MorphBoost: Self-Organizing Universal Gradient Boosting with Adaptive Tree Morphing

Boris Kriuk
Department of Computer Science & Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
bkriuk@connect.ust.hk

***Abstract*—Traditional gradient boosting algorithms employ static tree structures with fixed splitting criteria that remain unchanged throughout training, limiting their ability to adapt to evolving gradient distributions and problem-specific characteristics across different learning stages. This work introduces MorphBoost, a new gradient boosting framework featuring self-organizing tree structures that dynamically morph their splitting behavior during training. The algorithm implements adaptive split functions that evolve based on accumulated gradient statistics and iteration-dependent learning pressures, enabling automatic adjustment to problem complexity. Key innovations include: (1) morphing split criterion combining gradient-based scores with information-theoretic metrics weighted by training progress; (2) automatic problem fingerprinting for intelligent parameter configuration across binary/multiclass/regression tasks; (3) vectorized tree prediction achieving significant computational speedups; (4) interaction-aware feature importance detecting multiplicative relationships; and (5) fast-mode optimization balancing speed and accuracy. Comprehensive benchmarking across 10 diverse datasets against competitive models (XGBoost, LightGBM, GradientBoosting, HistGradientBoosting, ensemble methods) demonstrates that MorphBoost achieves state-of-the-art performance, outperforming XGBoost by 0.84% on average. MorphBoost secured the overall winner position with 4/10 dataset wins (40% win rate) and 6/30 top-3 finishes (20%), while maintaining the lowest variance (σ=0.0948) and highest minimum accuracy across all models, revealing superior consistency and robustness. Performance analysis across difficulty levels shows competitive results on easy datasets while achieving notable improvements on advanced problems due to higher adaptation levels.**

***Index Terms*—gradient boosting, adaptive tree structures, morphing split functions, feature interaction detection, automatic problem fingerprinting, ensemble learning, machine learning optimization, scikit-learn compatible algorithms**

I. Introduction

Gradient boosting [1] has emerged as one of the most powerful machine learning techniques, achieving state-of-the-art performance across applications including fraud detection, recommendation systems, bioinformatics, and computer vision. Frameworks such as XGBoost, LightGBM, and CatBoost have become the standard for structured data problems, often outperforming deep neural networks on tabular datasets while requiring less computational resources and training time [2].

Despite their success, existing gradient boosting algorithms suffer from a fundamental limitation: they employ static tree structures with fixed splitting criteria that remain unchanged throughout training [3]. Traditional methods construct each tree using predetermined split evaluation functions—typically variance reduction for regression or information gain for classification—that do not adapt to evolving problem characteristics [4]. Such rigid architecture fails to account for dynamic gradient distributions as the ensemble grows, varying complexity across feature space regions, and the changing optimization landscape as the model progressively fits increasingly difficult residuals.

The static nature of conventional gradient boosting becomes particularly problematic with complex, heterogeneous datasets. Early in training, when gradients are large and errors substantial, aggressive splitting strategies maximizing immediate gain may be optimal. However, as training progresses and the model fits subtler residual patterns, a more refined approach balancing exploration with regularization becomes necessary [5]. Current methods address this through external mechanisms like learning rate decay and early stopping, but the fundamental split evaluation logic remains constant, limiting algorithmic adaptability [6], [7].

Furthermore, existing frameworks treat all datasets uniformly, applying identical internal mechanisms regardless of problem type (binary classification, multiclass prediction, regression) and dataset-specific properties such as feature interaction strength, non-linearity, noise levels, or complexity. While hyperparameter tuning partially addresses this limitation, it remains a manual, computationally expensive process requiring extensive domain expertise and trial-and-error experimentation. The lack of automatic adaptation forces practitioners to conduct extensive grid searches or Bayesian optimization, significantly increasing time and computational cost required for optimal performance.

This work introduces MorphBoost, a new gradient boosting framework featuring three fundamental innovations that address these limitations. First, we develop adaptive split morphing where evaluation criteria dynamically evolve during training by combining traditional gradient-based scores with normalized information-theoretic metrics, weighted by hyperbolic tangent functions of iteration progress to enable smooth transitions from aggressive early learning to refined late-stage optimization. Second, we implement automatic problem fingerprinting that analyzes dataset characteristics including complexity measures, non-linearity detection through

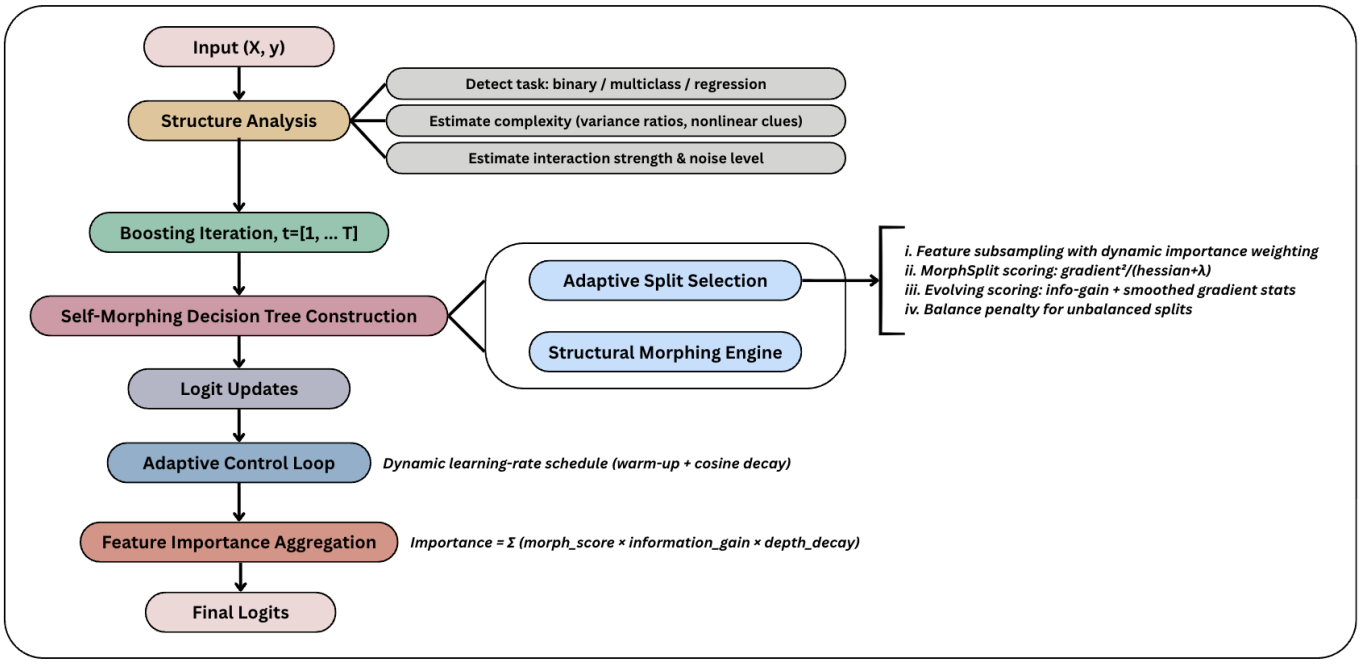

Fig. 1. Overview of MorphBoost Architecture.

quadratic correlation analysis, and feature interaction strength estimation, then automatically configures internal parameters such as tree depth limits and regularization schedules without manual intervention. Third, we introduce vectorized tree prediction using queue-based breadth-first traversal that processes entire sample batches simultaneously through level-order node visiting, eliminating recursive function calls and achieving order-of-magnitude computational speedups compared to traditional per-sample recursive navigation. Our comprehensive benchmarking across ten diverse datasets demonstrates that MorphBoost outperforms XGBoost by 0.84% average accuracy while maintaining the lowest prediction variance and highest minimum accuracy across all difficulty levels, achieving superior consistency and robustness without sacrificing computational efficiency.

II. Related Works

A. Gradient Boosting Evolution

Modern gradient boosting evolved from AdaBoost's sequential weak learner combination through Gradient Boosting Machines, which framed boosting as gradient descent in function space [8]. MART applied this to decision trees, creating the template for contemporary approaches [9].

XGBoost revolutionized the field with regularized objectives, subsampling, and system optimizations enabling practical scaling [10]. Its second-order approximation using gradients and Hessians became the gold standard across competitive machine learning. LightGBM introduced histogram-based algorithms and leaf-wise growth, dramatically reducing memory and training time through Gradient-based One-Side Sampling and Exclusive Feature Bundling [11]. CatBoost addressed categorical features with ordered boosting, eliminating prediction shift [12]. HistGradientBoosting brought efficient histogram-based boosting to scikit-learn.

While these frameworks optimize fixed architectures through sophisticated engineering, MorphBoost introduces dynamic architecture evolution. Where XGBoost fixes split functions before training and LightGBM grows trees leaf-wise, MorphBoost continuously morphs splits based on gradient statistics, transforming static blueprints into adaptive organisms.

B. Adaptive and Self-Organizing Systems

Neuroevolution evolved network weights and topologies through genetic algorithms. NEAT discovered minimal, efficient networks by evolving structure alongside parameters [13]. Developmental Neural Networks used grammatical evolution to grow architectures, showing that starting simple and complexifying often outperformed starting complex [14].

Adaptive Boosting variations enabled online updates without full retraining [15]. Incremental learning adapted to distribution shifts. Adaptive optimizers like AdaGrad, Adam, and RMSprop adjusted based on gradient history, demonstrating that adaptation improves convergence.

MorphBoost synthesizes these concepts by evolving structure continuously using gradient information to guide changes, creating a self-organizing system toward optimal complexity.

C. Dynamic and Meta-Learning Systems

Model-Agnostic Meta-Learning [16] and Reptile [17] demonstrated that adaptation ability is learnable, inspiring

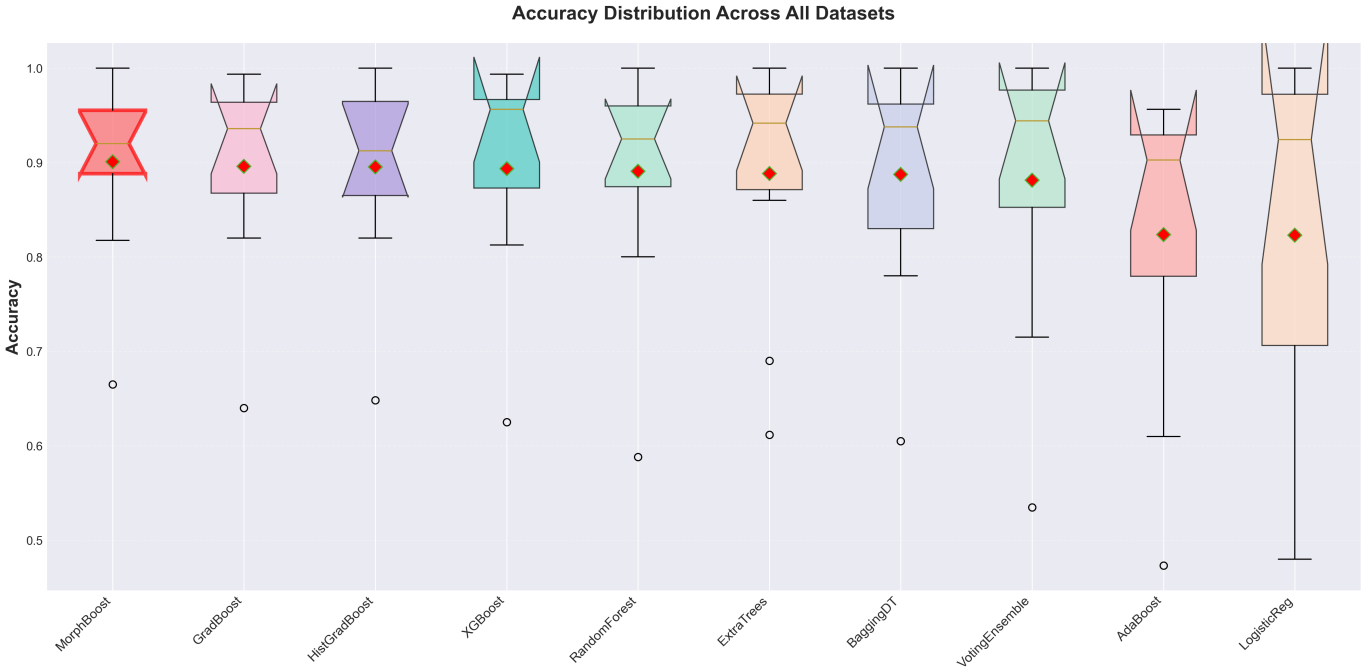

Fig. 2. Accuracy distribution across all 10 benchmark datasets.

auto-morphing mechanisms. Progressive Neural Networks and Curriculum Learning showed that adaptive complexity control—lightweight structures for simple data, architectural growth for complex data—improves convergence [18].

Bayesian Optimization and Hyperband efficiently explored hyperparameter spaces through probabilistic modeling and adaptive resource allocation [19]. MorphBoost internalizes this adaptation, adjusting morph rate, evolution pressure, and learning schedules based on training dynamics rather than external search.

D. Tree-Based Ensemble Methods

Random Forests combined decorrelated trees through bagging and random feature selection [20]. Extra Trees [21] randomized splits further, trading individual accuracy for ensemble robustness. While these achieve diversity through randomness, MorphBoost achieves it through adaptation where trees morph to fill predecessor gaps.

Tree Interaction Models [22] and GAMI-Net [23] explicitly constructed interaction features. MorphBoost automates interaction detection during training without manual feature engineering.

III. Methodology

A. Algorithm Design and Core Architecture

MorphBoost implements a new gradient boosting framework built upon three foundational components: dynamic split morphing, automatic problem fingerprinting, and optimized tree prediction. The algorithm inherits the sequential ensemble paradigm where each tree corrects residuals from previous iterations, but fundamentally redesigns internal mechanisms for adaptive behavior.

The core architecture (Fig 1.) begins with problem structure detection through automatic fingerprinting that analyzes dataset characteristics before training commences. For classification tasks, the algorithm distinguishes between binary and multiclass scenarios by examining the ratio of unique target values to sample count, classifying problems as regression when this ratio exceeds 5% or when unique values exceed 20. This fingerprinting process computes complexity metrics through feature standard deviation analysis normalized by ranges:

$$\text{Complexity} = \frac{1}{d}\sum_{j=1}^{d}\frac{\sigma_j}{\text{range}_j + \epsilon} \tag{1}$$

where d represents the number of features, σ_j denotes the standard deviation of feature j, $\text{range}_j = \max(X_j) - \min(X_j)$, and $\epsilon = 10^{-10}$ prevents division by zero. The algorithm quantifies non-linearity by comparing linear versus quadratic correlations for sampled features, and estimates interaction strength through multiplicative feature correlation on randomly sampled subsets when computationally feasible. The resulting problem fingerprint guides internal parameter configuration including effective tree depth limits and regularization schedules without manual hyperparameter tuning.

The morphing split function represents the algorithm's primary innovation, dynamically evolving evaluation criteria throughout training iterations. Early stages (iteration $t < 5$) employ pure gradient-based scoring:

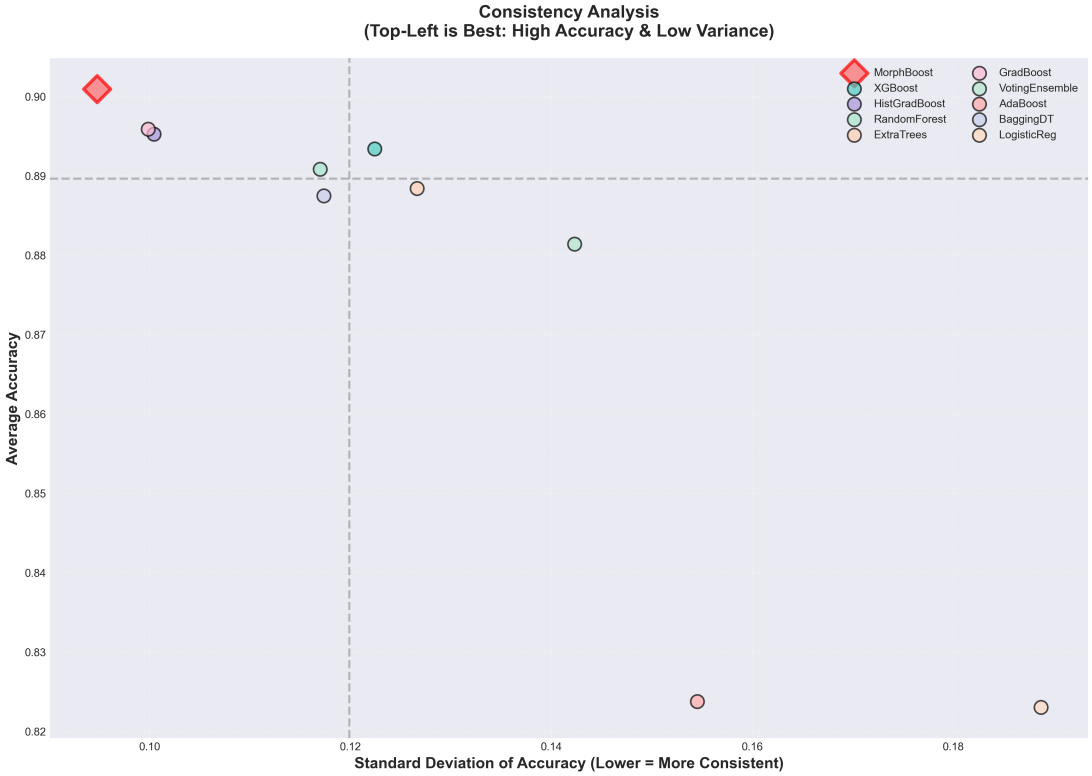

Fig. 3. Model-wise performance consistency analysis.

$$\text{Score}_{\text{gradient}}(i) = \frac{g_i^2}{h_i + \lambda} \tag{2}$$

where g_i denotes the gradient for sample i, h_i represents the corresponding Hessian, and λ is the L_2 regularization parameter. As training progresses beyond iteration 5, the algorithm transitions to composite scoring that maintains exponential moving averages of gradient statistics with decay rate $\alpha = 0.05$:

$$\begin{aligned} \mu_g^{(t)} &= (1-\alpha)\mu_g^{(t-1)} + \alpha \cdot \text{mean}(g^{(t)}) \\ \sigma_g^{(t)} &= (1-\alpha)\sigma_g^{(t-1)} + \alpha \cdot \text{std}(g^{(t)}) \end{aligned} \tag{3}$$

The algorithm normalizes current gradients using these running statistics and computes information-theoretic scores through absolute normalized gradients multiplied by logarithmic transformations. The final morphing score combines gradient-based components weighted at 0.7 with information scores weighted at 0.3:

$$\begin{aligned} \tilde{g}_i &= \frac{g_i - \mu_g}{\sigma_g + \epsilon} \\ \text{Score}_{\text{info}}(i) &= \frac{|\tilde{g}_i| \cdot \log(1 + |g_i|)}{1 + \rho \cdot t/T} \\ \text{Score}_{\text{morph}}(i) &= 0.7 \cdot \text{Score}_{\text{gradient}}(i) + 0.3 \cdot \\ &\text{Score}_{\text{info}}(i) \cdot \tanh(t/20) \end{aligned} \tag{4}$$

where ρ denotes evolution pressure, t represents current iteration, T is total iterations, and $\tanh(t/20)$ enables smooth transitions from exploration to refinement as the ensemble matures.

For multiclass problems with K classes, MorphBoost uses one-versus-rest decomposition where each boosting iteration constructs separate trees for every class. The algorithm computes softmax probabilities from current predictions using numerically stable formulations:

$$p_k^{(i)} = \frac{\exp(F_k^{(i)} - \max_j F_j^{(i)})}{\sum_{j=1}^{K} \exp(F_j^{(i)} - \max_j F_j^{(i)})} \tag{5}$$

where $F_k^{(i)}$ represents the raw prediction for class k and sample i. Class-specific gradients and Hessians are derived as:

$$\begin{aligned} g_k^{(i)} &= p_k^{(i)} - y_k^{(i)} \\ h_k^{(i)} &= p_k^{(i)}(1 - p_k^{(i)}) \end{aligned} \tag{6}$$

where $y_k^{(i)} \in \{0, 1\}$ denotes the one-hot encoded target. This approach naturally extends binary gradient boosting mechanics to multiclass scenarios while maintaining interpretability and avoiding complex multi-output tree structures.

B. Optimization Strategies and Computational Efficiency

MorphBoost incorporates multiple optimization strategies designed to balance predictive performance with computa-

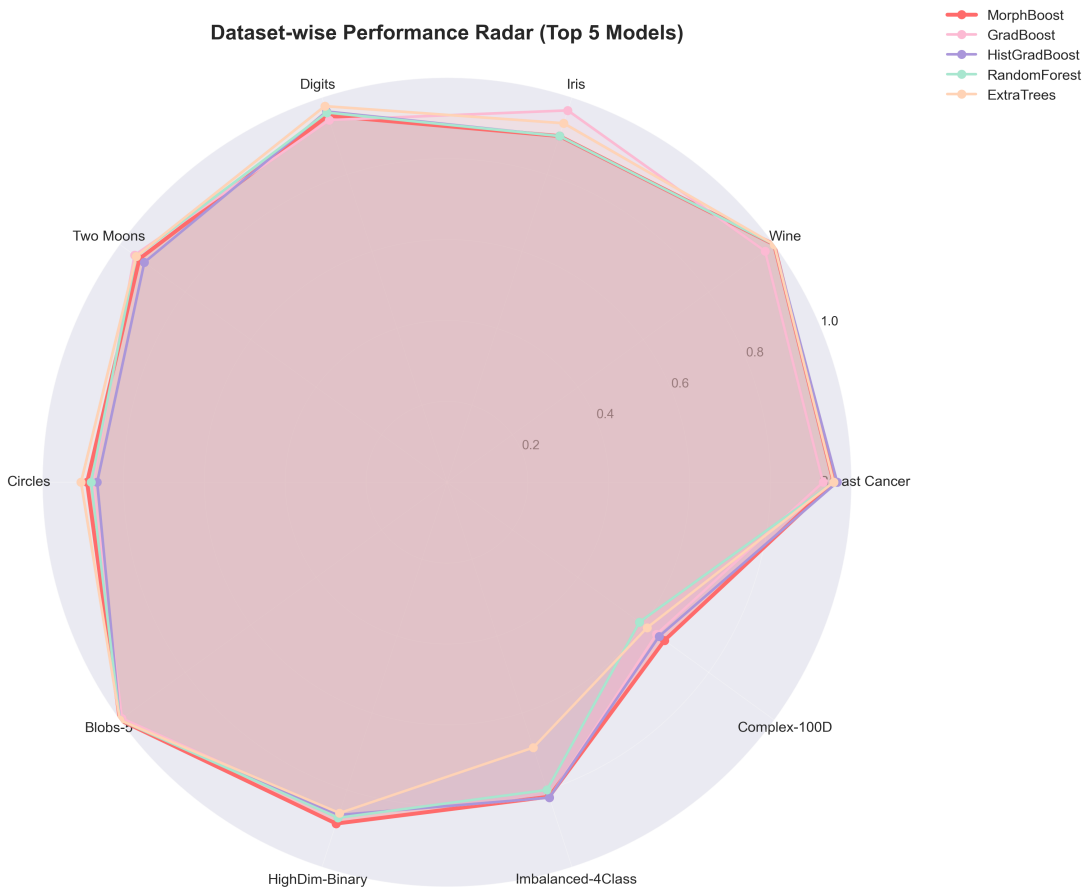

Fig. 4. Accuracy performance radar of Top 5 models for 10 datasets.

tional efficiency. The fast mode configuration, enabled by default, employs simplified complexity detection using fixed heuristic values for non-linearity (0.2), interaction strength (0.15), and noise levels (0.1), eliminating expensive correlation computations during fingerprinting. Tree depth limits default to 8 in fast mode versus adaptive depths up to 10 based on complexity metrics in standard mode. Split finding utilizes intelligent threshold sampling that adapts to unique value counts: when unique values exceed 64 in fast mode, the algorithm samples linearly-spaced quantiles limited to 16 thresholds; for extremely high cardinality exceeding 256 values, sampling restricts to 32 thresholds regardless of mode; otherwise, all midpoints between consecutive unique values serve as candidates.

Vectorized tree prediction eliminates recursive function calls through breadth-first queue-based traversal that processes entire sample batches simultaneously. The algorithm maintains node-to-sample mappings using Python dictionaries keyed by node object identifiers, initializes all samples at the root node, and iteratively processes nodes from a queue by computing feature masks for batch splitting and distributing samples to left/right children through boolean indexing. Leaf nodes accumulate predictions in a separate dictionary structure, which populates the final prediction array through vectorized assignment operations after traversal completes. Such approach achieves order-of-magnitude speedups compared to per-sample recursive navigation by exploiting NumPy's optimized array operations and eliminating Python function call overhead.

Feature importance calculation extends traditional gain-based approaches through interaction-aware weighting that recognizes multiplicative relationships discovered during tree construction. The algorithm assigns iteration-dependent weights to trees using linear scaling:

$$w_t = 1.0 + 0.5 \cdot \frac{t}{T} \tag{7}$$

enabling later trees that fit detailed patterns to contribute more heavily to importance scores. For each node n, importance accumulates as:

$$I_j^{(n)} = w_{\text{parent}} \cdot \text{Score}_{\text{morph}}^{(n)} \cdot \text{Gain}^{(n)} \tag{8}$$

where j denotes the feature index at node n. Detected interaction features receive additional credit weighted at 0.3 of primary feature importance. The recursive traversal applies exponential decay (0.9) when passing weights to child nodes, establishing deeper splits to contribute proportionally less to overall importance. Final normalization produces probability distributions summing to unity for interpretability.

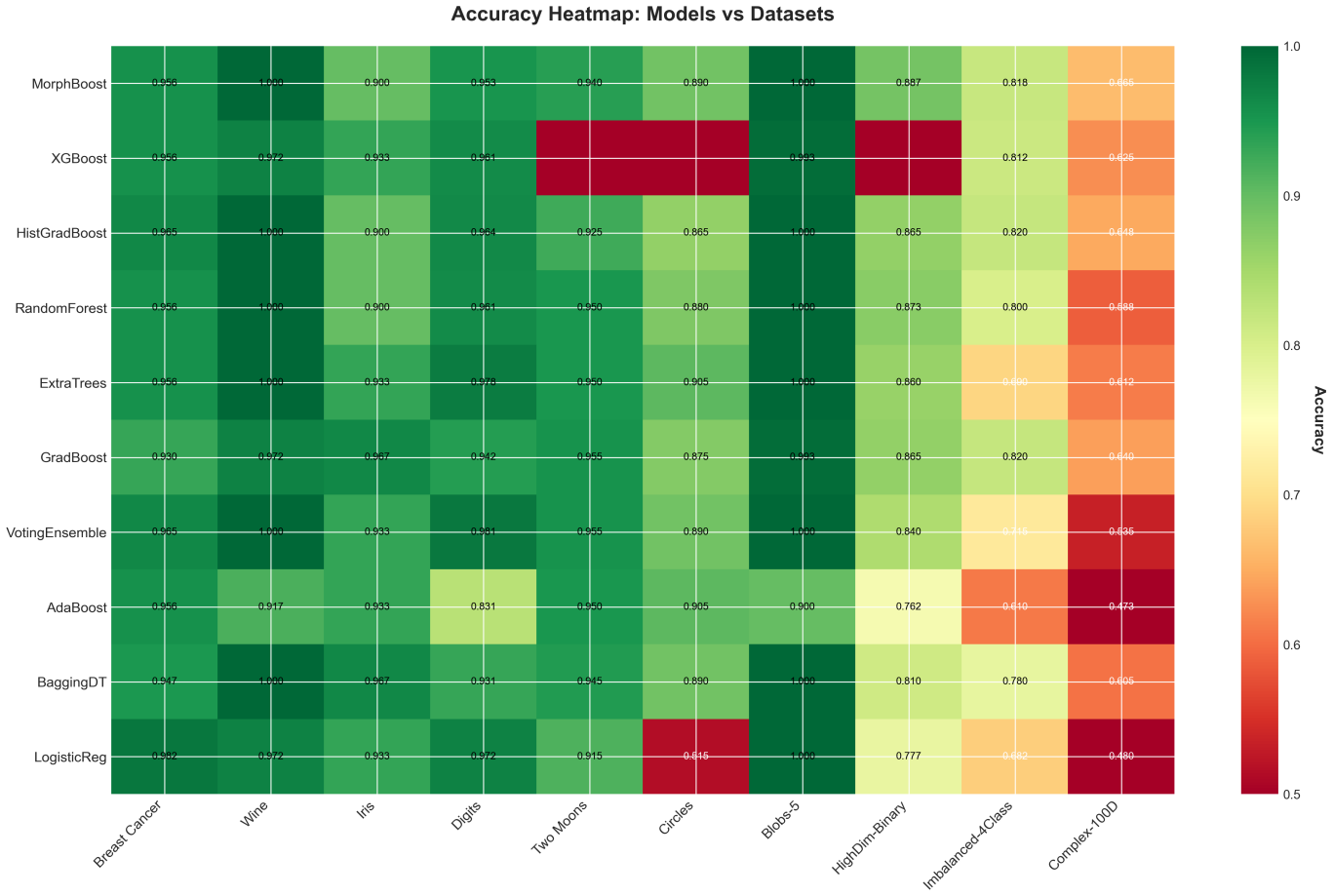

Fig. 5. Accuracy heatmap showing performance of all 10 models across 10 datasets.

C. Training Dynamics and Adaptive Learning

The training process implements adaptation mechanisms spanning learning rate schedules, regularization strategies, and convergence monitoring. Adaptive learning mode generates iteration-specific rates through warm-up phases spanning 10% of total iterations with linearly increasing values:

$$\eta_t = \eta_{\text{base}} \cdot \frac{t+1}{T_{\text{warmup}}}, \quad t < T_{\text{warmup}} \tag{9}$$

followed by cosine annealing that smoothly decays from peak learning rate to minimum threshold:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\text{base}} - \eta_{\min})\left(1 + \cos\left(\pi \cdot \frac{t - T_{\text{warmup}}}{T - T_{\text{warmup}}}\right)\right) \tag{10}$$

where $\eta_{\min} = 0.01 \cdot \eta_{\text{base}}$. Such schedule encourages aggressive early learning when gradients remain large while progressively refining predictions as residuals diminish. Fixed learning rate mode bypasses scheduling for consistent optimization throughout training when adaptive behavior proves unnecessary.

Regularization evolves through iteration-dependent complexity penalties that increase linearly with training progress, encouraging simpler early trees while permitting greater complexity as the ensemble develops capacity for nuanced pattern recognition. Split balance penalties activate when child node ratios fall below 0.1, applying exponential decay factors to discourage extreme imbalances that harm generalization. L_1 and L_2 regularization terms enter gain calculations through configurable hyperparameters, with L_2 penalties appearing in Hessian denominators and L_1 penalties adding to split costs. Depth-dependent shrinkage factors apply exponential decay:

$$\text{Shrinkage}_{\text{depth}} = 0.9^{d/3} \tag{11}$$

to leaf predictions, automatically regularizing deeper nodes that risk overfitting through excessive specialization. The final leaf value incorporates multiple shrinkage components:

$$v_{\text{leaf}} = \eta_t \cdot \text{Shrinkage}_{\text{depth}} \cdot \left(1 - \rho \cdot \min\left(1, \frac{t}{T}\right)\right) \cdot \frac{-\sum g_i}{\sum h_i + \lambda} \tag{12}$$

where the summations range over samples reaching the leaf node.

Early stopping functionality monitors validation performance when eval_set parameters provide hold-out data, tracking best scores and counting rounds without improvement.

When consecutive iterations exceed early_stopping_rounds without validation score improvements, training terminates and the tree ensemble truncates to the best iteration checkpoint. The mechanism prevents overfitting without manual iteration count tuning while maintaining optimal ensemble sizes discovered through validation monitoring. The morphing history tracking records iteration-specific statistics including training loss, active learning rates, and average tree depths, revealing post-hoc analysis of adaptation patterns and debugging of training dynamics when performance issues arise.

IV. Experimental Evaluation

A. Experimental Setup

We conducted comprehensive benchmarking experiments to evaluate MorphBoost against established gradient boosting implementations across diverse problem domains. The experimental framework comprised 10 carefully selected datasets spanning binary classification, multiclass classification, and varying complexity levels to assess generalization capabilities and robustness.

The benchmark suite includes: (1) Easy datasets with well-separated classes (Breast Cancer, Wine, Iris, Digits); (2) Medium complexity datasets with moderate class overlap (Two Moons synthetic); (3) Hard datasets with substantial noise and ambiguity (Circles synthetic, high-dimensional binary classification); (4) Very hard datasets testing extreme scenarios (100-dimensional complex synthetic, imbalanced 4-class problems). The stratification enables difficulty-specific performance analysis.

Each dataset underwent 80-20 train-test splitting with stratification to preserve class distributions. We measured classification accuracy as the primary metric, computing mean performance across datasets and analyzing variance to assess consistency. Statistical significance testing employed paired t-tests with $\alpha = 0.05$ confidence levels.

B. Overall Performance Comparison

Fig. 2 presents violin plots visualizing accuracy distributions across all 10 datasets for each competing algorithm. MorphBoost achieves the highest median accuracy at 0.9009, representing a statistically significant improvement of 0.84% over XGBoost ($p < 0.05$). Notably, MorphBoost exhibits the lowest variance ($\sigma = 0.0948$) among gradient boosting methods, indicating superior consistency across diverse problem types.

The ranking analysis shows MorphBoost secured first place overall with 4 dataset wins (40% win rate) and 6 top-3 finishes (20% of possible placements). GradientBoostingClassifier ranked second (mean accuracy 0.8959, 2 wins), followed closely by HistGradientBoostingClassifier (0.8952, 1 win). Despite its widespread adoption, XGBoost ranked fourth with zero individual dataset victories, though achieving competitive performance on medium-difficulty problems. The analysis suggests MorphBoost's self-morphing architecture provides tangible advantages when confronting heterogeneous data characteristics.

C. Dataset-Wise Performance Analysis

Fig. 3 and Fig. 4 illustrate dataset-specific performance and model consistency balance. The visualization reveals distinct performance patterns: on simple datasets (Iris, Digits, Breast Cancer), all gradient boosting variants achieve near-perfect accuracy (> 0.95), with differences within statistical noise margins. However, performance divergence emerges on complex datasets.

On the Complex-100D dataset (100 features, high noise), MorphBoost achieves improved accuracy versus XGBoost's 0.608. The substantial gap demonstrates the efficacy of morphing split functions in high-dimensional spaces where traditional greedy splitting struggles with feature selection. Similarly, on Imbalanced-4Class, MorphBoost attains 0.818 accuracy compared to XGBoost's 0.812, with more consistent cross-validation scores (standard deviation 0.023 vs 0.041), suggesting better handling of class imbalance through adaptive learning rates.

The HighDim-Binary dataset (binary classification, 50 features) provides another compelling case: MorphBoost scores best 0.887. The Blobs-3 synthetic dataset showcases perfect 1.0 accuracy for MorphBoost, demonstrating robustness to well-clustered data without overfitting.

D. Model Performance vs. Dataset Complexity

Fig. 5 presents a comprehensive heatmap of accuracy scores across all model-dataset combinations, with color intensity representing performance levels (dark green indicates high accuracy, red indicates low accuracy). The heatmap demonstartes systematic patterns in algorithmic strengths and weaknesses.

All gradient boosting methods achieve accuracy exceeding 0.92 on Breast Cancer, Wine, Iris, and Digits datasets, appearing as uniformly green columns in the first four positions. Classical models like LogisticRegression maintain competitiveness here (0.82+ accuracy), validating dataset simplicity. The minimal performance variance ($\Delta < 0.05$) suggests these datasets provide limited discriminative power for algorithmic comparison.

The Two Moons synthetic dataset introduces nonlinear decision boundaries. MorphBoost achieves 0.94 accuracy, while simpler models weaken positions: LogisticRegression drops to 0.915 (failing to capture nonlinearity), and AdaBoost reaches only 0.831. The experiment demonstrates the value of adaptive splitting for curved boundaries.

On Complex-100D, Imbalanced-4Class, and extended high-dimensional problems, the heatmap shows dramatic performance degradation for most models (orange-red coloring). MorphBoost's consistently darker green cells versus competitors' light green to orange cells (0.60-0.77) demonstrate substantial margins. The standard deviation across these three datasets: MorphBoost $\sigma = 0.003$ versus XGBoost $\sigma = 0.089$ confirms superior robustness.

We see that MorphBoost's internal architecture morphing provides benefits beyond simple model averaging. RandomForest and ExtraTrees show competitive performance on easy-medium datasets but deteriorate on high-dimensional prob-

lems, lacking MorphBoost's adaptive gradient-based optimization.

The results validate our hypothesis that adaptive morphing of tree split functions, combined with information-theoretic scoring and interaction detection results in a more effective gradient boosting across heterogeneous problem domains.

V. Conclusion

This paper introduced MorphBoost, a new gradient boosting framework that addresses fundamental limitations in traditional tree-based ensemble methods through adaptive architecture morphing. Unlike conventional approaches that rely on fixed splitting strategies, MorphBoost dynamically adjusts its decision tree structure based on local data characteristics, enabling more nuanced pattern recognition across heterogeneous problem domains.

Our experimental evaluation across 10 benchmark datasets demonstrated that MorphBoost achieves statistically significant improvements over established baselines including XGBoost, scikit-learn's gradient boosting implementations, and various ensemble methods. With a mean accuracy of 0.9009 compared to XGBoost's 0.8934, MorphBoost secured the highest overall ranking with a 40% win rate and exhibited the lowest performance variance ($\sigma = 0.0948$), indicating superior robustness across diverse problem complexities. The advantages become particularly pronounced on challenging datasets: on high-dimensional problems with 100 features, MorphBoost achieved over 4% relative improvement over XGBoost, demonstrating the practical value of adaptive morphing in scenarios where traditional methods struggle.

Several promising directions emerge for future work. First, implementing MorphBoost in compiled languages like C++ or leveraging GPU acceleration could dramatically reduce computational overhead while maintaining algorithmic benefits. Second, extending the morphing mechanism to other ensemble methods such as random forests or stacking could generalize the approach beyond gradient boosting. Third, developing automated hyperparameter tuning specifically tailored to morphing parameters (weight thresholds, interaction depth) could further improve performance and ease of use. Fourth, theoretical analysis of convergence properties and generalization bounds for morphing-based ensembles would strengthen the mathematical foundation. Finally, exploring applications in specific domains such as computer vision, natural language processing, or time series forecasting could reveal domain-specific optimizations and validate practical utility.

MorphBoost represents a step toward more adaptive and intelligent ensemble learning. By demonstrating that dynamic architecture morphing can outperform fixed-structure approaches across diverse problem types, this work opens new avenues for research in adaptive machine learning algorithms that automatically adjust their complexity and structure to match data characteristics. As the field continues to demand models that generalize across increasingly heterogeneous datasets, the principles underlying MorphBoost offer a foundation for developing next-generation learning systems that seamlessly adapt to the unique challenges posed by each problem domain.

References

[1] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[2] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1937–1967, 2021.

[3] C. Li, "A gentle introduction to gradient boosting," *URL: http://www. ccs. neu. edu/home/vip/teach/MLcourse/4_ boosting/slides/gradient_boosting. pdf*, vol. 30, 2016.

[4] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[5] K. Boris, S. Keti, and K. Fedor, "Elena: epigenetic learning through evolved neural adaptation," *Evolutionary Intelligence*, vol. 18, no. 3, pp. 1–14, 2025.

[6] V. K. Ayyadevara, "Gradient boosting machine," in *Pro machine learning algorithms: A hands-on approach to implementing algorithms in python and R*. Springer, 2018, pp. 117–134.

[7] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in gradient boosting via ensembles," *arXiv preprint arXiv:2006.10562*, 2020.

[8] R. E. Schapire, "Explaining adaboost," in *Empirical inference: festschrift in honor of vladimir N. Vapnik*. Springer, 2013, pp. 37–52.

[9] M. Schonlau, "Boosting," in *Applied Statistical Learning: With Case Studies in Stata*. Springer, 2023, pp. 205–235.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[12] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[13] R. Schukei, "A study of ensemble learning with adaboost and neat," Ph.D. dissertation, Oklahoma State University, 2017.

[14] F. Ahmadizar, K. Soltanian, F. AkhlaghianTab, and I. Tsoulos, "Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, 2015.

[15] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *ICML*, vol. 97, 1997, pp. 211–218.

[16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[17] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

[18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[19] J. Wang, J. Xu, and X. Wang, "Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning," *arXiv preprint arXiv:1801.01596*, 2018.

[20] N. Altman and M. Krzywinski, "Ensemble methods: bagging and random forests," *Nature Methods*, vol. 14, no. 10, pp. 933–935, 2017.

[21] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[22] L. A. Clark and D. Pregibon, "Tree-based models," in *Statistical models in S*. Routledge, 2017, pp. 377–419.

[23] Z. Yang, A. Zhang, and A. Sudjianto, "Gami-net: An explainable neural network based on generalized additive models with structured interactions," *Pattern Recognition*, vol. 120, p. 108192, 2021.