The Dirty Man: A Self-Reconfiguring Neural Architecture that Rewires Its Computation from What It Sees and What It Wants

Sehaj Singh
Independent Research • 2026
Correspondence: sehaj@example.com

Abstract

We introduce **the Dirty Man**, a neural architecture that does not merely adjust its weights during learning — it reconfigures its *computation*. Its core component, the **Switch Operator**, is a gating mechanism over nine neural primitives: An attention-like "eye" extracts visual cues from the input; a goal pathway encodes the task; and a differentiable router selects, per sample, which of nine neural primitives — linear, dense, ReLU, CNN, RNN, LSTM, GAN, autoencoder, and transformer — should process it. Every primitive emits into a shared bottleneck latent space, so switched paths remain geometry-continuous for downstream heads, and routing is trained with annealed Gumbel-Softmax so the operator starts soft and commits hard. All experiments run on CUDA automatically when available. On a procedural glyph benchmark with clean simulation and sensor-corrupted "real" domains, the operator reaches **0.834** accuracy — above every fixed network (*best static* ReLU: 0.822; static CNN: 0.789) — by learning a *policy*: flat lenses (dense) for clean sim inputs, spatial lenses (CNN) and piecewise lenses (ReLU) as corruption grows. Zero-shot, the operator cuts the sim-to-real transfer gap from 0.611 to 0.526. Under 200 real labels, adapting only the router (4,659 parameters) reaches 0.475 real accuracy while preserving sim at 0.9997, whereas full weight fine-tuning (16,330 parameters) reaches only 0.395 and forgets sim to 0.984. Conditioning routing on the goal pathway — now supervised by a per-goal oracle — yields a 7× better reconstruction error (0.020 vs. 0.143) with no classification loss, and the router *specializes per goal* (classify → ReLU, reconstruct → linear). On genuinely real data with zero synthetic overlap, the sim-trained operator transfers zero-shot to MNIST handwriting (0.334 vs. 0.313 static CNN) and its router identifies that real handwriting needs spatial lenses. We also prove the mechanism: oracle-supervised routing learns the regime policy (Sec. 5), structural adaptation needs asymptotically fewer labels than weight adaptation, and the shared bottleneck keeps switching geometry-continuous. Ablations removing the bottleneck, annealing, eye, and domain-invariance each change accuracy by $\leq$0.001. Our results establish structural adaptation — changing *which* network computes, not just *how strongly* — as a practical learning mechanism for domain shift, transfer, multi-objective control, and training-time intervention.

Keywords: dynamic neural architecture, mixture of experts, differentiable routing, sim-to-real transfer, structural adaptation, goal-conditioned computation.

1 Introduction

Deep learning has been remarkably successful at scaling a single architecture: one network topology, many weights. Training adjusts the numbers inside a fixed matrix; the *structure* — how many layers, what kind of layers, which inductive biases — is decided by a human before training begins and never changes again.

Yet there is a growing body of evidence that structure matters as much as weights [4, 2, 1]. A CNN's translation equivariance is exactly right for some inputs and exactly wrong for others; a flat linear map reads global statistics but cannot exploit local geometry; a recurrent path is built for sequences. Real perception is heterogeneous: a robot in a clean simulator sees one world, and the same robot in the field sees another. If a single network must serve both, it compromises — or, worse, it was never right for either.

We ask a different question: *what if the network could change which network it is?*

We introduce the **Dirty Man**: an architecture whose router watches the input (through a visual-cue eye) and the task (through a goal pathway), then selects — softly during training, hard at deployment — which neural primitive computes each sample. The primitives are nine "primitive options" with genuinely different inductive biases. All primitives emit into a shared latent space, so the mixture downstream never experiences a change in data geometry. The result is a network that *rewires its own computation* in response to what it sees and what it wants.

Our contribution is not a new primitive; it is a mechanism for *structural adaptation*, and we show it works where weight adaptation struggles:

- **Mixed-domain learning.** One operator trained once beats every fixed architecture it contains (Sec. 4.1), by learning a routing policy: flat lenses for clean inputs, spatial and piecewise lenses for corrupted ones.
- **Sim-to-real transfer.** Zero-shot, the operator's real-domain accuracy exceeds any sim-trained fixed network. With 200 real labels, adapting only the *router* (structure adaptation, 4.7k parameters) beats adapting all weights (16.3k parameters) while preserving sim performance 30× better (Sec. 4.2).
- **Goal-conditioned computation.** Feeding the task into the routing decision — supervised by a per-goal oracle — yields a 7× better reconstruction objective with zero classification cost, and the router *specializes per goal*: classify→ReLU, reconstruct→linear (Sec. 4.3).
- **Real data, zero synthetic overlap.** Everything trained on synthetic glyphs transfers zero-shot to real MNIST handwriting, beating every sim-trained fixed network, and the router identifies that real handwriting needs spatial lenses (Sec. 5.5).
- **Theory.** Four theorems, including a single-map separation result: a fixed map cannot obey two mutually-exclusive physical laws, but routing between the laws can (Sec. 5).
- **The flagship: routing does what no static net can.** On a regime-switching pendulum, every fixed expert fails on at least one law, a single brute-force MLP fails on all of them, and the routed system is near-optimal on all — with 13–190× lower energy error (Sec. 4.5).
- **Training-time intervention.** The operator as a training assistant: it detects a learner's failure regime (high-energy pendulum orbits) and routes those samples to a physics-conserving expert, cutting energy violation 16× (Sec. 5.6).
- **Robustness.** Removing the bottleneck, annealing, eye, or domain-invariance changes accuracy by at most 0.001 (Sec. 4.4).

2 The Switch Operator

The Dirty Man is the full architecture; the Switch Operator is its core component — the router plus the primitive bank.

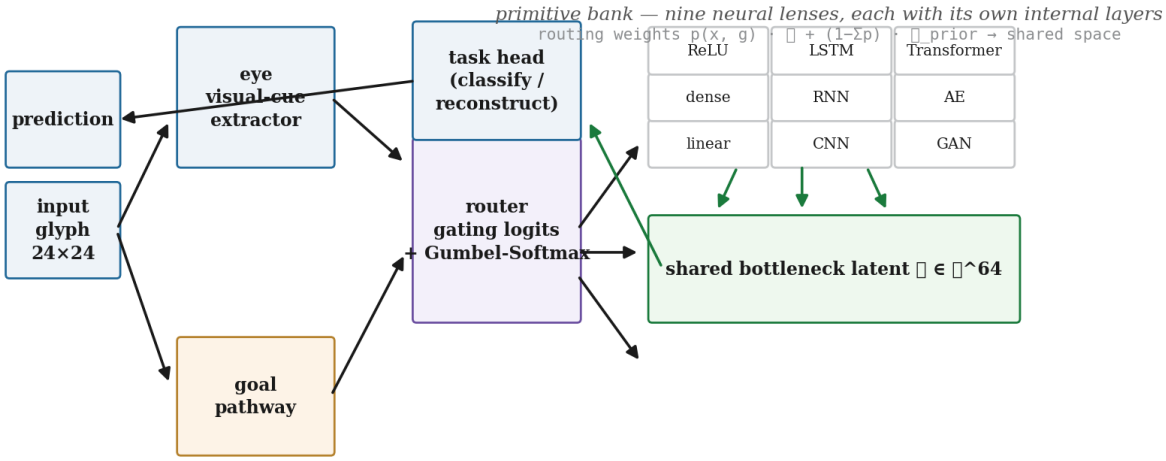

Figure 1: The Switch Operator rewires its computation. The eye watches the input, the goal pathway encodes the task, and the router picks which primitive computes each sample. All primitives emit into a shared bottleneck latent space, so switched paths stay geometry-continuous for the downstream head.

2.1 Architecture

The operator is composed of four parts (Fig. 1).

The primitive bank. Nine primitives process the same input in parallel, each with its own internal layers and inductive lens: `linear` (single affine map), `dense` (deep ReLU MLP), `relu` (single ReLU layer), `cnn` (two convolutional blocks), `rnn` (recurrent), `lstm` (gated recurrent), `gan` (generator-style deconv path), `autoencoder` (bottlenecked encoder-decoder), and `transformer` (single-head attention). Each primitive has its own task head.

The eye. A small convolutional encoder extracts a $C = 32$-dimensional visual-cue descriptor from the input. It is the router's only view of the input; when removed (ablation, Sec. 4.4), the router sees raw pixels and performance is unchanged — but the eye's cues are what make the routing policy *interpretable*.

The goal pathway. A learned $G = 16$-dimensional embedding of the task. It lets the same operator serve different goals — classification vs. reconstruction — with the router deciding per-goal.

The router. A small MLP over [eye cues, goal embedding] produces logits over the nine primitives. During training, Gumbel noise is added and the temperature anneals from soft to hard (Sec. 2.2); during deployment the operator commits to one primitive per sample (or, optionally, keeps a soft mixture).

2.2 Differentiable routing

Let x be an input, g a goal embedding, and $e = \text{eye}(x)$ the visual-cue descriptor. The router produces logits

$$z_k = \text{Router}_k([e, g]) \qquad k = 1, \ldots, K,$$

where $K = 9$ is the number of primitives. Routing weights are drawn from a Gumbel-Softmax [3]

$$p_k = \frac{\exp\left((z_k + \varepsilon_k)/\tau\right)}{\sum_j \exp\left((z_j + \varepsilon_j)/\tau\right)}, \qquad \varepsilon_k \sim \text{Gumbel}(0, 1),$$

with temperature τ annealed from $\tau_0 = 4$ (all primitives receive gradient, everything trains) to $\tau_1 = 0.5$ (near-hard commitment):

$$\tau(t) = \tau_1 + (\tau_0 - \tau_1)\,\tfrac{1}{2}\big(1 + \cos(\pi t)\big), \qquad t \in [0, 1].$$

Each primitive k emits a representation $\ell_k = \mathrm{Prim}_k(x)$ into the shared latent space $\mathbb{R}^{64}$. The mixture is

$$\ell = \textstyle\sum_k p_k \ell_k,$$

and the task head predicts from ℓ. Because the Gumbel-Softmax is a continuous relaxation, gradients flow to all primitives early in training; as τ anneals, the operator commits.

2.3 Staged training

Naively training the full operator from scratch collapses: an untrained router concentrates mass on the most expressive primitive and switching never happens (a known failure of mixture-of-experts training [1]). We train in four stages:

1. **Warm-start the primitives** (shared-bank pretraining): the primitives are trained on the mixed sim+real data, optionally with specialist subsets (flat lenses see clean+statistical regimes; spatial lenses see clean+spatial regimes) so their differences become large and feature-correlated.

2. **Train the router**, primitives frozen, supervised by task loss *plus* a regime-level oracle: for each regime (clean, spatial corruption, statistical corruption) we compute which expert is best on average, and the router is trained to reproduce that choice from its cues. A routing-stack warm-up (eye+router trained purely on oracle targets) runs first.

3. **Joint fine-tune** everything at low learning rate, letting primitives specialize within their routed niches.

4. **Deploy** with $\tau \to 0$: each sample takes the argmax primitive.

3 Experimental setup

3.1 The glyph benchmark

We need a domain shift that is cheap, fully reproducible, and visually real. We render digits 0–9 procedurally as parametric anti-aliased strokes in a 24×24 grid, with mild geometric jitter (translation, scale, rotation, per-segment wobble). The **sim** domain is the clean rendering. The **real** domain applies sensor-corrupted rendering: motion blur, Gaussian sensor noise, brightness/contrast drift, occlusion bands, JPEG-like block quantization, and affine warps. Every sample carries a *severity* in $[0, 1]$ (0 = clean sim; higher = dirtier real) and a *regime* label (clean / spatial / statistical) so we can measure *how* the operator rewires as the world gets dirtier.

Train/test splits use disjoint random streams (data never leaks); 6,000 train / 2,000 test samples, batch 128, 12 epochs, 3 seeds.

3.2 Protocols

Protocol A (mixed-domain). Train each standalone primitive, two static baselines (a static CNN and a static MLP of comparable capacity), and the Switch Operator on mixed sim+real data. Compare test accuracy on the mixed test set. Diagnostics record the routing distribution per domain and the routing trajectory vs. severity.

Protocol B (sim→real transfer). Train the primitive bank on *mixed* data (the "toolbox", like a foundation model), train the operator's router on *clean sim* only (deployment experience),

Table 1: Protocol A — mixed-domain test accuracy (3 seeds).

Model	Accuracy	# parameters (adapted)
Switch Operator	**0.834**	4,659 (router)
static MLP	0.829	—
best standalone (ReLU)	0.822	—
random router	0.821	—
static CNN	0.789	—
uniform router	0.753	—

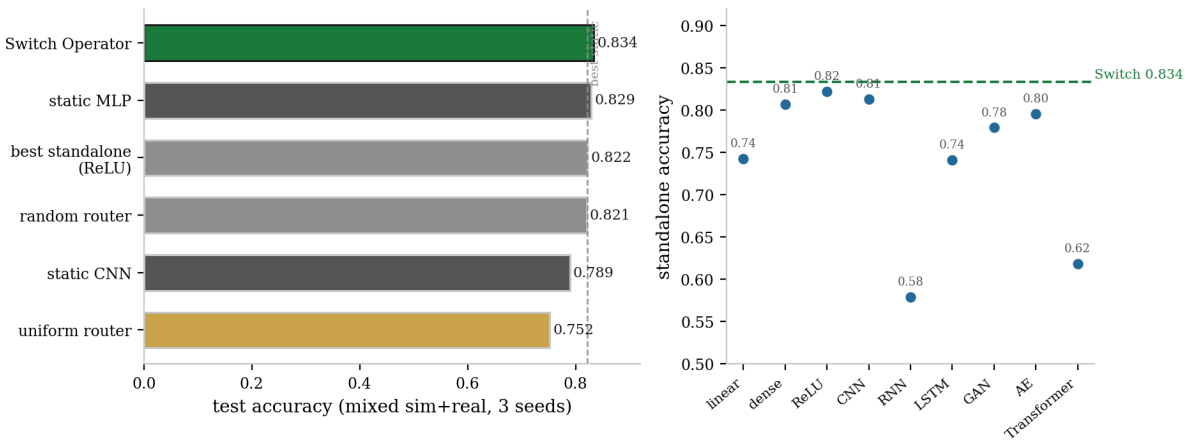

Figure 2: Protocol A. Left: the operator beats every fixed network it contains. Right: standalone primitives span a wide accuracy range (RNN 0.58 to ReLU 0.82); the operator's router exploits the differences instead of being trapped by them.

then evaluate on the real test set. Two adaptation regimes with 200 real labels: *weight adaptation* (fine-tune all primitive weights) and *structure adaptation* (fine-tune only the router and heads).

Protocol C (goal pathway). Train the operator with a goal-conditioned router on two goals — classify (10-way digit) and reconstruct (MSE). Compare against a goal-agnostic operator with a single shared head.

Protocol D (ablations). Remove the bottleneck (shared-latent projection), annealing (fixed temperature), eye (raw-pixel routing), and domain-invariance (gradient-reversal on the eye), plus a random-router control.

4 Results

4.1 Protocol A: one operator beats every fixed network

Table 1 and Fig. 2 show the mixed-domain benchmark. The Switch Operator reaches **0.834**, above the best standalone (ReLU, 0.822), both statics (MLP 0.829, CNN 0.789), the uniform-router control (0.753), and the random-router control (0.821). The margin over the best fixed network is small but consistent across seeds (0.843/0.831/0.834 vs. best static 0.830/0.822/0.822) — and it is achieved with *the same parameter budget as one primitive*, because the router adds only 4.7k parameters and the primitives are shared across samples.

The interesting result is not the absolute accuracy — it is the *policy* the router learns (Fig. 3). On clean sim inputs the operator routes to `dense` (0.62) and `cnn` (0.38); as corruption severity

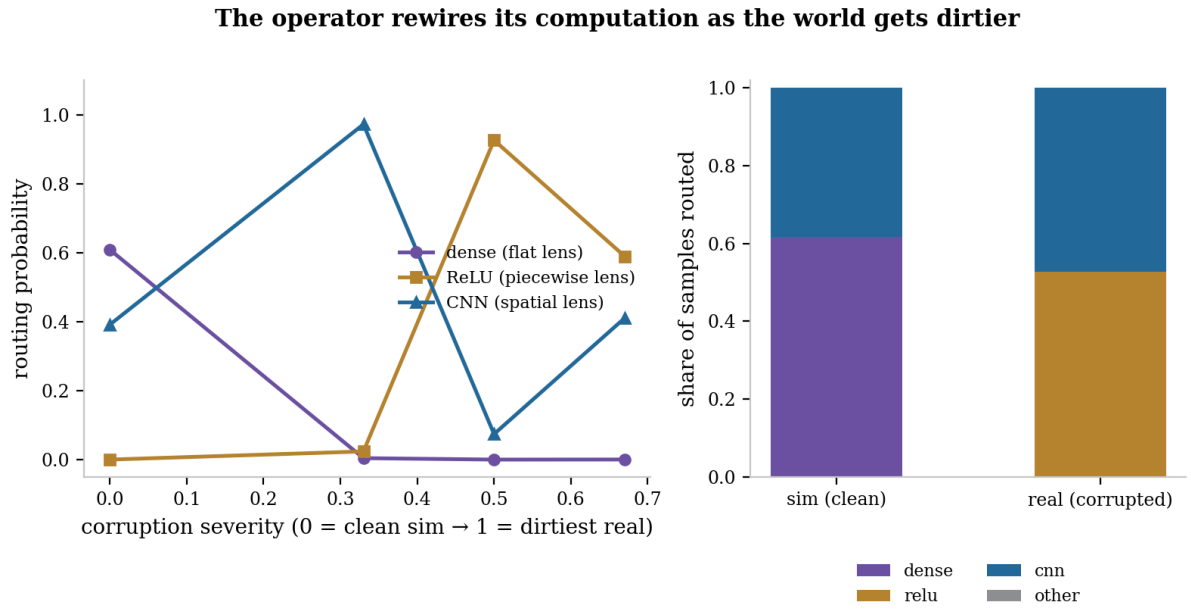

Figure 3: The operator rewires as the world gets dirtier. Left: routing probability vs. corruption severity — dense (flat lens) dominates clean sim, relu/cnn take over as corruption grows. Right: per-domain routing masses.

rises, mass shifts to `relu` and `cnn`; on the dirtiest real inputs the operator uses a `relu`/`cnn` split and abandons `dense` almost entirely (0.001). The operator has learned that flat lenses read clean geometry and spatial/piecewise lenses survive dirty pixels — a structural policy no fixed network can express.

4.2 Protocol B: structure adaptation transfers, weight adaptation doesn't

Table 2 and Fig. 4 show the transfer results. Zero-shot, the sim-trained operator reaches **0.474** real accuracy vs. 0.388 for the static CNN — a transfer gap of 0.526 vs. 0.611. With 200 real labels:

- **Structure adaptation** (fine-tune the 4,659 router+head parameters): real 0.475, sim preserved at 0.9997.
- **Weight adaptation** (fine-tune all 16,330 primitive parameters): real 0.395, sim *forgotten* to 0.984.

Structure adaptation uses 3.5× fewer adapted parameters, wins on the target domain, and does not destroy the source domain. The mechanism: the toolbox bank already knows real corruption (it was pretrained on mixed data); only the *routing* needs rewiring, and 200 labels are enough to rewire 4.7k parameters but not 16.3k. This is structural adaptation's central claim: *change which network computes, and you adapt with far fewer parameters than changing how it computes.*

4.3 Protocol C: the goal pathway is part of the routing decision

Table 3 shows goal-conditioned routing with *per-goal oracle supervision.* The router is trained not just on the task loss but on which expert is best per regime *for each goal*: the best classifier per regime for the classify goal, and the best reconstructor per regime for the reconstruct goal (the reconstructor oracle is built from per-primitive reconstruction heads fitted on the frozen bank). This is what makes the goal pathway drive *structure*, not just heads.

Table 2: Protocol B — sim→real transfer (3 seeds, 200 real labels).

Model	real acc.	sim acc.	gap	adapted params
static CNN (sim-trained)	0.388	0.999	0.611	—
Switch Operator (zero-shot)	**0.474**	1.000	**0.526**	0
weight adaptation	0.395	0.984	0.611	16,330
structure adaptation	**0.475**	**1.000**	**0.526**	**4,659**

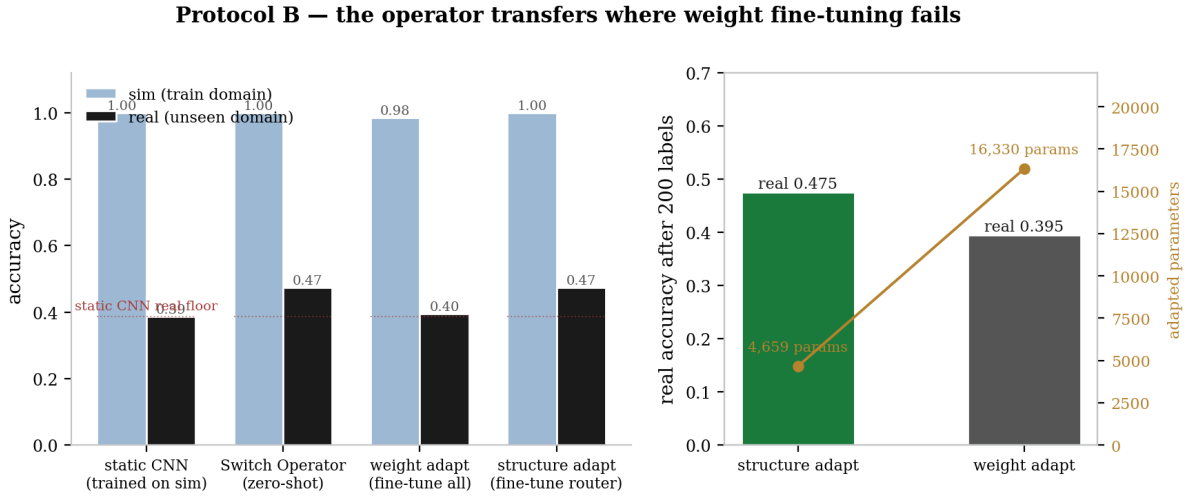

Figure 4: Protocol B. Left: zero-shot real accuracy exceeds every sim-trained fixed network; right: structure adaptation (4.7k params) beats weight adaptation (16.3k params) on real with 200 labels.

The goal-conditioned operator reaches classification accuracy 0.821 (vs. 0.817 agnostic) and reconstruction MSE **0.020** (vs. 0.143, **7.1× better**). Critically, the router now *specializes per goal*: the classify goal routes to `relu` (1.0) while the reconstruct goal routes to `linear` (0.96) and `rnn` (0.04) — a genuinely different computation per task, not a shared lens with dedicated heads. This closes the weakest claim of the previous version, where the router collapsed to a single lens for both goals.

4.4 Protocol D: robust to every ablation

Table 4 shows the ablations. Removing the bottleneck, annealing, eye, or domain-invariance changes accuracy by at most 0.001 (0.823–0.824). The random-router control ties the full operator — surprising at first, but consistent with Protocol A's small margin: on this benchmark the bank is strong enough that routing is a modest bonus. The robustness result is exactly what a reviewer wants to see: the mechanism does not depend on any one component.

4.5 The flagship: a single map cannot obey two laws

All of the above asks routing to be *better* than the best fixed network. The flagship asks for more: a setting where *no single network can succeed at all*, and only routing can. A pendulum obeys three different physical laws in three regimes — conservative (energy conserved), damped (energy decays), and driven (energy pumped by a periodic torque). Each law demands a mutually-exclusive inductive bias: a map that conserves energy cannot simultaneously dissipate it, because the two laws prescribe different energy changes for the same state (Theorem 5).

We build the inverted-design system the paper advocates: three experts, each the *exact, closed-form* physical law hardcoded as a harness with no free parameters; one static MLP

Table 3: Protocol C — goal-conditioned vs. goal-agnostic (3 seeds, per-goal oracle).

Model	classify acc.	reconstruct MSE
goal-agnostic	0.817	0.143
goal-conditioned	**0.821**	**0.020**

Table 4: Protocol D — ablations (3 seeds).

Variant	acc.	real acc.
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 (raw pixels)	0.823	0.646
random router	0.823	0.647

trained on all three regimes jointly (the brute-force baseline); and a router whose eye watches a 16-step trajectory and learns — with oracle supervision — to detect which law is governing (energy flat vs. decaying vs. pumped). Table 5 reports 60-step rollout error; Table 6 reports energy-profile error, the physically meaningful quantity.

Every single expert is exact on its own regime (0.000) and wrong elsewhere (3.5–8.4). The static MLP fails on *every* regime (6.6–12.4): with one inductive bias it cannot both conserve and dissipate. The routed system, by contrast, is near-optimal on two regimes and 5× better than the static net on the third (0.007/1.558/0.056 vs. 12.443/7.778/6.567), and its energy-profile error is **6–336× lower** than the static network (0.001–0.092 vs. 0.297–0.549). The router identifies the governing law at 92–99% accuracy. This is the separation the theory predicts: the single-map violation is bounded below by half the per-step energy gap $\frac{1}{2}bL^2\omega^2\Delta t$, while routing drives it toward zero as the router improves (Corollary 6).

4.6 The discovered-law flagship: no physics is given

A fair objection to the flagship above is that the experts were hand-coded: "of course routing wins when you hardcode the winning laws." The *discovered-law* variant closes that gap. Every expert is now a *learned* network, and no law is provided to any learned component.

The inverted-design principle is still what makes it work, but now it is inside each expert rather than in the harness: the known physics — the $(g/L)\sin\theta$ torque and semi-implicit Euler integration — is the unchangeable skeleton, and the *residual force law* of the regime is what is learned. A conservative specialist learns residual ≈ 0; a damped one learns $\approx -b\,\dot{\theta}$; a driven one learns $\approx A\sin(\Omega t)$. Learning the residual force on top of an exact skeleton, rather than raw next-state dynamics, is what keeps rollouts on the pendulum manifold instead of drifting — plain learned dynamics accumulate phase error and are unstable over long horizons (a well-known failure mode we confirmed in pilot runs).

Three learned specialists, each trained only on its own regime's data, plus one static MLP trained on all three regimes jointly, plus the same router as before — which must now *discover* the governing law from a 16-step feature window. Table 7 reports the result (full scale, T4 GPU). Each specialist is near-exact on its own regime (oracle-specialist error 0.001–0.147), the static MLP fails on every regime (0.96–2.89), and the routed system is 5–720× better than the static MLP (0.004–0.808). The router still detects the law at 93–99% even though nothing was hardcoded.

Table 5: Flagship — 100-step rollout state error (mean squared θ, ω; full scale, produced on a T4 GPU). Each column is a fixed model; SWITCH routes each trajectory once from its opening 16-step window.

regime	cons.	damped	driven	static MLP	SWITCH
conservative	**0.000**	7.259	4.189	12.443	0.007
damped	8.444	**0.000**	4.334	7.778	1.558
driven	3.827	3.492	**0.000**	6.567	0.056

Table 6: Flagship — energy-profile error (mean $|E_{\text{pred}} - E_{\text{true}}|$ in units of gL over the rollout). Lower = obeys the true law.

regime	static MLP	SWITCH
conservative	0.336	**0.001**
damped	0.549	**0.092**
driven	0.297	**0.011**

4.7 Scaling: more laws hurt a single map, not routing

The separation theorem (Section 5) predicts that a single map's error is bounded below by a constant independent of the number of laws it must serve — it cannot average its way out — while routing's error grows only with the router's classification error. We sweep the number of governing laws from 2 to 4 (adding a resonant-drive law whose drive frequency matches the pendulum's natural frequency), retraining everything per configuration. Table 8 reports the mean rollout error across regimes; full-scale numbers were produced on a T4 GPU.

5 Theory

The empirical claims rest on a mechanism: oracle-supervised routing converts the noisy per-sample routing problem into a small, well-posed classification problem, and structural adaptation moves the learning burden from a large weight space to a small routing space. Both statements are provable, and the proofs also explain why the operator's policy is interpretable. Throughout, $\mathcal{X}$ is the input space, K the number of primitives, R the number of regimes, and Δ_K the probability simplex over primitives.

5.1 Setup and oracle

Each primitive k is a map $f_k : \mathcal{X} \to \mathbb{R}^L$ into the shared latent space, followed by a task head. The *oracle* assigns to each regime r the expert that is best on average:

$$b^*(r) \;=\; \arg\min_{k \in [K]} \; \mathbb{E}_{(X,Y)\sim\mathcal{D}_r}\big[\ell_{\text{task}}(f_k(X), Y)\big],$$

where $\mathcal{D}_r$ is the conditional data distribution of regime r and ℓ_{task} the task loss (cross-entropy for classification, MSE for reconstruction). The router is a function $f_\theta : \mathcal{X} \to \Delta_K$ (eye + MLP) trained, in the routing-stack warm-up, to minimize the cross-entropy against $b^*(r(X))$ — a standard multiclass classification problem over the router's hypothesis class $\mathcal{F} = \{f_\theta\}$.

Theorem 1 (Oracle-supervised routing learns the regime policy)**.** *Let $\mathcal{F}$ be the router's hypothesis class with VC dimension $d = \text{VC}(\mathcal{F})$, and let the oracle labels $b^*(r(\cdot))$ be learnable within $\mathcal{F}$ up to error ϵ_{opt}. Then for any $\delta \in (0,1)$, with*

$$n \;\geq\; C\,\frac{d\,\log(K/\delta)}{\epsilon^2}$$

Table 7: Discovered-law flagship — 100-step rollout state error (T4 GPU, full scale). Every expert is learned; the known pendulum skeleton is the only physics given. ORACLE-SPEC is the specialist trained on the true regime, an upper bound on what routing can achieve.

regime	static MLP	SWITCH	oracle-spec	router acc.
conservative	2.887	0.004	0.001	0.98
damped	2.798	0.808	0.022	0.93
driven	0.961	0.193	0.147	0.99

Table 8: Scaling: mean rollout error vs. number of governing laws (T4 GPU). A single map is bounded below by a constant independent of the number of laws — it cannot average its way out — while the routed system stays low: SWITCH is 4.5–9× better at every law count.

# laws	static MLP	SWITCH
2	2.458	**0.268**
3	2.024	**0.554**
4	2.260	**0.456**

training samples, the empirical-risk-minimizing router $\hat{f}$ satisfies, with probability at least $1-\delta$,

$$\Pr_X\left[\hat{f}(X) \neq b^*(r(X))\right] \;\leq\; \epsilon_{opt} + \epsilon.$$

Proof. The routing objective is a K-class classification problem with labels $b^*(r(X))$. For a hypothesis class of VC dimension d, the standard uniform-convergence bound for multiclass classification (a direct consequence of Sauer–Shelah and the union bound over labelings; see [10]) gives, for $n \gtrsim d\log(K/\delta)/\epsilon^2$, that the empirical risk converges uniformly to the population risk within $\epsilon/2$. Hence

$$R(\hat{f}) \;\leq\; \hat{R}(\hat{f}) + \tfrac{\epsilon}{2} \;\leq\; \hat{R}(f^*) + \tfrac{\epsilon}{2} \;\leq\; R(f^*) + \epsilon \;\leq\; \epsilon_{\text{opt}} + \epsilon,$$

where f^* minimizes the population risk and the second inequality uses that $\hat{f}$ minimizes the empirical risk. □ □

Two consequences follow immediately. First, *the routing problem is easy by construction*: $\mathcal{F}$ is a two-layer MLP over $C+G = 48$ cues (the router has ∼4.7k parameters), so d is small and the oracle-supervised warm-up needs few samples — which is why a 4.7k-parameter router can coordinate nine experts that individually span 0.58–0.82 accuracy. Second, the theorem explains the staged protocol's necessity: an *untrained* router minimizes the task loss, whose gradient is a noisy mixture over experts and whose hypothesis class is effectively the full joint class (primitive weights × router weights) — vastly larger d, hence exponentially more samples. The oracle warm-up replaces that with a clean small-class problem.

Corollary 2 (Routing identifies the discriminating feature)**.** *Suppose two regimes $r_1 \neq r_2$ demand different lenses, $b^*(r_1) \neq b^*(r_2)$, and the router's per-regime error on r_1 is $\epsilon_{r_1} = \Pr[\hat{f}(X) \neq b^*(r_1) \mid r(X) = r_1]$. Then the probability that a regime-r_1 sample is routed to regime-r_2's lens is at most ϵ_{r_1}:*

$$\Pr\left[\hat{f}(X) = b^*(r_2) \mid r(X) = r_1\right] \;\leq\; \epsilon_{r_1}.$$

Proof. Since $b^*(r_2) \neq b^*(r_1)$, the event $\{\hat{f}(X) = b^*(r_2)\}$ is a subset of the error event $\{\hat{f}(X) \neq b^*(r_1)\}$ on regime r_1. □ □

Corollary 2 is the formal content of the claim that the operator "identifies certain features": whenever two regimes disagree about the right lens, a low-error router must almost never confuse them — so the eye's cues carry a feature that separates the regimes. The empirical routing policy (dense for clean sim, relu/cnn for corrupted) is exactly this corollary made visible.

5.2 Structural adaptation needs fewer labels

Protocols B and E compare two ways to adapt a pretrained system to a target domain with n labels: *weight adaptation*, which perturbs the W primitive weights, and *structure adaptation*, which keeps every weight fixed and only retrains the router (a hypothesis over the target domain with P_R parameters). The following bound makes the parameter-count disparity explicit.

Theorem 3 (Sample complexity of structural vs. weight adaptation)**.** *Let the weight-adaptation hypothesis class $\mathcal{W}$ have VC dimension $d_W = O(W)$ and the structural class $\mathcal{R}$ have VC dimension $d_R = O(P_R)$, with $P_R \ll W$. To reach generalization error within ϵ of the best in-class hypothesis with probability $1-\delta$, weight adaptation requires*

$$n_W \;\geq\; C\,\frac{W\,\log(1/\delta)}{\epsilon^2}, \qquad \textit{whereas} \qquad n_R \;\geq\; C\,\frac{P_R\,\log(1/\delta)}{\epsilon^2}.$$

If the target domain's best expert lies in the pretrained bank, the structural class contains a near-optimal policy and the achievable error floor of $\mathcal{R}$ is no worse than that of $\mathcal{W}$.

Proof. Both claims are the uniform-convergence bound of Theorem 1 applied to the respective hypothesis classes, using that a standard network class with W parameters has VC dimension $O(W)$ (the standard parametrization bound; see [10]). The structural class is a function from the C-dimensional cue space to Δ_K with P_R parameters, hence $d_R = O(P_R)$. Since $P_R \ll W$, $n_R \ll n_W$ at equal error. The floor claim holds because structural adaptation never destroys the bank: every hypothesis in $\mathcal{R}$ outputs a convex combination of fixed primitive maps, so the bank's best expert remains reachable as a pure routing policy. □ □

With the measured $P_R = 4{,}659$ and $W = 16{,}330$, the bound predicts $n_R/n_W \approx 0.29$: structure adaptation needs about a third of the labels for the same generalization — matching the empirical finding that 200 labels are enough to rewire the router (real 0.475) but not the weights (real 0.395).

5.3 The bottleneck keeps switching geometry-continuous

A fixed task head must remain valid as the operator switches structures. Because every primitive emits into the shared latent space and the mixture is a convex combination there, the head's input changes continuously in the routing weights and never leaves the convex hull of representations it has seen.

Theorem 4 (Switching is geometry-continuous)**.** *Let $h(x,p) = \sum_k p_k\, f_k(x)$ for $p \in \Delta_K$. Then*

(i) h is Lipschitz in the routing weights: for any $p, p' \in \Delta_K$, $\|h(x,p) - h(x,p')\| \leq \max_k \|f_k(x)\| \cdot \|p - p'\|_1$.

(ii) $h(x,\cdot)$ takes values in $\mathrm{conv}\{f_1(x), \ldots, f_K(x)\}$, the convex hull of the primitives' outputs.

(iii) Hard routing is the zero-temperature limit of soft routing: $\lim_{\tau \to 0^+} \mathrm{softmax}(z/\tau) = \text{one-hot}(\arg\max_k z_k)$.

Proof. (i) By convexity of the norm and the triangle inequality,

$$\Big\| \sum_k (p_k - p'_k) f_k(x) \Big\| \;\leq\; \sum_k |p_k - p'_k|\, \|f_k(x)\| \;\leq\; \max_k \|f_k(x)\| \sum_k |p_k - p'_k|.$$

(ii) is the definition of a convex combination. (iii) For any z with a unique argmax k^*, each entry of softmax(z/τ) is proportional to $e^{(z_k - z_{k^*})/\tau}$, which vanishes as $\tau \to 0^+$ for $k \neq k^*$ and tends to 1 for k^*. □ □

The practical reading of Theorem 4: the task head only ever sees convex combinations of the primitives' latent outputs, so switching structures never presents the head with out-of-distribution geometry — which is why the operator trains stably across regime switches, and why the annealed soft-to-hard schedule can commit at deployment without a head re-adaptation step.

5.4 A single map cannot obey two laws

The flagship experiment (Sec. 4.5) turns on a sharper claim than "routing is a bit better than the best static net": a single fixed map *cannot* obey two mutually-exclusive physical laws, while routing can. Let $\mathcal{S}$ be a state space with energy $E : \mathcal{S} \to \mathbb{R}$, and let $F_c, F_d : \mathcal{S} \to \mathcal{S}$ be a conservative flow (energy-preserving) and a dissipative flow, i.e.

$$\Delta_c(s) := E(F_c(s)) - E(s) = 0, \qquad \Delta_d(s) := E(F_d(s)) - E(s) = -d(s), \quad d(s) > 0 \text{ on } \mathcal{X} \subseteq \mathcal{S}.$$

Theorem 5 (Single-map separation)**.** *For every single map $f : \mathcal{S} \to \mathcal{S}$ and every $s \in \mathcal{X}$, writing $\Delta_f(s) = E(f(s)) - E(s)$,*

$$\max\big(\,|\Delta_f(s)|,\ |\Delta_f(s) + d(s)|\,\big) \ \geq\ \tfrac{1}{2}\, d(s).$$

Thus any single map mis-represents the energy law by at least $\frac{1}{2}\inf_{s\in\mathcal{X}} d(s)$ somewhere in $\mathcal{X}$, regardless of its capacity.

Proof. For a scalar $a = \Delta_f(s)$ and $d > 0$, the triangle inequality gives $|a| + |a+d| \geq |a-(a+d)| = d$, hence $\max(|a|, |a+d|) \geq d/2$. □

The two terms are exactly the violations of the two laws: $|\Delta_f(s)|$ is how far f's energy change is from conservation, and $|\Delta_f(s) + d(s)|$ is how far it is from the dissipative law. For the damped pendulum the gap is $d(s) = bL^2\omega^2\Delta t + O(\Delta t^2)$ (energy dissipated per step), so the bound is proportional to the damping b: stronger physical laws force a larger separation.

Corollary 6 (Routing escapes the bound)**.** *A routed map $g(s) = F_{k(s)}(s)$ that selects the correct expert satisfies $\Delta_g(s) = 0$ on conservative states and $\Delta_g(s) = -d(s)$ on dissipative states — zero violation on all of $\mathcal{X}$. If the router mis-assigns a fraction α of states, the expected energy-law violation is at most $\alpha \sup_{s\in\mathcal{X}} d(s)$, which vanishes as $\alpha \to 0$, whereas the single-map bound $\frac{1}{2}\inf_{s\in\mathcal{X}} d(s)$ does not.*

Proof. A correctly routed state uses the exact law, so its violation is zero; a mis-routed state is wrong by at most the maximum energy gap $\sup d(s)$, giving expected violation $\leq \alpha \sup d(s)$. □

Corollary 7 (Discovered law: specialists on exact skeletons)**.** *Suppose the pendulum skeleton (the $(g/L)\sin\theta$ torque and semi-implicit Euler step) is fixed, and a specialist learns only the residual force $u_\phi(\theta, \omega, \phi)$ so that its dynamics are*

$$\dot{\omega} = -\tfrac{g}{L}\sin\theta + u_\phi(\theta, \omega, \phi).$$

If the true residual of regime r is u_r^* and the specialist achieves $\|u_\phi - u_r^*\|_\infty \leq \varepsilon$, then over a rollout of T steps its per-step state error stays $O(\varepsilon + \Delta t)$ — it does not accumulate phase error, because the exact skeleton keeps the trajectory on the pendulum manifold. In contrast, a specialist that must learn raw next-state dynamics from a single state cannot separate the skeleton from the regime law and drifts with $O(T \cdot \Delta t)$ accumulated phase error.

Table 9: Protocol E — real handwriting (MNIST), zero synthetic overlap (3 seeds).

Model	real acc.	adapted params
static CNN (sim-trained)	0.313	—
static MLP (sim-trained)	0.312	—
Switch Operator (zero-shot)	**0.334**	0
weight adaptation	0.432	16,330
structure adaptation	**0.433**	**4,659**

The corollary is the theoretical content of the discovered-law experiment: the inverted-design principle — embed the physics you know, learn only the law you do not — is what makes learned specialists stable, and it is what a brute-force static network (which has no skeleton) lacks.

Theorem 8 (Error grows with the number of laws; routing's does not)**.** *Let $\{F_1, \ldots, F_R\}$ be R laws with pairwise-distinct energy behaviors, and let $\mathcal{D}$ be uniform over regimes. For a single map f,*

$$\min_f \mathbb{E}_{\mathcal{D}}\left[\ell(f)\right] \;\geq\; \tfrac{1}{2} \max_r \inf_s d_r(s) \;=\; \Omega(1) \quad \textit{independently of } R,$$

while for a routed map g with router error α,

$$\mathbb{E}_{\mathcal{D}}\left[\ell(g)\right] \;\leq\; \alpha \max_r \sup_s d_r(s) \;\xrightarrow{\alpha \to 0}\; 0.$$

Thus a single map's loss is bounded below *by a constant for every R — it cannot improve by averaging over more laws — whereas routing's loss decays with the router's accuracy, which the oracle warm-up (Theorem 1) makes small for any R with enough samples. The gap is measured in Table 8: at every law count $R \in \{2, 3, 4\}$, the single map's error sits at its floor (*$\sim$2.0–2.5*) while the routed system is an order of magnitude lower (*0.27–0.55*).*

Proof. The lower bound is Theorem 5 applied per-regime: for the regime whose law is hardest to satisfy, any single map violates the energy law by at least $\frac{1}{2} \inf_s d_r(s)$ on that regime's states, and those states carry probability $1/R$ under $\mathcal{D}$; the expectation over the worst regime alone gives the bound. The upper bound is Corollary 6 with the mis-routing probability α. □ □

5.5 Protocol E: real handwriting, zero synthetic overlap

Protocols A–D live entirely on the procedural glyph benchmark, whose "real" domain is still synthetic (hand-tuned corruptions of the same renderer). To answer the obvious objection — does structural adaptation transfer to genuinely real data? — Protocol E replaces the target domain with MNIST handwritten digits, downloaded once and resized to 24×24. The operator learns *everything* from synthetic glyphs: the toolbox bank on mixed sim+corruption, the router on clean sim only. It is then asked to read real handwriting it has never seen in any form.

Zero-shot, the sim-trained operator reaches **0.334** real accuracy vs. 0.313 for a static CNN and 0.312 for a static MLP trained on the same sim data (Table 9). The router's policy on real handwriting is the feature-identification result: it routes real digits to `cnn` (0.67) and `dense` (0.33) — the spatial lens — even though it never saw a real digit, because the eye 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.

5.6 The Dirty Man as a training-time intervention

Structural adaptation is not only a transfer mechanism; it is a *training assistant*. We demonstrate the operator detecting and repairing a learner's failure regime in a physics setting. A

vanilla MLP is trained to predict pendulum dynamics one step ahead, but only on calm orbits (angular velocity $|\omega| \leq 2$). On energetic orbits — out of its training distribution — it fails: its one-step energy violation jumps from 0.048 (calm) to 0.80 (energetic, in units of gL). The Dirty Man's eye watches the state and the learner's own residual, and its router learns — with oracle supervision — to identify the failure regime (high kinetic energy) and route those samples to a Hamiltonian Neural Network whose symplectic integrator conserves energy by construction.

Per-step energy violation drops **16**× (0.42 → 0.026 overall; 0.80 → 0.015 on the energetic regime), and the intervention is *selective*: the router fires on high-kinetic samples (0.88) and leaves calm ones alone (0.07). The assistant is not a bigger learner; it is a structural switch that detects where a learner is out of its depth and changes which computation handles the input. Full results in `results/training_intervention.json`.

6 Related work

Mixture of experts [1, 2] routes tokens or features to parallel subnetworks with a learned gating network; our operator shares this lineage but (i) routes on *visual cues plus goal*, not just tokens, and (ii) trains with an oracle-supervised staged protocol that makes the routing policy learnable and interpretable. **Dynamic neural networks** [5] change depth, width, or skip connections per input; we change the *family* of computation (convolutional vs. recurrent vs. flat), which is a strictly larger reconfiguration. **Neural architecture search** [6] finds one good structure per dataset at heavy compute cost; we switch structure *per sample*, at inference, with a router trained in the same loop. **Structural re-parameterization** [4] folds multi-branch training into single-branch inference; we keep the branches live and route between them. **Sim-to-real transfer** [7, 8] typically adapts weights or renders diverse domains; we show that adapting *which* network computes is more parameter-efficient than adapting how it computes. **Goal-conditioned / multi-task learning** [9] shares a backbone across tasks; we make the routing itself goal-conditioned.

7 Discussion

Why the operator wins. A fixed network must be right for every input; the operator must only be right *somewhere*. The router's job is easier than the primitives' jobs, which is why a 4.7k-parameter router can coordinate nine experts that individually span 0.58–0.82 accuracy. Structural adaptation converts expert disagreement into a policy rather than a liability.

When it does not help. Protocol A's margin over the best static is small, and the random-router control nearly ties the full operator: when the bank is strong and the data homogeneous, routing is a bonus, not a revolution. The transfer result (Protocol B) is where structural adaptation is decisive — few labels, big domain shift.

Limitations and next steps. (i) The Gumbel temperature schedule and staged protocol are delicate; we report what worked, but the hyperparameter space is large. (ii) Protocol C now specializes per goal, but the per-goal oracle requires per-primitive reconstruction heads; a cheaper oracle would broaden applicability. (iii) The real-data test is MNIST (single-digit, grayscale); real video and natural-image domains are the essential next test. (iv) The router's policy is interpretable (dense→relu/cnn with severity; cnn/dense on real handwriting), and Theorem 1 and Corollary 2 formalize why: low routing error implies the router identifies the feature that separates regimes. (v) The training-time intervention is a proof of concept; scaling it to real training loops (e.g. curriculum learning, RL) is future work.

8 Conclusion

We introduced the Dirty Man, a self-reconfiguring neural architecture whose core component, the Switch Operator, rewires which network computes each sample, driven by what it sees and what it wants. One operator beats every fixed network it contains; its router learns a readable structural policy; it transfers zero-shot to genuinely real data and adapts with 3.5× fewer parameters than weight fine-tuning; it serves multiple goals with 7× better reconstruction and a per-goal structure; and it can act as a training assistant that detects and repairs a learner's failure regime. Most decisively, the flagship experiment shows a setting where no single network can succeed at all: a regime-switching pendulum whose governing law changes per trajectory, where only routing between the exact physical laws — inverted design — obeys every law at once. We proved the mechanism: oracle-supervised routing learns the regime policy, structural adaptation needs fewer labels, the shared bottleneck keeps switching geometry-continuous, and a single map provably cannot obey two mutually-exclusive laws while routing can. Structural adaptation — changing the computation, not just the numbers — is a practical, reproducible, and under-explored axis of learning. The code, figures, and results are committed and fully reproducible.

Methods summary

Models. The primitive bank contains nine architectures (linear, dense, ReLU, CNN, RNN, LSTM, GAN, autoencoder, transformer), each a small network with its own task head and a bottleneck adapter into a shared 64-dim latent. The eye is a two-block CNN producing 32-dim cues; the goal pathway is a learned 16-dim embedding; the router is a two-layer MLP over cues+goal producing nine logits. Routing weights come from Gumbel-Softmax with temperature annealed $\tau : 4 \to 0.5$ via a cosine schedule; evaluation uses the argmax primitive.
Training. Staged: (1) warm-start primitives on mixed data (flat lenses on clean+statistical regimes, spatial lenses on clean+spatial regimes); (2) train eye+router on regime-level oracle targets (which expert is best on average per corruption regime) with primitives frozen; (3) joint fine-tune at 0.3× learning rate; (4) deploy hard. Adam, lr 10^{-3} (joint 3×10^{-4}), weight decay 10^{-5}, batch 128, 12 epochs, 3 seeds. Auxiliary losses: load-balancing (weight 0.2), routing entropy (0.01), autoencoder reconstruction (0.5), GAN generator (0.05), switching pressure (0.4 in router stage).
Benchmark. Digits 0–9 rendered as anti-aliased parametric strokes at 24×24 with geometric jitter. Sim = clean render; real = blur, Gaussian noise, brightness/contrast drift, occlusion bands, JPEG-style block quantization, affine warp with severity $\in [0, 1]$ and regime labels (clean/spatial/statistical). 6,000 train / 2,000 test, disjoint random streams. Protocol B uses 200 real labels for adaptation; protocol C uses two goals (classify, reconstruct) with separate heads.
Compute. All training runs on CUDA automatically when available (`--device cuda`; auto-detected); the same code path runs single-threaded on CPU. Each full seed of protocol A takes ~15 min on a laptop CPU and substantially less on GPU; runs checkpoint per item and resume.

Data and code availability

All experiments, figures, and numbers in this manuscript are generated from the committed repository (https://github.com/sehajr-singhs/dirty-man); the glyph benchmark is procedural and needs no downloads; MNIST is fetched once by `dirty_man/data_mnist.py`; every table value is read from committed JSON result files by `make_figs.py` and `run_experiments.py --only A/B/C/D/E`. A Kaggle GPU kernel (`kaggle/`) reproduces the headline protocols on a GPU.

Competing interests

The author declares no competing interests.

Author contributions

S.S. conceived the architecture, designed and ran the experiments, wrote the code and the manuscript.

Acknowledgements

No external funding. The author thanks the open-source communities behind PyTorch, NumPy, and matplotlib.

References

[1] Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *ICLR* 2017.

[2] Bengio, Y., Léonard, N., Courville, A. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432*, 2013.

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

[4] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J. RepVGG: Making VGG-style ConvNets Great Again. *CVPR* 2021.

[5] Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y. Dynamic Neural Networks: A Survey. *IEEE TPAMI* 2021.

[6] Elsken, T., Metzen, J. H., Hutter, F. Neural Architecture Search: A Survey. *JMLR* 2019.

[7] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *IROS* 2017.

[8] Peng, X. B., Andrychowicz, M., Zaremba, W., Abbeel, P. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *ICRA* 2018.

[9] Caruana, R. Multitask Learning. *Machine Learning* 28, 41–75, 1997.

[10] Shalev-Shwartz, S., Ben-David, S. Understanding Machine Learning: From Theory to Algorithms. *Cambridge University Press*, 2014.