) and the conditions fall on bits 13 and 15. In every evaluation with the conditions the transition held for every sample (1.0000 over 10^7). Without them it held with probability between 0.2499 and 0.2502 across the seven unconditioned evaluations, as expected for two independent bit conditions (`simeck/history/prepend.jsonl`, `simon/history/prepend.jsonl`).

Any r-round distinguisher for (`0x0000`, `0x0040`) is then an $(r{+}1)$-round chosen-plaintext distinguisher on the 2^{30} plaintexts that satisfy the two conditions. Both classes draw the first plaintext from that subspace, and the random class's second plaintext is uniform. This changes the round count, not the method. In Gohr's attacks Speck's first round is also free, but it needs no plaintext conditions, so a round count that includes the conditioned round is one higher than the Speck-equivalent count. We report both counts and label which is which.

The free first round is classical. Abed, List, Lucks and Wenzel [2] add Simon's first round for free for the same reason, that the subkey is injected after f, and they do it by choosing the plaintext words, which needs no bit conditions: for any left words L and $L' = L \oplus \texttt{0x0040}$, setting $R' = R \oplus f(L) \oplus f(L')$ gives the difference (`0x0000`, `0x0040`) after round one with

probability 1, for every key, over the full 2^{32} plaintext space, at the cost of a plaintext difference that varies from pair to pair. Our two-bit conditioning, which keeps a fixed plaintext difference, is one implementation of the same idea, not its price. Published neural key-recovery attacks on Simon32/64 and Simeck32/64 go further and prepend one to three rounds of a classical differential with neutral bits [20, 17, 18].

4.4 Simeck32/64: distinguishers

4.4.1 Basic pipeline

Table 1 gives the basic pipeline on Simeck32/64. Ten rounds still gives a genuine distinguisher, with both TPR and TNR away from the degenerate values. At 11 and 12 rounds the networks collapse to predicting a single class.

Table 1: Simeck32/64, basic pipeline: depth-1 network, 10^7 training samples, 10 epochs (5% of the paper's 200), input difference (`0x0000`, `0x0040`), accuracy on 10^6 fresh pairs.

Rounds	Accuracy	TPR	TNR
8	0.8636 ± 0.0007	0.9331	0.7941
9	0.6758 ± 0.0009	0.6975	0.6541
10	0.5384 ± 0.0010	0.5043	0.5724
11	0.4998 ± 0.0010	0.0000	1.0000
12	0.4992 ± 0.0010	0.8035	0.1953

Measured; `simeck/logs/simeck_{8..12}r.log`, SIMECK_REPORT §4.1.

At eight rounds, where the paper needs staged training to reach 0.514 on Speck32/64, the basic recipe gives 0.8636 on Simeck32/64. Table 2 lines the two ciphers up by distinguisher strength: Simeck32/64 is distinguishable about two rounds further than Speck32/64 across the whole range where both learn. The Speck rows are the paper's best networks, not all basic-pipeline ones: N_7 was trained on targets from a key-search teacher. The paper gives no basic-pipeline 7-round accuracy, so that row may favour Speck slightly; a weaker Speck figure would only widen the gap. Simeck32/64 has 32 rounds against Speck32/64's 22, so this compares how far differential-neural distinguishers reach, not the security margins of the two designs.

Table 2: Speck32/64 (paper, Table 2 of [1]) against Simeck32/64 (measured, Table 1), aligned by accuracy. The Speck N_6 is from the basic pipeline, N_7 was trained on KeyAveraging targets, and N_8 is staged (paper §4.4); every Simeck figure here is from the basic pipeline.

Speck32/64, paper		Simeck32/64, measured	
Rounds	N_r	Rounds	Accuracy
6	0.788	8	0.8636
7	0.616	9	0.6758
8	0.514	10	0.5384

Difference diffusion accounts for part of the gap. Table 3 counts the distinct output differences, and their entropy, after r rounds under each cipher's input difference. By entropy, Simeck lags Speck by about one round: at round 4 Speck has spread over 304,514 differences where Simeck occupies 3,672. We did not measure how much of the gap this lag explains. A second, structural candidate is that the last round of Simeck and Simon can be undone without the key (Section 4.5.1), so a distinguisher on r-round pairs effectively sees the $(r-1)$-round difference; for Speck, undoing the last round without the key recovers only part of the previous state. That alone could account for about one of the two rounds. Both are candidates, not measured explanations. The Simon columns are discussed in Section 4.7.

Table 3: Spread of the output difference after r rounds: number of distinct differences and their entropy in bits, 10^6 samples per cell, under `0x0040/0x0000` for Speck and (`0x0000`, `0x0040`) for Simeck and Simon. Estimates saturate near $\log_2 10^6 \approx 19.93$ bits, so only cells well below that are informative. Measured; SIMECK_REPORT §3, SIMON_REPORT §2.

	Speck32/64		Simeck32/64		Simon32/64	
Rounds	distinct	entropy	distinct	entropy	distinct	entropy
2	8	1.98	4	2.00	4	2.00
3	23,509	7.75	68	5.75	68	5.75
4	304,514	15.37	3,672	10.60	2,448	10.41
5	853,057	19.53	247,715	16.44	180,478	16.11

The 11-round wall is not a budget artefact. The sweep ran at 5% of the paper's training budget, and a reduced budget produced a false negative earlier in this project: a seven-round Speck network distilled by KeyAveraging at a reduced budget scored 0.5069 and looked like a failure of the method, then reached 0.6125 on the paper's schedule (`results/logs/train_N7_teacher{,300}.log`). Rounds 10 and 11 were therefore re-run at 100 epochs, half the paper's budget (Table 4). Ten times the training changes neither side of the boundary. The 10-round run sits at chance for about nine epochs, escapes, and is essentially converged by epoch 20 (0.5330; 0.5368 at epoch 100). The 11-round run stays within $[0.4999, 0.5015]$ for all 100 epochs, at the validation loss of a constant prediction of 0.5.

Table 4: Simeck32/64, effect of a tenfold training budget. Accuracy on 10^6 pairs.

Rounds	10 epochs	100 epochs
10	0.5384 ± 0.0010	0.5362 ± 0.0010
11	0.4998 ± 0.0010	0.5003 ± 0.0010

Measured; `simeck/history/simeck_1{0,1}r_d1{,_long}.json`, SIMECK_REPORT §4.3.

Escaping the constant solution is stochastic even at 10 rounds. Of the first six 10-round runs, five escaped and landed between 0.5317 and 0.5462, and one never left the plateau (0.50013 on 10^7 pairs, every sample labelled positive). Eight more 20-epoch runs with the same settings on GU605CM all escaped, at 0.52153 to 0.54653 on 10^7 pairs (`simeck/history/eval.jsonl`), so 13 of 14 runs on record escaped. The run-to-run spread at 10 rounds is therefore about 0.02, much larger than the error bars suggest, and the 0.5384/0.5362 budget comparison is inside it. At 11 rounds, none of 10 direct runs escaped: the two of Table 4 plus eight more at 20 epochs, whose best validation accuracy was 0.5008 (`simeck/logs/simeck_11r_direct_e20_{1..8}.log`). The per-run escape probability at 11 rounds is below 0.31 (upper end of the two-sided 95% Clopper–Pearson interval for 0 of 10, the convention used throughout; the one-sided 95% bound is 0.26), against 13 of 14 at 10 rounds. As the paper reports for Speck at eight rounds, direct training one round past the basic pipeline's limit learns nothing. The next two subsections show that the signal exists anyway.

4.4.2 Staged pre-training

Gohr's eight-round Speck recipe, moved three rounds up, is in Table 5.

Before any 11-round training, the 10-round seed already distinguishes 11-round data (0.50615, $z \approx 39$) and 12-round data (0.50164, $z \approx 10$). Staging roughly doubles the 11-round advantage, to 0.51274 ± 0.00032 (Table 6). Most of the gain arrives in stage 1 and in the first 10^8 samples of stage 2; the two learning-rate drops add about $+0.0015$ together, within the 10^6-pair error bar at each step, as in our Speck N_8 runs. The result is comparable in strength to the paper's

Table 5: Staged pre-training: the paper's Speck recipe and its Simeck transposition.

	Speck, 8 rounds (paper)	Simeck, 11 rounds
Seed	7-round net	10-round net (20 epochs, 0.5462)
Stage 1	5 rounds under `0x8000/840a`, 10 epochs, 10^7 samples, lr 10^{-4}	8 rounds under (`0x0140`, `0x0080`), same schedule
Stage 2	10^9 fresh 8-round samples, once each, at lr 10^{-4}, 10^{-5} and 10^{-6}, batch 10^4	same, at 11 rounds

Speck column as transcribed in SIMECK_REPORT §4.4 from the paper; Simeck column as run (`simeck/train_staged_simeck.py`).

Table 6: Simeck32/64, staged training at 11 rounds. "Seed" is the 10-round net before any 11-round training. Accuracy on 10^7 fresh pairs (± 0.0003). Paper: Gohr's staged Speck N_8 scores 0.514.

Net	10 rounds	11 rounds	12 rounds
Seed	0.54636	0.50615	0.50164
Staged	0.53573	**0.51274**	0.50215

Measured; `simeck/history/eval.jsonl`, SIMECK_REPORT §4.4.

staged Speck N_8 (0.514): staging moves the Simeck boundary from 10 to 11 rounds, as it moves Speck's from 7 to 8. The staged net is biased toward "random" (TPR 0.451, TNR 0.574 at 11 rounds) and lost 0.011 at 10 rounds relative to its seed, which is expected since it saw only 11-round data.

The stage-1 difference is a choice. Gohr's Speck recipe uses the 3-round output difference `0x8000/0x840a`; we did not measure whether it has close competitors. For Simeck, four differences tie near 2^{-4} ((`0x0140`, `0x0080`) 0.0629, (`0x1140`, `0x0080`) 0.0626, (`0x11c0`, `0x0080`) 0.0623, (`0x01c0`, `0x0080`) 0.0623, over 10^6 samples). Three were run from the same seed and gave 0.51274, 0.51280 and 0.51264 at 11 rounds on 10^7 pairs, a spread of 0.00016, inside one error bar (`simeck/history/eval.jsonl`).

The seed matters more than the difference. The fourth difference and a second seed net were run later on the GTX 1660 Ti with everything else identical. The second seed is another 20-epoch 10-round net, 0.53193 at 10 rounds against the first seed's 0.54636. All five staged nets were then re-evaluated in one pass on fresh samples (Table 7). The four differences from the first seed span 0.00079 at 11 rounds ($z \approx 3.5$ for the extreme pair). The second seed gives 0.50778, at least 0.0047 below every first-seed net ($z \approx 21$). It is also strongly biased toward "random" (TPR 0.309, TNR 0.707 at 11 rounds). The weaker seed stays weaker after staging.

Three more seeds were staged later on GU605CM with the first stage-1 difference: two of the eight new 10-round nets (seed d, 0.54023, and seed e, 0.53805, at 10 rounds on 10^7 pairs) and the one 10-round net that never left the plateau (seed c, 0.50013). Seeds d and e give 0.50918 and 0.51178 at 11 rounds, between seed b and the four seed-a nets. Seed c stays at chance (0.49980, and 0.49971 at 12 rounds with the free round): staging does not rescue a seed that learned nothing. The seed's 10-round accuracy does not fully predict the staged result either: seed d is the stronger seed (0.54023 against 0.53805) but gives the weaker staged net (0.50918 against 0.51178, $z \approx 12$). The weakest seed, b, still gives the weakest staged net and the strongest, a, the strongest. The headline 0.5127 is therefore the best of four working seeds; across them the staged 11-round accuracy ranges from 0.5078 to 0.5133.

Table 7: Staged 11-round Simeck nets: four stage-1 differences from seed a, one each from seeds b to e. Accuracy at 11 and 12 rounds, and at 12 rounds with the free first round ("12f"). 10^7 fresh pairs per cell, ± 0.00032 (2σ). Seeds a and b were evaluated in one pass on the GTX 1660 Ti, seeds c to e on GU605CM.

Seed	Stage-1 diff.	11	12	12f
a	(0x0140, 0x0080)	0.51277	0.50165	0.51272
a	(0x1140, 0x0080)	0.51276	0.50191	0.51302
a	(0x01c0, 0x0080)	0.51250	0.50189	0.51244
a	(0x11c0, 0x0080)	**0.51329**	0.50209	0.51328
b	(0x0140, 0x0080)	0.50778	0.50133	0.50796
c[a]	(0x0140, 0x0080)	0.49980	0.50012	0.49971
d	(0x0140, 0x0080)	0.50918	0.50172	0.50916
e	(0x0140, 0x0080)	0.51178	0.50154	0.51166

[a] Seed c is the 10-round run that never left the plateau. Measured; `simeck/history/eval.jsonl`, `simeck/history/prepend.jsonl`, `simeck/logs/eval_staged11_q4.log`, `simeck/logs/eval_staged11_q7.log`, SIMECK_REPORT §4.4, §4.14.

Staging once more fails. The same recipe one round further (seed: the staged 11-round net; stage 1 at 9 rounds; 3×10^9 12-round samples) does not work. Stage 1 reached only 0.5047 at 9 rounds, against 0.5247 at 8 rounds in the 11-round run. Stage 2 never left chance (0.4994–0.5005 at every 10^8-sample checkpoint), and the final net scores 0.50033 ± 0.00032 at 12 rounds, below its seed's 0.50215 (`simeck/history/simeck_12r_staged.jsonl`, SIMECK_REPORT §4.9). Training at 12 rounds erased the faint 12-round signal the seed had. Staging to 13 rounds from the staged 11-round net (stage 1 at 10 rounds, then 3×10^9 13-round samples; GTX 1660 Ti) fails the same way: the final net labels every input "random" (TPR 0, TNR 1 at 12, 13 and 14 rounds; `simeck/history/simeck_13r_staged.jsonl`).

4.4.3 The free first round

Table 8 prepends the conditioned round of Section 4.3. The 10-round net, unchanged, becomes an 11-round distinguisher at its full 10-round accuracy, stronger than the staged 11-round net. The staged net covers 12 rounds at 0.5129, and 13 rounds at 0.5018 when applied one round deeper than it was trained ($z \approx 11$). That is far too weak to use on its own, but enough for key ranking with 2^{14} pairs (Section 4.6). Without the conditioned round, the same nets at the same depths score 0.54636, 0.51274 and 0.50215 (Table 6). On 10^8 pairs the staged net applied at 13 rounds plus the conditioned round separates *14*-round data at 0.50064 ± 0.00010 ($z \approx 13$); without the conditions the same 14-round figure is 0.50011 ($z \approx 2$), and the staged net alone at 13 rounds scores 0.50052 ($z \approx 10$). These are statistical signals, not usable distinguishers.

4.5 Simeck32/64: differential baselines and difference search

4.5.1 Purely differential baselines

The exact difference distribution table is out of reach for the same reason as for Speck (Section 3.3). We therefore built concrete purely differential distinguishers. They are all lower bounds on the exact-DDT distinguisher D_r, and beating them does not establish what the paper establishes for Speck, that the networks beat D_r itself.

The first sampled tables use 10^9 real output differences per round count and classify a pair as real when its difference is more likely than uniform. They are tested on 10^6 real and 10^6 random fresh pairs, where the random pairs are encrypted rather than assumed uniform. One table uses the full 32-bit difference (with $10^9 \ll 2^{32}$ samples, this is the seen-at-least-once rule); the others project the difference onto 16 or 24 bits, including $\Delta L \oplus (\Delta R \lll 1)$. A depth-1 network trained on the difference alone is a further purely differential distinguisher.

Table 8: Simeck32/64 with the conditioned free first round. "Seed" is the 10-round net and "staged" the staged 11-round net; the number in parentheses is the depth the net is applied at, and the round count includes the free round. "Cond." marks whether the plaintext conditions are imposed. Accuracy on 10^7 pairs (± 0.0003).

Net (depth)	Rounds	Cond.	Accuracy
Seed (10)	11	yes	**0.54629**
Seed (10)	11	no	0.51306
Staged (11)	12	yes	**0.51293**
Staged (11)	12	no	0.50364
Staged (12)	13	yes	0.50180
Staged (13)	14	yes	0.50064[a]
Staged (13)	14	no	0.50011[a]

[a] 10^8 pairs (± 0.0001), GTX 1660 Ti. Measured; `simeck/history/prepend.jsonl`, SIMECK_REPORT §4.5, §4.9.

These tables miss a free round. Simeck adds the round key after f, and f has no key, so from a ciphertext pair (L, R), (L', R') the difference one round earlier is

$$\Delta L_{r-1} = R \oplus R', \tag{6}$$

$$\Delta R_{r-1} = L \oplus L' \oplus f(R) \oplus f(R'), \tag{7}$$

exactly, for every key. This is why the literature compares an $(r+1)$-round network with the r-round DDT [17]. A purely differential distinguisher may use it, and the first tables did not. An in-project review built "peeled" tables on this difference: a seen-at-least-once table of the full $(r-1)$-round difference from 4×10^8 real pairs, and count tables on 24- and 16-bit projections of it, with the projection chosen on a validation set and scored on an independent 10^7 pairs per class (`peel_baseline/peel_check.py`, `peel_proj.py`; CPU, about three minutes per round count). A smaller independent rerun (10^8 table pairs) gives the same ordering. Table 9 gives all of them.

Table 9: Simeck32/64: purely differential distinguishers against the full-pair networks. "Table" is the best of seven sampled tables on the ciphertext difference (± 0.0007); "peeled" is the best table on the keylessly peeled $(r-1)$-round difference (± 0.0003 or better). The difference-only net is trained at 8 rounds (10^7 pairs, ± 0.0003).

Rounds	Table	Peeled	Diff.-only net	Full-pair net
6	0.9880	—	—	—
7	0.8583	—	—	—
8	0.6855	0.7977	0.70532	0.8636
9	0.5613	**0.6831**[b]	0.54034	0.6758
10	0.5118	**0.5567**[b]	0.50955	0.5384[c]
11	0.5010	0.5090[b]	0.50209	0.51274[a]

[a] Staged net (Table 6); the other full-pair figures are basic-pipeline nets from Table 1. [b] 24-bit projection. [c] The best of 14 ten-round runs gives 0.5465 (10^7 pairs). Bold: the peeled table beats every network we have at that depth. Measured; `simeck/history/ddt_baseline.jsonl`, `simon/history/diffonly.jsonl`, `peel_baseline/peel_results.jsonl`, `peel_proj_results.jsonl`, SIMECK_REPORT §4.6, §4.9.

Against the tables on the ciphertext difference the networks lead by a wide margin from 8 rounds on, but those tables are the wrong reference. A peeled table built in a few CPU-minutes **beats every network we have at 9 and 10 rounds** (0.6831 against 0.6758; 0.5567 against the best of 14 ten-round runs, 0.5465). The networks lead it at 8 rounds (0.8636 against 0.7977) and the staged net at 11 rounds (0.51274 against 0.50904, $z \approx 19$), but two of the four staged seeds (Table 7) are level with or below it. The published exact single-pair DDT is above every

network (Section 4.9). So for Simeck we have no evidence that the networks use anything beyond the (peeled) difference distribution, and at 9 and 10 rounds they do not even reach a sampled approximation of it.

The difference-only net sees only the ciphertext difference, so it cannot peel a round. It reaches 0.7053 at 8 rounds, above every unpeeled sampled table, and at 11 rounds a purely differential signal (0.50209, $z \approx 13$) where the best unpeeled table is barely above chance (0.5010, $z \approx 3$). The 24-bit projections beat the full 32-bit table from 7 rounds on, because 10^9 samples are too few to estimate a 2^{32}-cell table.

4.5.2 Input-difference search

The port of the paper's Algorithm 3 (`simeck/diff_search_simeck.py`) trains a one-epoch auxiliary 3-round net on a random input difference, fits a ridge regression on its penultimate layer, and runs greedy search with exploration for 2000 steps per restart. Simeck's round function commutes with word rotation, so the 16 rotations of (`0x0000`, `0x0040`) are equivalent, and a restart counts as a find if it ends on any of them.

Table 10: Neural input-difference search. A find is a restart that ends on the literature difference: `0x0040/0x0000` for Speck, and (`0x0000`, `0x0040`) or one of its rotations for Simeck (two of the seven 5-round finds are exact, five are rotations). Intervals are 95% Clopper–Pearson. The Speck row is our run of the unmodified upstream script (Section 3).

Cipher	Scored at	Finds [95% CI]
Speck	3 rounds	3/10 [0.07, 0.65]
Simeck	3 rounds	0/20 [0, 0.17]
Simeck	5 rounds	7/10 [0.35, 0.93]

Measured; `results/logs/neural_diff_search.log`, `simeck/history/diff_search.jsonl`, `simeck/logs/diff_search_5r.log`, SIMECK_REPORT §4.7.

Scored at 3 rounds as in the paper, the search never finds the literature difference on Simeck (Table 10). The reason is visible in the scores: every restart ended between 0.980 and 0.999, so the 3-round objective is saturated by Simeck's slow diffusion and cannot separate a difference that is good at 3 rounds from one that extends. Scored at 5 rounds, 7 of 10 restarts end on the literature difference or a rotation of it, and all 7 extend to 8 rounds under the few-shot threshold. On Simeck the knob that matters is the scoring depth, not the optimiser.

Training a depth-1 net for 5 epochs at 8 rounds on each of the 20 differences found by the 3-round search, and on the literature difference, confirms the ranking: the literature difference reaches 0.8475, the best search result (`0x0000`, `0x3000`) reaches 0.6990, and 16 of the 20 are at chance (0.4996–0.5007; `simeck/history/rank_diffs.jsonl`). Five epochs can be too short to leave the plateau, so those rows mean "nothing found in 5 epochs", not "no signal".

4.6 Simeck32/64: key recovery

The last subkey still has a wrong-key profile. Simeck XORs the subkey in after f. Decrypting the last round under $k \oplus \delta$ then shifts the right word of both ciphertexts by δ and leaves the pair difference unchanged for every δ, so a ranking on the decrypted difference alone would be blind. One that also peels the next, keyless round is not, because that peel computes f of a word that δ has changed; the networks evidently do something equivalent. With the 10-round net on 11-round pairs, over all 2^{16} values of δ, the mean response is 0.5133 at $\delta = 0$ and falls with the Hamming weight of δ up to about weight 12 (0.5057 at weight 1, 0.5022 at weight 2, 0.4925 at weight 8), with 500 pairs per δ. At the paper's 3000 pairs per δ the true key scores 0.5094, and the two profiles correlate at $r = 0.70$. The true key is not the top-scoring guess in either profile: 31 and 52 wrong keys score higher, and in the 3000-pair profile the ten best wrong

Table 11: Rank of the true last subkey among all 2^{16} guesses. n counts pairs. Each group gives the cipher, the attack's round count, and the distinguisher with its accuracy at its own depth. Mean ranks carry 95% bootstrap intervals; seconds per trial are on the RTX 3050 (all trials in this table ran there). The first group is the paper's Table 3.

n	Trials	Mean rank [95% boot.]	Median	P[rank = 0]	P[rank < 32]	s/trial
Speck32/64, 9 rounds, N_7 (paper)						
64	1000	52.1	1	0.358	—	—
Simeck32/64, 11 rounds, 9-round net (0.676)						
16	50	678 [372, 1065]	220	0.00	0.14	0.3
32	50	103 [67, 149]	65.5	0.02	0.44	0.5
64	50	40.8 [28.1, 56.7]	23.5	0.00	0.56	0.9
128	50	23.6 [16.4, 31.9]	13	0.08	0.74	1.8
Simeck32/64, 12 rounds, 10-round net (0.546)						
64	50	4718 [3249, 6421]	2316	0.00	0.02	0.9
128	50	1150 [639, 1761]	349	0.02	0.12	1.9
256	50	307 [215, 414]	176.5	0.00	0.14	3.7
512	50	136 [82, 201]	34	0.04	0.50	7.5
1024	50	30.0 [22.4, 38.3]	19.5	0.00	0.66	15.0
Simeck32/64, 13 rounds, staged 11-round net (0.513)						
1024	20	2293 [1167, 3592]	1472	0.00	0.05	15.2
4096	20	42.2 [25.9, 61.0]	33	0.05	0.50	60.5
16384	20	8.7 [6.0, 11.4]	8	0.10	1.00	241

Measured, except the paper row; `simeck/history/key_rank.jsonl` via `simeck/summarise_key_rank.py`, SIMECK_REPORT §4.10.

keys have δ of Hamming weight 1 to 5 (`simeck/history/wkp.jsonl`, `wkp_last{,_n3000}.npy`; SIMECK_REPORT §4.8–4.9). The last subkey can therefore be narrowed to a candidate set but not pinned, and the attack results below show exactly that.

Attack layout. An R-round attack is one conditioned free round (Section 4.3), a distinguisher of depth $R-2$, and one last round undone by guessing k_{R-1}. The prepended round holds with probability 1, so every pair is usable and Gohr's neutral-bit structures are not needed. The last subkey is only 16 bits, so all 2^{16} guesses are scored instead of running Gohr's Bayesian key search. Each trial draws a fresh random master key (`simeck/key_rank_simeck.py`).

Ranks floor in the tens. More pairs push the true last subkey into the top few dozen, but rarely to the top: $P[\text{rank} = 0] \leq 0.10$ in every row of Table 11, against 0.358 in the paper's 9-round Speck attack. The top guess is usually a near miss. At 12 rounds and 1024 pairs it differs from the true subkey in 1 to 5 bits in all 50 trials. This is the profile above at work: guesses wrong by a small-weight δ look almost like the right one. The 13-round row reaches median rank 8 and puts the true subkey in the top 32 in all 20 trials with 2^{14} pairs.

Second subkey. Stage 2 decrypts the last round under each of the top 32 candidates and ranks all 2^{16} guesses of k_{R-2} with a distinguisher one round shallower, on a subset of the pairs. The attack returns the candidate pair with the highest stage-2 score. A wrong last-subkey candidate now corrupts the input to f in the next decryption, which the stage-2 network does see. Table 12 gives the results.

At 12 rounds and 1024 pairs (2^{10} chosen pairs, 2^{11} chosen plaintexts), the attack identifies k_{11} exactly in two thirds of trials and puts k_{10} in its top 32 whenever k_{11} survived stage 1. At 13 rounds, 1024 stage-2 pairs are too few: the 10-round net cannot separate the 32 candidates, and

Table 12: Two-subkey ranking on Simeck32/64. Thirty-two last-subkey candidates go to stage 2. n counts pairs (stage 1 / stage 2). Proportions carry 95% Clopper–Pearson intervals. All columns except the last ran on the RTX 3050; the last pools those 14 trials with 25 run later on the GTX 1660 Ti and 27 on GU605CM (same code and nets).

	12 rounds		13 rounds		
Nets, stage 1 / 2	10-round / 9-round	10-round / 9-round	staged 11-round / 10-round	staged 11-round / 10-round	staged 11-round / 10-round
n, stage 1 / 2	512 / 256	1024 / 256	16384 / 1024	16384 / 4096	16384 / 4096
Trials	30	30	4[a]	14	66[b] (pooled)
True k_{R-1} in top 32	9/30	22/30	4/4	14/14	66/66 [0.95, 1.00]
Stage 2 returns the true k_{R-1}	5/30 [0.06, 0.35]	20/30 [0.47, 0.83]	0/4	11/14 [0.49, 0.95]	**60/66** [0.81, 0.97]
Rank of true k_{R-2}, given true k_{R-1} among the candidates	median 7, all below 32 (9 trials)	median 4, all below 32 (22 trials)	181, 29, 57, 8	median 11, all below 32 (14 trials)	median 13.5, below 32 in 65/66
Both exact	0/30	1/30	0/4	2/14 [0.02, 0.43]	4/66 [0.02, 0.15]

[a] Interrupted after 4 of 10 planned trials when the host session ended. [b] The 14 trials of the previous column, 9 more two-stage trials and the 43 three-stage trials of Table 13, whose stages 1 and 2 are identical; summary by `simeck/summarise_stage3.py`. Measured; `simeck/history/key_rank.jsonl`, `simeck/logs/kr_simeck12_2stage.log`, `simeck/logs/kr_simeck13_2stage*.log`, SIMECK_REPORT §4.10.

stage 2 never chose the true k_{12}. With 4096 stage-2 pairs, **the last subkey k_{12} of 13-round Simeck32/64 was recovered exactly in 11 of 14 trials (95% CI $[0.49, 0.95]$), and k_{11} was in the top 32 in all 14**, from 2^{14} chosen pairs (2^{15} chosen plaintexts satisfying the two bit conditions). The first six trials gave 6/6 and the next eight gave 5/8, so the early estimate was optimistic. Fifty-two further trials (nine two-stage, and the stages 1–2 of the 43 three-stage trials below; 25 on the GTX 1660 Ti, 27 on GU605CM) bring the pooled count to **60 of 66** (95% CI $[0.81, 0.97]$), with the true k_{12} in the 32 candidates in all 66 and the true k_{11} in the top 32 given k_{12} in 65. Exact recovery of both subkeys by stage-2 argmax alone stays rare (4/66). Stages 1–2 take about 36 minutes on the RTX 3050 (mean 2198 s over its 14 trials) and 34 minutes on the GTX 1660 Ti (2043 s): $2^{16} \times 2^{14}$ network evaluations in stage 1 and $2^5 \times 2^{16} \times 2^{12}$ in stage 2, about 2^{33} in all.

Third subkey. Stage 3 takes the 128 best (k_{12}, k_{11}) pairs by stage-2 score, decrypts two rounds under each, and ranks all 2^{16} guesses of k_{10} with the 9-round net on the first 512 pairs; the attack returns the best-scoring triple. Forty-three trials ran, sixteen on the GTX 1660 Ti and twenty-seven on GU605CM (Table 13). The third subkey's score fixes the pair that stage 2 cannot: the true (k_{12}, k_{11}) has median stage-2 rank 35 (max 85), so 32 survivors keep it in only 19 of 43 trials, yet among 128 survivors the argmax triple carries both true k_{12} and k_{11} in **38 of 43**. The true triple is in the top 14 of 2^{23} in 41 trials (top 5 in 30), at rank 17 in one and at rank 63 in another, where the true pair ranked 4 in stage 2 but the true k_{10} only 28th. The full triple is the argmax in 11 of 43; the true k_{10} ranks 0 given the true pair in 12 of 43, median 2. The first sixteen trials alone gave 13/16 and 2/16, the twenty-seven later ones 25/27 and 9/27.

That floor has the tie structure of the last-subkey profile one layer down. We measured the wrong-key profile of the 9-round net on 11-round pairs, which is the stage-3 layout when the pair is right (`simeck/wrong_key_profile.py`, 2^{15} shared pairs; tag `simeck11_net9_n3000` in `simeck/history/wkp.jsonl`). Flipping bit 7 or bit 9 of the guess moves the mean logit by less

than two standard errors, bits 8 and 12 by three to four, and every other bit by 0.008 to 0.63. The guesses within two standard errors of the true key are exactly the span of bits 7 and 9, a tie floor of $(2^2-1)/2 = 1.5$, and the profile predicts mean rank 5.0 at 512 pairs against the measured 3.4 [2.1, 5.0] over 43 trials (median 2). Over the first 28 trials the measured mean was 2.7 and the prediction overshot it, as the Simeck last-subkey predictions do (SIMECK_REPORT §4.11, §4.14), because at this many pairs the faint guesses' mean differences are noisy estimates; the trial with rank 28 shows how heavy the tail is. Each trial's key, plaintexts and the 256 best k_{10} guesses of every survivor are saved (`simeck/history/stage3/`) for the master-key step below.

Table 13: Three-subkey ranking at 13 rounds. Stages 1–2 as in the last column of Table 12; stage 3 ranks k_{10} for the best 128 stage-2 pairs. Proportions with 95% Clopper–Pearson intervals, ranks with 95% bootstrap intervals. Sixteen trials on the GTX 1660 Ti, twenty-seven on GU605CM.

Quantity	Value
Stage-3 net and pairs	9-round net (0.676), 512
Trials	43
True k_{10} in top 32, given true (k_{12}, k_{11})	43/43 [0.92, 1.00]
Median rank of true k_{10}, given true (k_{12}, k_{11})	2 [1, 3] (mean 3.4 [2.1, 5.0], max 28)
True pair among the 128 survivors	43/43 (with 64: 40/43; with 32: 19/43)
Argmax triple has true k_{12} and k_{11}	**38/43** [0.75, 0.96]
Rank of true triple among 128×2^{16}	median 2 [1, 3], mean 4.7 [2.5, 8.1], max 63
All three subkeys exact	**11/43** [0.14, 0.41]
GPU time per trial, GTX 1660 Ti	3007 s (stages 235 / 1841 / 931 s)
GPU time per trial, GU605CM[a]	2189 s (stages 162 / 1323 / 704 s)

[a] Mean of 26 trials (median 2088 s); the first GU605CM trial shared the GPU with another program and took 4866 s. Measured; `simeck/history/key_rank.jsonl`, `simeck/history/key_rank_stage3.jsonl` via `simeck/summarise_stage3.py`, SIMECK_REPORT §4.12, §4.14.

Master key. Two consecutive round keys fix half of Simeck's 64-bit key-schedule state, and the schedule is invertible, so a candidate triple (k_{12}, k_{11}, k_{10}) leaves 2^{16} master keys, one for each value of k_9. The completion step (`simeck/recover_master_key.py`) walks the candidate triples of each trial in descending stage-3 score. For each triple it runs the schedule relation $k_{i+4} = k_i \oplus f(k_{i+1}) \oplus c_i$ backwards for all 2^{16} values of k_9 and checks each resulting master key by trial encryption of 4 stored plaintext pairs over 13 rounds. The first key that matches is the output. As a check, the routine applied to the stored true $k_9, \ldots, k_{12}$ returns the stored master key.

Table 14: Master-key completion from the stage-3 candidates of Table 13, 13-round Simeck32/64. Proportion with a 95% Clopper–Pearson interval. Cloud CPU, NumPy.

Quantity	Value
Trials	43
True master key returned	**43/43** [0.92, 1.00]
Triples walked before it (0-based)	median 2, max 63
Wrong keys that passed verification	0
Time, all 43 trials	3.8 s

Measured; `simeck/history/recover_master_key_2026-11-23.jsonl` (earlier runs: 28/28 and 29/29, `recover_master_key_28trials.jsonl`, `_29trials.jsonl`; the first sixteen trials alone: 16/16 [0.79, 1.00], `simeck/history/recover_master_key.jsonl`), SIMECK_REPORT §4.13.

The full 64-bit master key of 13-round Simeck32/64 was recovered in all 43 trials (95% CI $[0.92, 1.00]$, Table 14), from 2^{14} chosen pairs (2^{15} chosen plaintexts meeting the two bit

conditions), at most $64 \times 2^{16} = 2^{22}$ trial encryptions after the neural stages (at most 14×2^{16} in 41 of the 43 trials), and about 37 to 50 GPU-minutes per trial for those stages. The position at which the key is found equals the true triple's stage-3 rank in every trial, as it must: any triple with the true k_{12}, k_{11} and k_{10} completes to the true key. This step therefore adds no uncertainty beyond that of stage 3, and the end-to-end attack depends on walking a short list rather than on the stage-3 argmax, which is exact in only 11 of 43.

Two caveats. The ciphertexts were not stored with the candidate files, so the four verification pairs were re-encrypted from the stored plaintexts under each trial's stored master key, which plays the encryption oracle; the search itself reads only the candidate lists and those pairs. The candidate list is capped at 4096 triples of 128×256; no trial needed more than 64.

Deviations from Gohr's attack. There are no neutral bits and no Bayesian key search; both are made unnecessary by the probability-1 prepended round and the exhaustive 16-bit scan. Gohr's c_1/c_2 cutoffs and iteration budget therefore have no counterpart. Success is reported as rank statistics rather than under the paper's criterion (last subkey exact, penultimate within Hamming distance 2), so these figures are not directly comparable with T10, except for the master-key result, which is end-to-end. The data complexity assumes a chosen-plaintext attacker who can satisfy the two plaintext-bit conditions.

4.7 Simon32/64

The Simon scripts are copies of the Simeck ones with the cipher module swapped. The only cipher-specific edits are the plaintext conditions of Section 4.3 and the linear part removed in the baselines' projected feature, $\Delta L \oplus (\Delta R \lll 2)$.

4.7.1 Basic pipeline

Table 15: Simon32/64, basic pipeline, with the settings of Table 1, and Simeck alongside. Accuracy on 10^6 fresh pairs; the 2σ error bar is ± 0.0004 at 7 rounds, ± 0.0008 at 8 and ± 0.0010 from 9 on.

Rounds	Simon	TPR	TNR	Simeck
7	0.9651	0.9980	0.9322	—
8	0.7726	0.7607	0.7845	0.8636
9	0.6037	0.4871	0.7201	0.6758
10	0.5000	0.0000	1.0000	0.5384
11	0.5002	0.1006	0.9002	0.4998
12	0.5001	0.7143	0.2861	0.4992

Measured; `simon/sweep.log`, `simon/logs/simon_7r.log` to `simon_12r.log`, SIMON_REPORT §3.

At our training budget the basic pipeline reaches 9 rounds on Simon32/64, against 10 on Simeck32/64 (Table 15). Wherever both learn, Simon is the weaker of the two by less than one round: its 8- and 9-round accuracies fall between Simeck's at r and $r+1$ rounds. A 20-epoch 9-round run reached 0.61262 ± 0.00031 on 10^7 pairs and is the seed for staging. At 10 rounds, direct training failed in all four attempts: the sweep run, plus three 20-epoch runs that ended at 0.5007, 0.4993 (all negative) and 0.5006 (all positive) (`simon/logs/simon_10r_direct_e20_{1,2,3}.log`). Since a 10-round Simeck run fails to start about one time in six, 0 of 4 is suggestive rather than conclusive (95% CI for the escape probability $[0, 0.60]$), and with 10 epochs on 10^7 samples it says nothing about larger budgets. The staged results below are the stronger evidence that the signal exists but direct training at this budget does not find it.

Difference diffusion does not predict this ordering. Table 3 shows Simon's difference distribution flattening slightly *more slowly* than Simeck's: 2,448 against 3,672 distinct differences at

Table 16: Simon32/64, staged nets evaluated at several depths. Accuracy on 10^7 fresh pairs (± 0.0003) unless marked. The last column adds the conditioned free first round to the net at its own depth, so it covers one round more (round count in parentheses).

	Rounds					
Net	9	10	11	12	13	With free round
Staged 10-round	0.49885	**0.55628**	0.50017	0.50103	—	0.55624 (11)
Staged 11-round	—	0.50021	**0.50894**	0.50020	0.49990	0.50862 (12)
Staged 12-round	—	—	—	0.5002[a]	—	—
Staged 10-round, seed b[b]	0.49953	**0.55527**	0.50006	0.50094	—	0.55507 (11)
Staged 11-round, seed b[b]	—	0.50029	**0.50897**	0.50011	0.50010	0.50893 (12)

[a] 10^6 pairs only (± 0.0010); this run is in the record but not yet written up in SIMON_REPORT.md. [b] Second seed, trained on GU605CM. Measured; `simon/logs/eval_seedb_q7.log`, SIMON_REPORT §5.4, `simon/history/eval.jsonl`, `simon/history/prepend.jsonl`, `simon/history/simon_12r_staged.jsonl`, SIMON_REPORT §5, §5.1.

round 4, and 16.11 against 16.44 bits at round 5. Yet its networks are weaker. The entropy of the difference distribution is a crude summary, and the peeled differential tables of Section 4.7.3 are a better guide: at 9 rounds they score 0.6179 on Simon and 0.6831 on Simeck, the same ordering as the networks. We have not measured what separates the two ciphers.

4.7.2 Free first round and staged training

With Simon's own bit conditions (Section 4.3), the 20-epoch 9-round net covers 10 rounds at 0.61264 ± 0.00031, its full 9-round accuracy, against 0.53575 without the conditions (`simon/history/prepend.jsonl`). The 10-epoch net gives 0.60398 and 0.53312.

Staged training follows Section 4.4.2, one round lower than for Simeck. The seed is the 20-epoch 9-round net, stage 1 runs at 7 rounds under (`0x0441`, `0x0100`), and stage 2 uses 10^9 fresh 10-round samples at each of the three learning rates. Like Simeck's, Simon's 3-round output difference has four tied modes ((`0x0441`, `0x0100`) 0.0625, (`0x0440`, `0x0100`) 0.0624, (`0x0641`, `0x0100`) 0.0623, (`0x0640`, `0x0100`) 0.0622). The first is used throughout; two others were tried for the 11-round stage (below). Table 16 gives the results of staging once, twice and three times.

Unlike Simeck, where the 10-round seed already beat chance at 11 rounds, Simon's 9-round seed is at chance at 10 rounds (0.50006 on 10^7 pairs), so staging creates the signal rather than amplifying it. Stage 1 does most of the work (10-round accuracy $0.5004 \to 0.5344$ on 10^6 pairs) and the first 10^9 samples of stage 2 the rest (0.5560). The two learning-rate drops change nothing (0.5561 final). The staged net reaches 0.55628 at 10 rounds, much stronger at its depth than Simeck's staged 11-round net (0.51274); both are one round past the basic pipeline's last success. With the free round it covers 11 rounds at 0.55624.

Staging a second time works on Simon, where it failed on Simeck: from the staged 10-round net, which is at chance at 11 rounds, the same recipe reaches 0.50894 at 11 rounds and, with the free round, 0.50862 at 12. Simon then reaches the same round counts as Simeck at somewhat lower accuracy (0.5089 against 0.5127 at 11 rounds, 0.5086 against 0.5129 at 12 with the free round).

Two of the other tied modes were later run for the 11-round stage on the GTX 1660 Ti, with everything else identical. (`0x0440`, `0x0100`) gives 0.50902 at 11 rounds and 0.50883 at 12 with the free round; (`0x0641`, `0x0100`) gives 0.50874 and 0.50876 (10^7 pairs, ± 0.00032; `simon/history/eval.jsonl`, `simon/history/prepend.jsonl`). The three 11-round results span 0.00028 ($z \approx 1.3$ for the extreme pair), so the stage-1 difference does not matter for Simon either. All three share one seed (SIMON_REPORT §5.1).

A second 9-round seed, trained on GU605CM with the same settings (0.6157 ± 0.0010 on 10^6

pairs; `simon/logs/simon_9r_d1_e20_b.log`), gives a staged 10-round net at 0.55527 against the first seed's 0.55628 ($z \approx 4.5$), and 0.55507 at 11 rounds with the free round. Staged again, it reaches 0.50897 at 11 rounds against 0.50894 from the first seed, and 0.50893 at 12 rounds with the free round. Unlike Simeck's seeds (Table 7), the two Simon seeds give the same staged nets to within 0.001 at 10 rounds and 0.00003 at 11.

Staging a third time fails. With the staged 11-round net as seed, stage 1 at 9 rounds learned nothing (0.49995 on 10^6 pairs), stage 2 never left chance, and the final 12-round net scores 0.5002 ± 0.0010 on 10^6 pairs (`simon/history/simon_12r_staged.jsonl`, `simon/logs/simon_12r_staged.log`). Both ciphers' staged chains stop at 11 rounds of network depth.

4.7.3 Purely differential baselines

The baselines are built exactly as in Section 4.5.1, including the tables on the keylessly peeled $(r-1)$-round difference (with $\Delta R_{r-1} = \Delta L \oplus f(R) \oplus f(R')$ for Simon's f), and are lower bounds on the exact-DDT distinguisher.

Table 17: Simon32/64: sampled difference tables against the full-pair networks. "Table" is the best of seven tables on the ciphertext difference (± 0.0007); "peeled" is the best table on the keylessly peeled difference (± 0.0003 or better).

Rounds	Table	Peeled	Full-pair net
6	0.9876	—	—
7	0.7887	**0.9769**[c]	0.9651
8	0.6220	0.7570[d]	0.7726
9	0.5520	**0.6179**[d]	0.6126[a]
10	0.5082	0.5402[d]	0.5563[b]
11	0.5010	0.5079[e]	0.5089[b]

[a] 20-epoch net. [b] Staged nets. [c] Full 32-bit peeled difference. [d] 24-bit projection. [e] 16-bit projection $\Delta L_{r-1} \oplus (\Delta R_{r-1} \lll 2)$. Bold: the peeled table beats the network. Measured; `simon/history/ddt_baseline.jsonl`, `peel_baseline/peel_results.jsonl`, `peel_proj_results.jsonl`, SIMON_REPORT §6.

Against the tables on the ciphertext difference the networks lead from 7 rounds on, but a peeled table **beats the network at 7 and 9 rounds** (0.9769 against 0.9651; 0.6179 against 0.6126; Table 17). The networks lead it at 8 rounds and the staged nets at 10 and 11 rounds, at 11 by only 0.0010 ($z \approx 5$). Every published DDT figure is above every network (Section 4.9). This is what Gohr, Leander and Neumann [21] argue for ciphers of Simon's type: a network can only use differential features, so the DDT is the optimum. A difference-only net trained at 7 rounds, which cannot peel a round, reaches 0.83521 on 10^7 pairs (`simon/history/diffonly.jsonl`).

4.7.4 An unexplained two-round effect

Every basic-pipeline Simon network we evaluated is a better distinguisher two rounds past its training depth than one round past it (Table 18). The Simeck nets show no such pattern and decay monotonically.

The 11-round figure for the 9-round net replicated on three independent samples (0.50682, 0.50679, 0.50692) and with independent uniformly random subkeys in place of the key schedule (0.50703), so the effect is a property of the round function, not of the key schedule. It also appears in a network that sees only the ciphertext difference, and in purely differential tables: a peeled table of 6-round differences used as a 7-round distinguisher scores 0.96059 at 7 rounds, 0.50030 at 8 and 0.50351 at 9 (10^7 pairs per class), and a 7-round table used at 8 rounds gives 0.50073 at 9 and 0.50319 at 10. The same construction on Simeck decays monotonically (0.93606, 0.52403, 0.50920, 0.50262; `peel_baseline/tworound.log`). The effect is therefore a property of Simon's difference propagation, not of anything the networks add, and the open question

Table 18: Accuracy at and beyond each net's training depth r, on 10^7 fresh pairs (± 0.0003). "Real differences" is Gohr's task in which both classes are real pairs and class 0 is blinded by one random mask on both ciphertexts. The mask keeps the ciphertext difference but not the peeled difference, so for Simon a purely differential distinguisher can solve it. "Difference only" nets see $(0, 0, \Delta L, \Delta R)$.

Cipher	Net, task	r	$r+1$	$r+2$	$r+3$	$r+4$
Simon	7-round	0.96494	0.50420	**0.52697**	0.50006	—
Simon	8-round	—	0.50081	**0.52575**	0.49995	—
Simon	9-round (20 epochs)	0.61262	0.50006	**0.50679**	0.50005	0.50011
Simon	9-round (10 epochs)	—	0.50000	**0.50688**	—	—
Simon	7-round, real differences	0.89066	0.50386	**0.52170**	0.50012	—
Simon	difference only, trained at 7	0.83521	0.50602	**0.51948**	0.50044	—
Simeck	10-round (control)	0.54636	0.50615	0.50164	—	—
Simeck	difference only, trained at 8 (control)	0.70532	0.54034	0.50955	0.50209	—

Measured; `simon/history/eval.jsonl`, `realdiff.jsonl`, `diffonly.jsonl`, `subkey_check.jsonl`, `simeck/history/eval.jsonl`; SIMON_REPORT §7.

becomes why the $(r{+}2)$-round difference distribution resembles the r-round one more than the $(r{+}1)$-round one does. That could be checked from the DDT itself; we have not done so. The real-differences task does not separate the two channels for Simon (caption of Table 18). The staged 10-round net shows it faintly (0.50103 at 12 rounds, $z \approx 6.5$, against 0.50017 at 11), and the staged 11-round net does not (0.49990 at 13 rounds, two past its depth).

We have no explanation for it. Two candidates were Simon's $(1, 8, 2)$ rotation amounts, since rotating a 16-bit word by 8 twice is the identity, and an interaction with the Feistel word swap. We tested the second on the GTX 1660 Ti: if a net keyed on the round-r left word, it should still work on $(r{+}1)$-round pairs whose two words are exchanged in each ciphertext. It does not. Every full-pair net, Simon's five and three Simeck controls, is at chance on swapped inputs at every offset r to $r{+}3$ (all within 0.0009 of 0.5 on 10^7 pairs; `simon/history/swap.jsonl`, `simon/swap_eval.py`). Swapping removes the $r{+}2$ signal rather than moving it to $r{+}1$, so the effect is not the word swap. A difference-only net keeps a small signal at swapped $r{+}1$ (0.50320, about a sixth of its plain $r{+}2$ advantage), so in the difference channel too the swap explains at most a small part (SIMON_REPORT §7).

We tested the rotation candidate on GU605CM by changing only the rotations of the round function, $f(x) = ((x \lll a) \wedge (x \lll b)) \oplus (x \lll c)$, with Simon's key schedule, the input difference (`0x0000`, `0x0040`), and the depth-1 network trained for 10 epochs on 10^7 samples (`simon/rotation_variant.py`). These variants are toy ciphers with no test vector; the script checks that $(1, 8, 2)$ reproduces Simon's official vector and that every variant decrypts correctly. Table 19 gives the results.

The Simon control reproduces the effect at both depths. The variant $(1, 7, 2)$, which has no rotation by 8, shows it too: its $r{+}2$ cells are further from 0.5 than its $r{+}1$ cells (0.47786 against 0.49040 at depth 7; 0.50438 against 0.49789 at depth 8). An accuracy below 0.5 is still a distinguisher with its decision inverted, so the $r{+}2$ signal is present in both. The Simeck round function decays monotonically, as the Simeck nets do. $(1, 9, 2)$ and $(1, 8, 3)$ diffuse so slowly that their nets are near-perfect at 7 and 8 rounds and learn nothing that carries a round further, so they say nothing either way. Within what was run, a rotation by 8 is not needed for the effect, and the effect remains unexplained. Each cell is a single training run, and the sign changes of $(1, 7, 2)$ between depths are themselves unexplained and not replicated. Simon's 0.8907 at 7 rounds on the real-differences task is not a non-differential signal: a peeled difference table scores 0.9526 on the same task (`peel_baseline/peel_results.jsonl`). Gohr's argument that this task is difference-blind holds for Speck, where undoing the last round without the key recovers

Table 19: Simon-like round functions with altered rotations (a, b, c). One depth-1 net per row, trained at r rounds; accuracy at r to r+3 rounds on 10^7 fresh pairs (± 0.00032). $(1, 8, 2)$ is Simon and $(5, 0, 1)$ is Simeck's round function. GU605CM.

(a, b, c)	r	r	r+1	r+2	r+3
(1, 8, 2)	7	0.97216	0.50201	**0.51821**	0.50023
	8	0.73523	0.50078	**0.51843**	0.50005
(1, 7, 2)	7	0.90497	0.49040	**0.47786**	0.49954
	8	0.74363	0.49789	**0.50438**	0.49988
(1, 9, 2)	7	1.00000	0.49971	0.50002	0.49991
	8	0.99999	0.50010	0.50011	0.50025
(1, 8, 3)	7	0.99708	0.49922	0.49695	0.49984
	8	0.94502	0.50322	0.50061	0.49995
(5, 0, 1)	7	0.98744	0.55287	0.52027	0.50434
	8	0.85797	0.57602	0.51895	0.50360

Measured; `simon/history/rotation_variant.jsonl`, `simon/logs/rotation_variant.log`, SIMON_REPORT §7.

Table 20: Simon32/64: rank of the true last subkey among all 2^{16} guesses, laid out as Table 11.

n	Trials	Mean rank [95% boot.]	Median	P[rank = 0]	P[rank < 32]	s/trial
Simon32/64, 12 rounds, staged 10-round net (0.556)						
64	50	6319 [3774, 9353]	1882.5	0.00	0.04	0.9
128	50	2160 [1080, 3461]	640	0.00	0.06	1.9
256	50	487 [213, 849]	124	0.00	0.16	3.8
512	50	107 [78, 143]	70.5	0.00	0.30	7.6
1024	50	67.9 [52.4, 84.3]	47.5	0.04	0.32	15.1
Simon32/64, 13 rounds, staged 11-round net (0.509)						
1024	20	5577 [2154, 11077]	2378	0.00	0.00	15.4
4096	20	1507 [702, 2849]	966.5	0.00	0.05	60.7
16384	20	906 [684, 1136]	809.5	0.00	0.00	245

Measured; `simeck/history/key_rank.jsonl` via `simeck/summarise_key_rank.py`, SIMON_REPORT §5.2.

only a word whose difference is a linear function of the ciphertext difference, but not for Simon or Simeck.

4.7.5 Last-subkey ranking

The attack layout and script are those of Section 4.6, with Simon's conditioned round (`simeck/key_rank_simeck.py--ciphersimon`). Only the last subkey was ranked; no two-subkey attack was run on Simon.

Simon's last subkey is much harder to rank than Simeck's at the same round count, even though the 12-round Simon distinguisher is the stronger of the two (0.556 against 0.546). At 1024 pairs the median rank is 47.5 against Simeck's 19.5. At 13 rounds Simon stalls near rank 800 with 2^{14} pairs, where Simeck reaches 8 (Tables 11 and 20). Both ciphers add the last subkey linearly, so a wrong guess only XORs δ into one word of both ciphertexts. A ranking on the decrypted difference alone would be blind; the networks can rank because, in effect, they also undo the next round, whose f sees the wrong δ.

Why the 13-round rank stalls. We measured the wrong-key profile in exactly the attack layout (GTX 1660 Ti, `simeck/wrong_key_profile.py`): one set of 2^{15} pairs shared by every guess $k \oplus \delta$, with $m(\delta)$ the mean difference of pre-sigmoid logits from the true key. The attack ranks δ above the true key with probability about $\Phi(\sqrt{n}\, m(\delta)/s(\delta))$, so the expected rank at n pairs is the sum of that over δ. The staged 11-round Simon net reads only 5 of the 16 bits of the

decrypted right word (bits 3, 4, 10, 11, 13). Flipping any other bit changes the mean logit by at most 3.4×10^{-5}, within two standard errors of zero, and the guesses within two standard errors of the true key are exactly the $2^{11} - 1$ nonzero elements of the span of those eleven bits. They tie with the true key in expectation, so its expected rank floors near 1024 however many pairs are used. The profile predicts mean rank 4516, 1120 and 1035 at 1024, 4096 and 16384 pairs against the measured 5577, 1507 and 906, each inside the measurement's bootstrap interval. At 12 rounds the staged 10-round net ignores only four bits (6, 7, 13, 15), and its predictions (115 and 83 at 512 and 1024 pairs) also match (107, 68). The staged 11-round net from the second Simon seed reads the same five bits (tag `simon13_seedb_n3000` in `simon/history/wkp.jsonl`): 1983 guesses lie within two standard errors of the true key, against 2047 for the first seed, so the floor is a property of where staged Simon nets look, not of one training run. The second seed's net was not used in key-ranking trials. The corresponding Simeck nets ignore five bits at 12 rounds and four to six at 13, which is why Simeck's 13-round rank reaches about 10 (SIMON_REPORT §5.3, SIMECK_REPORT §4.11). Gohr's fresh 3000-pair protocol cannot show this structure at 13 rounds: its per-δ standard error is as large as the whole profile.

Deviations (Simon). The basic-pipeline sweep ran at 10 epochs, 5% of the paper's budget, as for Simeck. Staged training carries the same deviations as for Simeck (Section 4.10), and its stage-1 difference is one of four tied modes; two others give the same 11-round result and the fourth was not tried. Two seeds were staged to 10 and 11 rounds. The differential baselines are sampled lower bounds, not the exact DDT. Several CPU evaluations ran alongside GPU training, which affects the wall-clock times in the logs but not the accuracies. The 12-round staged run was evaluated only on 10^6 pairs.

4.8 Cross-cipher summary

Table 21 collects the results. Five conclusions are supported by the measurements above.

1. With the method held fixed and the ciphers aligned by accuracy, Simeck32/64 is distinguishable about two rounds further than Speck32/64, and Simon32/64 slightly less than two (Simon at $r + 2$ rounds scores 0.7726 and 0.6037 against the paper's 0.788 and 0.616 for Speck at $r = 6, 7$; the Speck 0.616 is the distilled N_7, which if anything favours Speck). This holds at our training budget. Two candidate explanations of the Simeck–Speck gap, neither measured, are Simeck's slower difference diffusion and the keyless last round of Simon and Simeck, which gives their networks the previous round's difference for free.

2. Staged pre-training transfers: it moves each cipher's boundary by one round. Chaining it a second time worked on Simon (10 to 11 rounds) and failed on Simeck (11 to 12); the next step failed on Simon too (11 to 12). Both chains stop at a network depth of 11 rounds.

3. For ciphers that add the round key after f, the first round can be made free by choosing plaintexts, as Abed et al. did for Simon [2]. This adds one round to every distinguisher. We implement it with two plaintext-bit conditions and a fixed plaintext difference; choosing the right word per pair needs no conditions.

4. Key ranking transfers even though the last subkey enters linearly: a wrong last-round guess leaves the decrypted difference unchanged but corrupts the next keyless peel, which the networks evidently use. On Simeck it supports a 13-round attack that recovers the full master key in 43 of 43 trials. On Simon the 13-round ranking stalls because the staged 11-round net reads only 5 of the 16 bits of the decrypted word, so most wrong guesses tie with the true one.

5. The paper's central Speck result, networks beating the optimal purely differential distinguisher, does *not* carry over in our measurements. Sampled tables on the keylessly peeled

Table 21: Gohr's pipeline on three 32/64 ciphers. Round counts are for the distinguisher's own input difference; "free round" rows add the conditioned first round of Section 4.3. Speck values are from the paper [1]; Simeck and Simon values are measured and repeat the tables above.

	Speck32/64 (paper)	Simeck32/64	Simon32/64
Full rounds	22	32	32
Basic pipeline, last round learned	7 rounds; the paper's N_7 (0.616) is distilled[a]	10 rounds, 0.5384	9 rounds, 0.6037
Direct training one round further	8 rounds: fails	11 rounds: 0 of 10 runs	10 rounds: 0 of 4 runs
Staged training	8 rounds, 0.514	11 rounds, 0.5127 (four seeds: 0.5078–0.5133)	10 rounds, 0.5563; again to 11, 0.5089
Staging one round further	—	12 and 13 rounds: fail	12 rounds: fails
With the free round	first round free without conditions	12 rounds, 0.5129; 13 rounds, 0.5018; 14 rounds, 0.50064 (10^8 pairs)	11 rounds, 0.5562; 12 rounds, 0.5086
Last-subkey rank, largest attack	9 rounds, 64 pairs: median 1, $P[0] = 0.358$	13 rounds, 2^{14} pairs: median 8, top-32 in 20/20	13 rounds, 2^{14} pairs: median 809.5, top-32 in 0/20
Two-subkey attack	11 rounds: 52.1% (paper's criterion)	13 rounds: k_{12} exact 60/66; with a third stage, true (k_{12}, k_{11}) 38/43	not run
Full key recovery	remaining work $\approx 2^{38}$, estimated	13 rounds: 43/43 [0.92, 1.00], 2^{14} pairs	not run

[a] The paper's basic pipeline learns 7 rounds (§4.3, "Training Cost"), but the reported N_7 is not a basic-pipeline network. It is a single-block network trained for 300 epochs on 10^6 seven-round samples labelled by KeyAveraging (Algorithm 1), using a six-round single-block network distilled from N_6 (§4.3–4.4). N_5 and N_6 are basic-pipeline networks.

difference beat our networks at 7 and 9 rounds on Simon and at 9 and 10 rounds on Simeck, and lag them by at most 0.004 at the staged depths. Published exact single-pair DDT figures are above every network we trained, for both ciphers (Section 4.9). We have no evidence that our networks use anything beyond the difference distribution; the networks are lower bounds on what this kind of network can reach, not the state of the art.

4.9 Comparison with published results

The figures in this subsection are *quoted* from the papers named and were not measured here. They come from a reading of the full texts on 2026-11-01 (`notes/literature_survey.md` in the project files); only values read directly from a table or sentence of a paper are used. Published work exists for both ciphers, and our networks are below it.

Simon32/64. Bao et al. [20] report single-pair Simon32/64 distinguishers with the same input difference: Gohr's network with Gohr's learning-rate schedule reaches 0.6110 at 9 rounds, 0.6296 with a tuned schedule, and 0.6515 with an SE-ResNeXt network. Their staged 10- and 11-round networks reach 0.5610 and 0.5174, trained on 2^{28} to 2^{30} samples per stage against our 10^7 and 10^9. Gohr, Leander and Neumann [21] report 0.661, 0.567 and 0.520 at 9, 10 and 11 rounds, and Bellini et al.'s DBitNet [22] 0.5166 at 11 rounds. Bao et al.'s purely differential distinguishers, lined up with the network one round higher as in their paper, give 0.6628, 0.5683 and 0.5203 for 9, 10 and 11 rounds. Our figures are 0.6037 (basic pipeline, 10 epochs), 0.5563 (staged) and 0.5089 (staged). At 9 rounds we are close to Gohr's untuned network and about 0.06 below the differential bound; at 11 rounds our advantage over 0.5 is about half of what the published networks and the differential bound reach. Gohr, Leander and Neumann argue that for ciphers of Simon's type the network can use only differential features, which is consistent with the best networks matching the DDT and with our own peeled tables (Section 4.7.3). Our two-round effect (Section 4.7.4), which the peeled tables show too, is not discussed in the papers we read.

Simeck32/64. Zhang et al. [17] computed the exact purely differential distinguisher for (`0x0000`, `0x0040`) on single pairs. Because the last Simeck round can be peeled off a ciphertext pair without the key, they compare an (R+1)-round network with the R-round table. On that alignment the table gives 0.9040, 0.5738 and 0.5194 for 8-, 10- and 11-round networks, against our 0.8636 (basic), 0.5384 (basic) and 0.5127 (staged). Our Simeck networks are therefore also below the differential bound, by 0.04 at 8 rounds and 0.007 at 11. Our difference-only network sees only the ciphertext difference and cannot peel a round, so its bound is the 8-round table, 0.7105; it reaches 0.7053. The published Simeck networks we found [19, 17] use 8 pairs per sample, so their accuracies are not comparable with ours.

Released networks on our data. To separate a training gap from an implementation difference, the released Simon32/64 networks of Bao et al. and the released 11-round DBitNet of Bellini et al. were evaluated on 10^7 samples from our own `simon.py` on GU605CM (Table 22; `followups/FOLLOWUPS_REPORT.md`). All fifteen Bao et al. networks land within 0.0004 of the accuracies in the authors' own evaluation log, and DBitNet gives 0.51765 against the paper's 0.5166 and the 0.5182 in its repository's README. Our Simon implementation therefore agrees with theirs in distribution, and the gap to the published networks is one of training and architecture, not a bug. A depth-10 ResNet of Bao et al. trained with Gohr's own learning-rate schedule gives 0.61119 at 9 rounds, the same as our depth-1 network's 0.6126; their lower-rate schedule adds about 0.02.

Combining pairs. Summing single-pair logits over m pairs under one key (`followups/multipair.py`, 2^{22} pairs per cell) reproduces the pattern Gohr, Leander and Neumann describe.

Table 22: Released Simon32/64 networks evaluated on data from our `simon.py`, 10^7 samples (± 0.00032), against the authors' own figures. DBitNet uses input difference (`0x0000`, `0x0400`); the others (`0x0000`, `0x0040`).

Network	r	Ours	Authors
SE-ResNeXt [20]	9	0.65129	0.65148[a]
	10[b]	0.56068	0.56100[a]
	11[b]	0.51757	0.51740[a]
ResNet, Gohr's schedule [20]	9	0.61119	0.6110
DBitNet [22]	11	0.51765	0.5166
DBitNet, trained here[c]	10	0.55863	—
	11	0.51454	—
	12	0.50023	—
Ours, depth 1	9	0.6126	—
Ours, staged	11	0.5089	—

[a] The authors' evaluation log, 5×10^7 samples on their generator; other author figures are from the papers. [b] Staged. [c] Their training pipeline re-implemented (`followups/autond_simon.py`), round-by-round warm start from 7 rounds, 10 epochs per round, one run; the round-12 net stays at chance, so it reaches 11 rounds and not 12. Measured; `followups/history/bao_simon.jsonl`, `followups/history/autond_eval.jsonl`, `followups/FOLLOWUPS_REPORT.md` (T17, T21).

Bao et al.'s 9-round Simon network combined over 16 and 64 pairs gives 0.9513 and 0.9989, against their figures of 0.963 and 1.00 for a single-pair network 0.01 stronger. Our 9-round network gives 0.8886 and 0.9920. Lu et al.'s dedicated 8-pair networks [19] (0.9176, 0.6975, 0.5609 at 9 to 11 rounds) beat 8-pair combinations of the best released single-pair network (0.8802, 0.6686, 0.5568), so on this evidence they do more than combine; we did not resolve why. For Simeck, our 8-pair combinations (0.9163, 0.6528, 0.5439 at 9 to 11 rounds) stay below the exact 8-pair DDT of Zhang et al. (0.9518, 0.7221, 0.5677), the same gap as for single pairs (`followups/history/multipair.jsonl`).

Key recovery. Published Simeck32/64 attacks reach 13 rounds with 2^{16} chosen plaintexts at 88% success [18], as quoted by Zhang et al. [17], and 15, 16 and 17 rounds at 99.17%, 100% and 30% [17]. Bao et al. attack 16-round Simon32/64 with 2^{21} chosen plaintexts at 49% success. These attacks prepend one to three classical rounds and use neutral bits and Bayesian key search. Our 13-round Simeck master-key recovery (43/43 with 2^{15} chosen plaintexts) is a different design point, a probability-1 conditioned round and exhaustive 16-bit scans with no neutral bits, not a new record. Note that published data complexities count chosen plaintexts, while n here counts pairs.

Not checked. The paper of Lyu et al. [18] could not be retrieved, so its figures are quoted second-hand. The released attack scripts of Bao et al. were not run. The bibliography entries were checked against OpenAlex in an in-project review, except the Gohr–Leander–Neumann preprint, which OpenAlex does not list.

4.10 Deviations and limits

The deviations are stated next to each result above. Collected here:

- *Training budget.* The basic-pipeline sweeps ran at 10 epochs, 5% of the paper's 200. At the Simeck boundary a tenfold budget changed nothing (Table 4). Seed nets for staging were trained for 20 epochs.
- *Run-to-run spread.* At 10 rounds on Simeck the spread between runs is about 0.02, and one run in 14 never left the constant solution. Staged Simeck nets from four working seeds

span 0.5078 to 0.5133 at 11 rounds, so the headline 0.5127 is near the top of that range. Simon's two seeds give staged nets within 0.001 of each other at 10 and 11 rounds.

- *Staged training.* Each 10^9-sample pass was generated and consumed in 10^7-sample chunks because of the 7 GB memory limit under WSL; every sample is still seen once at batch size 10^4. Adam's moment estimates are reset at each learning-rate change. The stage-1 difference is one of four tied 3-round modes on both ciphers; the others tried (three on Simeck, two on Simon) give the same 11-round accuracy, while a second seed on Simeck does not.

- *Baselines.* Our own purely differential baselines are sampled lower bounds, not the exact DDT. The first ones ignored the keyless last round and so understated the differential reference by up to 0.19 (Simon, 7 rounds); the peeled tables, built by an in-project review, correct that.

- *Key recovery.* There are no neutral bits, no Bayesian key search and no c_1/c_2 cutoffs, results are rank statistics rather than the paper's success criterion, and the attacks assume a chosen-plaintext attacker who can meet two plaintext-bit conditions. The 13-round two-subkey result pools 66 trials from three hosts, and the three-stage and master-key results rest on 43 trials from two hosts. In the master-key step the stored master key stood in for the encryption oracle when the verification pairs were re-encrypted.

- *Corrections to the working reports.* Some significance figures in SIMECK_REPORT.md and SIMON_REPORT.md divide by the 2σ error bar and so understate z by a factor of two. For example, the 13-round free-round distinguisher, 0.50180 on 10^7 pairs, is reported there as "about 6σ" and is $z \approx 11$. The sampled-table baselines are tested on 2×10^6 pairs and carry ± 0.0007, not the ± 0.0010 printed there. The z values and error bars in this section are recomputed. Bootstrap intervals were recomputed as described in Section 4.1.

4.11 Observations on the agent

The extension work was done by Claude, and five episodes from its record bear on how such work should be checked.

An implementation error that no result would have revealed. The Simeck constant-sequence error of Section 4.2 was the agent's own. It was caught only because the agent checked the test vector before training, and it was fixed by reading the specification after a search over 320 variants failed.

An experiment that measured something other than its label. A wrong-key profile meant to probe the second-to-last Simeck subkey decrypted the last round with the correct key first, so it repeated the last-subkey experiment one round deeper. The agent caught this and relabelled the result as a replicate rather than reporting it as a new measurement (SIMECK_REPORT §4.8).

Early estimates that were optimistic. The first six 13-round two-subkey trials returned the true last subkey 6 times out of 6; the next eight returned it 5 times. The working report keeps both figures and says that the first was optimistic. A reader given only the first batch would have overstated the attack.

A baseline that missed a free round. The agent built the Simeck and Simon differential baselines on the ciphertext difference, carrying over the Speck construction, and every write-up until this review said that the networks beat every purely differential distinguisher by a wide margin. A separate review session pointed out that the last round of these ciphers can be undone without the key; tables on the peeled difference beat the networks at several depths. The same oversight made the real-differences task look difference-blind for Simon. A comparison against a baseline is only as good as the baseline's use of what is free.

A false negative avoided by an earlier lesson. At the Simeck 11-round boundary, the sweep's degenerate, chance-level predictions looked exactly like the reduced-budget false negative of the Speck N_7 distillation run. The agent re-ran the boundary at ten times the budget, and then ran eight more direct attempts, before concluding that direct training fails.

5 A defect in the supplementary code

5.1 Symptom

Our 9-round key-rank measurement returned a $20.5\% \pm 5.7$ success rate against the paper's 35.8%, while simultaneously reproducing the mean key rank (40.1 ± 33.5 against 52.1 ± 34.7). A distinguisher that were simply weaker than the paper's would degrade both statistics together; degrading one while matching the other is anomalous.

5.2 Diagnosis

Our first hypothesis was numerical. The scoring computes $\log_2(Z/(1-Z))$ on float32 network output, so any Z rounding to exactly 1.0 becomes $+\infty$, and keys with infinite scores would tie with one another and be broken randomly. Instrumenting the run disproved this: zero non-finite scores across 4.19×10^6 raw predictions per trial.

The same instrumentation revealed the actual cause. The upstream `key_rank.py` computes

```
rank0 = np.sum(Z > Z[k1]) # subkeys scoring strictly higher than the right key
rank1 = np.sum(Z >= Z[k1]) # subkeys scoring at least as high -- includes the right
    key
...
r[i] = randint(a, b) # Python randint is inclusive at both endpoints
```

The m subkeys tied with the right key occupy positions `rank0`…`rank1` $- 1$, where $m =$ `rank1` $-$ `rank0` ≥ 1 because the right key ties with itself. The paper's stated convention is that the right key sits at a uniformly random position among them, which is `randint(rank0, rank1-1)`. Using `rank1` as the upper bound admits one position no key occupies.

We measured the tie count directly: $m = 1$ **in every one of 200 trials.** A neural distinguisher essentially never assigns two subkeys identical scores. So the draw is `randint(rank0, rank0+1)`—a coin flip adding 0 or 1 to every reported rank, which inflates the mean by 0.5 and halves the rate of reporting rank 0.

5.3 Three independent confirmations

(i) Correcting the convention reproduces the paper. 200 trials, same shipped network:

(ii) The author's own arrays cannot have come from the shipped code. The repository ships the key-rank arrays behind Table 3, and they reproduce it exactly (median 1.0, mean 52.1, 35.8% at rank 0 for the neural distinguisher; 9.0, 263.9 and 13.0% for the DDT). Under the shipped convention, every trial truly at rank 0 is bumped to rank 1 with probability $\frac{1}{2}$, so the

Statistic	Upstream convention	Corrected	Paper
median key rank	2.0	**1.0**	1.0
mean key rank	40.1 ± 33.5	$\mathbf{39.6 \pm 33.5}$	52.1 ± 34.7
success rate	$20.5\% \pm 5.7$	$\mathbf{39.5\% \pm 6.9}$	35.8%

reported count at rank 1 must be *at least* the count at rank 0. Formally, writing T_k for the true count at rank k,

$$N_0 = \tfrac{1}{2}T_0, \qquad N_1 = \tfrac{1}{2}T_0 + \tfrac{1}{2}T_1 \;\geq\; N_0.$$

The author's array shows

```
rank 0:  358     rank 1:  150     rank 2:  81     rank 3:  59
```

and $150 < 358$. The array is inconsistent with the published code path.

(iii) The arithmetic closes at two independent data sizes. Halving the author's true rates predicts what the buggy convention should yield, and that is what we measured:

Pairs	Author's array (true)	Buggy convention predicts	Measured (buggy)	Measured (corrected)
32	10.0%	5.0%	**5.5%**	—
64	35.8%	17.9%	$\mathbf{20.5\% \pm 5.7}$	$\mathbf{39.5\% \pm 6.9}$

Every cell agrees within its error bar.

A fourth view makes the mechanism visible. Writing the reported rank as $R = T + B$ with $B \sim \text{Bernoulli}(\frac{1}{2})$,

$$\Pr[R \leq k] \;=\; \Pr[T \leq k] - \tfrac{1}{2}\Pr[T = k],$$

so the cumulative distribution is depressed by exactly half the probability mass at k. That mass is large at $k = 0$ and negligible by $k = 9$:

$\Pr[\text{rank} \leq k]$	k=0	k=1	k=2	k=4	k=9
author's array	0.358	0.508	0.589	0.695	0.780
ours, corrected	0.395	0.550	0.645	0.715	0.815
ours, upstream convention	**0.205**	0.455	0.585	0.685	0.815

The three curves converge by $k = 9$. No plausible difference in distinguisher quality produces that signature—a weaker distinguisher lags at every k.

5.4 Impact

The paper's Table 3 is correct and reproduces cleanly. The script published to generate it does not reproduce it, and errs by approximately a factor of two in the direction of understating the attack. A reader who runs `key_rank.py` as shipped and compares against Table 3 will conclude that the paper overstates its 9-round result. Since this repository is the reference implementation on which subsequent work in neural cryptanalysis builds, the defect is worth correcting at the source.

We note in passing that the same shipped arrays confirm a further claim from Section 4.4: at 256 chosen plaintexts (128 pairs) the attack succeeds $71.5\% \pm 2.9$ of the time, exceeding the 50% ceiling that Proposition 1 places on any purely differential full-key-recovery attack. The DDT distinguisher at the same data size reaches 31.8%.

6 Discussion

6.1 What reproduces easily, and what does not

The distinguishers reproduce almost trivially—the shipped networks evaluate to within 0.002 of every published accuracy, and retrained networks reach comparable numbers on a fraction of the intended budget. The parts that resist reproduction are not the machine learning but the classical cryptanalysis surrounding it: the full difference distribution table is a 300-core-day, 70 GB computation, and the perfect five-round distinguisher took 13.5 hours on ten threads. An observation worth recording is that in this paper the *neural* components are by a wide margin the cheapest to reproduce.

6.2 Two methodological traps

Both of these produced results that looked like failed reproductions and were artefacts of our own process. We document them because both generalise.

Reduced training budgets must scale by gradient steps, not epochs. Constraining every training run to five epochs cut N7 by roughly 60× relative to the others, because its dataset is an order of magnitude smaller *and* its schedule is 300 epochs rather than 200. It produced 0.5069—a number that reads as a categorical failure of the KeyAveraging-distillation method. At the paper's own schedule the same code reached 0.6125. An epoch is not a unit of learning; it is a unit of dataset traversal, and the two coincide only when dataset sizes match.

Two TensorFlow processes on one GPU degrade throughput by 6–15×, not 2×. Measured for the same depth-10 network: 124 s per epoch alone, ≈810 s sharing with an inference job, 1804 s sharing with two. Under WSL2 the GPU is virtualised through the Windows WDDM driver, and once the combined allocations approached the 6 GB limit the driver appears to page memory between host and device. Related, and non-obvious: `SIGSTOP` does not release a process's GPU allocation, so pausing a job to "measure the card alone" silently measures a contended card. Three of our diagnostic measurements were invalid for this reason before we noticed.

6.3 Threats to validity

Single-seed training. Each network was trained once. We compare against the paper at four significant figures without an empirical estimate of run-to-run variance, so a 0.003 discrepancy cannot presently be distinguished from seed noise. Repeated training of the single-block networks would resolve this cheaply and is the most valuable outstanding work.

Limited trial counts on the stochastic attacks. The 11-round success rate rests on 83 trials (±10.9) and the key rank on 200 (±6.9), against the paper's 1000 in both cases. The point estimates agree; the intervals are wider than we would like. The 11-round figure moved from 44.0% to 56.6% when we went from 25 trials to 83, which is a useful reminder of how little a small-sample success rate constrains.

The 8-round distinguisher's margin. N8 reaches 0.5135 against a classical baseline of 0.512. Both our reproduction and the paper's original are only a few standard errors above the baseline, and the result depends on three stages of bootstrapping from other distinguishers. This is the paper's weakest claim, and reproducing it does not make it strong.

We did not verify the DDT rows. Claims about D6, D7 and D8 rest on the paper's own computation. Our confirmation of D5 by an independent route provides some assurance that the method is sound, but it is not the same as checking them.

7 Conclusion

Gohr's results reproduce. Eighteen claims tested, eighteen confirmed, on hardware roughly half the capability of the original and a software stack six years newer. The paper's central scientific claim—that neural distinguishers exploit structure in the ciphertext pair that no purely differential distinguisher can access—is supported by three independent lines of evidence in our hands: the real-differences experiment, the disagreement analysis against the difference distribution table, and the 9-round attack exceeding the ceiling that Proposition 1 places on purely differential attacks.

The reproduction also surfaced a defect that a reproduction is uniquely positioned to find. The paper is right and its published script is wrong, in a way that makes the paper look worse than it is, and that could only be detected by someone running the code and comparing carefully against the table it claims to produce. We regard this as the strongest argument for reproduction studies as a genre: not that they catch bad papers, but that they catch the gap between what a paper claims and what its artifacts do.

A Defects in the supplementary code

1. **`key_rank_ddt.py` cannot execute.** Calls `sp.ex_key_test` and `sp.enc_test`, which do not exist under those names (`expand_key`, `encrypt`), and uses `randint` without importing it—three errors in 29 lines.

2. **`key_rank.py` crashes on a clean checkout.** Writes to `./data_9r_attack/`, absent from the archive.

3. **`test_key_recovery.py` launches 20 twelve-round attacks at import**, so the module cannot be imported to reuse its functions.

4. **The 12-round attack path applies the wrong wrong-key profile.** It passes `m=m6, s=s6` to the inner key search despite accepting `m_help`/`s_help` for that purpose, applying the 6-round profile to the 7-round network.

5. **Two `np.save` calls in `train_nets.py` write to the same filename**; the validation-accuracy history is overwritten by the loss history.

6. **`keyschedule is 'free'`**—identity comparison against a string literal.

7. **`key_nonuniformity.test` swaps its return values**, printing accuracy and MSE under each other's labels.

8. **`key_rank.py` off-by-one in the rank statistic**—see Section 5. Halves the reported success rate.

9. **Two operations exceed a 6 GB card** where an 11 GB card would not: handing `predict` arrays of 10^6–4×10^6 rows materialises a single eager tensor promoted to float32. An environment issue rather than an upstream defect, but it silently caps smaller hardware.

10. **`speck.readcsv` is unusable at its intended scale.** Per-column hex converters inside `np.genfromtxt` take over 10 minutes on 10^5 rows against 5.6 s for a direct parse; the paper's own analysis used a 10^6-row set.

Only defect 8 affects a published number. The remainder are packaging faults, but collectively they suggest the supplementary code was assembled after the fact rather than being the code that produced the paper.

B Compute profile

Wall-clock, single machine, jobs run strictly sequentially on the GPU:

Stage	Duration
C++ Markov analysis (10 threads, CPU)	13.5 h
KeyAveraging teacher set (10^9 predictions)	1.6 h
N7, 300 epochs	3.0 h*
N6 depth-10, 5 epochs	0.9 h
N8 staged training	0.9 h
Key rank, 200 trials at $n = 64$	2.1 h
Algorithm 1, all 2^{16} keys	0.9 h
N5 depth-10, 38 epochs	8.6 h
Input-difference search, 10 restarts	0.6 h
11-round attack, 83 trials	15.4 h**
Wrong-key profile, few-shot, distillation, evaluations	0.7 h

* includes a suspend/resume cycle; actual compute is lower.
** largely run concurrently with training, and therefore heavily degraded—see Section 6.2.

Total elapsed for the study: approximately 32 hours on one laptop. Roughly 6 hours of that was lost to the GPU contention described in Section 6.2, and a further 3 hours to the N7 budget error.

References

[1] A. Gohr. Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. *CRYPTO 2019*, LNCS 11693, pp. 150–179.

[2] F. Abed, E. List, S. Lucks, J. Wenzel. Differential Cryptanalysis of Round-Reduced Simon and Speck. *FSE 2014*, LNCS 8540, pp. 525–545.

[3] M. R. Albrecht, G. Leander. An All-In-One Approach to Differential Cryptanalysis for Small Block Ciphers. *SAC 2012*, LNCS 7707, pp. 1–15.

[4] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, L. Wingers. The SIMON and SPECK Lightweight Block Ciphers. *DAC 2015*, pp. 1–6.

[5] E. Biham, R. Chen. Near-Collisions of SHA-0. *CRYPTO 2004*, LNCS 3152, pp. 290–305.

[6] A. Biryukov, V. Velichkov, Y. Le Corre. Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. *FSE 2016*, LNCS 9783, pp. 289–310.

[7] C. Blondeau, B. Gérard. Multiple Differential Cryptanalysis: Theory and Practice. *FSE 2011*, LNCS 6733, pp. 35–54.

[8] I. Dinur. Improved Differential Cryptanalysis of Round-Reduced Speck. *SAC 2014*, LNCS 8781, pp. 147–164.

[9] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. *CVPR 2016*, pp. 770–778.

[10] G. Hinton, O. Vinyals, J. Dean. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

[11] D. P. Kingma, J. Ba. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

[12] X. Lai, J. L. Massey, S. Murphy. Markov Ciphers and Differential Cryptanalysis. *EUROCRYPT 1991*, LNCS 547, pp. 17–38.

[13] H. Lipmaa, S. Moriai. Efficient Algorithms for Computing Differential Properties of Addition. *FSE 2001*, LNCS 2355, pp. 336–350.

[14] M. Pelikan, D. E. Goldberg, E. Cantú-Paz. BOA: The Bayesian Optimization Algorithm. *GECCO 1999*, vol. 1, pp. 525–532.

[15] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, G. Gong. The Simeck Family of Lightweight Block Ciphers. *CHES 2015*.

[16] B. Zhu. Reference implementations of the Simeck family. `https://github.com/bozhu/Simeck`, 2015.

[17] L. Zhang, J. Lu, Z. Wang, C. Li. Improved Differential-neural Cryptanalysis for Round-reduced Simeck32/64. *Frontiers of Computer Science*, 2023.

[18] L. Lyu, Y. Tu, Y. Zhang. Deep Learning Assisted Key Recovery Attack for Round-Reduced Simeck32/64. *ISC 2022*.

[19] J. Lu, G. Liu, B. Sun, C. Li, L. Liu. Improved (Related-key) Differential-based Neural Distinguishers for SIMON and SIMECK Block Ciphers. arXiv:2201.03767; IACR ePrint 2022/030.

[20] Z. Bao, J. Guo, M. Liu, L. Ma, Y. Tu. Enhancing Differential-Neural Cryptanalysis. *ASIACRYPT 2022*.

[21] A. Gohr, G. Leander, P. Neumann. An Assessment of Differential-Neural Distinguishers. IACR ePrint 2022/1521.

[22] E. Bellini, D. Gerault, A. Hambitzer, M. Rossi. A Cipher-Agnostic Neural Training Pipeline with Automated Finding of Good Input Differences. *IACR Transactions on Symmetric Cryptology*, 2023.