SCULPT: BACKPROPAGATION-FREE LINEAR-HEAD ADAPTATION WITH SYMBOLIC CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-differentiable solvers and rule engines provide structured feedback on neural predictions, yet translating this feedback into lightweight post-deployment updates remains challenging. We introduce Symbolic Constraint Updates for Lightweight Predictor Tuning (SCULPT), a backpropagation-free framework for adapting linear prediction heads over frozen representations. SCULPT leverages an exact correspondence between linear inequalities on prediction scores and half-spaces in the head-parameter space, enabling direct computation of parameter updates. To accommodate feedback with varying reliability and structure, SCULPT combines reliability-gated directions from violated constraints with covariance-aware fitting of consensus-filtered solver targets. A shared feedback router supports either feedback type or their combination, with an optional trust projection for drift control. Experiments on routing, visual Sudoku, and corrupted digit addition demonstrate that SCULPT effectively incorporates symbolic feedback to improve prediction without retraining representations or differentiating through the checker.

1 INTRODUCTION

Many deployed neural systems make a prediction and then invoke a program that checks whether the prediction is usable. Shortest-path solvers, Sudoku engines, schedulers, and collision monitors can provide exact task feedback, yet their branching, search, and contact logic is not naturally differentiable. This paper studies a practical repair problem: given a useful frozen representation, can a non-differentiable checker improve only the deployed linear head without retraining the backbone or differentiating through the checker?

Existing approaches expose different interfaces. Logical satisfaction objectives and probabilistic logic provide differentiable learning signals (Xu et al., 2018; Manhaeve et al., 2018), while optimization layers and continuous relaxations expose gradients through constrained or discrete choices (Amos & Kolter, 2017; Jang et al., 2017; Maddison et al., 2017). Evolutionary strategies estimate update directions through repeated black-box evaluations (Salimans et al., 2017). Solver self-training converts decoded structures into pseudo-labels, but its gradient update can reinforce a confident yet wrong decode. The unresolved issue is therefore not merely the absence of gradients; it is how to exploit directional checker feedback when its scale, coherence, and reliability vary across tasks.

Our key observation is that the frozen-head setting exposes exact geometry. If $y = W\phi(x)$ and a checker returns $a^\top y \geq b$, then the acceptable parameters satisfy $\langle a\phi(x)^\top, W\rangle_F \geq b$. This converts a non-differentiable program response into a linear constraint on the editable weights. The remaining design question is how several such constraints should update the head. Coherent constraints benefit from their joint feature covariance, whereas conflicting constraints require bounded influence and reliability filtering.

We instantiate this observation as a unified **S**ymbolic **C**onstraint **U**pdates for **L**ightweight **P**redictor **T**uning (SCULPT) update family with a shared feedback router. The router forms a reliability-gated moment direction from structured cuts and a covariance-aware direction from consensus-filtered solver targets, then activates the branch exposed by a task while validation selects only scale and preconditioning. An optional output- or parameter-space trust projection limits drift. This formulation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

supports path-feature cuts for routing and perturb-and-MAP consensus targets for Sudoku and MNIST Addition without differentiating through their solvers.

We evaluate SCULPT on routing, visual Sudoku, and corrupted MNIST Addition. The studies identify early adaptation gains and benefits from refreshing solver targets on Sudoku, and characterize accuracy-resource trade-offs on MNIST Addition.

Our contributions are threefold:

- We propose SCULPT, a backpropagation-free framework that adapts a linear prediction head over frozen representations using symbolic feedback from non-differentiable checkers or solvers.
- SCULPT introduces an exact checker-to-parameter half-space lift and a unified feedback router that combines reliability-gated directional cuts with covariance-aware feasible targets, together with guarantees for trust-region drift and bounded checker errors.
- Experiments across diverse structured-prediction tasks and convolutional, residual, and transformer backbones characterize the adaptation quality, feedback-budget behavior, robustness, and representation-level transferability of SCULPT.

2 Related Work

Differentiable learning with symbolic programs and solvers. Neuro-symbolic learning uses structured knowledge to guide neural predictions. Semantic Loss turns logical satisfaction into a differentiable objective, DeepProbLog combines neural predicates with probabilistic logic, and SATNet integrates a differentiable satisfiability solver into a neural model (Xu et al., 2018; Manhaeve et al., 2018; Wang et al., 2019). Continuous relaxations provide another route to gradients through discrete choices, while ISED and DOLPHIN support learning with symbolic programs (Jang et al., 2017; Maddison et al., 2017; Solko-Breslin et al., 2024; Naik et al., 2025). Decision-focused learning instead optimizes predictions for downstream decision quality (Wilder et al., 2019; Elmachtoub & Grigas, 2022). Differentiable optimization layers, black-box solver differentiation, solution caching, and DataSP connect optimization outcomes to neural training through task-specific learning objectives (Amos & Kolter, 2017; Agrawal et al., 2019; Pogančić et al., 2020; Mulamba et al., 2021; Lahoud et al., 2024). These approaches chiefly use symbolic computation to define or propagate a training signal. SCULPT starts from an already deployed, frozen representation and uses checker-returned cuts or feasible targets to update only the linear head, without differentiating through the checker.

Test-time and post-deployment adaptation. Test-time training adapts a pretrained model through auxiliary self-supervision; TENT minimizes prediction entropy, whereas T3A adjusts class supports using confident test predictions (Sun et al., 2020; Wang et al., 2021; Iwasawa & Matsuo, 2021). Continual adaptation and methods such as EATA and SAR address instability from noisy test samples or repeated updates (Wang et al., 2022; Niu et al., 2022; 2023). Backpropagation-free alternatives further reduce update costs through operations on representations, normalization statistics, or estimated class distributions (Wang et al., 2024; Dong et al., 2025; Zhang et al., 2025). Their adaptation signals primarily come from unlabeled inputs and the model's own predictions. SCULPT uses an additional source of task information: reference-path comparisons in routing and constraint-consistent assignments in Sudoku and Addition. This lets the checker specify a correction even when the current prediction is confident but violates a task constraint.

Online updates from structured feedback. Passive-Aggressive learning and online projection show how violated linear constraints can produce direct parameter updates (Crammer et al., 2006; Bubeck, 2015). Perturb-and-MAP generates structured candidates through repeated discrete solves (Papandreou & Yuille, 2011). SCULPT uses the feedback available from its checkers in two ways: a linear constraint on prediction scores induces an exact half-space for the editable head, while agreement among feasible solver assignments supplies weighted targets for covariance-aware fitting. Reliability-gated aggregation limits the influence of conflicting cuts, and optional trust control bounds drift from the deployed head. Together, these mechanisms connect online constraint updates and solver-generated targets within one frozen-head adaptation interface.

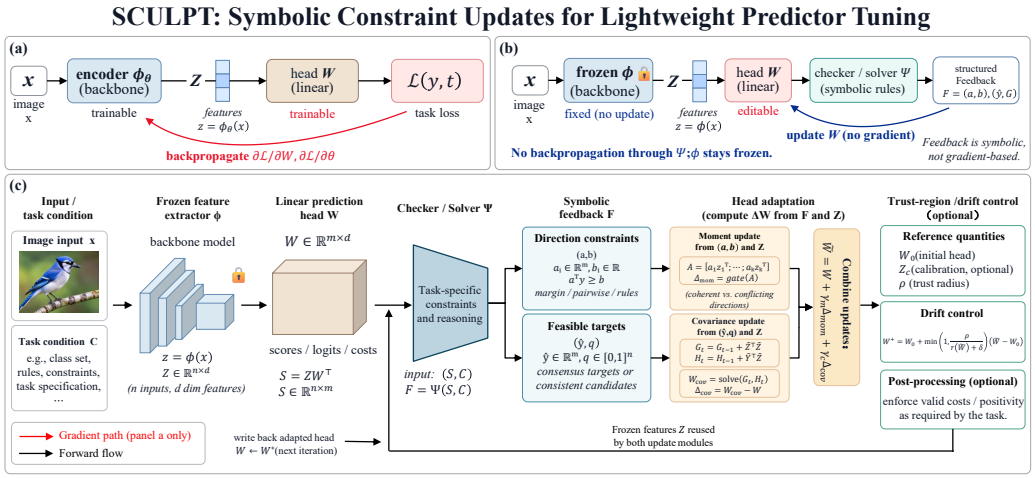

Figure 1: **Overview of SCULPT.** (a) Standard adaptation backpropagates through both the encoder and prediction head. (b) SCULPT freezes the encoder and updates only the linear head using symbolic checker/solver feedback, without differentiating through the checker. (c) Directional constraints and feasible targets produce moment- and covariance-based updates, which are combined, optionally trust-projected, and written back to the head.

3 METHOD

3.1 PROBLEM SETTING AND OVERVIEW

Let $\phi : \mathcal{X} \to \mathbb{R}^d$ be a frozen feature extractor and $W \in \mathbb{R}^{m\times d}$ an editable linear head. The prediction is

$$y_W(x) = W\phi(x). \tag{1}$$

We use *checker* to mean an executable constraint-checking program that inspects a prediction and determines whether it satisfies the task rules. The checker is not another neural network and need not be differentiable. Examples include a shortest-path routine comparing a proposed route with an optimal route, CP-SAT checking Sudoku row, column, and block validity, enumeration checking whether two predicted digits have the observed sum, and a geometric monitor testing whether an action enters a hazard. Unlike a scalar accept/reject test, an actionable checker also reports *which rule was violated* and in what direction the output must move.

Formally, a checker Ψ either accepts $y_W(x)$ or returns a violated local output constraint

$$a^\top y \geq b, \tag{2}$$

where $a \in \mathbb{R}^m$ identifies a direction in output space and $b \in \mathbb{R}$ is the required margin. The checker may contain discrete search, enumeration, argmin operations, or hard physical thresholds; none of these operations is placed in the automatic-differentiation graph. Substituting the linear head into Eq. (2) gives

$$\langle A, W\rangle_F \geq b, \qquad A = a\phi(x)^\top, \tag{3}$$

because $a^\top W\phi(x) = \text{tr}((a\phi(x)^\top)^\top W)$. Thus, although the checker is non-differentiable, Eq. (3) shows that its directional response defines an *exact* half-space in the editable parameters. The normal A requires only one outer product and can therefore be used directly as a head-update direction. This construction requires directional feedback; a Boolean accept/reject signal alone is insufficient.

The checker may alternatively return consensus targets (Z, Y, q). Each row of $Z \in \mathbb{R}^{n\times d}$ is a frozen feature, the corresponding row of $Y \in \mathbb{R}^{n\times m}$ is a feasible solver target, and $q \in [0,1]^n$ records agreement across perturbed checker calls. These targets are not assumed to be ground-truth labels, and low-agreement rows are discarded before updating. A task may return directional cuts, consensus targets, or both from the same minibatch.

For spatial tasks, the head acts on each local feature, and a structured cut sums the resulting feature normals. Exactness refers specifically to the induced linear score constraint. A nonlinear cost

Algorithm 1 SCULPT: one frozen-head adaptation minibatch

Require: Batch X, frozen ϕ, head W, deployed head W_0, checker Ψ
Require: Persistent moment/covariance state; validation-selected configuration
Compute detached features $Z = \phi(X)$ and scores $S = ZW^\top$
Query Ψ for directional cuts and/or consensus targets with reliability
Initialize $\Delta_{\mathrm{mom}} \leftarrow 0$, $\Delta_{\mathrm{cov}} \leftarrow 0$
if directional cuts are available **then**
    Pull cuts back to head space; update consensus and gate using prior history
    Compute Δ_{mom} by Eq. (9); update direction history
end if
if reliable solver targets are available **then**
    Filter/weight targets and accumulate covariance sufficient statistics
    Solve Eq. (11); set $\Delta_{\mathrm{cov}} \leftarrow W_{\mathrm{cov}} - W$
end if
Mix available directions into $\widetilde{W}$ using Eq. (12)
Set $W^+ \leftarrow \mathcal{T}_\rho(\widetilde{W}; W_0)$ if enabled, else $W^+ \leftarrow \widetilde{W}$
Apply task-specific decoder corrections; copy W^+ into the head
return updated head and persistent state

transform or positivity correction used by the decoder need not preserve the same feasible set, and a solver target need not be a ground-truth label. The half-space identity therefore does not, by itself, guarantee improved task accuracy.

Figure 1 (a), (b) contrasts standard gradient-based adaptation with SCULPT. Standard adaptation backpropagates a task loss through the model, whereas SCULPT freezes the feature extractor and updates only the linear head using symbolic feedback from a task-specific checker or solver. Figure 1(c) presents the complete SCULPT architecture. Given an adaptation minibatch, SCULPT computes detached features $Z = \phi(X)$ and scores $S = ZW^\top$, after which Ψ returns directional cuts, consensus targets, or both. Directional cuts produce a reliability-gated moment direction, while consensus targets produce a covariance-aware direction. The available directions are combined, optionally restricted by trust-region control and task-specific post-processing, and written back to the linear head.

Algorithm 1 formalizes one adaptation minibatch. Throughout adaptation, ϕ remains frozen, while the moment history and covariance sufficient statistics persist across minibatches. All adaptation operations are performed outside autograd. The following sections describe the two update directions and their shared feedback router; Appendix A specifies the task-level configurations.

3.2 Reliability-gated moment direction

Sparse checker feedback should move the head in the indicated direction without allowing one unreliable decode to dominate. For a violated cut (A_i, b_i), define its non-negative violation $v_i(W) = [b_i - \langle A_i, W\rangle_F]_+$, normalized normal $\widehat{A}_i = A_i/(\|A_i\|_F + \delta)$, and capped projection coefficient

$$\alpha_i = \min\left\{\frac{v_i(W)}{\|A_i\|_F^2 + \delta}, \tau\right\}. \tag{4}$$

The batch consensus and its exponential moving average are

$$s_t = \left\| B^{-1} \sum_i \widehat{A}_i \right\|_F, \qquad \bar{s}_t = \beta_s \bar{s}_{t-1} + (1-\beta_s) s_t, \tag{5}$$

$$h_t = \beta_h h_{t-1} + (1-\beta_h) B^{-1} \sum_i \widehat{A}_i. \tag{6}$$

Before updating h_t, each cut receives history agreement

$$q_i = \sigma\left(\frac{\langle \widehat{A}_i, \widehat{h}_{t-1}\rangle_F + \mu}{T_h}\right), \qquad \widehat{h} = h/(\|h\|_F + \delta). \tag{7}$$

Let $c_t = \mathbf{1}[\bar{s}_t > \kappa]$ and $g_t = \sigma((s_t - s_0)/T_s)$. The two candidate directions are

$$\Delta_{\text{coh}} = \eta_f g_t \mathcal{P}_t \left(B^{-1} \sum_i A_i \right) + \eta_p B^{-1} \sum_i \alpha_i A_i, \tag{8}$$

$$\Delta_{\text{conf}} = \eta_h \frac{\sum_i q_i A_i}{\sum_i q_i + \delta}, \qquad \Delta_{\text{mom}} = c_t \Delta_{\text{coh}} + (1 - c_t)\Delta_{\text{conf}}. \tag{9}$$

Validation selects identity or Adam preconditioning for $\mathcal{P}_t$. Coherent cuts share an aggregate direction and capped correction; conflicting cuts are filtered by historical agreement. Dijkstra uses identity preconditioning and Warcraft uses Adam.

The consensus statistics serve different purposes. The current-batch score s_t measures whether returned cuts point in similar directions, while the history agreement q_i favors cuts aligned with earlier feedback when the batch is conflicting. The capped coefficient α_i limits the contribution of a single large violation. The gate therefore changes how the available cuts are aggregated according to their coherence; it cannot detect a checker whose errors remain consistent over time. We examine the effect of corrupted checker decisions in Section 4.5.

3.3 Covariance-aware direction

When perturb-and-MAP samples agree on a coherent target, treating accepted rows independently discards useful feature correlation. Let $Z \in \mathbb{R}^{n \times d}$ contain accepted frozen features, let $Y \in \mathbb{R}^{n \times m}$ contain consensus targets, and let W_0 be the deployed head. Reliability enters through $\bar{Z} = \text{diag}(\sqrt{q})Z$ and $\bar{Y} = \text{diag}(\sqrt{q})Y$. Accepted rows accumulate in sufficient statistics across adaptation minibatches. We solve a ridge-regularized least-squares problem optionally anchored at W_0:

$$\min_W \ \|\bar{Z}W^\top - \bar{Y}\|_F^2 + \lambda \|W - W_0\|_F^2, \qquad \lambda \geq 0, \tag{10}$$

whose closed form is

$$W_{\text{cov}} = (\bar{Y}^\top \bar{Z} + \lambda W_0)(\bar{Z}^\top \bar{Z} + \lambda I)^{-1}. \tag{11}$$

We obtain W_{cov} by solving the corresponding linear system and define $\Delta_{\text{cov}} = W_{\text{cov}} - W$. The Gram matrix $\bar{Z}^\top \bar{Z}$ captures interactions among feature dimensions. For Sudoku, Y contains consensus-filtered CP-SAT cell assignments, with their agreement encoded by q.

3.4 Shared feedback router and trust projection

The moment and covariance terms form a shared feedback router. Given validation-selected weights $\gamma_m, \gamma_c \geq 0$, the candidate is

$$\widetilde{W} = W + \gamma_m \Delta_{\text{mom}} + \gamma_c \Delta_{\text{cov}}. \tag{12}$$

Unavailable directions are zero: cut-only tasks use $\gamma_c = 0$, target-only tasks use $\gamma_m = 0$, and mixed feedback can activate both. Validation selects scale and preconditioning within the fixed checker interface.

The optional trust projection is explicit. For calibration features Z_c, define

$$r_Z(U) = (|Z_c|m)^{-1/2} \, \|Z_c(U - W_0)^\top\|_F, \tag{13}$$

and use $r_W(U) = \|U - W_0\|_F$ when validation selects parameter-space control. For either $r \in \{r_Z, r_W\}$,

$$\mathcal{T}_\rho(U; W_0) = W_0 + \min\left\{1, \frac{\rho}{r(U) + \delta}\right\}(U - W_0), \qquad W^+ = \mathcal{T}_\rho(\widetilde{W}; W_0). \tag{14}$$

The projection is inactive below radius ρ and otherwise rescales the candidate toward the deployed head. MNIST Addition and Sudoku use output drift r_Z, Warcraft uses r_W, and Dijkstra selects no trust projection. We treat this safeguard as drift control rather than a universal source of accuracy gain.

Branch-level guarantees. The two branches admit complementary one-step statements. A valid-cut projection reaches its half-space and decreases distance to every feasible comparator. With unit-normal error at most ε, decrease still holds when the true margin satisfies $\gamma > \varepsilon D$ and $0 < \eta < 2(\gamma - \varepsilon D)/(1+\varepsilon)^2$; Eq. (14) independently enforces $r(W^+) \le \rho$. For fixed accepted targets, every fractional step $\eta \in (0, 1]$ toward W_{cov} decreases the regularized fitting objective. These are branch-local guarantees, not convergence of the mixed rule under systematically incorrect feedback; Appendix A gives formal statements and proofs.

Computational profile. The moment branch reduces B normals over the $m \times d$ head. The covariance branch forms statistics in $O(nd^2 + nmd)$ time and solves a $d \times d$ system in $O(d^3)$; trust evaluation costs $O(n_c dm)$. No backbone activations are retained for backward, and the only additional mutable state is a $d \times d$ accumulator when covariance fitting is active. Task-specific checkers and hyperparameters remain in the appendix.

4 Experiments

4.1 Setup and reporting protocol

We evaluate frozen-head adaptation on Dijkstra routing, official Warcraft shortest paths, clue-free visual Sudoku, and corrupted MNIST Addition. All adaptation methods operate on frozen features. Routing methods receive reference paths; DataSP and some decision-focused methods additionally use true costs. Addition methods receive observed sums, whereas test-time adaptation baselines use predictions alone. The supervised references use individual labels or costs. Appendix B details the data splits, validation choices, and update budgets.

We report mean±sample standard deviation across random seeds. For routing, exact path or mask match measures agreement with a reference route, while path IoU and tie-aware optimality provide complementary measures of route quality. For the visual tasks, we distinguish raw predictions from outputs after constraint decoding. Table 1 pairs SCULPT with the highest-mean evaluated non-SCULPT comparator for each metric; Appendix Tables 6 and 7 report all evaluated methods. Appendix D describes the paired multiple-comparison analysis.

Table 1: Cross-task performance against the highest-mean evaluated non-SCULPT comparator. Entries are mean±sample SD (%, higher is better); Δ is SCULPT minus the comparator in percentage points.

Setting / metric	Comparator	Comparator score	SCULPT	Δ
Dijkstra routing (exact path match)				
12×12 ID	Perceptron	85.88±3.79	**87.13±3.66**	+1.25
18×18 OOD	Perceptron	79.00±4.22	**82.83±3.52**	+3.83
Warcraft (exact path-mask match)				
12×12	DataSP (2024)	**40.06±1.20**	36.90±0.96	−3.16
18×18	PG-Adam	18.84±1.07	**19.26±0.71**	+0.42
24×24	PG-Adam	**11.94±0.97**	11.88±0.83	−0.06
30×30	PG-Adam	7.50±0.59	**8.22±0.54**	+0.72
Clue-free visual Sudoku				
Raw cell accuracy	Supervised†	64.05±0.78	**70.49±0.33**	+6.44
Decoded board EM	Supervised† / Self-training	88.75±4.19	**96.25±4.07**	+7.50
Corrupted MNIST Addition (raw pair exact match)				
Single-digit	ISED (2024)	77.63±1.10	**78.74±1.07**	+1.11
2-digit ID	Exact marginal	**62.49±0.92**	61.33±0.55	−1.16
3-digit OOD	Exact marginal	**48.45±1.19**	47.47±0.65	−0.98

Bold marks the higher displayed mean, not statistical significance. † denotes a label-using reference. Full comparisons appear in Appendix Tables 6 and 7.

4.2 Routing accuracy and size transfer

On synthetic Dijkstra routing, we adapt a 12-dimensional terrain-cost head using 384 maps of size 12×12, then evaluate on 160 ID and 120 size-OOD maps per run. Reference paths provide the path-feature differences used for directional updates. SCULPT obtains 87.13% exact path match ID and 82.83% at 18×18, exceeding the Perceptron mean by 1.25 and 3.83 points, respectively. Neither comparison remains significant after Holm correction within the primary comparison family (Appendix D).

For Warcraft, methods use the official splits and a shared frozen image encoder (Pogančić et al., 2020). At 12×12, SCULPT reaches 36.90% exact-mask match, below DataSP (40.06%) and the matched path-gap Adam control (37.88%). It exceeds DBB by 9.04 points; this DBB comparison remains significant after Holm correction (Appendix D). The separate DataSP comparison in Table 1 reports the highest comparator mean and is not part of that significance claim. SCULPT also reaches 80.75% path IoU (Appendix D), indicating that exact agreement with a single reference mask does not fully describe route similarity.

The larger-map study keeps the frozen encoder and selects the SCULPT moment and covariance weights, 1 and 0.10, on 18×18 validation maps. The same weights are used at 24×24 and 30×30, with head adaptation performed separately at each size. SCULPT has a higher exact-match mean than PG-Adam at sizes 18 and 30, and a lower mean at size 24; none of the three paired comparisons remains significant after Holm correction (Appendix D). DataSP was evaluated only at 12×12.

Exact-mask agreement is strict because it compares a predicted path with one reference mask. On the larger maps, SCULPT reaches path IoU of 75.82%, 72.83%, and 70.86% at sizes 18, 24, and 30, versus 75.69%, 72.74%, and 70.72% for PG-Adam. Its tie-aware optimality is 99.86%, 99.80%, and 99.82% (Appendix D). These measures distinguish route quality from agreement with a particular reference mask. The IoU differences are small, so they do not alter the conclusion that the larger-map comparison with PG-Adam is close.

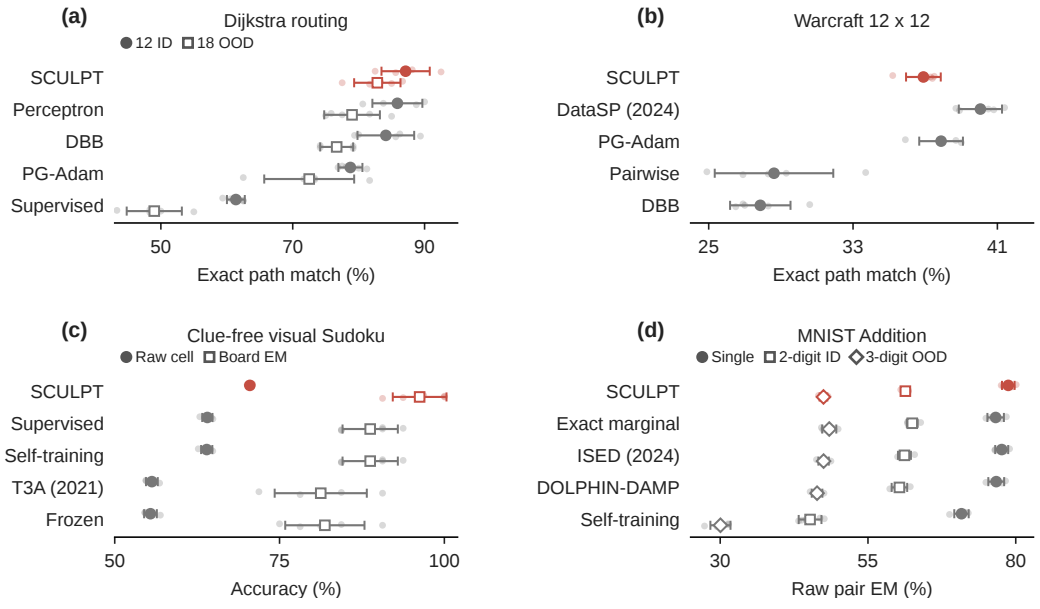

Figure 2: Cross-task baseline comparison. Large markers show means, whiskers show sample standard deviations, and small points show individual runs. (a) Dijkstra exact path match on 12×12 ID and 18×18 OOD grids. (b) Warcraft 12×12 exact path-mask match. (c) Clue-free visual Sudoku raw-cell accuracy and decoded-board exact match. (d) MNIST Addition raw pair exact match for single-digit, two-digit ID, and three-digit OOD operands. Red marks SCULPT; marker shapes distinguish settings or metrics. The figure shows representative strong and reference comparators; Appendix Tables 6 and 7 give complete baseline coverage.

4.3 VISUAL CONSTRAINTS AND ARITHMETIC RECOVERY

The two visual tasks expose feasible assignments rather than path-feature cuts, but they test different levels of structure. A Sudoku board contains 81 digit images, and CP-SAT enforces row, column, and block constraints; MNIST Addition observes only the sum of corrupted digit images, not their individual labels. Perturb-and-MAP agreement filters solver targets in both cases. We therefore separate local recognition from structured recovery: Table 1 reports raw Sudoku cell accuracy and decoded-board EM, and raw pair EM at each operand length for Addition.

On clue-free Sudoku, SCULPT improves raw-cell accuracy over the supervised head by 6.44 points; this paired comparison remains significant after Holm correction (Appendix D). It also improves decoded-board EM by 7.50 points. On Addition, it leads the single-digit raw pair comparison, whereas exact marginalization leads when the same position-shared head transfers from two- to three-digit operands. This contrast matters: solver decoding can enforce a valid board or sum after adaptation, but the raw metrics test whether symbolic feedback actually repairs the deployed predictor. Appendix Table 7 retains every evaluated baseline and both raw and decoded views.

For Addition, the validation and evaluation generators share the official MNIST test-image pool, so OOD denotes unseen operand length rather than image-disjoint evaluation. Fixed configurations, step counts, and MAP-call budgets are recorded in Appendix B; the feedback-budget study below uses a separate validation protocol.

4.4 BACKBONE GENERALIZATION

We replace each task's original CNN with ResNet-18 or a ViT while freezing the resulting representation and retaining the same checker interface. For each backbone, SCULPT parameters are selected on one source configuration and then fixed: 12×12 Warcraft, corrupt-head Sudoku, or single-digit Addition. We transfer them without target-configuration tuning to larger maps, alternative Sudoku corruptions, and longer operands. Every cell in Table 2 pairs Frozen and SCULPT from the same head and features; comparisons are within a backbone rather than across absolute starting points.

Table 2: Backbone and task-configuration transfer. Each cell shows the SCULPT mean (%) and its paired gain over Frozen in parentheses (percentage points). The three task groups report exact path-mask match, raw cell accuracy, and raw pair exact match, respectively.

Task / configuration	CNN	ResNet-18	ViT
Warcraft (exact path-mask match)			
12×12	40.36 (+22.50)	57.82 (+38.14)	29.88 (+11.74)
18×18	21.54 (+14.98)	39.26 (+32.12)	11.42 (+5.40)
24×24	13.44 (+10.24)	30.54 (+25.88)	5.40 (+2.32)
30×30	8.84 (+7.32)	24.52 (+22.60)	2.04 (+0.52)
Clue-free visual Sudoku (raw cell accuracy)			
Corrupt head	70.04 (+14.45)	21.77 (+10.60)	66.67 (+29.53)
Input only	69.90 (+14.24)	21.45 (+10.28)	65.05 (+27.80)
Occlusion transfer	89.64 (+8.79)	89.20 (+18.92)	78.70 (+11.11)
Corrupted MNIST Addition (raw pair exact match)			
Single-digit	78.46 (+22.21)	34.60 (+32.49)	60.56 (+27.29)
Two-digit ID	61.14 (+26.16)	0.19 (+0.17)	33.83 (+21.34)
Three-digit OOD	46.83 (+26.93)	0.01 (+0.01)	19.80 (+15.75)

All 21 held-out backbone-configuration cells have positive mean gains. Because each cell compares the same frozen features and initial head, the gain measures whether checker feedback repairs that representation; absolute scores across architectures are not directly comparable. The fixed-source protocol tests whether validation-selected settings remain useful across backbones and changes in map size, corruption, or operand length.

Some gains are small: Warcraft ViT improves by 0.52 points at 30×30, while ResNet-18 Addition improves by 0.17/0.01 points in raw pair exact match at two/three digits. These results support positive mean adaptation gains across the evaluated configurations but do not establish architecture-invariant accuracy or reliable recovery when structured errors compound.

4.5 Feedback efficiency and adaptation mechanisms

Figure 3 examines the covariance branch under limited symbolic feedback on Sudoku and MNIST Addition. Each curve uses validation-fixed configurations, restarts from the same deployed head at every budget, and matches solver calls between SCULPT and self-training. On Sudoku, SCULPT leads self-training at smaller budgets, but the gap narrows and reverses at the largest plotted budget. On Addition, SCULPT maintains a higher raw pair exact-match mean across the plotted budgets. The effect of covariance fitting therefore depends on the task and adaptation horizon.

Within the separate Sudoku feedback-budget protocol, querying the current head yields higher raw-cell accuracy than fitting targets from the initial head at 160 calls, indicating a benefit from refreshing solver targets at that budget.

Figure 3(d,e) complements the target experiments by corrupting Dijkstra checker decisions. SCULPT and Perceptron are similar at 10-20% corruption, while all methods degrade as unreliable decisions become more frequent; broad variation at 40% limits stronger robustness claims. More generally, SCULPT requires directional or feasible-target feedback rather than Boolean acceptance. Reliability is uncalibrated, scale and preconditioning are validation-selected, and the branch-level guarantees do not establish convergence of the mixed rule. Warcraft exact masks are tie-sensitive, and MNIST Addition uses controlled corruption.

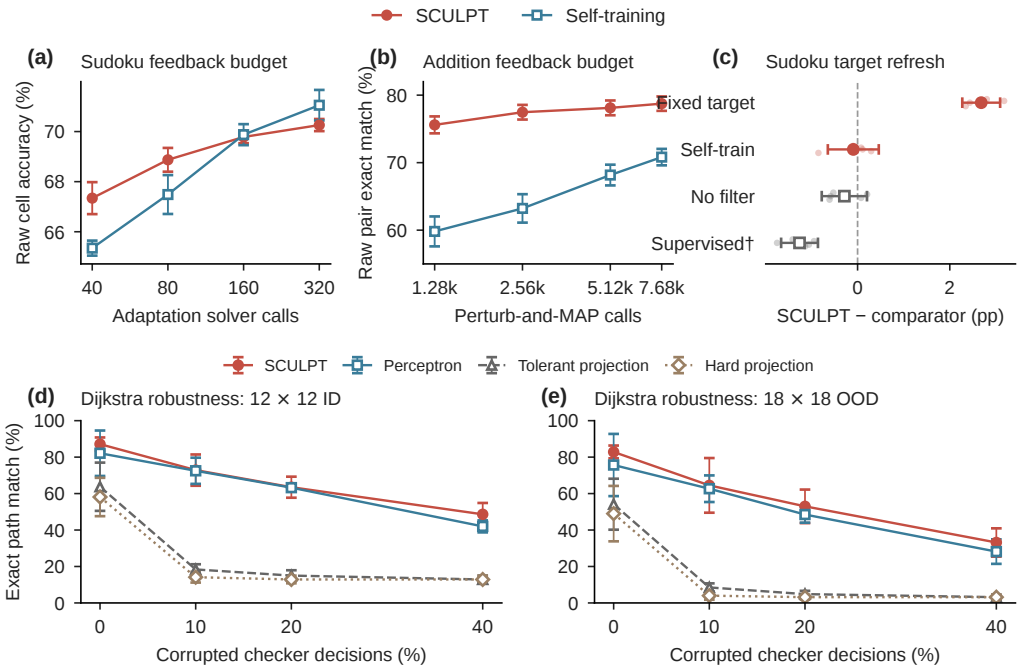

Figure 3: Feedback-budget and robustness analysis. (a,b) Sudoku raw-cell accuracy and Addition raw pair EM versus matched solver calls. (c) Paired Sudoku contrasts at 160 calls (95% intervals; †: label-using reference). (d,e) Dijkstra exact path match under corrupted checker decisions, evaluated against clean paths. Curves show mean±sample SD; small points show runs.

5 Conclusion

SCULPT converts symbolic checker feedback into backpropagation-free frozen-head updates. Directional cuts help when the checker supplies a stable correction; covariance fitting helps when repeated solves agree. The experiments show strong Dijkstra and Sudoku results, similar mean performance to a matched Warcraft path-gap optimizer on larger maps, and an advantage for exact marginalization on multi-digit Addition. Performance therefore depends on feedback quality, ambiguity, and budget: SCULPT is an adaptation interface, not a universally superior optimizer.

AI USE STATEMENT

Generative AI tools were used to assist with preliminary literature discovery and metadata checking, manuscript language editing and formatting, figure and table preparation, and experiment-code inspection, editing, and orchestration. The core research idea, methodology, theoretical derivations, and mathematical claims were developed by the authors without generative AI. The authors reviewed all AI-assisted text and code changes, checked citations against publisher or official venue records, and verified reported results against executable code and stored result artifacts. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work uses public benchmark data and synthetic instances; it involves no human subjects or personal data.

REPRODUCIBILITY STATEMENT

The implementation, exact experiment commands, environment versions, data paths, seed sets, and artifact locations are documented in the accompanying repository README. Appendix A specifies the task-level checker implementations and numerical update configurations, and the remaining appendix sections document data processing, validation choices, statistical procedures, additional results, and limitations.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9558–9570, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/9ce3c52fc54362e22053399d3181c638-Abstract.html.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 136–145, 2017. URL https://proceedings.mlr.press/v70/amos17a.html.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3–4):231–357, 2015. doi: 10.1561/2200000050.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006. URL https://jmlr.org/papers/v7/crammer06a.html.

Cynthia Dong, Hong Jia, Young D. Kwon, Georgios Rizos, and Cecilia Mascolo. LeanTTA: A backpropagation-free and stateless approach to quantized test-time adaptation on edge devices. *arXiv preprint arXiv:2503.15889*, 2025. doi: 10.48550/arXiv.2503.15889.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1): 9–26, 2022. doi: 10.1287/mnsc.2020.3922.

Vishal Gupta and Michael Huang. Decision-focused learning with directional gradients. In *Advances in Neural Information Processing Systems*, volume 37, pp. 79194–79220, 2024. doi: 10.52202/079017-2514.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 2427–2440, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/1415fe9fea0fa1e45dddcff5682239a0-Abstract.html.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Alan Lahoud, Erik Schaffernicht, and Johannes Stork. DataSP: A differential all-to-all shortest path algorithm for learning costs and predicting paths with context. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pp. 2094–2112, 2024. URL https://proceedings.mlr.press/v244/lahoud24a.html.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJKYvt5lg.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html.

Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Víctor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pp. 2833–2840, 2021. doi: 10.24963/ijcai.2021/390.

Aaditya Naik, Jason Liu, Claire Wang, Amish Sethi, Saikat Dutta, Mayur Naik, and Eric Wong. DOLPHIN: A programmable framework for scalable neurosymbolic learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45461–45483, 2025. URL https://proceedings.mlr.press/v267/naik25a.html.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905, 2022. URL https://proceedings.mlr.press/v162/niu22a.html.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g2YraF75Tj.

George Papandreou and Alan L. Yuille. Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 193–200, 2011. doi: 10.1109/ICCV.2011.6126242.

Marin Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkevoJSYPB.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017. doi: 10.48550/arXiv.1703.03864.

Alaia Solko-Breslin, Seewon Choi, Ziyang Li, Neelay Velingker, Rajeev Alur, Mayur Naik, and Eric Wong. Data-efficient learning with neural programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14666–14689, 2024. doi: 10.52202/079017-0468.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=uXl3bZLkr3c.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6545–6554, 2019. URL https://proceedings.mlr.press/v97/wang19e.html.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022. doi: 10.1109/CVPR52688.2022.00706.

Yanshuo Wang, Ali Cheraghian, Zeeshan Hayder, Jie Hong, Sameera Ramasinghe, Shafin Rahman, David Ahmedt-Aristizabal, Xuesong Li, Lars Petersson, and Mehrtash Harandi. Backpropagation-free network for 3D test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23231–23241, 2024. doi: 10.1109/CVPR52733.2024.02192.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):1658–1665, 2019. doi: 10.1609/aaai.v33i01.33011658.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5502–5511, 2018. URL https://proceedings.mlr.press/v80/xu18h.html.

Youjia Zhang, Youngeun Kim, Young-Geun Choi, Hongyeob Kim, Huiling Liu, and Sungeun Hong. Backpropagation-free test-time adaptation via probabilistic Gaussian alignment. In *Advances in Neural Information Processing Systems*, volume 38, pp. 84921–84951, 2025. doi: 10.52202/085713-2562.

A Additional Method Details

A.1 Task-specific checkers

Algorithm 1 executes under `torch.no_grad()` and copies the update into `head.weight`. The task-specific checker and decoder choices are as follows.

Routing. For Dijkstra and Warcraft, the checker compares the predicted and oracle paths. Their difference in accumulated frozen cell features yields a separating structured constraint. Dijkstra uses Eq. (9) with $\beta_s = 0.9$, $\kappa = 0.45$, $s_0 = 0.20$, $T_s = 0.05$, $\beta_h = 0.8$, $\mu = 0.15$, $T_h = 0.12$, $\tau = 0.10$, $(\eta_f, \eta_p, \eta_h) = (0.018, 0.10, 0.036)$, and identity $\mathcal{P}_t$. Warcraft uses current-batch consensus ($\beta_s = 0$), $\kappa = 0.05$, $g_t = 1$, $\beta_h = 0.8$, $\mu = 0.10$, $T_h = 0.15$, $\tau = 0.08$, Adam preconditioning with step 0.010, and projection coefficient 0.05. Its conflict step is 1.15 times the validated base cut step. Larger-map Warcraft adds a ridge residual solve: cells present only in the oracle path receive desired cost change -0.05, cells present only in the predicted path receive $+0.05$, and the solved delta is weighted by 0.10. Ties are evaluated by true path cost as well as exact pixel match.

Warcraft applies positive-cost corrections after the trust operation; Proposition 2 bounds the trust operator, not the displacement after these corrections.

Visual Sudoku. CP-SAT enforces one digit per cell and all-different constraints for every row, column, and 3×3 block. Perturbing logits generates alternative feasible boards; solver and cell consensus filter ambiguous pseudo-labels before the proximal solve.

MNIST Addition. Given two digit logits and their observed sum, an enumerative checker returns the highest-scoring digit pair consistent with that sum. Perturb-and-MAP agreement filters ambiguous pairs before the same proximal covariance update; individual digit labels are reserved for evaluation and the supervised baseline.

A.2 Single-constraint projection guarantee

Proposition 1. *Assume the half-space $\mathcal{H} = \{U : \langle A, U\rangle_F \geq b\}$ contains a comparator $W^\star$. The Euclidean projection*

$$W^+ = W + \frac{b - \langle A, W\rangle_F}{\|A\|_F^2} A \tag{15}$$

for a violated cut satisfies

$$\|W^+ - W^\star\|_F^2 \leq \|W - W^\star\|_F^2 - \frac{(b - \langle A, W\rangle_F)^2}{\|A\|_F^2}. \tag{16}$$

This property applies to the normalized projection of one valid cut. It does not establish convergence of the complete mixed update, jointly trained deep networks, or updates based on systematically incorrect checker feedback.

Proof. Let $v = b - \langle A, W\rangle_F > 0$ and $\alpha = v/\|A\|_F^2$. Feasibility of $W^\star$ gives $\langle A, W - W^\star\rangle_F \leq -v$. Therefore $\|W^+ - W^\star\|_F^2 = \|W - W^\star\|_F^2 + 2\alpha\langle A, W - W^\star\rangle_F + \alpha^2\|A\|_F^2 \leq \|W - W^\star\|_F^2 - v^2/\|A\|_F^2$.

Proposition 2 (Trust projection and bounded checker error)**.** *For any candidate U, the trust operator in Eq.* (14) *satisfies $r_Z(\mathcal{T}_\rho(U; W_0)) \leq \rho$ (and analogously for r_W). In addition, let A be a unit-norm valid cut, let $\widetilde{A}$ be the implemented normal with $\|\widetilde{A} - A\|_F \leq \varepsilon$, and suppose a comparator satisfies $\langle A, W^\star - W\rangle_F \geq \gamma$. For $W^+ = W + \eta\widetilde{A}$ and $D = \|W - W^\star\|_F$,*

$$\|W^+ - W^\star\|_F^2 \leq D^2 - 2\eta(\gamma - \varepsilon D) + \eta^2(1 + \varepsilon)^2. \tag{17}$$

In particular, if $\gamma > \varepsilon D$ and $0 < \eta < 2(\gamma - \varepsilon D)/(1+\varepsilon)^2$, the comparator distance strictly decreases.

Proof. The trust claim follows because Eq. (14) scales $U - W_0$ by at most $\rho/(r(U) + \delta)$. For Eq. (17), write $\widetilde{A} = A + E$. Then $\langle \widetilde{A}, W - W^\star\rangle_F \leq -\gamma + \|E\|_F D \leq -\gamma + \varepsilon D$ and $\|\widetilde{A}\|_F \leq 1 + \varepsilon$. Expanding $\|W + \eta\widetilde{A} - W^\star\|_F^2$ gives Eq. (17). This is a one-step robustness statement; it does not imply convergence of the mixed adaptive rule.

A.3 COVARIANCE-OBJECTIVE DECREASE

The covariance solve has a complementary guarantee for a fixed accepted-target set. It concerns the fitted target objective, rather than accuracy of the solver's pseudo-labels.

Proposition 3 (Fixed-target covariance update)**.** *Let* $F(W) = \|\widetilde{Z}W^\top - \widetilde{Y}\|_F^2 + \lambda\|W - W_0\|_F^2$ *and* $G = \widetilde{Z}^\top\widetilde{Z} + \lambda I \succ 0$*. Let* $W_\star$ *minimize* F*, as in Eq.* (11)*. For* $W_\eta = W + \eta(W_\star - W)$ *and* $0 \le \eta \le 1$*,*

$$F(W) - F(W_\eta) = \eta(2-\eta)\|(W - W_\star)G^{1/2}\|_F^2 \ge 0. \tag{18}$$

The decrease is strict for $\eta > 0$ *when* $W \ne W_\star$*.*

Proof. The normal equation is $W_\star G = \widetilde{Y}^\top\widetilde{Z} + \lambda W_0$. Expanding the quadratic around $W_\star$ cancels the cross term and gives $F(W) - F(W_\star) = \|(W - W_\star)G^{1/2}\|_F^2$. Since $W_\eta - W_\star = (1-\eta)(W - W_\star)$, subtracting the two objective gaps yields the identity. The result applies before adding the moment direction or applying a projection; changing the accepted targets changes the objective.

B EXPERIMENTAL SETUP AND HYPERPARAMETERS

Table 3 summarizes data and update budgets; Table 5 gives the primary covariance settings. Runs use matched randomization across methods, and all representations remain frozen during adaptation.

Table 3: SCULPT data and adaptation budgets per run. Evaluation sizes are per split; steps count minibatch updates. Solver calls exclude evaluation-time decoding.

Task / cohort	Adaptation	Evaluation	Steps	Batch	Calls
Dijkstra	384 maps	160 ID / 120 OOD	100	32	3,200
Warcraft, each size	2,000 maps	1,000 maps	100	32	3,200
Sudoku	128 boards	64 raw / 32 decoded	20	4	160
MNIST Addition single	4,096 pairs	2,048 pairs	30	64	7,680
MNIST Addition multi	4,096 pairs	2,048 pairs	60	64	15,360

Warcraft uses 500 official validation maps per size. Sudoku uses two CP-SAT calls per selected board; MNIST Addition uses four MAP calls per pair. Dijkstra and Warcraft count one predicted-path query per sampled map with reference paths available.

B.1 DATA, MODELS, AND COMPUTATION

Hardware and software. Experiments use an NVIDIA L20 (PyTorch 2.5.1/CUDA 12.4); Sudoku budget and configuration-transfer runs use an L20 with PyTorch 2.13.0/CUDA 13.0 and OR-Tools 9.15.6755. Every comparison is made within one matched environment.

Frozen backbones and editable heads. Dijkstra uses a 12-dimensional terrain feature and cost vector. Warcraft uses a four-layer CNN and scalar 1×1 head; Sudoku and Addition use separate two-layer digit CNNs with 96-dimensional features and bias-free nine-/ten-class heads. All backbones are pretrained before adaptation and then frozen.

Backbone-generalization protocol. Table 2 adds ResNet-18 (He et al., 2016) and a ViT (Dosovitskiy et al., 2021). Digit ResNet-18 uses a gray-scale 3×3 stem without initial max pooling; the ViT uses 4×4 patches, width 128, four layers, and four heads. Warcraft ResNet-18 ends after `layer2`; its ViT uses 8×8 patches and bicubic interpolation of the learned 12×12 positional grid. Both emit normalized 128-channel grids. Only the final linear or 1×1 head is editable. Digit ResNet/ViT models train for 10/14 epochs and Warcraft models for 8/16; one checkpoint per architecture is shared by all adaptation runs.

Head corruption is normalized across parameter dimensions; a disjoint cohort selects one configuration per backbone on the source setting (12×12 Warcraft, corrupt-head Sudoku, or single-digit Addition), and that configuration is reused on every target setting. Test results never select parameters.

Table 4: Validation-selected settings for the backbone-generalization study. Warcraft uses a parameter-space trust radius; Sudoku and Addition use an output-space trust radius. For Addition, $\lambda = 0$ is implemented with numerical jitter. Dashes indicate parameters that do not apply.

Task	Backbone	Steps	Adam step	Target scale	Cut cap τ	Ridge λ	Trust ρ
Warcraft	CNN	100	0.020	–	0.050	–	2.5
	ResNet-18	100	0.020	–	0.050	–	2.5
	ViT	75	0.0003	–	0.050	–	8.0
Sudoku	CNN	20	–	6.0	–	0.30	2.0
	ResNet-18	20	–	1.5	–	0.10	8.0
	ViT	20	–	3.0	–	0.10	2.0
MNIST Addition	CNN	20	–	9.0	–	0	3.5
	ResNet-18	30	–	3.0	–	0	3.5
	ViT	30	–	9.0	–	0	3.5

The routing, Sudoku, Addition, feedback-budget, and multi-digit protocols use the cohorts and budgets in Table 3; all methods receive matched inputs and clean evaluation checkers. The backbone-transfer table uses the same frozen-head interface across source and target configurations. The official Warcraft archive is identified by DOI `10.17617/3.YJCQ5S`; the repository records exact image corruption, solver limits, and checkpoint commands.

B.2 SCULPT HYPERPARAMETERS AND CONFIGURATION IDENTITY

Table 5: SCULPT settings for the primary covariance comparisons. The role column explains each hyperparameter; values are selected on validation data before test evaluation. All columns use $(\gamma_m, \gamma_c) = (0, 1)$.

Parameter	Role	Sudoku	Addition single	Addition multi
Ridge λ	pull toward deployed head	1	0	0.35
Target scale	solver-target magnitude	3	6	2.75
Trust radius ρ	output-drift cap	2	2.25	1.75
Post-solve anchor a	final pull toward W_0	0.002	0	0
Consensus threshold	required agreement	1	1	1
Solver samples K	MAP samples per example	2	4	4
Perturbation scale	solver-logit noise	0.02	0.03	0.02
Steps	adaptation minibatches	20	30	60
Batch size	examples per update	4	64	64

The covariance solve uses consensus-weighted one-hot targets and accumulates sufficient statistics across minibatches; the unanchored cases add $10^{-5}I$ numerical jitter.

SCULPT settings are selected on disjoint validation cohorts; test estimates never select parameters. Dijkstra and Warcraft at 12×12 use the moment branch, larger Warcraft uses the mixed branch, and Sudoku/Addition use covariance fitting. Baseline learning rates and solver-specific options are kept in the reproduction scripts; the comparison tables report their resulting test metrics under the same data and checker budgets.

C COMPLETE BASELINE COMPARISONS

Tables 6 and 7 contain every evaluated comparator behind the main summary, including frozen and label-using references.

D ADDITIONAL RESULTS AND STATISTICAL DETAILS

Figure 4 complements the main tables with size and length transfer. At Warcraft sizes 18/24/30, SCULPT obtains path IoU of 75.82/72.83/70.86%, compared with 75.69/72.74/70.72% for PG-Adam;

Table 6: Complete routing comparisons: exact path/mask match (%, mean±sample SD; higher is better). Dijkstra transfers from 12×12 to 18×18. Warcraft uses the same frozen 12×12 backbone at all map sizes. Each block lists all evaluated methods for its setting.

(a) Dijkstra

Method	12 ID	18 OOD
Frozen	1.63±2.64	0.50±1.12
Supervised†	61.38±1.35	49.00±4.18
Perceptron	85.88±3.79	79.00±4.22
PA (2006)	69.13±7.59	59.33±11.54
Tolerant proj.	68.75±10.64	57.17±12.14
PG-Adam	78.75±1.82	72.50±6.82
DBB (2020)	84.13±4.30	76.67±2.50
PGB (2024)	7.13±7.53	2.00±3.61
SCULPT	**87.13±3.66**	**82.83±3.52**

(b) Warcraft 12×12

Method	Exact match
Frozen	17.78±2.71
Supervised†	14.28±0.79
Perceptron	19.16±2.67
PA (2006)	3.20±0.00
Tolerant proj.	24.98±2.39
PG-Adam	37.88±1.21
DBB (2020)	27.86±1.67
MAP-C (2021)	17.64±4.16
Pairwise	28.62±3.29
PGB (2024)	18.16±2.50
DataSP (2024)	**40.06±1.20**
SCULPT	36.90±0.96

(c) Warcraft size transfer

Method	18×18	24×24
Frozen	6.54±0.90	3.26±0.71
Supervised†	5.54±0.64	2.10±0.16
Perceptron	7.48±1.06	4.70±0.81
Tolerant proj.	12.20±0.67	7.70±0.57
PG-Adam	18.84±1.07	**11.94±0.97**
SCULPT	**19.26±0.71**	11.88±0.83

(d) Warcraft 30×30

Method	Exact match
Frozen	1.50±0.83
PG-Adam	7.50±0.59
DBB (2020)	5.90±0.41
MAP-C (2021)	1.96±0.57
Pairwise	5.82±0.29
SCULPT	**8.22±0.54**

Bold/underline: best/second-best displayed mean per setting; ties share marks, not significance. Years identify source papers; unyearmarked comparators are controls; †: label-using reference. Methods absent from a block were not evaluated in that setting. All routing entries use the fixed configurations described in Appendix B.

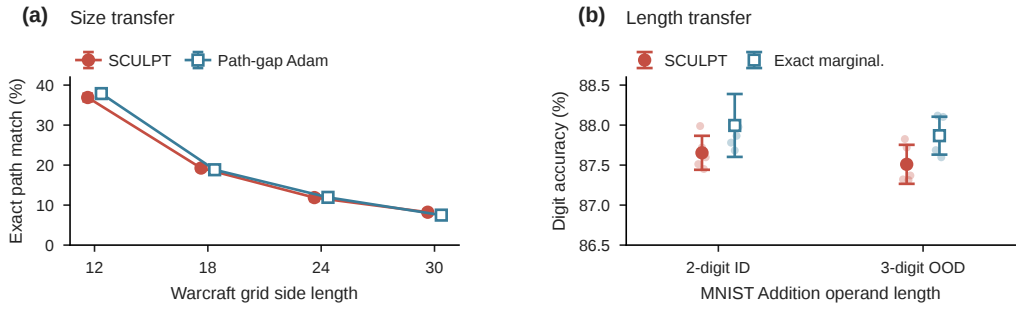

Figure 4: Size and length generalization. Large markers show means and whiskers show sample standard deviations. (a) Exact path match when a 12×12-trained frozen Warcraft backbone transfers to official larger maps; lines connect means for each method. (b) Digit accuracy under two-to-three-digit MNIST Addition transfer; small translucent points show individual runs. Exact marginalization has a 0.34-0.36 percentage-point higher mean. Horizontal offsets separate overlapping methods and do not indicate different grid sizes or operand lengths.

its tie-aware optimality is 99.86/99.80/99.82%. At size 12, SCULPT reaches 80.75% path IoU despite lower exact-mask agreement, indicating that reference-mask match can understate path quality under cost ties. For Addition, exact marginalization has a 0.34-0.36-point higher mean digit accuracy across two- and three-digit operands.

The 11 primary paired comparisons use 95% t intervals, Cohen's d_z, and Holm correction. Only Warcraft 12×12 versus DBB (raw $p = 0.0023$) and Sudoku versus the supervised head ($d_z = 10.06$,

Table 7: Complete visual-constraint comparisons (%, mean±sample SD; higher is better). (a) Clue-free visual Sudoku: raw cell accuracy and CP-SAT decoded-board exact match. (b) Corrupted MNIST Addition: raw pair exact match at each operand length.

(a) Clue-free visual Sudoku

Method	Raw cell accuracy	Decoded board EM
Frozen	55.42±0.97	81.88±6.01
Supervised†	<u>64.05±0.78</u>	<u>88.75±4.19</u>
Perceptron	56.03±0.93	81.88±6.01
Tolerant proj.	56.25±0.87	82.50±5.23
Self-training	63.95±0.85	<u>88.75±4.19</u>
TENT (2021)	55.05±1.03	81.88±6.01
T3A (2021)	55.64±0.89	81.25±6.99
SCULPT	**70.49±0.33**	**96.25±4.07**

(b) Corrupted MNIST Addition

Method	Single-digit	2-digit ID	3-digit OOD
Frozen	56.19±2.24	34.97±1.84	19.88±1.66
Supervised†	73.88±1.38	39.69±2.36	24.12±1.87
Self-training	70.82±1.22	45.21±1.93	30.04±1.70
TENT (2021)	56.10±2.13	34.79±1.81	19.70±1.69
T3A (2021)	57.26±1.63	35.99±1.73	20.78±1.48
EATA-lite (2022)	55.98±2.11	34.79±1.83	19.70±1.69
SAR-lite (2023)	56.18±2.23	34.97±1.84	19.88±1.66
Exact marginal	76.60±1.39	**62.49±0.92**	**48.45±1.19**
ISED (2024)	<u>77.63±1.10</u>	61.17±1.10	<u>47.49±0.88</u>
DOLPHIN-DAMP (2025)	76.70±1.34	60.32±1.31	46.39±0.90
SCULPT	**78.74±1.07**	<u>61.33±0.55</u>	47.47±0.65

Bold/underline: best/second-best displayed mean per metric/setting; ties share marks, not significance. Years identify source papers; unyearmarked comparators are controls; †: label-using reference. Each block lists every evaluated method; absent methods were not evaluated. Only the prediction head is adapted. MNIST Addition shares validation/evaluation image pools; length-selection access and adaptation budgets are detailed in the main text and Appendix B.

raw $p = 2.54 \times 10^{-4}$) remain significant after correction. Adjusted p-values are 1.000/0.772/0.391 for Dijkstra versus Perceptron at ID/OOD/40% corruption and 0.772/1.000/0.772 for Warcraft versus PG-Adam at sizes 18/24/30. Other baseline and feedback-budget comparisons are outside this family.

Exact marginal sums the predicted probabilities of digit assignments consistent with the observed sum; it is an exact-enumeration control for the MNIST Addition setting studied by Manhaeve et al. (2018).

PGB implements the perturbation-gradient baseline of Gupta & Huang (2024) and searches $h \in \{0.001, 0.01, 0.035\}$ and Adam rates $\{0.003, 0.01\}$, selecting $(h, \eta) = (0.01, 0.01)$ for Dijkstra and $(0.035, 0.01)$ for Warcraft, with 100 batches of 32. ISED and DOLPHIN select rate 0.05 from $\{0.001, 0.01, 0.05\}$ and use 4,096 adaptation pairs, 2,048 evaluation pairs, and 30 batches of 64; ISED uses ADD_MULT, while DOLPHIN uses DAMP without top-k truncation. DataSP retains 70 nodes and selects $(\eta, \beta) = (0.01, 100)$ from $\{(0.001, 100), (0.01, 100), (0.001, 10)\}$. For Warcraft, DBB uses perturbation 0.25, MAP-C uses an eight-solution cache, and Pairwise regresses predicted path gaps to true-cost gaps; these controls share frozen features, splits, and evaluation decoders.