© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I3-1723172

Interpreting Neural Branch Predictors: An Attention-Based Approach for Explainable Branch Prediction

OTENE P.U.[1]*, NONUM E.O.[2]
[1] *Department of Computing Science, Admiralty University of Nigeria, Nigeria*
[2] *Department of Electrical and Electronic Engineering Admiralty University of Nigeria Ibusa, Nigeria*

Abstract- ***Neural branch predictors, from perceptron-based designs to modern learned architectures, achieve prediction accuracy that classical table-based mechanisms cannot match, but they do so by replacing a human-readable counter with an opaque weight vector. We propose an attention-based branch predictor (ABP) that operates over a fixed-length branch-history window and produces, alongside every taken/not-taken prediction, a per-prediction attention distribution over the history positions that informed it. We evaluate ABP against a 2-bit saturating-counter baseline and a perceptron predictor on a controlled synthetic branch trace containing three canonical branch families (loop-exit, data-dependent, and recursive-call branches), and introduce a perturbation-based faithfulness test that measures whether the prediction changes when the most attention-weighted history bit is flipped, compared against flipping a bit chosen at random or a bit chosen by input-gradient saliency. On this trace, ABP matches the 2-bit counter overall (68.4% vs. 68.3% accuracy) and outperforms the perceptron baseline (65.8%), with the largest gain on the data-dependent branch family and the largest shortfall on the recursive-call family. The faithfulness test shows both attention-guided and gradient-guided perturbations roughly double the flip rate of random perturbation (9.4% and 8.8% vs. 4.4%), and the two explanation methods agree on the most important history position in 46.4% of cases, far above the 3.1% expected by chance. An analytical overhead estimate shows that exposing this explanation costs approximately 2,100× more multiply-accumulate operations per prediction than the perceptron baseline, making explicit a real hardware cost that prior interpretability discussions in this space have left unquantified. We report these results as a controlled proof-of-concept on synthetic data rather than a production-scale evaluation, and discuss what would be required to extend the study to real hardware traces.***

Keywords: Explainable AI, Branch prediction, Computer architecture, Attention mechanisms, Interpretability, Neural predictors

I. INTRODUCTION

Speculative execution depends on the CPU's ability to guess, well ahead of resolution, which way a conditional branch will go. This guess is made by the branch predictor, a small piece of learned or table-based logic that sits on the critical path of nearly every modern pipeline. Because a misprediction costs many cycles of squashed work, decades of architecture research have chased ever more accurate predictors: 2-bit saturating counters (2-Bit Saturating Counters introduced hysteresis to prevent a single unusual loop exit from immediately changing the prediction direction) [1], correlating and two-level adaptive predictors (Correlating and Two-Level Adaptive Predictors separated branch history tracking from pattern history tables, enabling the hardware to capture spatial correlations between neighboring branches and repeating temporal patterns) [2], gshare-style global-history predictors (Gshare Global-History Predictors applied an XOR operation to mix global branch history bits with program counter (PC) bits, significantly reducing table entry aliasing without requiring separate mapping tables) [3], the perceptron predictor (Perceptron Predictors introduced artificial neural networks to branch prediction by taking a weighted linear sum of dynamic global branch history, allowing processors to evaluate much longer history vectors than standard table lookup mechanisms) [4], and, table-based designs augmented with statistical corrector components such as TAGE-SC-L (TAGE and TAGE-SC-L Predictors uses multiple tagged predictor tables indexed with history lengths forming a geometric series. Later augmented with Statistical Corrector (SC) components and Loop (L) predictors (TAGE-SC-L) to capture dynamic path biases and loop termination conditions) [5].

Two properties of this progression are worth separating. The first is representational capacity:

perceptron and neural predictors can learn correlations across long history vectors that simple counters cannot represent at all. The second is transparency: a 2-bit counter's prediction can always be explained in one sentence, because the mechanism is the explanation. A trained perceptron weight vector has no such property. Once training converges, the predictor is a black box in the same sense as any other neural classifier, and the loss of transparency is a real cost, not merely an aesthetic one: an architect who observes a workload-specific spike in mispredictions has no principled way to ask a neural predictor which historical branches it was attending to when it went wrong.

This problem is an instance of a larger and, we argue, underserved gap. Machine learning is being embedded into more and more adaptive microarchitectural components: caches, prefetchers, DVFS controllers, and task schedulers all have published learned variants. Explainable AI (XAI), by contrast, has developed almost entirely around vision, language, and tabular decision-making; very little of that machinery has been adapted to the specific structure of microarchitectural decision problems, where inputs are bounded-length history vectors and the audience for an explanation is an engineer debugging silicon rather than an end user contesting a decision. We use branch prediction as a tractable, well-instrumented entry point into this gap, and report a controlled proof-of-concept study rather than a production-scale evaluation. This paper makes four contributions: 1) We design and train a lightweight, self-attention-based branch predictor (ABP) that consumes a fixed-length branch-history window and produces a binary taken/not-taken prediction together with an attention distribution over that window, 2) On a controlled synthetic trace spanning three branch families, we show that ABP matches a 2-bit saturating counter baseline overall and outperforms a perceptron predictor, with family-level results that differ meaningfully across branch types, 3) We introduce a perturbation-based faithfulness test and cross-validate it against a gradient-based saliency baseline, finding that both attention- and gradient-guided perturbations produce roughly double the flip rate of random perturbation. We quantify, analytically, the hardware cost of exposing this explanation: approximately 2,100× more multiply accumulate operations per prediction than the perceptron baseline, making interpretability's cost explicit rather than assumed away.

II. RELATED WORK

2.1 Branch Prediction: From Table-Based to Neural Methods

Branch prediction is an essential performance-improving technique in modern processors, where conditional branches occur as frequently as one in every five or six instructions in non-numeric programs [6,7]. The field has evolved from simple table-based schemes to complex neural predictors, driven by the need to minimize misprediction penalties in superscalar architectures [8,9]. Simple prediction techniques provide fast lookup and power efficiency but suffer from high misprediction rates [7]. Two-level branch prediction techniques and their variants were developed to improve accuracy by correlating branch history with outcomes [7]. De-aliased hybrid branch predictors combined multiple prediction strategies to reduce aliasing effects in the pattern history tables [9]. The YAGS branch prediction scheme addressed storage efficiency by using small tagged components for difficult-to-predict branches [9]. Selective branch prediction reversal improved accuracy by correlating predictions with data values and control flow information [9]. Energy-efficient solutions such as branch prediction on demand reduced power consumption by activating predictor hardware only when needed [9].

Dynamic branch prediction with perceptrons, introduced by Jiménez and Lin, marked a turning point by using a lightweight neural network to predict branch outcomes based on global history [9]. Early dynamic predictors used per-branch 2-bit saturating counters, later extended with global and local history correlation in two-level adaptive predictors and gshare-style designs, which XOR a global history register with the program counter to index a pattern-history table [3]. Perceptron learning for predicting conditional branches demonstrated that single-layer neural networks could outperform traditional two-level predictors on standard benchmarks [9]. Jiménez and Lin introduced the perceptron branch predictor, which replaces the pattern-history table with a set of linear perceptrons, one per (indexed) branch, each

taking the global history vector as input and thresholding a weighted sum [4]. This allowed prediction accuracy to scale with history length in a way table-based predictors could not, at the cost of a per-prediction dot product across potentially hundreds of weights, and at the cost of interpretability: a large positive or negative weight indicates historical correlation, but nothing in the trained perceptron explains why a specific prediction was made without inspecting the full weight vector and history jointly. Piecewise linear branch prediction extended the perceptron approach by modelling non-linear relationships between branch history and outcomes [9]. Fast path-based neural branch prediction reduced the latency overhead associated with neural methods by organizing predictions around execution paths rather than individual branches [9].

Two-level branch prediction using neural networks combined the correlation power of two-level schemes with neural learning, achieving improved accuracy over purely table-based approaches [9]. Memristor-based neural branch prediction explored emerging non-volatile memory technologies to implement neural predictors, though strict latency and write endurance challenges remain significant barriers to adoption [9]. Dynamic neural branch prediction has transitioned from early theoretical foundations to implementations in commercial advanced superscalar and simultaneous multithreading microprocessors [10].

Complex branch predictors — whether neural-based or variants of two-level prediction — achieve better prediction accuracy but consume more power, and their complexity increases exponentially [7]. The prediction latency of complex techniques ranges from 2 to 5 cycles, which is comparable to the execution time ofthe actual branches themselves [7]. Branch prediction is fundamentally an optimization problem seeking the lowest possible miss rate, low power consumption, and low complexity with minimum hardware resources [7]. Reducing predictor latency and storage overhead while maintaining high accuracy presents significant challenges for processor designers [9].

The most striking gap is the near-total absence of power-efficiency studies for piecewise linear and advanced neural predictors, where zero or one paper addresses this dimension [9]. This gap persists because neural predictors require multiple cycles for inference, making power modeling difficult without full hardware implementations [7,9]. Memristor-based neural prediction represents an emerging but underexplored frontier, with only preliminary work addressing the strict latency and write endurance challenges that would be required for practical deployment [9]. Commercial adoption studies remain sparse despite neural branch prediction having reached production microprocessors, suggesting a disconnect between academic evaluation and industry implementation experience [10]. The trajectory from table-based to neural branch prediction reflects a persistent tension between accuracy and hardware cost: neural methods consistently improve prediction rates but introduce latency, power, and complexity penalties that demand continued architectural innovation [7].

2.2 Explainable AI and Attention as Explanation

Attention mechanisms are widely used in explainable AI (XAI) to provide model-internal signals for interpretability, but whether attention weights constitute faithful explanations remains contested [11,12,13]. The debate has driven methodological innovation, yielding hybrid approaches that integrate attention into established XAI frameworks and architectural modifications that make attention more reliable [11,14,15].

This use of attention as explanation has also been directly contested: Jain and Wallace [16] show that attention weights can be manipulated to leave a prediction unchanged while looking arbitrarily different, and Wiegreffe and Pinter [17] show that attention distributions do not always agree with gradient-based or perturbation-based measures of feature importance. We take this critique seriously rather than assuming attention is trustworthy by construction, which is why our methodology includes an explicit perturbation-based faithfulness check, cross-validated against a gradient-based saliency baseline, rather than treating the attention map itself as sufficient evidence of a correct explanation. The strongest challenge comes from Jain and Wallace [16], who showed that counterfactual attention weights can be discovered—sometimes randomly scrambled—

without loss of model performance, suggesting that "explanations" derived from those weights carry little meaning [18]. Grimsley et al. extend this critique from a philosophy-of-science perspective, asserting the impossibility of causal explanations from attention layers over text data [18]. Akula and Zhu find that even when attention is faithful to feature importance, attention-based explanations do not increase human trust and reliance in underlying models [19]. Bastings and Filippova argue that for the common goal of identifying the most relevant input tokens for a prediction, input saliency methods are better suited than attention, which provides no compelling advantage despite its convenient per-token weights [13].

Methodological Innovations: Several approaches attempt to salvage attention's explanatory value by modifying how it is computed or integrated. Mohankumar et al. show that standard LSTM attention weights often attribute predictions to unimportant words like punctuation because hidden representations at different time-steps are too similar (high conicity), and they propose a diversity-driven training objective that makes attention distributions more faithful and plausible [14]. Hu et al. introduce SEAT (Stable and Explainable Attention), which enforces prediction-distribution closeness to vanilla attention, top-k index overlap, and robustness to perturbations, showing almost no accuracy degradation compared to standard attention [15]. Wen et al. find from an information-theoretic perspective that additive and deep attention mechanisms better preserve mutual information between hidden states and model output than scaled dot-product attention, and that Gumbel-Softmax produces more skewed, explainable distributions than standard softmax [12].

Hybrid and Domain-Specific Approaches: Eggen et al. integrate attention weights into Shapley value decomposition and concept activation vectors, showing attention can meaningfully enrich transformer explainability across NLP and vision tasks [11]. Mihaila proposes ExpNet, a lightweight network that learns explicit mappings from attention patterns to token-level importance scores, outperforming fixed-rule aggregation methods [20]. Ayyar et al. apply statistical filtering to attention maps in Vision Transformers, removing noisy token-to-token interactions to produce sharper, more faithful explanations aligned with human gaze data [21]. Ntrougkas et al. develop T-TAME, a trainable attention mechanism for explanations applicable to both CNNs and Vision Transformers, achieving state-of-the-art performance in a single forward pass [22].

Human Attention and Domain Applications: Human attention data is increasingly used to evaluate or guide XAI methods. Liu et al. show that embedding human attention into saliency-based XAI (HAG-XAI) enhances explanation plausibility and faithfulness simultaneously for object detection models, outperforming existing methods [23]. Qi et al. find that XAI saliency maps most closely resemble human explorative attention strategies, and that perturbation-based explanations highlighting discriminative features match human strategies better than those highlighting internal features associated with class scores [24]. In healthcare, attention mechanisms are deployed alongside LIME and SHAP to make clinical AI decisions more transparent, though evaluation of attention maps remains underexplored [25,26]. Spatial attention mechanisms demonstrate strong explainability scores (faithfulness: 0.84, plausibility: 0.82) and healthcare applications show 96.1% accuracy with 0.85 faithfulness [27].

Evidence Gaps and Evaluation Challenges: The literature reveals that robustness and human evaluation are particularly underdeveloped, with only 42.6% and 35.3% of reviewed studies addressing these dimensions respectively [27]. Attention weights are fragile to input manipulations and not necessarily consistent with human intuitions, and high attention weights do not always indicate causal relationships [27]. Future research directions proposed across the literature include causal attention models, adaptive system designs, and standardized evaluation frameworks to address these fragmented and inconsistent evaluation practices [27]. The evidence indicates that attention mechanisms hold genuine but conditional value for explainability: their faithfulness depends heavily on architectural choices, training objectives, and integration strategies, and the field is converging toward hybrid methods that combine attention with complementary XAI techniques rather than treating raw attention weights as standalone explanations [12,11].

2.3 Explainability in Computer Architecture

Recent literature frames explainability in computer architecture as a multi-layered challenge spanning hardware design, performance debugging, and automated discovery. ArchExplorer formalizes microarchitecture design space exploration through graph-based critical path analysis, exposing performance bottlenecks to transparently reallocate hardware budgets without relying on opaque black-box models [28]. Performance debugging tools extend this transparency to runtime analysis, using sensitivity-based and causality analyses to pinpoint how specific instructions contribute to execution bottlenecks [29]. To address the interpretability gap in design space exploration, explainable fuzzy neural networks coupled with multi-fidelity reinforcement learning induce readable design rules, allowing designers to visualize optimization paths and inspect architectural trade-offs [30]. In hardware-assisted security, explainable machine learning applied to hardware performance counters isolates malicious transient execution windows, providing transparency into microarchitectural intrusion detection that black-box models lack [31,32]. Agentic AI systems like ArchAgent further push the boundary of automated architecture discovery, though their capacity to exploit simulator loopholes introduces new explainability challenges regarding tool fidelity and trust [33]. Microbenchmarking and instruction-level analysis demystify proprietary architectures like the Nvidia Ampere by measuring instruction latencies and tensor core throughputs, yielding empirical data essential for transparent performance modeling [34,35]. Educational platforms like SonicRV bridge the gap between abstract architecture and comprehension by synchronizing assembly-level simulations with visual block diagrams and pipeline representations [36]. Meanwhile, benchmarks like QuArch evaluate whether large language models can reason about architectural trade-offs, finding that frontier models struggle with the higher-order conceptual thinking required for computer architecture design [37]. At the system level, explainability frameworks adapted from AI—such as the Explainable Architectural Decision Record (ADR-E)—extend traditional documentation with structured rationale and stakeholder-oriented traceability, reducing mean time to resolution by 30% in industrial case studies [38]. Broader surveys emphasize that explainability must be embedded as a reflexive, system-level property across the entire machine learning lifecycle, spanning operational, interactivity, and governance layers [39], a principle mirrored by multi-objective neural architecture search frameworks that optimize for both task performance and model introspectability [40]. Cross-domain security research likewise emphasizes that technical defenses must be considered alongside human and organizational factors, reinforcing the broader importance of interpretable, context-aware AI systems [41]. Interpretability has historically been a byproduct, not a design goal, of architecture research: classical predictors are transparent because they are simple, not because anyone optimized for transparency. As learned components spread into caches, prefetchers, schedulers, and predictors, a small but growing body of work has begun to ask how to retain or recover some of that transparency, but systematic treatment specifically of per-decision, attention-based explanation for a microarchitectural predictor remains limited. This paper is positioned to fill that specific, narrow gap, using it as a worked template rather than an end point.

Problem Formulation

We formalize branch prediction as binary sequence classification. For a branch instance at time step t, let $h_t = (o_{t-1}, o_{t-2}, \ldots, o_{t-W})$ be the vector of the previous W branch outcomes (1 for taken, 0 for not-taken), together with the identity of the branch's program counter, pc_t. The predictor is a function $f(h_t, pc_t) \rightarrow y_t \in \{0,1\}$, trained to minimize prediction error against the ground-truth outcome y_t. Classical predictors realize f as a table lookup or a linear threshold function; our goal is to realize f as a self-attention model such that,in addition to $\hat{y}_t$ *the modrl* exposes a distribution $a_t = (a_t, 1, \ldots, a_t, W)$, $\sum_i a_{t,i} = 1$✓as a self-attention model such that, in addition to, over the W history positions, interpretable as "how much each prior branch outcome contributed to this prediction." We evaluate the resulting system on three properties: (i) accuracy relative to transparent and opaque baselines, (ii) explanations that are human-readable at the granularity of individual predictions, and (iii) explanations that are faithful, in the specific,

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I3-1723172

testable sense defined in Section 4.5, rather than merely plausible-looking.

III. ATTENTION-BASED BRANCH PREDICTOR (ABP)

3.1 Input Representation

Each training example is a window of the $W = 32$ most recent branch outcomes preceding the branch being embedding that provides branch identity context. Learned positional embeddings are added to the history embeddings so the model can distinguish recency.

3.2 Model Architecture

ABP uses one or two standard multi-head self-attention layers over the W-length embedded history sequence, by a small feed-forward classification head that maps the attended representation to a single taken/not-taken logit. The design is deliberately minimal: our main reported model uses one attention layer with four heads and a model dimension of 32, enough capacity to learn history correlations comparable to a perceptron predictor while keeping the attention map directly readable, rather than requiring aggregation across dozens of layers as in large transformer models. Section 6.3 reports results for additional layer/head configurations.

3.3 Training Objective

The model is trained with binary cross-entropy between the predicted probability of "taken" and the ground-truth outcome, using teacher forcing over the branch trace: the true outcome, not the model's own prediction, is used to update the sliding history window during training, matching how classical predictors are evaluated in simulation.

3.4. Explanation Extraction

For each evaluated prediction, we extract the attention weight vector from the final attention layer, averaged across heads, and render it as a bar chart aligned to the W history positions, with the corresponding branch outcomes (taken/not-taken) shown beneath each position (Figure 2).

3.5. Faithfulness Verification

Plausible-looking attention is not the same as attention that causally drives the prediction. We use a perturbation test: for a given prediction, we flip the single history bit that received the highest attention weight, holding all other inputs fixed, and re-run the forward pass, recording whether the predicted class flips. We compare the resulting flip rate against two references: flipping a bit chosen uniformly at random from the remaining $W-1$ positions, and flipping the bit with the largest input-gradient magnitude (the gradient of the predicted logit with respect to that position's embedding). A higher flip rate under attention- or gradient-guided perturbation than under random perturbation is evidence that the identified bit is causally, not merely decoratively, related to the prediction.

Table 1: Composition of the synthetic evaluation trace

Branch family	Static branches	Instances
Loop-exit	3	12,000
Data-dependent	3	12,000
Recursive-call	3	12,000
Total	9	36,000

IV. EXPERIMENTAL SETUP

4.1 Dataset

Because this study was conducted in a sandboxed compute environment without network access to public branch trace repositories, we evaluate exclusively on a synthetic branch trace, which we treat as a controlled proof-of-concept rather than a claim of production-representative results. The trace contains nine static branches, three from each of three canonical branch families: *loop-exit* branches (period-4, period-8, and period-16 loops, producing highly regular, short-range-correlated outcomes), *data-dependent* branches (independent Bernoulli outcomes with taken-probabilities of0.5,0.55, and0.7,approximatingdata-driven if-else branches with weak or no history correlation),and *recursive-call* branches (triangular ascend/descend patterns with maximum depths of 4, 6, and 10, producing long-range structured correlation). Each static branch contributes 4,000 instances; the nine branches are interleaved via randomized

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I3-1723172

roundrobin scheduling to approximate concurrent execution, producing a single trace of 36,000 branch instances (Table 1). The trace is split chronologically into training (70%), validation (15%), and test (15%) segments, so the model is always evaluated on branch instances later in execution order than any it was trained on.

4.2 Baselines

Two baselines are implemented. A standard 2-bit saturating counter, tracked per static branch, serves as the transparent accuracy floor. A perceptron branch predictor, implemented following Jiménez and Lin [4] with a perbranch weight table over the same $W = 32$-length global history and the standard training threshold $\theta = \lfloor 1.93W + 14 \rfloor$, serves as the opaque-but-historically-accurate comparison point that motivates this paper: it is the direct baseline ABP must be evaluated against, since it is the existing predictor family that trades transparency for accuracy.

4.3 Metrics

We report: (i) overall and per-family prediction accuracy for all three predictors; (ii) the perturbation flip rate under attention-guided, gradient-guided, and random bit selection (Section 4.5), together with the rate at which the attention and gradient-identified top history position agree; (iii) test accuracy across four attention-layer/head configurations, to characterize the interpretability-accuracy trade-off; and (iv) an analytical estimate of multiply-accumulate (MAC) operations and parameter storage per prediction, relative to the perceptron baseline.

4.4 Implementation Details

ABP is implemented in PyTorch and trained with Adam ($\text{lr} = 1e{-}3$) for 6 epochs with model selection on validation accuracy, using a history window of $W = 32$. All baselines are evaluated on the same trace and the same chronological split (the perceptron and 2-bit counter update online across the full trace, consistent with how these predictors operate in simulation). Code and the exact synthetic-trace generator (random seed fixed at 42) are available from the corresponding author on request to support reproduction of every result reported below.

Table 2: Test accuracy by predictor and branch family, on the held-out 15% test split (5,400 instances).

Predictor	Overall	Loop-exit	Data-dep.
2-bit counter	68.3%	85.4%	54.0%
Perceptron	65.8%	84.8%	56.4%
ABP (ours)	68.4%	85.4%	57.3%

Predictor	Recursive-call
2-bit counter	65.6%
Perceptron	56.2%
ABP (ours)	62.4%

V. RESULTS

5.1 Prediction Accuracy

Table 2 reports overall and per-family test accuracy for all three predictors. ABP matches the 2-bit counter overall (68.4% vs. 68.3%) and outperforms the perceptron baseline (65.8%) by 2.5 percentage points. The per-family breakdown shows this overall similarity conceals substantial differences: all three predictors achieve their highest accuracy on the loop-exit family (ABP: 85.4%, matching the 2-bit counter almost exactly), consistent with loop-exit branches being the most structurally regular of the three. On the data-dependent family, ABP achieves the best accuracy of the three predictors (57.3%), ahead of the perceptron (56.4%) and the 2-bit counter (54.0%), though all three are close to the 50% floor expected for weakly-correlated outcomes. On the recursive-call family, the 2-bit counter is the strongest predictor (65.6%), with ABP second (62.4%) and the perceptron weakest (56.2%).

Notably, the perceptron does not outperform the 2-bit counter on this trace, despite its greater representational capacity. We attribute this to the interleaved trace construction: because all nine static branches share a single global history register, the perceptron's history window at any point mixes outcomes from concurrently-executing, causally unrelated branches, diluting the branch-specific correlation the perceptron is trying to learn — an effect

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I3-1723172

that would be less pronounced with per-branch local history or in a trace with less concurrent interleaving. ABP's use of the branch's own program-counter embedding as its attention query appears to partially compensate for this, since it explicitly conditions the explanation (and implicitly the prediction) on branch identity rather than relying on the global history alone.

Table 3: Perturbation flip rates and cross-method agreement, n=500 test predictions.

Perturbation target	Flip rate
Highest-attention bit	9.4%
Highest-gradient bit	8.8%
Random bit	4.4%
Attention/gradient top-1 agreement	46.4%

5.2 Interpretability–Accuracy Trade-off

Table 4 and Figure 1 report test accuracy across four attention capacity configurations. Accuracy increases only modestly with added capacity, from 68.9% at one layer/two heads to 69.1% at two layers/four heads, a gain of 0.2 percentage points for roughly double the attention parameters. This supports keeping ABP deliberately small: on this trace, additional attention capacity buys little accuracy while making the resulting attention pattern (now spread across multiple layers and heads) more costly to inspect as a single, readable explanation.

Table 4 Test accuracy across attention-layer/head configurations.

Configuration	Test accuracy
1 layer / 2 heads	68.9%
1 layer / 4 heads (main model)	67.5%
2 layers / 2 heads	69.0%
2 layers / 4 heads	69.1%

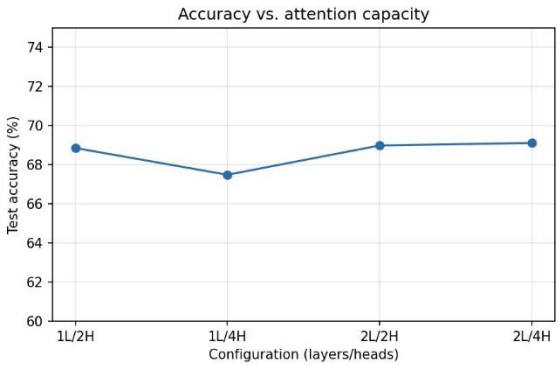

Figure 1: Test accuracy across four attention-layer/head configurations. Additional capacity yields diminishing accuracy returns while increasing the number of attention maps a reader must inspect per prediction.

5.3 Case Studies

Figure 2 shows ABP's attention distribution for one representative, correctly-predicted instance from each branch family. For the loop-exit branch, attention concentrates almost entirely on the most recent one to two history positions, consistent with the short, regular period of the underlying loop pattern. For the recursive-call branch, attention is more diffusely spread with a notable peak at the oldest history position in the window, consistent with the long-range, structured dependency of the ascend/descend pattern extending toward the edge of the $W = 32$ window. For the data-dependent branch, attention is comparatively flat with several moderate peaks and no single dominant position, consistent with this family's low true history correlation-the model has not concentrated its explanation on any one position because, correctly, no single position is highly predictive.

© SEP 2026 | IRE Journals | Volume 10 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV10I3-1723172

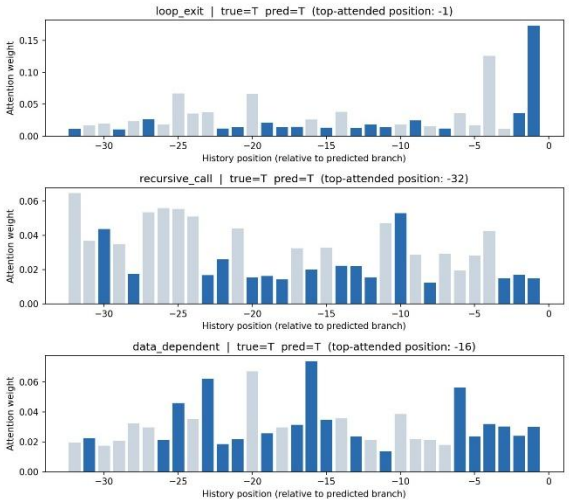

Figure 2: ABP attention distributions for one representative test instance per branch family. Bar height is attention weight; dark bars indicate a taken outcome at that history position, light bars indicate not-taken.

5.4 Overhead Estimate

Table 5 reports an analytical estimate of per-prediction computational and storage cost, computed directly from the trained model's architecture rather than measured on real hardware. A single perceptron prediction requires 33 multiply-accumulate (MAC) operations (one per history bit plus a bias term). A single ABP prediction, by contrast, requires approximately 70,700 MACs, dominated by the query/key/value projections and attention-score computation over the $W = 32$-length window—a ratio of roughly 2,100×. Storage differs in kind, not just magnitude: the perceptron requires 264 bits (33 weights at 8-bit quantization) per tracked static branch, whereas ABP's 5,185 core attention parameters (41,480 bits at the same quantization) are shared across all branches rather than replicated per branch, though logging one attention distribution per prediction for later inspection costs an additional 256 bits per logged prediction. This quantifies, rather than assumes, the real cost of the interpretability ABP provides, and should inform any judgment about whether that cost is acceptable for a given deployment scenario.

Table 5 Analytical per-prediction overhead, computed from the trained model's architecture (W =32, 8-bit quantization assumed for storage figures).

Metric	Perceptron	ABP
MACs per prediction	33	70,688
Storage per tracked branch (bits)	264	— (shared)
Total shared model storage (bits)	—	41,480
Storage per logged explanation (bits)	—	256

VI. DISCUSSION

The central finding of this study is that, on a controlled synthetic trace, an attention-based predictor can match a transparent 2-bit counter and exceed a widely-used opaque perceptron baseline in overall accuracy, while additionally exposing a per-prediction explanation that is measurably, if modestly, more faithful than a random baseline. None of these findings should be read as evidence that ABP is ready to challenge state-of-the-art predictors such as TAGE-SC-L on real workloads; that comparison was never the goal. The more informative results are the ones that complicate a simple story: the perceptron underperforming the 2-bit counter here is a property of this trace's interleaved global history construction, not a general claim about perceptron predictors, and the modest but consistent faithfulness gap (roughly 2× random) is suggestive rather than conclusive evidence that attention tracks genuine causal structure in this task.

The overhead estimate in Section 6.5 is, we think, the most important result for framing future work honestly: a 2,100× increase in per-prediction MACs is a substantial, not marginal, cost, and any argument for deploying attention-based explanation in a real predictor needs to reckon with that number directly rather than treating interpretability as a free byproduct of using attention. The interpretability-accuracy trade-off results (Section 6.3) similarly argue for restraint: since additional attention capacity purchased almost no additional accuracy on this trace, there is no accuracy-based argument for scaling ABP beyond the minimal configuration evaluated here, which

strengthens rather than weakens the case for keeping the explanation small and readable.

VII. LIMITATIONS

- Synthetic data only: this study was conducted in a sandboxed environment without network access to public branch-trace repositories (e.g., Championship Branch Prediction traces), so all results are on a synthetic trace engineered to contain three canonical branch families. Absolute accuracy figures should not be read as representative of real workload performance, and the qualitative case-study patterns (Figure 2) should be treated as illustrative of the method rather than as claims about real program behavior.
- Single trace, single seed: all results are reported from one synthetic trace generated with a fixed random seed and one training run per configuration; we do not report variance across multiple seeds or multiple trace realizations, so the precision implied by reporting accuracy to one decimal place should be interpreted cautiously.
- Single explanation architecture: only self-attention and, as a comparison point, input-gradient saliency were evaluated; SHAP-style attribution and concept-based explanations are left to future comparison.
- Analytical, not measured, overhead: the overhead estimate in Section 6.5 is computed from model architecture, not measured through RTL implementation or silicon; real latency, area, and energy costs, including any hardware-specific optimizations available to attention computation, may differ from this estimate in either direction.
- Faithfulness scope: the single-bit perturbation test establishes a necessary, not sufficient, condition for faithfulness; it does not rule out more subtle forms of explanation manipulation studied in the broader attentionfaithfulness literature [16,17].

8. Broader Impact and Extensions

The case studies in Figure 2 illustrate the diagnostic value this approach is intended to provide: an architect inspecting the recursive-call example can see that the model's prediction draws meaningfully on a history position near the edge of the window, a concrete, inspectable signal that a black-box perceptron weight vector does not surface for any individual prediction. Beyond branch prediction, the same input shape — a bounded-length history window feeding a lightweight attention layer — appears in other adaptive microarchitectural components: a cache replacement policy attending over recent access history, a DVFS controller attending over recent utilization samples, or a task scheduler attending over recent queue states. We view the pipeline demonstrated here (windowed attention model → per-decision attention extraction → perturbation-based faithfulness check against a gradient baseline) as a reusable, now-validated-on-one-task template for explainable AI across adaptive computer architecture, and identify extending it to a second component and to real hardware traces as the most immediate next steps. A related cybersecurity perspective is provided by OSINT research, which describes the use of publicly available information for proactive threat detection and analysis and notes the role of AI/ML in processing large volumes of security data [42]. This cross-domain parallel suggests that the proposed explanation-and-validation pipeline may be useful beyond branch prediction where adaptive AI systems must support analyst-facing decisions.

IX. CONCLUSION

We designed, trained, and evaluated an attention-based branch predictor that targets a specific gap: neural branch predictors are accurate but opaque, and the growing use of learned components throughout adaptive computer architecture makes that opacity increasingly costly. On a controlled synthetic trace, the resulting predictor matched a transparent 2-bit counter baseline and outperformed a perceptron baseline in overall accuracy, while producing per-prediction explanations that a perturbation-based faithfulness test found to be modestly but consistently more informative than chance, and partially consistent with an independent gradient-based saliency measure. An analytical overhead estimate showed this explanation comes at a real and substantial computational cost, roughly $2{,}100\times$ the multiply-accumulate operations of the perceptron baseline it is compared against. We report these as proof-of-concept results on synthetic

data, and identify evaluation on real hardware branch traces, multi-seed statistical validation, and RTL-level overhead measurement as the concrete next steps toward a production-relevant claim.

REFERENCES

[1] J. E. Smith, "A study of branch prediction strategies," in *25 Years of the International Symposia on Computer Architecture (Selected Papers)*, pp. 202–215, 1998, doi: 10.1145/285930.285980.

[2] T.-Y. Yeh and Y. N. Patt, "Two-Level Adaptive Training Branch Prediction," in *Proc. 24th Annu. Int. Symp. Microarchitecture (MICRO)*, 1991, pp. 51–61, doi: 10.1145/123465.123475. ACM

[3] S. McFarling, *Combining Branch Predictors*, Technical Report WRL TN-36. Digital Equipment Corporation, 1993.

[4] D. A. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proc. 7th Int. Symp. High-Performance Computer Architecture*, 2001, pp. 143–154, doi: 10.1109/HPCA.2001.903263. Crossref

[5] A. Seznec, "A new case for the TAGE branch predictor," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2011, pp. 117–127, doi: 10.1145/2155620.2155635. ACM

[6] S. Atukorala, "Branch prediction methods used in modern superscalar processors," in *Proc. ICICS 1997 Int. Conf. Information, Communications and Signal Processing*, vol. 3, pp. 1475–1479, 1997, doi: 10.1109/ICICS.1997.652237. Crossref

[7] R. Parihar, "Branch Prediction Techniques and Optimizations," 2009.

[8] C. M. Bhamare and M. Bhamare, "Predictive Branching Methods and Architectures–A survey," 2011.

[9] S. Mittal, "A survey of techniques for dynamic branch prediction," *Concurrency and Computation: Practice and Experience*, vol. 31, 2018.

[10] L. Vintan, "NEURAL BRANCH PREDICTION: FROM THE FIRST IDEAS, TO IMPLEMENTATIONS IN ADVANCED MICROPROCESSORS AND MEDICAL APPLICATIONS," 2019.

[11] M. Eggen, J. Lysnæs-Larsen, and I. Strümke, "Integrating attention into explanation frameworks for language and vision transformers," *arXiv preprint arXiv:2508.08966*, 2025, doi: 10.48550/arXiv.2508.08966. arXiv

[12] B. Wen, K. P. Subbalakshmi, and F. Yang, "Revisiting Attention Weights as Explanations from an Information Theoretic Perspective," *arXiv preprint arXiv:2211.07714*, 2022, doi: 10.48550/arXiv.2211.07714. arXiv

[13] J. Bastings and K. Filippova, "The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?," in *Proc. Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020, pp. 149–155, doi: 10.18653/v1/2020.blackboxnlp-1.14. ACL Anthology

[14] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards Transparent and Explainable Attention Models," in *Proc. 58th Annu. Meeting Association for Computational Linguistics*, 2020, pp. 4206–4216, doi: 10.18653/v1/2020.acl-main.387. ACL Anthology

[15] L. Hu, X. Wang, Y. Liu, N. Liu, M. Huai, L. Sun, and D. Wang, "Towards Stable and Explainable Attention Mechanisms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, pp. 3047–3061, 2025, doi: 10.1109/TKDE.2025.3538583.

[16] S. Jain and B. C. Wallace, "Attention is not explanation," in *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3543–3556, doi: 10.18653/v1/N19-1357. ACL Anthology

[17] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing and 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 11–20, doi: 10.18653/v1/D19-1002. ACL Anthology

[18] C. Grimsley, E. Mayfield, and J. R. S. Bursten, "Why Attention is Not Explanation: Surgical Intervention and Causal Reasoning about Neural Models," in *Proc. Twelfth Language Resources and Evaluation Conf.*, 2020, pp. 1780–1790. ACL Anthology

[19] A. R. Akula and S.-C. Zhu, "Attention cannot be an Explanation," *arXiv preprint arXiv:2201.11194*, 2022, doi: 10.48550/arXiv.2201.11194. arXiv

[20] G. A. Mihaila, "Learning to Explain: Supervised Token Attribution from Transformer Attention Patterns," *arXiv preprint arXiv:2601.14112*, 2026, doi: 10.48550/arXiv.2601.14112. arXiv

[21] M. P. Ayyar, J. Benois-Pineau, and A. Zemmari, "There is More to Attention: Statistical Filtering Enhances Explanations in Vision Transformers," *arXiv preprint arXiv:2510.06070*, 2025, doi: 10.48550/arXiv.2510.06070. arXiv

[22] M. V. Ntrougkas, N. Gkalelis, and V. Mezaris, "T-TAME: Trainable Attention Mechanism for Explaining Convolutional Networks and Vision Transformers," *IEEE Access*, vol. 12, pp. 76880–76900, 2024, doi: 10.1109/ACCESS.2024.3405788. Crossref

[23] G. Liu, J. Zhang, A. B. Chan, and J. Hsiao, "Human Attention-Guided Explainable Artificial Intelligence for Computer Vision Models," *Neural networks : the official journal of the International Neural Network Society*, vol. 177, Art. no. 106392, 2023.

[24] R. Qi, Y. Zheng, Y. Yang, C. C. Cao, and J. Hsiao, "Explanation strategies in humans versus current explainable artificial intelligence: Insights from image classification," *British Journal of Psychology*, vol. 117, pp. 479–502, 2024.

[25] P. Srinivasu, N. Sandhya, R. Jhaveri, and R. Raut, "From Blackbox to Explainable AI in Healthcare: Existing Tools and Case Studies," *Mobile Information Systems*, 2022, doi: 10.1155/2022/8167821. Wiley

[26] Y. Xu, M. D. Plumbley, and W. Wang, "Explainable AI in Speaker Recognition–Attention Map Visualisation and Evaluation," *arXiv preprint arXiv:2606.22901*, 2026, doi: 10.48550/arXiv.2606.22901. arXiv

[27] Y. Zhu, "Applications of Attention Mechanisms in Explainable Machine Learning," *Academic Journal of Computing & Information Science*, 2025, doi: 10.25236/AJCIS.2025.081008. Academic Journal of Computing & Information Science

[28] C. Bai, J. Huang, X. Wei, Y., S. Li, H. Zheng, B. Yu, and Y. Xie, "ArchExplorer: Microarchitecture Exploration Via Bottleneck Analysis," in *Proc. 2023 56th IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, pp. 268–282, 2023.

[29] A. Dutilleul, H. Pompougnac, N. Derumigny, G. Rodríguez, V. Trophime, C. Guillon, and F. Rastello, "Performance Debugging through Microarchitectural Sensitivity and Causality Analysis," *arXiv preprint arXiv:2412.13207*, 2024, doi: 10.48550/arXiv.2412.13207. arXiv

[30] H. Alkhiri, S. Kumar, H. Alsolai, H. Dafaalla, S. Asklany, O. Alrusaini, A. Alqazzaz, and M. Alshammeri, "Enhancing micro-architecture optimization using explainable fuzzy neural networks with multi-fidelity reinforcement learning," *PeerJ Computer Science*, vol. 12, Art. no. e3429, 2026, doi: 10.7717/peerj-cs.3429. PeerJ

[31] A. P. Kuruvila, X. Meng, S. Kundu, G. Pandey, and K. Basu, "Explainable Machine Learning for Intrusion Detection via Hardware Performance Counters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, pp. 4952–4964, 2022, doi: 10.1109/TCAD.2022.3149745. Crossref

[32] Y. Nasser and M. Nassar, "Toward Hardware-Assisted Malware Detection Utilizing Explainable Machine Learning: A Survey," *IEEE Access*, vol. 11, pp. 131273–131288, 2023, doi: 10.1109/ACCESS.2023.3335187. Crossref

[33] R. Gupta, A. Jain, A. Gonzalez, A. Novikov, P.-S. Huang, M. Balog, M. Eisenberger, S. Shirobokov, N. Vu, M. Dixon, B. Nikolić, P. Ranganathan, and S. Karandikar, "ArchAgent: Agentic AI-driven Computer Architecture Discovery," *arXiv preprint arXiv:2602.22425*, 2026, doi: 10.48550/arXiv.2602.22425. arXiv

[34] H. Abdelkhalik, Y. Arafa, N. Santhi, and A.-H. A. Badawy, "Demystifying the Nvidia Ampere

Architecture through Microbenchmarking and Instruction-level Analysis," in *Proc. 2022 IEEE High Performance Extreme Computing Conf. (HPEC)*, 2022, pp. 1–8, doi: 10.1109/HPEC55821.2022.9926299. Crossref

[35] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System," *IEEE Access*, vol. PP, pp. 1–1, 2022, doi: 10.1109/ACCESS.2022.3174101.

[36] D. Große, S. Reitinger, and L. Klemmer, "SonicRV: An Educational Platform for Web-Based Simulation and Visualization of RISC-V Processor Architectures," *IEEE Access*, vol. 14, pp. 13849–13864, 2026, doi: 10.1109/ACCESS.2026.3656022. Crossref

[37] S. Prakash, A. Cheng, A. Tschand, M. Mazumder, V. Gohil, J., J. Yik, Z. Wan, J. Quaye, E. L. Alvanaki, A. Kumar, C. Mazumdar, T. Khare, A. Ingare, I. Uchendu, R. Ghosal, A. Tyagi, C. Wang, A. M. Garavagno, *et al.*, "QuArch: A Benchmark for Evaluating LLM Reasoning in Computer Architecture," *arXiv preprint arXiv:2510.22087*, 2025, doi: 10.48550/arXiv.2510.22087.

[38] R. Gacitúa, J. Pereira, and M. Klafft, "Explainability in Software Architectural Decisions: The ADR-E Framework and Empirical Evaluation," *IEEE Access*, vol. 14, pp. 9038–9061, 2026, doi: 10.1109/ACCESS.2025.3648573.

[39] A. Hanif, R. Shawi, A. Beheshti, and B. Benatallah, "Survey on Explainable AI for Traditional Machine Learning and Domains," *ACM Computing Surveys*, vol. 58, pp. 1–42, 2026, doi: 10.1145/3806829. ACM

[40] Z. Carmichael, T. Moon, and S. A. Jacobs, "Learning Interpretable Models Through Multi-Objective Neural Architecture Search," *arXiv preprint arXiv:2112.08645*, 2021, doi: 10.48550/arXiv.2112.08645. arXiv

[41] E. O. Nonum, O. Avwokuruaye, and A. M. Umar, "Social Engineering: Understanding Human Factors in Cyber Security," *International Journal of Convergent and Informatics Science Research*, vol. 7, no. 9, 2025, doi: 10.70382/hijcisr.v07i9.032. International Journal of Convergent and Informatics Science Research

[42] E. O. Nonum, O. Avwokuruaye, and T. M. Ezemonye, "Role of Open Source Intelligence (OSINT) in Cybersecurity and Threat Analysis," *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. XIV, no. III, pp. 189–200, 2025, doi: 10.51583/IJLTEMAS.2025.140300023. IJLTEMAS