Reverse-engineering a trained network for all-optical inference via random-feature kernels

W. Schulz

Jun 27, 2026

Abstract

Can inference on an already-trained network run entirely in light? An envelope exists, bounded by a quantified depth→width feature wall — and an honest requote against a digital ASIC collapses the headline speedup to near parity. Two genuine advantages survive: precision robustness and zero-forgetting frozen features.

A falsification study. Status: model + single pre-registered run complete. Verdict: narrowly supported — an envelope exists and is bounded by a quantified depth→width feature wall.

> ⚠ Addendum (10-step follow-up program — **`docs/experiment-plan-10-steps.md`**). A ten-experiment program (empirical distillation + cost-model stress) ran after this paper and materially revises two conclusions: 1. Requote everything against a digital ASIC, not an H100 (step 9). Against a fused INT8 ASIC the headline 48× collapses to 1.23×, winning at only 1/5 complexities. The large speedups below were mostly the GPU's launch overhead and bf16 energy, not a fundamental optical edge. The defensible claim shrinks to "near parity at trivial complexity, low precision, batch-1." 2. The "free" square-law feature map is only a degree-2 polynomial kernel (step 1, empirically). It has a hard expressivity floor that grows with target depth; universality needs a true (non-free) phase nonlinearity. The §3/§5 envelope is therefore optimistic for deep targets. The backbone *held*: the depth→width conservation law (§3) is empirically real (measured base 1.7–2.4 vs assumed 4). Two genuine advantages were also found — random features are precision-robust (~3000× the matmul cliff, step 8) and frozen features give zero-forgetting continual learning for free (step 10). See that doc for all ten.

Cost model: `photonic/costmodel/kernel.py` · Harness: `photonic/kernel_inference.py` · Results: `docs/results/kernel-inference-results.{md,csv}` · Tests: `photonic/costmodel/tests/test_kernel.py` (10) + the suite (174 total).

---

0. The question

> Can you take an already-trained neural network and transform it so that inference runs *completely in light* — and if so, where does that stop working?

This paper answers the constructive half (yes, via a specific transformation) and, more importantly, builds the falsification machine that finds the boundary. It is the companion to the speed-of-light SSM study (`docs/experiment-speed-of-light.md`): that one asked whether an optics-native *recurrence*

beats a GPU; this one asks whether an optics-native *kernel approximation of an arbitrary trained function* does.

The work is scoped to inference only. We are not learning representations; the target network's weights are frozen. That scoping is load-bearing — it neutralizes the usual objection to fixed kernels (that they do no representation learning), because at inference no substrate learns anything.

---

1. Why most of a trained network resists optics, in one paragraph

A photonic matrix engine wins only when a weight matrix is loaded once into the mesh and reused across many inputs — the cost is amortized over the reuse. Sorting a trained network's inference operations by *reuse* rather than by *static-vs-dynamic*:

- Learned projections / FFN weights — loaded once, reused for every input forever. Optics-friendly.
- Attention $\mathbf{Q \cdot K^T}$ and $\mathbf{softmax \cdot V}$ — built fresh from the current input's activations, used once, then discarded (reuse factor ≈ 1). These are *activation×activation* products with no resident operand. Optics-hostile — not because weights change, but because there is no reused matrix to make resident.
- Softmax — a global reduction (row-sum normalization) with no clean optical form; forces a detect→digital→re-modulate round trip every layer.
- Depth — a long cascade of nonlinearities that (a) needs a clean cascadable all-optical nonlinearity (immature) and (b) accumulates analog noise with no free re-clocking.

A trained Transformer spends its critical-path compute precisely on the hostile operations. So a *direct* port loses. The interesting question is whether a *transformation* of the trained function can move the work onto optics-native primitives.

---

2. The transformation: a random-feature kernel

Replace the trained network's fixed input→output map `f(x)` with a random-feature kernel approximation:

$$f(x) \approx W \cdot \varphi(x), \qquad \varphi(x) = \sigma(\Omega x), \qquad \Omega \text{ fixed random.}$$

- Ω — a large *fixed random* projection `d_in → D`.
- σ — a point-wise nonlinearity (the feature map).
- W — a small linear readout `D → out_dim`, the only trained operator (fit by least-squares against `f`'s outputs; this is distillation, done offline, once).

This is the random-features view of kernel machines (Rahimi & Recht, 2007). Structurally it is one nonlinearity deep, which is the whole point: it collapses the depth cascade (defusing the nonlinearity-cascade and noise-accumulation problems) into a single layer.

2.1 Why this specific transformation is optics-native

Each piece lands on a primitive optics does *natively*:

Piece	Operation	Optical realization	Cost behavior
Ω	fixed random `d_in→D`	a multiple-scattering medium (diffuser / disordered cavity) — a fixed random transmission matrix realized in one passive optical transit	no tiling, no weight reload; cost ∝ readout `D`, not `D·d_in` MACs
σ	point-wise feature map	square-law detection `|E|²` — the nonlinearity rides on the photodetector	free (mature; it *is* the ADC front-end)
W	small trained readout	ordinary weight-resident MZI mesh	standard tiling cost, but small (`D→out_dim`)

The middle column is the crux. The single *large* operation — the projection — is the one big linear map a photonic substrate can do without the tiling tax that kills programmed matmuls. A scattering medium does not encode a programmed `L×L` mesh tiled `⌈D/L⌉·⌈d_in/L⌉` times; it performs the entire fixed random `D×d_in` map in bulk, passively, in a single pass. This is the LightOn-OPU / Saade-et-al. *"random projections at the speed of light"* regime. And the feature-map nonlinearity is the one nonlinearity that is genuinely free in optics — square-law detection — so φ needs zero optical-electronic round trips internally.

So the pipeline is, end-to-end: encode `x` into light once → passive scatter to `D` features → square-law detect (free nonlinearity) → small resident readout → out. The electronics touch it at the converters only.

---

3. The catch, made quantitative: the depth→width conservation law

Collapsing depth is not free. The expressivity the deep network packed into depth must reappear somewhere, and for a fixed kernel it reappears as the number of random features **`D`**. Two distinct results bound it, and at inference both are pure approximation theory (no training dynamics involved):

1. Random-feature sampling. Matching the target kernel to relative error `eps` needs `D ∝ 1/eps²` features (Rahimi & Recht's `O(1/√D)` sup-norm convergence, inverted).
2. Depth separation. A shallow/kernel model needs width growing super-polynomially (generally exponentially) in the target's effective depth to represent it within a fixed error (Telgarsky 2016; Eldan & Shamir 2016; Montúfar et al. 2014 on linear-region counting).

We fuse these into a single stylized law (`kernel.features_needed`):

```
D(complexity, eps)  =  c_rf · base^(complexity − 1) / eps²
```

- `complexity` = the target's effective depth (1 = already kernel-easy).
- `base` = the per-depth feature multiplier (`DEPTH_WIDTH_BASE`, default 4, deliberately mild — the real blow-up is worse, so a small base is *conservative for the falsification*).

This law's job is not to predict the exact D for a specific network — it is to locate the crossover where the optical envelope closes. We show (§5.3) the qualitative conclusion survives sweeping base ∈ {2, 4, 8}.

A second, independent cost grows with D: readout. The passive medium makes the *multiply* free, but every one of the D feature modes still has to be detected — an ADC per mode. So the projection's interface cost is O(D). The scattering medium removes the D·d_in MAC term; it does not remove the D readout term.

Net: two D-driven walls. An economic wall (the O(D) readout tax eventually exceeds the GPU) and a feasibility wall (D exceeds the optical feature budget, MAX_OPTICAL_FEATURES ≈ 10^6 modes, optimistic free-space).

---

4. The cost model

We extend the existing photonic cost model (the same Op/HardwareConfig/CostResult contract, the same calibrated ~2 fJ/MAC and ~10× interface-tax assumptions) with one sub-model, kernel.py:

- **scattering_projection_cost** — prices Ω as a passive transit: inject d_in modulated inputs, one bulk transit, detect D outputs over the L-wide converter array. No tiling term, no reload term. Energy = DAC·d_in + ADC·D + passive-optical·D; latency = inject + transit + readout, dominated by the ⌈D/L⌉ readout groups.
- **features_needed** — the conservation law above.
- **optical_feature_budget** — the feasibility cap.
- **make_matmul_cost_fn** — a dispatcher: a matmul tagged optical_random_projection → scattering model; every other matmul → the ordinary tiling.cost. (Ω is cheap; the readout W still pays full tiling.)

The harness (kernel_inference.py) scores three contenders per (complexity, eps, substrate) point:

1. optical-kernel — the all-optical pipeline at the D the law demands.
2. gpu-kernel — the *same* kernel as plain matmuls on an H100 (control: is optics' edge real, or is the kernel just cheap everywhere?).
3. gpu-original — the original depth-**complexity** MLP on an H100. This is the honest deploy baseline: the thing a practitioner would otherwise run.

All at batch-1 / M=1 (the decode regime optics targets).

The iso-accuracy assumption (main threat to validity). eps is treated as the task's *accepted* error tolerance: the fidelity-eps kernel and the exact GPU network are *both assumed to satisfy the task* at that eps, so comparing their cost is fair. The cost model scores latency/energy at this assumed iso-task-accuracy; it does not train the kernel or measure a real accuracy gap. Discharging this assumption (actually distill a trained net into a random-feature kernel and measure D vs. accuracy) is the obvious next experiment and the one thing that would convert this from a cost argument into an empirical one.

---

5. Results (single pre-registered run)

5.1 The envelope exists

At low complexity and loose fidelity the all-optical kernel wins decisively. Headline point: complexity 1, eps 0.1, D = 100 → 44.8 ns on the MZI mesh vs 2.16 μs for the depth-1 network on the H100 — 48× faster. The win persists out to eps ≈ 0.03 and across complexities 1–4 (e.g. complexity 3, eps 0.1, D = 1,600 → 22.6×).

5.2 The envelope closes — exactly as the conservation law predicts

As either `eps` tightens or `complexity` grows, `D` explodes and both walls bite:

complexity	eps	D	optical latency (MZI)	vs GPU-original	status
1	0.1	100	44.8 ns	48.1×	★ win
1	0.03	1,112	335 ns	6.4×	★ win
1	0.01	10,000	2.86 μs	0.8×	lose (readout tax)
3	0.1	1,600	462 ns	22.6×	★ win
3	0.01	160,000	45.4 μs	0.2×	lose
6	0.1	102,400	29.1 μs	0.8×	lose
6	0.001	1.0×10^9	291 ms	0.0×	infeasible (D > budget)

The frontier (tightest winning `eps` per complexity, base 4): `1→0.03, 2→0.03, 3→0.03, 4→0.1, 6→closed`. By effective depth 6 the kernel never wins at any swept fidelity — the depth→width wall has overtaken the GPU before the tolerance is met.

5.3 The conclusion is not an artifact of the chosen constant

Re-running the entire verdict at `base ∈ {2, 4, 8}`:

base	thesis	best speedup	frontier (complexity → tightest winning eps)
2	✓	77.8×	1:0.03, 2:0.01, 3:0.03, 4:0.03, 6:0.03
4	✓	48.1×	1:0.03, 2:0.03, 3:0.03, 4:0.1, 6:closed

base	thesis	best speedup	frontier (complexity → tightest winning eps)
8	✓	48.1×	1:0.03, 2:0.03, 3:0.1, 4:0.1, 6:closed

In every case the same shape holds: an envelope at low complexity / loose fidelity that closes as either tightens. Only the boundary moves. (The mild non-monotonicity — e.g. complexity 2 sometimes winning at tighter `eps` than complexity 1 — is real: a deeper original network is also slower on the GPU, raising the bar the kernel must clear *and* the cost it competes against, which partially offsets the feature growth until the wall dominates.)

5.4 An honest wrinkle: "all-optical" ≠ "optical-energy-dominated"

The reported optical energy fraction of the projection is only ~1%. The compute is genuinely optical, but the energy ledger is dominated by the DAC/ADC conversions — the same interface-tax story that runs through the whole project, here concentrated in the `O(D)` readout. "All-optical" describes where the *multiply* happens, not where the *joules* go. This is not a flaw in the kernel idea; it is the precise reason large `D` is fatal.

---

6. What this says about the original question

- Yes, a trained network's *inference function* can be transformed to run end-to-end in light — via a random-feature kernel — and the transformation is uniquely optics-native because its one large operation (the random projection) is the one big linear map optics does without tiling, and its nonlinearity is the free square-law detector.
- But the regime is bounded, and the bound is the depth→width conservation law: the feature count to hit a tolerance grows like `base^(depth)/eps²`, and two `D`-driven walls (readout tax, feature budget) close the envelope as the target gets deeper or the required fidelity tighter.
- The practical sweet spot the model identifies: shallow effective-complexity targets at modest fidelity, batch-1, latency-bound — exactly the corner (control loops, RF front-ends, cheap edge perception) every track of this project keeps converging on. It is *not* a path to running a frontier-capability Transformer in light, because such models have high effective complexity and demand high fidelity, which is precisely where the envelope is closed.

The kernel transformation is the strongest optics-alignment lever we found: it is the one move that helps the nonlinearity-cascade, noise-accumulation, *and* tiling problems at once (the random projection is the one form of width that does not trigger tiling). Its price is paid entirely in the conservation law — and this study makes that price a number.

---

7. Reproduce

```
cd photonics
PYTHONPATH=. ../.venv/bin/python -m photonic.costmodel.kernel        # sub-model self-check
PYTHONPATH=. ../.venv/bin/python -m photonic.kernel_inference        # full sweep -> docs/resul
PYTHONPATH=. ../.venv/bin/python -m photonic.costmodel.tests.test_kernel   # tests
PYTHONPATH=. ../.venv/bin/python -m pytest -q                        # full suite (174)
```

8. Threats to validity / open work

1. Iso-accuracy is assumed, not measured (§4). The single most valuable follow-up: distill a real trained net into a random-feature kernel and plot measured accuracy vs. `D`, overlaying the optical budget. That replaces the stylized `features_needed` law with an empirical curve and either widens or collapses the envelope for real.
2. Effective complexity is a free parameter. We sweep it; we do not measure it for any real model. Mapping a given trained network to an effective `complexity` is itself a research question (intrinsic dimension / kernel-bias estimation).
3. The scattering medium is idealized. Real OPUs have finite contrast, fixed (un-tunable) `Ω` you must calibrate against, and `1/f` drift; the model charges only the readout, not calibration overhead.
4. Readout **`D`** may be reducible. Structured/Fastfood projections, learned feature selection, or analog readout (optical `W` fused with the detection) could blunt the `O(D)` tax — the most promising way to push the frontier outward, and a clean next module.

Published by NeuroGenesys (AW3 Technology, Inc.). Canonical version: https://neurogenesys.si/research/optical-kernel-inference