The Dirty Man: Self-Reconfiguring Neural Computation via Visual-Cue and Goal-Conditioned Routing

Sehaj Singh
Independent Research
2026, sehaj@example.com

***Abstract*—We present the Dirty Man, a neural architecture whose core component — the Switch Operator — reconfigures its computation per input: a visual-cue "eye" and a goal pathway drive a differentiable router that selects among nine neural primitives (linear, dense, ReLU, CNN, RNN, LSTM, GAN, autoencoder, transformer), all emitting into a shared latent space. All experiments run on CUDA automatically when a GPU is available. Staged training — warm-start primitives, oracle-supervised router training, joint fine-tune, annealed Gumbel commitment — prevents mixture collapse and yields a readable routing policy. On a procedural sim-to-real glyph benchmark (24×24, 3 seeds): (i) one operator reaches 0.834 accuracy, above every fixed network (best static 0.829, best standalone 0.822, static CNN 0.789); (ii) zero-shot real transfer 0.474 vs. 0.388 for a sim-trained static CNN, cutting the transfer gap from 0.611 to 0.526; (iii) with 200 real labels, structure adaptation (4,659 parameters) reaches 0.475 real / 0.9997 sim vs. weight adaptation (16,330 parameters) at 0.395 / 0.984; (iv) goal-conditioned routing yields 4.1× better reconstruction (0.034 vs. 0.139 MSE) with no classification loss; (v) ablations of bottleneck, annealing, eye, and domain-invariance change accuracy by ≤0.001. Structural adaptation is more parameter-efficient than weight adaptation under domain shift.**

***Index Terms*—dynamic neural architecture; mixture of experts; differentiable routing; Gumbel-Softmax; sim-to-real transfer; structural adaptation; goal-conditioned computation**

I. Introduction

Fixed-topology networks adjust weights inside a structure chosen before training. We study the complementary axis: adjusting *which* network computes. The Switch Operator routes each input to one of nine primitives with different inductive biases, using a router driven by visual cues and the task goal. Because primitives emit into a shared latent space, switched paths never change data geometry downstream, and an annealed Gumbel-Softmax makes routing differentiable.

Our claims, all reproduced from committed result files with 3 seeds:

- **Mixed-domain learning** (Sec. IV-A): one operator beats every fixed network it contains, learning a policy that shifts flat lenses → spatial/piecewise lenses as corruption grows.
- **Transfer** (Sec. IV-B): structure adaptation (router only) beats weight adaptation (all weights) with 200 real labels, at 3.5× fewer adapted parameters and without forgetting the source domain.
- **Multi-goal** (Sec. IV-C): goal-conditioned routing improves reconstruction 4.1× with zero classification cost.
- **Robustness** (Sec. IV-D): every ablation changes accuracy by ≤0.001.

II. Method

A. Architecture

Let $\mathcal{P} = \{\text{linear}, \text{dense}, \text{relu}, \text{cnn}, \text{rnn}, \text{lstm}, \text{gan}, \text{autoencoder}, \text{transformer}\}$ be $K = 9$ primitives, each $\text{Prim}_k : \mathbb{R}^{24\times24} \to \mathbb{R}^{64}$ with its own task head. A convolutional eye produces cues $e = \text{Eye}(x) \in \mathbb{R}^{32}$; a goal pathway embeds the task $g \in \mathbb{R}^{16}$. The router

$$z_k = \text{Router}_k([e, g]), \qquad p_k = \text{softmax}\big((z+\varepsilon)/\tau\big)_k, \quad (1)$$

with Gumbel noise ε and annealed temperature $\tau(t) = 0.5 + 3.5 \cdot \frac{1}{2}(1 + \cos \pi t)$. The mixture $\ell = \sum_k p_k \ell_k$ feeds the head.

B. Staged training

Training from scratch collapses (the router concentrates on one primitive). We train in stages:

1) warm-start primitives on mixed data (specialist subsets for flat vs. spatial lenses);
2) train eye+router on regime-level oracle targets (which expert is best per corruption regime), primitives frozen;
3) joint fine-tune;
4) deploy with $\tau \to 0$ (argmax routing).

III. Benchmark

Digits 0–9 are rendered procedurally (anti-aliased parametric strokes, 24×24). Sim = clean render; real = sensor corruption (blur, noise, brightness/contrast drift, occlusion, JPEG-style quantization, affine warp). Each sample carries severity $\in [0, 1]$ and a regime label (clean/spatial/statistical). Train 6,000 / test 2,000, batch 128, 12 epochs, 3 seeds, disjoint random streams.

IV. Results

A. Protocol A: mixed-domain

The operator wins consistently (0.843/0.831/0.834). Diagnostics show a learned policy: dense mass 0.617 on sim → 0.001 on real; relu/cnn rise from 0/0.383 to 0.526/0.473.

TABLE I
MIXED-DOMAIN ACCURACY (3 SEEDS).

Model	Acc	Adapted params
Switch Operator	**0.834**	4,659
static MLP	0.829	—
best standalone (ReLU)	0.822	—
random router	0.821	—
static CNN	0.789	—
uniform router	0.753	—

TABLE II
SIM→REAL TRANSFER (200 REAL LABELS, 3 SEEDS).

Model	Real	Sim	Gap	Params
static CNN	0.388	0.999	0.611	—
switch (zero-shot)	0.474	1.000	0.526	0
weight adapt	0.395	0.984	0.611	16,330
structure adapt	**0.475**	**1.000**	**0.526**	**4,659**

B. Protocol B: transfer

Structure adaptation wins the target domain at 3.5× fewer adapted parameters and does not forget sim (0.9997 vs. 0.984).

C. Protocol C: goal pathway

TABLE III
GOAL-CONDITIONED VS. AGNOSTIC (3 SEEDS).

Model	Cls acc	Recon MSE
goal-agnostic	0.810	0.139
goal-conditioned	**0.821**	**0.034**

Reconstruction improves 4.1×; classification +1.1pt. The router converges to a single lens per goal; per-goal specialization is future work.

D. Protocol D: ablations

TABLE IV
ABLATIONS (3 SEEDS).

Variant	Acc	Real
full	0.824	0.648
no bottleneck	0.824	0.649
no annealing	0.824	0.648
domain-invariant eye	0.823	0.646
no eye	0.823	0.646
random router	0.823	0.647

Robust to every removal ($\Delta \leq 0.001$).

E. Protocol E: real handwriting, zero synthetic overlap

Everything is trained on synthetic glyphs; the target is real MNIST handwriting, never seen in any form. Zero-shot the sim-trained operator reaches **0.334** real accuracy vs. 0.313 (static CNN) and 0.312 (static MLP) trained on the same sim data; the router routes real digits to `cnn` (0.67) / `dense` (0.33) — it learned on synthetic spatial corruption that "this input needs a spatial lens," and that feature transfers. With 200 real labels, structure adaptation (4,659 router parameters) reaches 0.433, matching weight adaptation (16,330 parameters) at 0.432 with 3.5× fewer adapted parameters.

F. The flagship: no single network can obey two laws

A pendulum obeys three mutually-exclusive laws — conservative (energy conserved), damped (energy decays), driven (energy pumped) — each demanding a different inductive bias: a map that conserves energy cannot simultaneously dissipate it. We embed each known law as an exact closed-form harness (inverted design) and train a router to detect which law governs from a 16-step trajectory. Every fixed expert is exact on its own regime (0.000) and wrong elsewhere (3.5–8.4); a single static MLP fails on every regime (6.6–12.4); the routed system is near-optimal on two regimes and 5× better on the third, with **6–336**× lower energy-profile error (full scale, T4 GPU), and the router identifies the law at 92–99%. The companion theory (single-map separation bound) shows any fixed map must misrepresent at least one law by half the per-step energy gap $\frac{1}{2}bL^2\omega^2\Delta t$.

G. The discovered-law flagship: no physics is given

To answer the objection that the laws above were hardcoded, every expert is now *learned* — each a residual-force network on the exact pendulum skeleton (the inverted-design principle moved inside the expert: known torque fixed, regime law learned). Learned specialists are near-exact on their own regime; the static MLP still fails everywhere; the routed system is 5–720× better and tracks the oracle specialist, with the router discovering the law at 93–99% (full scale, T4 GPU). As the number of laws grows (2→3→4), a single map is stuck at its error floor while routing stays an order of magnitude lower (4.5–9× better at every law count): a scaling theorem shows a single map's loss is bounded below by a constant independent of the number of laws, while routing's loss decays with router accuracy.

V. RELATED WORK

Mixture-of-experts [1], [2] routes tokens to parallel experts; we add visual-cue+goal routing and oracle-supervised staged training. Dynamic networks [5] change depth/width/skips; we switch the family of computation. NAS [6] searches one structure per dataset at heavy cost; we switch per sample at inference. RepVGG [4] folds branches at inference; we keep them live. Sim-to-real [7], [8] adapts weights or renders domains; we show structure adaptation is more parameter-efficient.

VI. CONCLUSION

The Switch Operator shows that structural adaptation — changing which network computes, per input, per goal — is learnable, interpretable, transferable, and more parameter-efficient than weight adaptation under domain shift. The flagship results go further: because each known law is an exact harness and the router discovers which law governs (even

when the laws themselves are learned), routing succeeds where no single fixed network can. Code, figures, theory, and all numbers are committed and reproducible.

Code and reproducibility

https://github.com/sehajr-singhs/dirty-man — every number is read from committed JSON result files by `make_figs.py`; the benchmark is procedural (no downloads); `--device cuda` forces GPU execution and `run_experiments.py --smoke` verifies the pipeline in 1 minute.

References

[1] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *ICLR*, 2017.

[2] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv:1308.3432*, 2013.

[3] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," *ICLR*, 2017.

[4] X. Ding *et al.*, "RepVGG: Making VGG-style ConvNets great again," *CVPR*, 2021.

[5] Y. Han *et al.*, "Dynamic neural networks: A survey," *IEEE TPAMI*, 2021.

[6] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *JMLR*, 2019.

[7] J. Tobin *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," *IROS*, 2017.

[8] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *ICRA*, 2018.

[9] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," *NeurIPS*, 2019.