Selective ML-Guided Beam Search for Differential-Trail Discovery in GIFT-64

Alireza Gholizadeh Shahrbejari*
Department of Computer Engineering, University of Guilan, Rasht, Iran

Reza Ebrahimi Atani†
Department of Computer Engineering, University of Guilan, Rasht, Iran

Abstract

Beam search bounds differential-trail exploration, but may still expand many intermediate differences. We test whether a learned short-horizon tail-cost residual can reduce this work while leaving classical selection in control. A regressor estimates a residual adjustment to an active-nibble heuristic. With four or two rounds remaining, the post-selection policy protects the best 25% of the classical frontier, removes at most 30% of accepted states, and abstains when fewer than five states can be pruned. It cannot restore candidates already rejected by classical selection.

We develop the method on a state-grouped GIFT-64 corpus and evaluate it on $D = 8$ and $D = 12$ holdouts and a $D = 16$ stress test. Target recovery and the best classical cost are preserved on all 48 holdout cases and all 18 stress cases. Aggregate node reductions are 6.832%, 0.099%, and 0.040% at $D = 8$, $D = 12$, and $D = 16$. An equal-budget ablation fixes the frontier, protected core, and pruning quota while changing only the removed-state ranking. Learned and classical ranking preserve all 48 holdout cases, with aggregate reductions of 2.359% and 2.411%. Their removal sets differ on 57 of 85 active frontiers, but the learned score does not establish a measurable advantage over the classical control. Random ranking produces larger raw reductions but loses quality. All comparisons use a bounded classical baseline; no globally optimal trails are claimed.

Keywords: differential cryptanalysis; beam search; machine learning; GIFT-64; automated trail search; controlled ablation

1 Introduction

Differential cryptanalysis studies how differences introduced at the input of an iterated cipher propagate to its output and remains one of the central tools for evaluating the security margins of symmetric-key primitives [3]. For a substitution–permutation network (SPN), the differential distribution table (DDT) of the S-box gives the probability of each local transition, while the linear diffusion layer determines how active components are redistributed between rounds. A differential *trail* fixes one complete sequence of intermediate differences, and its probability is obtained from the product of its local transition probabilities under the usual Markov-cipher model. This object is distinct from a differential or differential hull, which aggregates the contributions of multiple trails sharing the same external differences. The present work concerns the search for individual low-cost trails between fixed input and output differences; it does not estimate differential-hull probabilities.

*Email: gholizadeh.a2000@gmail.com
†Email: rebrahimi@guilan.ac.ir

The local transition structure of an SPN is compact, but the number of multi-round combinations grows rapidly with the state size and the number of rounds. This tension is especially visible in lightweight bit-permutation ciphers such as PRESENT [4] and GIFT [2]. Their 4-bit S-boxes permit inexpensive local DDT enumeration, yet the interaction of many parallel S-boxes with a bitwise permutation produces a large global trail space. Consequently, identifying a high-probability trail is primarily a search problem rather than a local probability-computation problem.

Automatic trail search has therefore become a standard component of block cipher analysis. Exact or proof-oriented approaches formulate the problem through branch-and-bound, mixed-integer linear programming (MILP), Boolean satisfiability (SAT/SMT), or constraint programming. These approaches can provide strong bounds or optimality statements, but their computational cost can become substantial for long trails and large state spaces. Bounded heuristic methods, including beam search, instead retain only a limited frontier of promising partial trails. They offer direct control over memory and runtime, but a state removed from the frontier cannot be recovered later. The ranking rule is therefore crucial: a cheap heuristic can keep the search tractable, while a single harmful ranking decision can eliminate the ancestor of the desired trail.

These limitations motivate a cautious test of learned search guidance. Short bounded tail searches may contain state–target information not represented by a simple active-nibble count, but average regression accuracy does not ensure that every search-critical state will be ranked correctly. In cryptanalytic search, pruning one indispensable ancestor can change the final result. The present method therefore does not attempt to repair earlier classical rejection. It addresses the narrower question of whether a learned residual can support controlled second-stage pruning among states already accepted by the classical policy.

Fixed-endpoint trail search is useful when an analyst must connect selected input and output differences rather than enumerate unrestricted characteristics. Examples include screening candidate endpoints, extending components of boomerang or rectangle constructions, and exploratory security evaluation under a fixed computational budget. In these settings, even a moderate reduction can permit more endpoint pairs to be examined, provided that pruning does not trade away the classical search result. This quality–efficiency requirement motivates both the protected integration and the controlled attribution experiments in this work.

1.1 Related Work and Research Gap

Matsui introduced a generic branch-and-bound procedure for finding strong differential and linear characteristics by combining accumulated trail weight with bounds for the remaining rounds [12]. Subsequent work improved both the formulation and implementation of automatic search. Ji, Zhang, and Ding proposed three accelerations of Matsui's algorithm and applied them to DESL and GIFT [9]. GIFT-specific studies have further used automated characteristic search for single-key and related-key differential analysis, clustering, and boomerang constructions [11, 10]. These works demonstrate the cryptanalytic value of carefully engineered trail search on the same cipher family considered here.

Solver-based methods provide another major line of work. MILP models have been used to count active S-boxes in bit-permutation SPNs and to encode probability-aware S-box behavior [14, 1]. SAT encodings incorporating Matsui-style bounding conditions have likewise accelerated the search for differential and linear characteristics, including applications to PRESENT, GIFT-64, and other lightweight ciphers [13]. At the implementation level, GPU-parallel branch-and-bound and meet-in-the-middle enumeration have provided substantial speedups and have been applied to PRESENT and GIFT-64 [15]. Broadly, these methods improve exact encodings, bounds, enumeration strategies, or hardware utilization while retaining cryptographically defined objectives.

Machine learning has entered differential cryptanalysis mainly through a different objective.

Gohr's neural differential cryptanalysis uses a neural network as a ciphertext-pair distinguisher and combines it with key search [6]; later work has developed this distinguisher-centered direction extensively [5]. Closer to automatic security evaluation, Idris et al. formulated active-S-box prediction as a regression problem for lightweight generalized Feistel ciphers [7]. Their later DiffGen framework combines metaheuristic generation with a neural validator to produce truncated differentials [8].

Our setting differs from both lines. We do not train a ciphertext distinguisher, and we do not ask a model to generate a truncated differential directly. We also do not replace the cipher's DDT probabilities or claim an exact learned bound. Instead, the model estimates a short-horizon residual tail cost for a concrete intermediate state and target endpoint. This estimate is inserted only after the original bounded classical policy has formed its frontier, and it is allowed to prune only a protected and explicitly capped subset. The research gap addressed here is thus the conservative integration and controlled evaluation of a learned post-selection score within an existing bounded trail search. The method is not presented as a mechanism for recovering states rejected during classical selection, and any loss of the classical result is treated as a failure rather than as an efficiency gain.

1.2 Approach and Contributions

We study this integration problem on GIFT-64. The classical reference is a deterministic bounded beam search using accumulated negative-logarithmic probability, a lightweight active-nibble heuristic, score-based pruning, beam-width constraints, and cost-based fallback states. Supervision is generated by separate bounded tail searches. For each supported horizon, a lightweight regressor predicts only the residual between the observed bounded tail cost and the classical activity heuristic. At inference time, the residuals are centered within the current frontier before being fused with the classical scores.

The main contributions are as follows.

1. We formulate short-horizon residual tail-cost estimation for differential-trail search, conditioned on the current difference and the target endpoint. The labels are explicitly treated as bounded-search observations rather than exact optimal costs, and the corpus is split by intermediate state to prevent endpoint-level leakage.

2. We introduce a selective integration policy that leaves the complete classical selection stage intact. Learned pruning is activated only with four or two rounds remaining, protects the best 25% of the frontier according to the classical score, removes at most 30% of the accepted states, and abstains when the pruning quota is smaller than five. This limits the model's authority without claiming a formal preservation guarantee.

3. We separate model development from search evaluation through state-grouped learning splits, independent pair construction, a frozen GIFT-64 holdout, and a deeper $D = 16$ stress test. Search failures are reported explicitly and are excluded from efficiency gains.

4. We add an equal-budget attribution experiment in which the frontier, protected core, and exact pruning quota are held fixed while only the removed-state ranking changes. This control tests whether the learned score contributes beyond the additional pruning stage itself. A removal-set audit distinguishes different ranking decisions from measurable downstream utility by recording which accepted states each score removes.

The implementation, frozen models, evaluation pairs, and result artifacts are publicly available on Github[1].

[1] `https://github.com/Gholizadeh-sh/ml-guided_beam-search`

The remainder of the paper is organized as follows. Section 2 defines the classical search and the selective learned intervention. Section 3 describes corpus construction, feature representations, and model selection. Section 4 specifies the frozen evaluation protocol, and Section 5 reports the search-level results. Section 6 discusses depth dependence, attribution, and the limits of empirical preservation, and Section 7 concludes the paper.

2 Methodology

This section presents our selective machine-learning-assisted framework for differential-trail search. The central design principle is that a learned heuristic should not replace the classical search policy. Instead, the classical beam search first determines the admissible working frontier, after which a learned tail-cost estimator is allowed to remove only a bounded fraction of low-priority states at selected late-search horizons. This design limits the authority of the learned component and permits the search to abstain from using it when the candidate frontier is too small.

2.1 Differential-Trail Search

Let a differential trail over D rounds be denoted by

$$\tau = (\Delta_0, \Delta_1, \ldots, \Delta_D)\,, \tag{1}$$

where Δ_r denotes the difference at the input of round r. For a transition from Δ_r to Δ_{r+1}, we express its cost in negative-logarithmic form as

$$w(\Delta_r, \Delta_{r+1}) = -\log_2 \Pr[\Delta_r \to \Delta_{r+1}]\,. \tag{2}$$

The accumulated cost of a partial trail reaching Δ_r is therefore

$$C_r = \sum_{i=0}^{r-1} w(\Delta_i, \Delta_{i+1})\,. \tag{3}$$

For a fixed input difference Δ_0 and target output difference $\Delta_D^\star$, the search attempts to find a low-cost trail satisfying

$$\Delta_D = \Delta_D^\star. \tag{4}$$

Lower negative-logarithmic cost corresponds to a higher-probability differential trail.

At each round, the current frontier is expanded through feasible differential transitions of the S-box layer and the cipher's diffusion layer. Duplicate states are merged by retaining only the smallest accumulated cost. Since the number of reachable states grows rapidly with the number of rounds, a bounded beam-search policy is used to control the frontier size.

2.2 Classical Search Baseline

For a candidate state Δ with h rounds remaining, we use the activity heuristic

$$\mathrm{LB}(\Delta, h) = h \cdot \mathrm{HW}_{\mathrm{nib}}(\Delta), \tag{5}$$

where $\mathrm{HW}_{\mathrm{nib}}(\Delta)$ is the number of active nibbles in Δ. We retain the notation LB because it is used throughout our implementation; however, it should be interpreted as a cheap search heuristic rather than as a formally proven admissible lower bound.

The classical score of a state is

$$S_{\mathrm{cl}}(\Delta) = C(\Delta) + \mathrm{LB}(\Delta, h), \tag{6}$$

where $C(\Delta)$ is the accumulated trail cost up to the current round.

After expansion and deduplication, candidates are ordered according to S_{cl}. The baseline search then applies its fixed pruning policy, including the score-based pruning threshold, the beam-width constraint, and a small set of cost-based fallback states ("lifeboats"). We denote the output of this complete classical selection stage by

$$\mathcal{K}_r = \mathrm{Select}_{\mathrm{cl}}\left(\mathcal{E}_r\right), \tag{7}$$

where $\mathcal{E}_r$ is the deduplicated set of states generated at round r.

An important property of our proposed method is that machine learning is applied *after* this classical selection. Consequently, the learned component cannot introduce a state rejected by the classical search or enlarge the classical frontier.

2.3 Learning a Residual Tail-Cost Estimate

A direct prediction of the complete remaining trail cost forces the model to learn both simple activity information and more subtle state-dependent structure. Instead, we use the classical activity heuristic as a baseline and learn only its residual.

For a state Δ, target difference $\Delta_D^\star$, and remaining horizon h, let

$$C_{\mathrm{tail}}^{(B)}\left(\Delta, \Delta_D^\star, h\right) \tag{8}$$

denote the best tail cost found by a fixed bounded tail-search procedure with budget B. The training target is defined as

$$R\left(\Delta, \Delta_D^\star, h\right) = C_{\mathrm{tail}}^{(B)}\left(\Delta, \Delta_D^\star, h\right) - \mathrm{LB}(\Delta, h). \tag{9}$$

The superscript (B) is important: the labels are produced by bounded search and are therefore empirical search labels rather than exact optimal tail costs. Accordingly, neither the learning target nor the resulting estimator is claimed to provide an admissible heuristic.

For each supported horizon h, a regression model f_h estimates

$$\widehat{R}_h = f_h\left(\Delta, \Delta_D^\star\right). \tag{10}$$

Separate horizon-specific experts are used because the statistical structure of a short remaining trail changes substantially with h.

At inference time, predictions are centered within the current classical frontier:

$$\widetilde{R}_h(\Delta) = \widehat{R}_h(\Delta) - \mathrm{median}_{\Delta' \in \mathcal{K}_r}\, \widehat{R}_h(\Delta'). \tag{11}$$

Centering removes the layer-dependent offset of the regressor and makes the learned signal primarily affect the relative ordering of candidate states.

The resulting learned score is

$$S_{\mathrm{ML}}(\Delta) = S_{\mathrm{cl}}(\Delta) + \lambda_h \widetilde{R}_h(\Delta), \tag{12}$$

where λ_h controls the influence of the learned residual at horizon h.

2.4 Selective and Bounded ML Intervention

Prediction accuracy alone does not guarantee safe search control: even a regressor with good average ranking performance may incorrectly demote a state that is required by the final trail. We therefore restrict the learned component through three mechanisms: horizon selection, a protected classical core, and bounded pruning authority.

First, the learned component is activated only at a predefined set of remaining horizons

$$\mathcal{H}_{\mathrm{ML}} = \{4, 2\}. \tag{13}$$

At every other horizon, including $h = 6$, the algorithm follows the classical search unchanged.

Second, after obtaining $\mathcal{K}_r$, the states are ordered according to their classical score. The best fraction ρ is designated as a protected classical core,

$$\mathcal{P}_r = \mathrm{Top}_\rho\left(\mathcal{K}_r; S_{\mathrm{cl}}\right). \tag{14}$$

States in $\mathcal{P}_r$ cannot be removed by the learned component.

Third, the maximum number of states that ML may remove is

$$q_r = \lfloor \eta \, |\mathcal{K}_r| \rfloor, \tag{15}$$

where η is the maximum pruning fraction. If

$$q_r < q_{\min}, \tag{16}$$

the model is not invoked and the algorithm abstains from learned pruning at that frontier.

Otherwise, the unprotected states

$$\mathcal{U}_r = \mathcal{K}_r \setminus \mathcal{P}_r \tag{17}$$

are ranked by S_{ML}, and at most q_r states with the worst learned scores are removed. The next frontier is therefore

$$\mathcal{F}_{r+1} = \mathcal{K}_r \setminus \mathrm{Worst}_{q_r}\left(\mathcal{U}_r; S_{\mathrm{ML}}\right). \tag{18}$$

This construction is deliberately asymmetric. Machine learning may only remove states from the frontier already accepted by the classical policy; it cannot rescue classically rejected candidates, alter the underlying differential transition probabilities, or expand the beam.

2.5 Final Search Procedure

Algorithm 1 summarizes the complete procedure.

Algorithm 1 Selective bounded ML-assisted differential-trail search

1. **Input:** input difference Δ_0, target $\Delta_D^\star$, total depth D, classical beam policy, experts $\{f_h\}$, active horizons $\mathcal{H}_{\mathrm{ML}}$, protected fraction ρ, pruning fraction η, minimum quota $q_{\min}$, and weights $\{\lambda_h\}$.
2. Initialize $\mathcal{F}_0 \leftarrow \{(\Delta_0, 0)\}$.
3. For $r = 0$ to $D - 1$:
   (a) Expand all states in $\mathcal{F}_r$ by one differential round and deduplicate equal output differences, keeping the minimum accumulated cost.
   (b) Set $h \leftarrow D - (r + 1)$ and $\mathcal{K}_r \leftarrow \mathrm{Select}_{\mathrm{cl}}(\mathcal{E}_r)$.
   (c) If $h \notin \mathcal{H}_{\mathrm{ML}}$, set $\mathcal{F}_{r+1} \leftarrow \mathcal{K}_r$ and continue.
   (d) Set $q_r \leftarrow \lfloor \eta |\mathcal{K}_r| \rfloor$. If $q_r < q_{\min}$, set $\mathcal{F}_{r+1} \leftarrow \mathcal{K}_r$ and continue.
   (e) Protect $\mathcal{P}_r \leftarrow \mathrm{Top}_\rho(\mathcal{K}_r; S_{\mathrm{cl}})$.
   (f) Predict $\widehat{R}_h(\Delta)$ for $\Delta \in \mathcal{K}_r$ and set $m_h \leftarrow \mathrm{median}_{\Delta \in \mathcal{K}_r} \widehat{R}_h(\Delta)$.
   (g) For every $\Delta \in \mathcal{K}_r$, compute

$$S_{\mathrm{ML}}(\Delta) \leftarrow S_{\mathrm{cl}}(\Delta) + \lambda_h\big(\widehat{R}_h(\Delta) - m_h\big).$$

   (h) Set $\mathcal{U}_r \leftarrow \mathcal{K}_r \setminus \mathcal{P}_r$ and remove $\mathcal{R}_r \leftarrow \mathrm{Worst}_{q_r}(\mathcal{U}_r; S_{\mathrm{ML}})$.
   (i) Update $\mathcal{F}_{r+1} \leftarrow \mathcal{K}_r \setminus \mathcal{R}_r$.
4. Return the lowest-cost trail reaching $\Delta_D^\star$, if found.

The final frozen configuration uses

$$\rho = 0.25, \qquad \eta = 0.30, \qquad q_{\min} = 5, \tag{19}$$

with

$$\lambda_4 = 1.0, \qquad \lambda_2 = 0.10. \tag{20}$$

These values are kept unchanged during the final validation experiments.

2.6 Scope of the Method

The proposed procedure is intended as a conservative learned modification of a bounded classical search, not as a replacement for exact cryptanalytic optimization. The protected core and pruning cap restrict the influence of the learned estimator, but they do not provide a formal guarantee that the same trail will always be retained. Consequently, search quality is evaluated empirically against the deterministic classical baseline under identical search budgets.

This distinction is essential to our evaluation. A reduction in expanded nodes is considered beneficial only when the learned search preserves the quality of the corresponding classical result. Cases in which learned pruning causes the classical target trail to be missed are treated as search failures, rather than as computational savings.

3 Dataset and Learning Setup

The learned component is trained to estimate the residual tail cost defined in Equation 9. Since exact optimal tail costs are expensive to obtain at the scale required for supervised learning, we construct training labels using bounded differential-tail searches. The resulting labels describe the best tail found under a fixed search budget and are therefore treated as bounded-search approximations rather than exact cryptanalytic optima.

We develop and select the learned experts on GIFT-64. The complete search-control policy is frozen before the final holdout and depth stress experiments.

3.1 Tail-Cost Corpus Construction

Each training example has the form

$$\left(\Delta, \Delta_D^\star, h, C_{\text{tail}}^{(B)}\right), \tag{21}$$

where Δ is an intermediate difference, $\Delta_D^\star$ is a reachable tail endpoint, h is the number of remaining rounds, and $C_{\text{tail}}^{(B)}$ is the minimum tail cost found under a fixed bounded search budget B.

Intermediate states are obtained by running short prefix searches from sparse input differences. For each selected intermediate state, a bounded forward tail search is executed for the required horizon. Rather than retaining only the best endpoint, endpoints are sampled across different positions of the resulting tail-cost ranking. This produces examples ranging from high-quality tails to weaker boundary cases and reduces the concentration of the training set around only the easiest endpoints.

For each retained state–endpoint pair, the regression target is

$$y = C_{\text{tail}}^{(B)} - h\,\mathrm{HW}_{\text{nib}}(\Delta). \tag{22}$$

Thus, the model is trained to predict only the deviation from the activity heuristic used by the classical search.

Importantly, a row in the corpus states only that the corresponding endpoint was found with the reported cost under the recorded bounded-search policy. We do not mark these costs as exact, and we make no claim that a lower-cost tail cannot exist outside the bounded search.

Table 1: GIFT-64 bounded tail-cost corpus after global deduplication.

Remaining horizon	Raw rows	Unique rows
$h = 2$	60,000	56,398
$h = 4$	30,000	28,735
$h = 6$	12,000	11,639
Total	102,000	96,772

3.2 GIFT-64 Development Corpus

The GIFT-64 production corpus was constructed for remaining horizons

$$h \in \{2, 4, 6\}. \tag{23}$$

Although the final method uses only $h = 2$ and $h = 4$, the $h = 6$ corpus was included during development to study whether earlier learned intervention could be useful.

The raw corpus contains 102,000 samples distributed as follows:

$$\begin{aligned} h = 2 &: 60{,}000, \\ h = 4 &: 30{,}000, \\ h = 6 &: 12{,}000. \end{aligned} \tag{24}$$

The horizons were further stratified by the nibble activity of the intermediate state. The raw target allocation was 10,000 examples for each activity bucket at $h = 2$, 6,000 at $h = 4$, and 3,000 at $h = 6$.

After global deduplication using the tuple

$$(\text{cipher}, \Delta, \Delta^{\star}_{D}, h, B)\,, \tag{25}$$

the corpus contains 96,772 unique examples. No conflicting tail-cost labels were observed among duplicate records. The resulting horizon distribution is shown in Table 1.

The tail-labeling budget is horizon specific. For $h = 2$, the bounded search uses an activity-cap schedule of $(6, 8)$, beam width 512, and a per-nibble transition cap of 8. For $h = 4$, the corresponding configuration uses the activity-cap schedule $(5, 6, 7, 8)$, beam width 1024, and per-nibble cap 4. For $h = 6$, the schedule is $(4, 5, 6, 7, 8, 9)$ with beam width 512 and per-nibble cap 4. Each horizon therefore has a single fixed label-generation policy throughout the corpus.

3.3 State-Grouped Data Splitting

Random row-level splitting would be inappropriate because several endpoints may originate from the same intermediate state. If different endpoints of the same state appeared in both training and validation data, the reported model quality could partially reflect memorization of the state rather than generalization to unseen search states.

We therefore split the GIFT-64 corpus by the pair

$$(\text{cipher}, \Delta), \tag{26}$$

so that every example derived from the same intermediate state belongs to exactly one partition. The resulting split is summarized in Table 2.

The three partitions contain no overlapping intermediate states. Model and feature-family decisions are made using development and validation evidence; the state grouping prevents direct state leakage across these partitions.

Table 2: State-grouped split of the deduplicated GIFT-64 corpus.

Partition	Rows	Unique states
Training	76,636	12,871
Validation	9,574	1,631
Test	10,562	1,593

3.4 Feature Representations

The models receive only information available at search time: the current intermediate difference Δ, the target output difference $\Delta_D^\star$, and the remaining horizon h. The original input difference and the accumulated partial-trail cost are deliberately excluded from the regression features.

We consider two feature representations that are retained in the final method.

Compact representation. Let

$$\Delta = (s_0, \ldots, s_{15}), \qquad \Delta_D^\star = (o_0, \ldots, o_{15}), \tag{27}$$

where each $s_i, o_i \in \{0, \ldots, 15\}$ denotes a nibble value. The compact representation concatenates the normalized nibble values, their binary activity masks, and four global scalar features:

$$\begin{aligned} \mathbf{x}_{\text{compact}} = [&s_0/15, \ldots, s_{15}/15, \\ &o_0/15, \ldots, o_{15}/15, \\ &\mathbf{1}_{s_0 \neq 0}, \ldots, \mathbf{1}_{s_{15} \neq 0}, \\ &\mathbf{1}_{o_0 \neq 0}, \ldots, \mathbf{1}_{o_{15} \neq 0}, \\ &\mathrm{HW}_{\text{nib}}(\Delta)/16, \\ &\mathrm{HW}_{\text{nib}}(\Delta_D^\star)/16, \\ &h/32,\ \mathrm{LB}(\Delta, h)/128]. \end{aligned} \tag{28}$$

This produces a 68-dimensional feature vector.

One-hot representation. The second representation independently one-hot encodes every nibble of Δ and $\Delta_D^\star$ over the 16 possible nibble values and appends four scalar features:

$$\mathbf{x}_{\text{onehot}} = [\mathrm{OH}(\Delta), \mathrm{OH}(\Delta_D^\star), \mathrm{HW}_{\text{nib}}(\Delta), \mathrm{HW}_{\text{nib}}(\Delta_D^\star), h, \mathrm{LB}(\Delta, h)]. \tag{29}$$

Since each 64-bit difference contains 16 nibbles, this representation has

$$16 \times 16 + 16 \times 16 + 4 = 516 \tag{30}$$

features.

3.5 Expert Screening and Final Models

During GIFT-64 development, we screen several lightweight regression families, including ridge regression, decision trees, and histogram gradient boosting. The screening considers both predictive quality and inference cost, because a model that provides slightly better regression accuracy may still be undesirable when invoked repeatedly inside the search loop.

The final $h = 2$ expert is a decision-tree regressor using the one-hot representation. The tree has maximum depth 10 and minimum leaf size 5. The final $h = 4$ expert is ridge regression with regularization parameter

$$\alpha_{\text{ridge}} = 1, \tag{31}$$

using the compact representation with feature standardization.

For reference, on the GIFT-64 validation split, the selected $h = 2$ tree achieves an MAE of 1.1518, $R^2 = 0.8647$, and Spearman correlation 0.9230. The selected $h = 4$ ridge model obtains an MAE of 3.6489, $R^2 = 0.6359$, and Spearman correlation 0.8041.

More expressive gradient-boosted models obtain better regression error in several settings, but their inference cost is roughly an order of magnitude larger. Since the learned component is embedded inside an iterative cryptanalytic search, we prefer the lightweight experts once their search-level behavior is found to be competitive.

The $h = 6$ expert is retained only during development experiments. Subsequent search-level diagnostics show that learned intervention at this horizon is less stable than at $h = 2$ and $h = 4$. It is therefore excluded from the frozen final search policy rather than replaced or retuned.

3.6 Separation Between Learning and Final Search Evaluation

The learning corpus and the final search benchmarks serve different purposes. Tail-corpus examples supervise the residual-cost estimators, whereas final search cases are independently generated input–output differential pairs on which the complete classical and ML-assisted algorithms are compared.

In particular, model regression metrics are not used as substitutes for search-level evaluation. A model can achieve good average prediction or ranking performance while still pruning a state required by a particular trail. For this reason, the primary evaluation of the proposed method is performed at the search level, measuring target recovery, trail cost, expanded nodes, and wall-clock time. These metrics and the final benchmark construction are described in Section 4.

4 Experimental Protocol

We evaluate the proposed method at the level of the complete differential-trail search rather than only through regression metrics. For every benchmark pair, the deterministic classical search and the selective ML-assisted search are executed under the same expansion, beam, and pruning configuration. The only difference is the bounded learned intervention described in Section 2.

The experimental design separates parameter development from final validation. This distinction is particularly important because the learned component may reduce the search effort while accidentally removing a state required to reach the target difference. We therefore treat preservation of the classical search result as the primary requirement and search-cost reduction as a secondary objective.

4.1 Common Search Configuration

Unless explicitly stated otherwise, both the classical and ML-assisted searches use the parameters in Table 3.

The number of cost-based lifeboat states is scheduled by search level. The schedules used at the three GIFT-64 search depths are

$$\mathcal{L}_8 = (2, 2, 4, 4, 8, 8, 16, 32), \tag{32}$$

$$\mathcal{L}_{12} = (2, 2, 4, 4, 8, 8, 16, 32, 64, 96, 128, 192), \tag{33}$$

$$\mathcal{L}_{16} = (2, 2, 4, 4, 8, 8, 16, 32, 64, 96, 128, 192, 192, 192, 192, 192). \tag{34}$$

The $D = 16$ schedule preserves the complete $D = 12$ prefix and extends the four additional levels using the existing maximum value of 192. This choice was fixed before inspecting the $D = 16$ ML-assisted results.

Table 3: Common bounded-search configuration used for the final experiments.

Parameter	Value
Beam width B	4096
Active-nibble cap t	3
Per-nibble transition cap	8
Score-threshold factor α	1.08
Score-cap fraction	0.5
Maximum lifeboat pool	512
Activity-heuristic floor scale	0.6

For the selective method, the final search-control configuration is fixed to

$$\mathcal{H}_{\mathrm{ML}} = \{4, 2\}, \qquad \rho = 0.25, \qquad \eta = 0.30, \qquad q_{\min} = 5, \tag{35}$$

with

$$\lambda_4 = 1.0, \qquad \lambda_2 = 0.10. \tag{36}$$

No learned intervention is performed at $h = 6$ or at any other remaining horizon.

4.2 Construction of Search Benchmarks

The search benchmarks consist of input–output differential pairs $(\Delta_0, \Delta_D^\star)$ generated independently by the classical bounded search. We begin from sparse input differences containing between one and three active nibbles. For each input difference, the classical search is run to the requested depth D, and target output differences are sampled from the resulting final frontier.

Selecting only the best-ranked endpoint would produce a benchmark dominated by particularly easy targets. We therefore stratify target outputs according to their rank in the final classical frontier. For GIFT-64, the three difficulty strata are

$$\begin{aligned} \text{easy:} \quad & 2 \le r < 4, \\ \text{medium:} \quad & 4 \le r < 33, \\ \text{hard:} \quad & 33 \le r < 80, \end{aligned} \tag{37}$$

where r denotes the one-based endpoint rank. Endpoints are sampled randomly within the corresponding rank interval.

The terms *easy*, *medium*, and *hard* therefore describe frontier-rank strata rather than a theoretical measure of differential difficulty. In particular, they should not be interpreted as formal complexity classes.

Within each benchmark collection, input and target output differences are kept unique. The final GIFT-64 holdout is also explicitly checked to be disjoint from all previously used development and diagnostic collections.

4.3 Development and Holdout Discipline

The final method was obtained through a sequence of development experiments on GIFT-64. These experiments included model screening, horizon-specific search tests, frontier diagnostics, bounded intervention-strength tests, and failure analysis.

An important methodological distinction is made between development evidence and final validation evidence. An initially fresh 36-pair $D = 8/D = 12$ evaluation exposed a search

failure under the earlier $\{6, 4, 2\}$ intervention policy. Once this failure was inspected and used to modify the algorithm, that collection was no longer considered a holdout and was reclassified as development data.

The identified failure was subsequently analyzed using horizon-specific interventions. The failure occurred for every tested configuration containing the $h = 6$ intervention, whereas the $h = 4$, $h = 2$, and combined $h = 4 + h = 2$ variants retained the classical target. This evidence motivated removal of $h = 6$ from the final policy.

The resulting $h = 4 + h = 2$ configuration was then confirmed on two existing development collections totaling 72 GIFT-64 pairs. Only after this configuration passed the predefined development criteria was the method frozen for final validation.

No parameter of the selective method was modified after inspection of the final GIFT-64 holdout.

4.4 Final GIFT-64 Holdout

The primary final validation set contains 48 previously unused GIFT-64 input–output pairs. It is balanced across two search depths and three difficulty strata:

$$\begin{aligned} D = 8: &\quad 8 \text{ easy} + 8 \text{ medium} + 8 \text{ hard}, \\ D = 12: &\quad 8 \text{ easy} + 8 \text{ medium} + 8 \text{ hard}. \end{aligned} \tag{38}$$

Thus, each depth contains 24 cases, and each difficulty class contains 16 cases overall.

All 48 input differences are unique, all 48 target output differences are unique, and the set is disjoint from the previously used development and evaluation collections.

The following criteria were specified for a successful final holdout:

1. 100% empirical quality preservation overall and separately at $D = 8$ and $D = 12$;
2. no benchmark case with a node-count regression;
3. positive mean per-case node reduction; and
4. positive aggregate node reduction.

The configuration is not modified if one of these criteria is violated.

4.5 GIFT-64 Depth Stress Test

After the $D = 8/D = 12$ method was frozen, we perform an additional $D = 16$ stress test to study whether the behavior persists at a longer search depth.

The stress set contains 18 independently generated pairs:

$$6 \text{ easy} + 6 \text{ medium} + 6 \text{ hard}. \tag{39}$$

The same $h = 4 + h = 2$ intervention policy, model weights, protected-core fraction, pruning authority, abstention threshold, and learned-score weights are used without modification.

The purpose of this experiment is primarily to test depth generalization and non-regression rather than to tune the method for $D = 16$. Accordingly, the predefined stress-test criteria are

1. 100% empirical quality preservation;
2. zero node-count regressions; and
3. non-negative aggregate node reduction.

A positive efficiency gain is reported when present, but it is not required for the depth-generalization test to be considered non-regressive. No $D = 16$-specific ML tuning is performed after the results are observed.

4.6 Equal-Budget Attribution Protocol

The main comparison establishes the behavior of the complete selective method, but it does not by itself show whether node reduction is caused by the learned ranking or simply by adding another bounded pruning stage. We therefore run an equal-budget ablation on the same 48 final GIFT-64 holdout pairs. The protected-core rule, pruning cap, minimum quota, active horizons, and exact quota calculation are identical across policies. Only the ranking used to choose states from the unprotected set is changed:

1. *learned*: remove the worst states under the learned score S_{ML};
2. *classical-score*: remove the same number of states using only the classical score S_{cl}; and
3. *random*: remove the same number uniformly from the unprotected set, repeated with 30 deterministic seeds.

We use two complementary comparisons. In the full-search ablation, each ranking policy runs end to end under the same quota rule. Later frontiers may then differ because earlier removal decisions change the search trajectory. In the matched-frontier ablation, all policies receive the same classical frontier, the same protected core, and the exact same prune count at each eligible intervention. The search is then continued classically from each resulting frontier. This second design isolates the immediate causal effect of removed-state ranking from differences in upstream trajectories.

For an active pair–horizon frontier, let $\mathcal{R}^{\mathrm{ML}}_{i,h}$ and $\mathcal{R}^{\mathrm{CL}}_{i,h}$ denote the states removed by learned and classical-score ranking under their common quota $q_{i,h}$. We measure removal agreement using

$$A_{i,h} = \frac{|\mathcal{R}^{\mathrm{ML}}_{i,h} \cap \mathcal{R}^{\mathrm{CL}}_{i,h}|}{q_{i,h}}, \tag{40}$$

and the Jaccard coefficient

$$J_{i,h} = \frac{|\mathcal{R}^{\mathrm{ML}}_{i,h} \cap \mathcal{R}^{\mathrm{CL}}_{i,h}|}{|\mathcal{R}^{\mathrm{ML}}_{i,h} \cup \mathcal{R}^{\mathrm{CL}}_{i,h}|}. \tag{41}$$

The different-decision fraction is $1 - A_{i,h}$. Frontiers on which the quota falls below $q_{\min}$ are recorded as inactive and excluded from numerical overlap averages. We additionally audit that both second-stage removal sets are disjoint from the states already rejected by the initial classical selection. This disjointness is expected by construction because both second-stage controls operate only on the accepted frontier.

Quality preservation remains the primary criterion. Random-policy node reductions are reported as a raw pruning–quality trade-off across all trials; they are not interpreted as valid efficiency gains when a target is lost.

4.7 Evaluation Metrics

Let C_i^{cl} and C_i^{ML} denote the best target-trail costs returned by the classical and ML-assisted searches for benchmark case i, respectively.

We define empirical quality preservation as

$$Q_i = \begin{cases} 1, & \text{if the ML-assisted search reaches the target and } C_i^{\mathrm{ML}} \le C_i^{\mathrm{cl}}, \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

The quality-preservation rate for a benchmark set $\mathcal{S}$ is

$$\mathrm{QPR}(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} Q_i. \tag{43}$$

This metric is relative to the deterministic bounded classical baseline. It does not imply that either method has found the mathematically optimal differential trail.

Let N_i^{cl} and N_i^{ML} denote the numbers of generated search nodes. The per-case node reduction is

$$R_i^{(N)} = 100 \left(1 - \frac{N_i^{\mathrm{ML}}}{N_i^{\mathrm{cl}}} \right). \tag{44}$$

A positive value denotes a reduction in generated nodes. We report both the arithmetic mean of $R_i^{(N)}$ and its median.

Because a simple average gives equal weight to small and large searches, we also report aggregate node reduction:

$$R_{\mathrm{agg}}^{(N)} = 100 \left(1 - \frac{\sum_{i \in \mathcal{S}} N_i^{\mathrm{ML}}}{\sum_{i \in \mathcal{S}} N_i^{\mathrm{cl}}} \right). \tag{45}$$

This metric is especially useful for detecting situations in which a small number of large regressions outweigh many small improvements.

Wall-clock measurements include model-inference overhead. Each final search case is executed five times for each method, and the median runtime is used for the timing comparison. Let $\widetilde{T}_i^{\mathrm{cl}}$ and $\widetilde{T}_i^{\mathrm{ML}}$ denote these median times. The corresponding time reduction is

$$R_i^{(T)} = 100 \left(1 - \frac{\widetilde{T}_i^{\mathrm{ML}}}{\widetilde{T}_i^{\mathrm{cl}}} \right). \tag{46}$$

Node counts are treated as the primary efficiency measure because they are deterministic under the fixed search configuration, whereas wall-clock measurements are more sensitive to system noise.

4.8 Treatment of Search Failures

A smaller search tree is not considered an efficiency improvement when it is obtained by pruning away the target trail. Consequently, a case with $Q_i = 0$ is reported as a quality failure regardless of how many nodes or how much runtime it saves.

For benchmark sets containing such failures, we distinguish between

1. the unconditional quality-preservation rate over all cases; and
2. efficiency statistics computed over the quality-preserved cases.

This prevents an early search failure from being counted as a successful speedup merely because the algorithm terminated after exploring fewer states. Raw node counts for failed cases are retained for diagnostic purposes but are not interpreted as valid efficiency gains.

4.9 Summary of Final Evaluation Sets

Final evaluation is performed on 66 independent GIFT-64 input–output pairs distributed across three search depths and three difficulty strata. The complete distribution is summarized in Table 4. Development collections used for method selection are not included in the final evaluation.

Table 4: Final search-level evaluation sets. Development collections used for method selection are not included.

Cipher	Depth	Easy	Medium	Hard	Total
GIFT-64	8	8	8	8	24
GIFT-64	12	8	8	8	24
GIFT-64	16	6	6	6	18
				Total	66

Table 5: Horizon-attribution experiment on the GIFT-64 development case that failed under the original $h = 6 + h = 4 + h = 2$ selective policy. "MISS" indicates that the target output difference was not recovered.

Configuration	Target found	Best cost	Node reduction
Classical	Yes	52.000	0.00%
$h = 6$	No	MISS	7.27%
$h = 4$	Yes	52.000	7.32%
$h = 2$	Yes	52.000	3.69%
$h = 6 + 4$	No	MISS	10.33%
$h = 6 + 2$	No	MISS	7.80%
$h = 4 + 2$	Yes	52.000	11.02%
$h = 6 + 4 + 2$	No	MISS	10.69%

5 Results

We now evaluate the proposed selective ML-assisted search at the level of complete differential-trail discovery. Following the protocol of Section 4, search quality is treated as the primary criterion. Node reduction is interpreted as an efficiency improvement only when the ML-assisted search preserves the classical result.

We first summarize the development evidence that motivated the final $h = 4 + h = 2$ policy, then report the frozen GIFT-64 holdout results, the $D = 16$ depth stress test, and the equal-budget attribution analysis.

5.1 Development Evidence and Horizon Selection

An earlier selective configuration allowed learned intervention at

$$h \in \{6, 4, 2\}. \tag{47}$$

Although this configuration reduced the number of generated nodes on many development cases, it failed to preserve the classical search result on one previously unseen $D = 8$ pair. Since this failure was subsequently inspected, the corresponding benchmark collection was reclassified as development data, as described in Section 4.3.

To identify which intervention horizon was associated with the failure, we performed a controlled horizon-attribution experiment on the affected pair. All other search parameters were held fixed, while the active learned horizons were varied.

A clear pattern emerges in this case: every tested configuration containing the $h = 6$ intervention loses the target, whereas every configuration without $h = 6$ preserves the classical best cost. This experiment does not prove that $h = 6$ is universally unsafe; however, together with the earlier frontier diagnostics, it provides sufficient development evidence to exclude $h = 6$ from the final frozen policy.

Table 6: Final GIFT-64 holdout results. Node reduction is measured relative to the deterministic bounded classical baseline.

D	n	Quality preservation	Positive cases	Mean node reduction	Aggregate node reduction	Median node reduction
8	24	100%	15/24	5.119%	6.832%	2.121%
12	24	100%	1/24	0.096%	0.099%	0.000%
All	48	100%	16/48	2.607%	2.359%	0.000%

Table 7: GIFT-64 final-holdout results grouped by endpoint-rank difficulty.

Difficulty	n	Quality preservation	Mean node reduction	Aggregate node reduction
Easy	16	100%	1.330%	1.022%
Medium	16	100%	2.937%	2.399%
Hard	16	100%	3.556%	3.787%

The resulting $h = 4 + h = 2$ configuration was subsequently evaluated on 72 development pairs collected from two independent development sets. It preserved the classical result on all 72 cases and produced no node-count regression. The mean per-case node reduction was 3.200%, while the aggregate node reduction was 2.044%. These results were used only to confirm the final configuration before freezing it; they are not part of the final holdout evaluation.

5.2 Final GIFT-64 Holdout

We next evaluate the frozen $h = 4 + h = 2$ method on the final 48-pair GIFT-64 holdout. The complete selective configuration was fixed before these pairs were inspected.

The primary result is that the selective method preserves the classical search quality on all 48 holdout cases. There are also no node-count regressions. Thus, the frozen method satisfies all predefined final-holdout criteria.

The efficiency effect, however, depends strongly on the search depth. At $D = 8$, 15 of the 24 cases produce a positive node reduction. The mean reduction is 5.119%, while the aggregate reduction reaches 6.832%. The median reduction is 2.121%, and the largest individual reduction is 33.740%.

At $D = 12$, the behavior is substantially different. Only one of the 24 cases produces a positive reduction in generated nodes. The mean and aggregate reductions fall to 0.096% and 0.099%, respectively, with a median of zero.

These results show that preserving the search result does not imply that the same intervention provides the same computational benefit at different depths. The method remains empirically non-regressive at $D = 12$, but its node-level efficiency advantage is almost completely neutralized.

5.3 Effect of Target Difficulty

The final $D = 8/D = 12$ holdout is balanced across easy, medium, and hard target rank strata. Table 7 reports the corresponding results after combining both depths.

All three strata retain the classical result in every case. Interestingly, the largest aggregate reduction is observed in the hard stratum rather than the easy stratum. We therefore do not observe evidence that the learned intervention is useful only on particularly easy differential pairs.

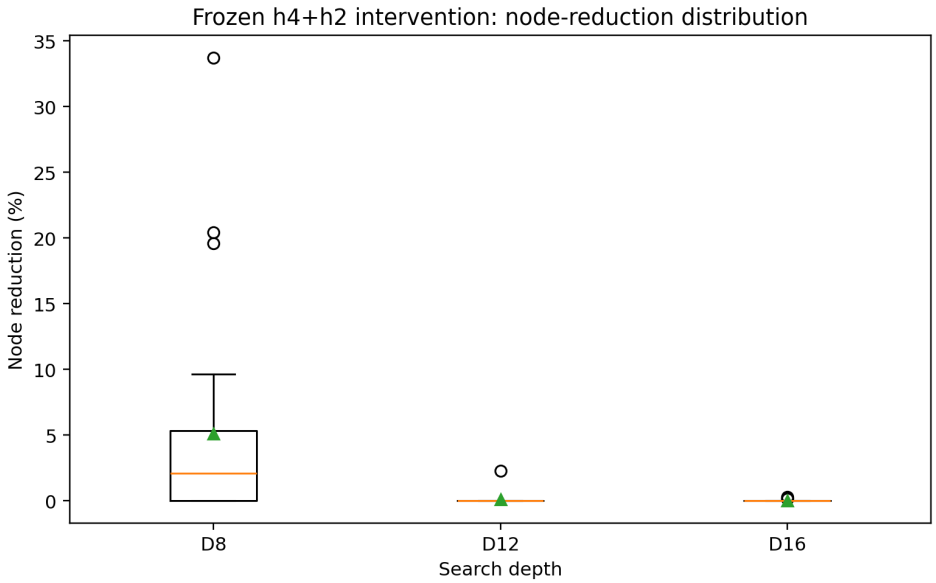

Figure 1: Distribution of node-count reductions produced by the frozen $h = 4 + h = 2$ policy at different GIFT-64 search depths.

Table 8: GIFT-64 $D = 16$ stress-test results.

Metric	Result
Number of cases	18
Quality preservation	100%
Positive / zero / negative	3 / 15 / 0
Mean node reduction	0.036%
Aggregate node reduction	0.040%
Median node reduction	0.000%
Maximum node reduction	0.311%
Median wall-clock reduction	2.141%

At the same time, the difficulty groups combine $D = 8$ and $D = 12$, and the strong depth dependence observed in Table 6 remains a major confounding factor. We consequently avoid interpreting the difficulty trend as a general relationship between endpoint rank and learned-search effectiveness.

5.4 GIFT-64 $D = 16$ Stress Test

The frozen policy is next evaluated at $D = 16$ without retraining or retuning. The purpose of this experiment is to test whether the method remains non-regressive when the total search depth is extended beyond the development and primary holdout depths.

All 18 targets are recovered with the same best cost as the classical baseline, and no node-count regression occurs. The predefined stress-test criteria are therefore satisfied.

The efficiency gain, however, is effectively negligible. The aggregate node reduction is only 0.040%, and 15 of the 18 cases have exactly zero node reduction.

Taken together with the $D = 8$ and $D = 12$ results, this establishes a clear empirical depth trend:

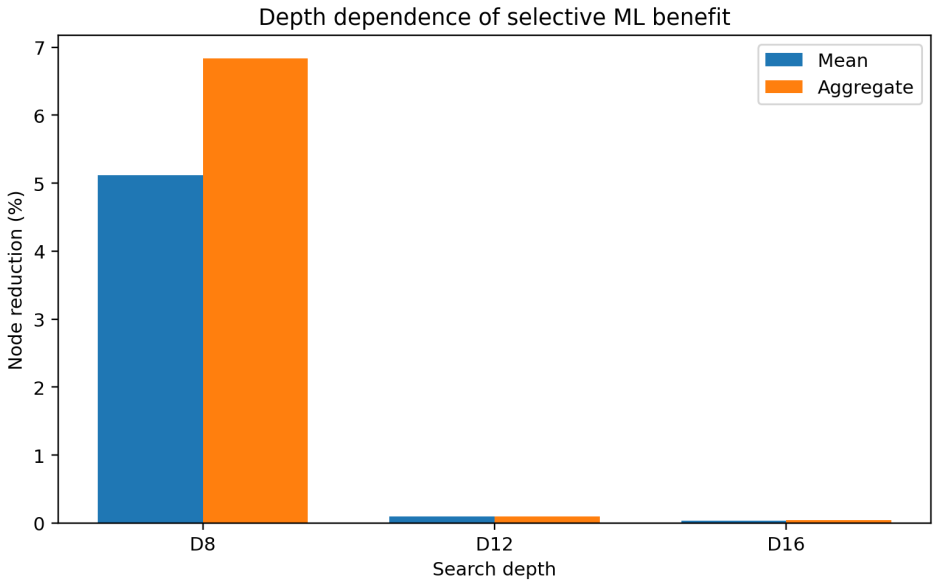

Figure 2: Mean and aggregate node reductions across the three evaluated GIFT-64 search depths. The efficiency benefit becomes increasingly small as the total depth grows.

$$\begin{aligned} D = 8: &\quad 6.832\% \text{ aggregate node reduction,} \\ D = 12: &\quad 0.099\% \text{ aggregate node reduction,} \\ D = 16: &\quad 0.040\% \text{ aggregate node reduction.} \end{aligned} \tag{48}$$

A natural explanation is that the learned intervention is intentionally restricted to the final four and two remaining rounds. As the total search depth grows, an increasing fraction of the overall expansion cost is incurred before the algorithm reaches either learned intervention point. Therefore, even effective pruning near the end of the search can account for only a small fraction of the total number of generated nodes.

This explanation is consistent with the observed trend, although the present experiments do not establish it as a formal scaling law.

5.5 Wall-Clock Behavior

Median wall-clock reductions on the final GIFT-64 benchmarks are 4.678%, 3.549%, and 2.141% for $D = 8$, $D = 12$, and $D = 16$, respectively.

We interpret these timing measurements cautiously. In particular, the $D = 12$ and $D = 16$ node reductions are nearly zero, while the measured median runtime remains slightly lower for the selective implementation. Such small timing differences may be affected by caching, memory behavior, interpreter overhead, and system noise.

For this reason, generated-node counts constitute our primary efficiency metric. Runtime results are included as complementary implementation-level evidence rather than as the basis for the main efficiency claim.

5.6 Equal-Budget Attribution and Empirical Preservation

Table 9 compares learned, classical-score, and random ranking under the controlled protocol of Section 4.6. Both deterministic rankings preserve the classical result on every one of the 48 holdout cases in the end-to-end experiment. Their aggregate node reductions are also nearly identical: 2.359% for learned ranking and 2.411% for classical-score ranking.

Table 9: Equal-budget ablation on the 48-pair GIFT-64 holdout. Random ranking uses 30 deterministic trials per pair. Its node reductions are raw trade-off measurements and include trials that fail quality preservation.

	Full search		Matched frontier	
Ranking policy	Quality	Aggregate reduction	Quality	Aggregate reduction
Learned score	100.00%	2.359%	100.00%	1.230%
Classical score	100.00%	2.411%	100.00%	1.237%
Random	96.39%	9.900%	98.09%	5.087%

Table 10: Overlap between learned and classical-score removal sets under an identical matched-frontier pruning budget. Micro statistics weight each removed state equally across active frontiers.

Horizon	Active frontiers	Different sets	Identical sets	Micro overlap	Micro Jaccard
$h = 4$	37	34	3	79.98%	66.64%
$h = 2$	48	23	25	92.91%	86.75%
All	85	57	28	91.18%	83.78%

The matched-frontier experiment strengthens this attribution check because all methods start from exactly the same frontier and remove exactly the same number of unprotected states. Across 96 pair–horizon frontiers, learned and classical-score ranking again preserve quality throughout, with aggregate reductions of 1.230% and 1.237%, respectively. Pairwise reduction comparisons favor learned ranking on 7 frontiers, classical ranking on 14, and tie on 75; a two-sided sign test gives $p = 0.189$. In the full-search comparison, the corresponding counts are 6, 9, and 33 ($p = 0.607$).

The removal-set audit in Table 10 examines whether the near-equal aggregate reductions conceal different ranking decisions. Eleven $h = 4$ frontiers are inactive because their quota is below $q_{\min} = 5$; the remaining 85 pair–horizon frontiers are included in the overlap statistics.

The learned and classical-score policies select different removal sets on 57 of the 85 active frontiers. The distinction is strongest at $h = 4$, where 34 of 37 active frontiers differ and the micro overlap is 79.98%. Across both horizons, each policy removes 6,901 states, of which 6,292 are common and 609 are exclusive to each ranking. The audit also confirms zero intersection between either second-stage removal set and the states rejected during initial classical selection, as expected because the second stage operates only on the accepted frontier. The learned and classical scores therefore make different pruning decisions among accepted states, especially at the earlier $h = 4$ horizon. This behavioral difference alone does not show that the learned decisions are better.

Random ranking removes the same number of states at every intervention and shows that an identical quota can lead to different empirical preservation outcomes. It produces larger raw node reductions, but quality falls to 96.39% end to end and 98.09% in the matched-frontier trials. These reductions cannot be counted as successful acceleration because some target trajectories are lost.

The controlled result is narrower than a claim that ML itself creates the observed node reduction. In this configuration, the protected core, capped quota, and late intervention explain most of the stable behavior, while the incremental effect of replacing classical-score ranking with learned ranking is not statistically or practically distinguishable on the evaluated holdout. The ablation nevertheless identifies a useful design boundary: both deterministic scores preserve

more evaluated trials than random removal, but the present residual model has not translated its different decisions into an advantage over the simpler classical ranking control.

6 Discussion and Limitations

The experiments characterize one restricted way to integrate learned guidance into differential-trail search without replacing the classical beam-search policy. The classical search first determines the admissible frontier, while an auxiliary ranking may remove only a bounded subset of states at short remaining horizons. The equal-budget controls show that the empirical stability of this design should not be attributed to the learned score alone.

This distinction is important. Several development experiments showed that good regression quality alone does not guarantee that a learned score can safely control the search. The final method therefore uses machine learning as a limited intervention mechanism rather than as the primary search policy, and its incremental value is evaluated against an equal-authority classical control.

6.1 Rationale for Restricting Learned Authority

The original motivation of this work was to use a learned tail-cost estimate more directly inside beam-search ranking. In principle, such an estimator can capture structural information that is not represented by a simple active-nibble heuristic. In practice, however, ranking errors are highly asymmetric.

A prediction error may only change the order of states that do not contribute to the selected target trail. An error affecting a critical ancestor, in contrast, can permanently remove the desired trail from the beam. Average regression metrics therefore do not fully characterize the risk introduced by a learned search heuristic.

The final design addresses this problem by limiting the model's authority in three ways. First, intervention occurs only after classical selection. Second, the highest-ranked classical states are protected from learned pruning. Third, the number of removable states is explicitly bounded. The model also abstains when the available pruning quota is too small.

These mechanisms do not provide a formal safety guarantee, but they reduce the number of decisions for which prediction error can affect the trajectory of the search. Quality is preserved on all evaluated $D = 8$, $D = 12$, and $D = 16$ cases. This result is consistent with the restricted design, although it does not isolate which safeguard is responsible for preservation.

6.2 Depth-Dependent Efficiency

The most pronounced limitation observed on GIFT-64 is the strong dependence of the efficiency gain on total search depth. The aggregate node reductions are

$$6.832\%, \qquad 0.099\%, \qquad 0.040\% \tag{49}$$

for $D = 8$, $D = 12$, and $D = 16$, respectively.

The learned policy itself is horizon-relative: it intervenes only when four or two rounds remain. Consequently, increasing the total number of rounds does not create additional learned intervention points. Instead, more of the overall search is completed entirely by the classical algorithm before the first learned decision is reached.

This provides a natural explanation for the observed trend. Let $N_{\mathrm{pre}}(D)$ denote the number of nodes generated before the search reaches the first learned horizon, and let $N_{\mathrm{late}}(D)$ denote the remaining search work. The maximum fraction of the computation that a purely late-stage intervention can influence is approximately bounded by

$$\frac{N_{\text{late}}(D)}{N_{\text{pre}}(D) + N_{\text{late}}(D)}. \tag{50}$$

As D increases while the intervention horizons remain fixed, this fraction can become progressively smaller. The empirical results are consistent with this interpretation.

We emphasize, however, that Equation 50 is an explanatory view rather than a proven scaling law. Search-frontier growth is highly cipher- and difference-dependent, and our experiments do not establish a general mathematical relationship between D and the attainable node reduction.

The final method is not a depth-independent acceleration technique. Its clearest efficiency benefit appears in the shorter search; the longer GIFT-64 experiments mainly demonstrate empirical non-regression.

6.3 Why Earlier Intervention Was Not Retained

A straightforward response to the depth-dependent limitation would be to activate machine learning earlier in the search. Our development experiments show why this is not necessarily safe.

The initially considered policy included an expert at $h = 6$. Frontier diagnostics indicated that ancestor retention at this horizon was weaker than at $h = 4$ and $h = 2$. More importantly, an independently generated $D = 8$ development case was missed by the combined $h = 6 + h = 4 + h = 2$ policy.

The subsequent horizon-attribution experiment produced a particularly clear development pattern: every tested configuration containing $h = 6$ missed the target, whereas the corresponding configurations without $h = 6$ retained the classical result.

This observation provides strong empirical motivation for disabling the $h = 6$ intervention in the final configuration. It does not establish that an $h = 6$ expert is inherently unsafe, nor that a different model, representation, training corpus, or integration rule could not use this horizon successfully.

Instead, the result exposes a broader trade-off. Earlier intervention has the potential to influence a larger fraction of the search tree and therefore offer greater efficiency gains, but prediction errors made at these horizons also propagate through more subsequent rounds. Restricting intervention to short horizons sacrifices some possible speedup in exchange for greater empirical stability.

6.4 Predictive Accuracy Is Not Equivalent to Search Safety

Another important observation is that model-level predictive performance and search-level behavior should not be conflated.

On the GIFT-64 validation data, the selected experts show relatively strong regression and ranking performance. Nevertheless, the development experiments demonstrate that even a model with useful average predictive signal can make an isolated ranking error that changes the final search result.

In principle, a pruning score need not estimate the absolute tail cost exactly; it would instead need to rank removable states well enough to improve the quality–efficiency trade-off. Whether the present learned score meets that condition must be determined by the equal-budget search experiment rather than by prediction metrics alone.

This is why the final evaluation emphasizes complete-search quantities such as target recovery, best trail cost, and generated nodes rather than selecting the final algorithm solely from MAE, R^2, or Spearman correlation.

The equal-budget ablation makes the same distinction from another direction. The learned models show held-out predictive signal for their bounded labels, but replacing the learned removed-state ranking with the classical score produces nearly the same node reduction and

the same quality preservation. Predictive quality is therefore evidence of statistical association with the labels, but it is not evidence by itself that the learned score improves the complete search.

6.5 What the Equal-Budget Control Attributes

An uncontrolled comparison between the classical baseline and the complete selective method combines two changes: an additional pruning stage is added, and a learned score ranks the removable states. The equal-budget experiment separates these effects by holding the pruning authority fixed.

The near equality between learned and classical-score ranking means that the 2.359% overall reduction of the complete learned method cannot be presented as an ML-specific gain. In this comparison, most of the stable reduction is attributable to the selective mechanism itself: protect a classical core, intervene late, cap the quota, and abstain on small frontiers. Random ranking shows that the same quota can remove a critical ancestor and reduce quality, but the current learned residual does not show an advantage over the classical ranking available at no training cost.

The removal-set audit refines this conclusion. Although the aggregate efficiency values are nearly equal, learned and classical-score ranking do not make identical decisions: their removal sets differ on 57 of 85 active frontiers and on 34 of 37 active $h = 4$ frontiers. This establishes only that the learned residual does not always reproduce the classical ordering. It does not establish that the changed decisions are more useful. The open problem is to convert a distinct ordering into a better quality–efficiency trade-off under the same pruning authority.

This is a useful outcome for method design. It identifies the appropriate baseline for future learned search and prevents model metrics from being used as a proxy for causal search benefit. A stronger learned component should be required to dominate the classical-score control at matched pruning budgets, not merely to reduce nodes relative to a search without the extra pruning stage.

6.6 Limits of the Labels and Classical Baseline

The tail-cost labels are the best results returned by a fixed search budget, not exact minimum costs. The regressors consequently learn some properties of the labeling procedure alongside properties of the cipher. A direct audit during development also found frontier states that could be recovered under a larger budget but were absent from the positive training corpus, particularly at longer horizons. More exhaustive labeling could narrow this mismatch, but at a substantially higher computational cost.

The classical reference has a similar limitation: it is a deterministic bounded heuristic, not an exact trail optimizer. Beam width, activity restrictions, transition caps, score thresholds, and the lifeboat schedule all affect the explored state set. Our comparisons are therefore paired comparisons against this implemented baseline, not against globally optimal differential trails.

The lifeboat schedule introduces an additional cross-depth qualification. Its counts are indexed by absolute search level rather than solely by the remaining horizon. This does not affect a comparison at a fixed D, where both searches use the same schedule, but it makes comparisons across depths less direct. We accordingly use the $D = 16$ experiment as a stress test of the frozen policy under the implemented baseline.

6.7 Empirical Preservation and Scope

Protecting a classical core and capping learned pruning reduce the number of decisions exposed to prediction error, but they cannot guarantee that every critical ancestor will survive. A formal equivalence guarantee would require identifying all such ancestors in advance or retaining the

entire classical frontier, which would remove the pruning benefit. The preservation results in this paper are therefore empirical. The random equal-budget trials provide a concrete example of the remaining risk: even a fixed, capped quota can remove a critical ancestor when the removed-state ranking is unsuitable.

Cipher coverage is limited to GIFT-64. The observed behavior need not extend to different S-box sizes, substantially different diffusion layers, ARX designs, or other trail representations. Cross-cipher evaluation is therefore left as future work rather than inferred from the single-cipher study.

6.8 Runtime and Implications for Learned Search

Generated-node counts are the primary efficiency measure because they are deterministic for a fixed configuration. Wall-clock results are secondary: Python overhead, memory allocation, scheduling, and cache effects can dominate small timing differences even after repeated runs. The timing experiments are useful mainly for checking that model inference does not introduce a prohibitive implementation cost.

More broadly, the experiments suggest that model authority is as important as predictive accuracy. Late intervention is easier to control, but it can affect only a small portion of a long search. Earlier intervention has greater potential influence and greater consequences when a ranking error occurs. This trade-off points toward confidence-aware abstention and learned decisions about when to intervene, rather than simply replacing the lightweight regressors with larger models.

7 Conclusion

We asked whether a lightweight tail-cost predictor could reduce a bounded search for differential trails while leaving the classical policy in control. The proposed method estimates residual adjustments to an active-nibble heuristic and applies them only after classical selection. Intervention is restricted to the $h = 4$ and $h = 2$ horizons, a protected classical core is retained, and the model abstains when the pruning quota is too small.

On GIFT-64, the frozen policy preserves target recovery and the best classical cost on all 48 holdout cases and all 18 $D = 16$ stress cases. The efficiency effect, however, is concentrated at the shorter depth: aggregate node reduction falls from 6.832% at $D = 8$ to 0.099% at $D = 12$ and 0.040% at $D = 16$. A late intervention can influence only the final part of the search, so the longer experiments mainly demonstrate non-regression against the implemented baseline rather than depth-independent acceleration.

The equal-budget ablation changes how this reduction should be attributed. Learned and classical-score ranking both preserve all 48 holdout cases and produce nearly identical overall aggregate reductions, 2.359% and 2.411%, respectively. Random ranking produces larger raw reductions but loses quality in repeated trials. The protected core, capped quota, and late intervention therefore account for most of the stable behavior, while the incremental advantage of the present learned ranking over a matched classical control is not established.

The overlap audit shows that the learned score changes the removed-state set on 57 of 85 active frontiers, particularly at $h = 4$. This is evidence of a different ranking among classically accepted states, not evidence that the changed decisions are better. The current integration rule has not translated that behavioral difference into a measurable downstream advantage.

The conclusions are tied to bounded labels, a bounded classical reference, and one 64-bit SPN; neither search is claimed to recover globally optimal trails. The most useful next step is not simply a larger regressor, but a learned decision rule that can demonstrate a better quality–efficiency trade-off than classical-score pruning under an identical quota. Possible directions include confidence-aware abstention, adaptive intervention horizons, stronger tail-labeling

procedures, and later cross-cipher evaluation. A further direction is to extend the present pruning-only design into a rescue-capable policy. Such a policy would expose the model to the preselection candidate pool and allow it, under a separate and strictly bounded budget, to retain a small number of candidates that classical selection would otherwise reject. Matched-budget experiments would then be needed to determine whether these rescued candidates improve final trail quality or efficiency without destabilizing the search. The present study contributes a controlled integration and evaluation framework, together with evidence about when its learned component changes pruning decisions and what must be controlled before future efficiency gains can be attributed specifically to machine learning.

References

[1] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) S-boxes to optimize probability of differential characteristics. *IACR Transactions on Symmetric Cryptology*, 2017(4):99–129, 2017.

[2] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present – towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[3] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[4] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[5] David Gerault, Anna Hambitzer, Moritz Huppert, and Stjepan Picek. Survey: Six years of neural differential cryptanalysis. *IACR Communications in Cryptology*, 3(2), 2026.

[6] Aron Gohr. Improving attacks on round-reduced Speck32/64 using deep learning. In *Advances in Cryptology – CRYPTO 2019*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[7] Mohamed Fadl Idris, Je Sen Teh, Jasy Liew Suet Yan, and Wei-Zhu Yeoh. A deep learning approach for active S-box prediction of lightweight generalized feistel block ciphers. *IEEE Access*, 9:104205–104216, 2021.

[8] Mohamed Fadl Idris, Je Sen Teh, and Mohd Najwadi Yusoff. DiffGen: A data-driven framework for generating truncated differentials. *Applied Intelligence*, 55(5):329, 2025.

[9] Fulei Ji, Wentao Zhang, and Tianyou Ding. Improving matsui's search algorithm for the best differential/linear trails and its applications for DES, DESL and GIFT. *The Computer Journal*, 64(4):610–627, 2021.

[10] Fulei Ji, Wentao Zhang, Chunning Zhou, and Tianyou Ding. Improved (related-key) differential cryptanalysis on GIFT. In *Selected Areas in Cryptography – SAC 2020*, volume 12804 of *Lecture Notes in Computer Science*, pages 198–228. Springer, 2020.

[11] Yunwen Liu and Yu Sasaki. Related-key boomerang attacks on GIFT with automated trail search including BCT effect. Cryptology ePrint Archive, Paper 2019/669, 2019.

[12] Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1995.

[13] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Transactions on Symmetric Cryptology*, 2021(1):269–315, 2021.

[14] Siwei Sun, Lei Hu, Ling Song, Yonghong Xie, and Peng Wang. Automatic security evaluation of block ciphers with s-bP structures against related-key differential attacks. Cryptology ePrint Archive, Paper 2013/547, 2013.

[15] Wei-Zhu Yeoh, Je Sen Teh, and Jiageng Chen. Automated enumeration of block cipher differentials: An optimized branch-and-bound GPU framework. *Journal of Information Security and Applications*, 65:103087, 2022.