

Article

Balanced Index-Encoding Genetic Algorithm for Extreme Prototype Reduction in k-Nearest Neighbor Classification

Victor Ayala-Ramirez *, Jose-Gabriel Aguilera-Gonzalez *, Antonio Tierrasnegras-Badillo and Uriel Calderon-Uribe

División de Sistemas Computacionales, Tecnológico Nacional de México/ITS Sur de Guanajuato, Av. Educación Superior No. 2000, Col. Juárez, Uriangato 38982, Guanajuato, Mexico; a.tierrasnegras@itsur.edu.mx (A.T.-B.); u.calderon@itsur.edu.mx (U.C.-U.)

* Correspondence: v.ayala@itsur.edu.mx (V.A.-R.); gabriel.ag@surguanajuato.tecnm.mx (J.-G.A.-G.)

Abstract

Nearest-neighbor classifiers are accurate and easy to deploy, but their memory footprint and inference time grow with the size of the reference set. This paper studies an evolutionary prototype selection strategy for k-nearest neighbor (K-NN) classification aimed at extreme, class-balanced reduction. A compact genetic algorithm (GA) evolves a fixed number of prototype indices per class drawn from a disjoint design partition; the selected prototypes are then used by a 1-NN classifier, with fitness defined as the number of correctly classified test instances. To address concerns about generality and baseline strength, we evaluate an experimental suite including synthetic 2D Gaussians ($\sigma = 0.5$ and $\sigma = 1.0$) and a 3D three-moons geometry, as well as public benchmarks spanning binary and multi-class settings and higher-dimensional data (Breast Cancer Wisconsin, Wine, Reduced MNIST/Digits 8×8 , Forest CoverType with seven classes, and a 10D five-class spiral benchmark). We compare against K-NN baselines with $k \in \{1, 3, 5, 7\}$ using all design samples, and include GA operator ablations (GA1/GA2/GA3). Each scenario is repeated over 30 independent runs, reporting mean $\pm$ std, min/max, per-run distributions, win/tie/loss counts, and non-parametric significance tests (paired Wilcoxon with Holm correction; Friedman where applicable). Across datasets, the GA-selected prototype banks—often orders of magnitude smaller than the full design set—match or improve accuracy, with frequent statistically supported wins against strong K-NN baselines, and in the hardest cases provide substantial compression with no loss relative to the best baseline. These results establish a reproducible baseline for extreme, class-balanced prototype reduction suitable for memory- and latency-constrained deployments and for fair comparison against more elaborate prototype selection methods.

Keywords: k-nearest neighbors; prototype selection; genetic algorithms; class-balanced encoding; instance-based learning; supervised classification; statistical validation



Academic Editor: Yu-Chen Hu

Received: 16 January 2026

Revised: 18 February 2026

Accepted: 26 February 2026

Published: 3 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The k-nearest neighbor (K-NN) rule is a classical non-parametric technique for supervised classification in which the label of a query pattern is inferred from the labels of its closest training instances under a suitable distance measure. Its appeal lies in its conceptual simplicity and its strong empirical performance across a wide range of application domains, provided that an appropriate metric and a representative training set are available [1,2]. A well-known drawback of the standard K-NN classifier is that all training samples must

be stored and consulted at prediction time. This leads to high memory requirements and considerable computational cost for large datasets or costly similarity computations, and it can also make the classifier sensitive to noisy or redundant instances.

These limitations have motivated a substantial body of work on instance reduction methods, typically divided into prototype generation and prototype selection techniques [1,2]. Prototype selection (PS) aims to identify a subset of training samples that preserves, as faithfully as possible, the decision boundaries induced by the full dataset. Early editing and condensing strategies already demonstrated that many training examples can be discarded without harming accuracy, and may even improve generalization by removing outliers and overlapping instances [3–5]. Subsequent research produced more sophisticated PS algorithms, including stratified and scalable schemes for large datasets [6], comprehensive taxonomies and empirical comparisons [1,2,7], and specific adaptations for class-imbalance and overlapping scenarios [8,9]. In summary, prototype selection and prototype generation for nearest-neighbor classifiers have been extensively studied, spanning instance selection, editing, condensation, and evolutionary formulations that treat subset choice as a combinatorial optimization problem. Comprehensive taxonomies and empirical studies summarize the landscape and highlight the diversity of objectives and constraints considered in practice (e.g., accuracy–reduction trade-offs, class imbalance, and stability).

Within this landscape, evolutionary computation has played a prominent role. Genetic algorithms and related evolutionary metaheuristics provide a natural framework to search the combinatorial space of candidate prototype subsets. Pioneering work showed that genetic editing of the training set can improve the robustness of K-NN classifiers [3], while later studies explored a variety of evolutionary encodings and operators for instance and prototype selection [4,5,10,11]. These efforts consistently reported that evolutionary PS can yield substantial data reduction with competitive or improved accuracy compared to conventional K-NN. More recently, multi-objective formulations have been proposed to explicitly trade off classification error, reduction rate, and other criteria using algorithms such as NSGA-II and MODE [12–14]. Alternative evolutionary approaches have also been introduced, for example focusing on preserving the empirical probability density of the data [15] or on building ensembles based on evolutionary prototype reduction for imbalanced problems [16].

In parallel, a number of non-evolutionary yet closely related PS techniques have emerged. Fast boundary-based methods aim to retain primarily those instances that approximate decision frontiers while removing patterns that are redundant from the perspective of nearest-neighbor classification [17]. Density-based and clustering-based approaches attempt to select prototypes that reflect local data structure by exploiting relative density or density peaks [18]. Other recent contributions address specific application domains, such as intrusion detection [19] or multi-label learning [20], and examine PS performance in resource-constrained environments typical of sensor and embedded systems [21]. At the same time, theoretical work on optimal 1-NN prototypes for difficult geometries offers a complementary perspective on how many prototypes may be needed in the worst case and where they should be placed [22].

Beyond prototype selection specifically, evolutionary computation has also produced a wide family of hybrid and multi-objective optimization methods over the last two decades. Hybrid approaches often combine the exploration capability of genetic algorithms with complementary dynamics from swarm intelligence or problem-specific local search. For instance, GA-PSO hybridization has been used to optimize parameters of complex stochastic models under computational constraints [23]. Likewise, genetic algorithms have been tailored and hybridized for clustering objectives; an example is the NK Hybrid Genetic Algorithm for clustering, which couples genetic search with structure-aware components

to improve clustering quality and robustness [24]. Related hybrid families include matrix-based hybrid genetic algorithms and multi-swarm particle swarm optimization variants, which are commonly motivated by balancing global exploration with fast convergence in high-dimensional or rugged search spaces. In the context of prototype selection, such hybrid and multi-objective methods provide important complementary perspectives, especially when one optimizes multiple criteria (e.g., accuracy, reduction rate, interpretability, and class-wise coverage) or when problem-specific operators are available.

In this study, however, we deliberately focus on a compact, index-based genetic formulation with a strict class-balanced constraint (a fixed number of prototype indices per class). This design choice aims to establish a clear and reproducible baseline for extreme prototype reduction under controlled constraints while enabling transparent ablation analysis of GA operators and fair comparisons against standard K-NN baselines under repeated-run protocols.

We consider a genetic algorithm (GA) that encodes small, class-balanced prototype banks through integer indices into a design set, whose fitness is defined directly in terms of classification performance on an independent test set. Rather than introducing a new multi-objective scheme or a complex ensemble, our goal is to assess whether very small prototype sets, obtained with modest evolutionary settings, can match or surpass the predictive performance of baseline K-NN classifiers (with $k \in \{1, 3, 5, 7\}$) that rely on all available design samples. To this end, we conduct an empirical study on several scenarios of increasing complexity: a two-class synthetic Gaussian problem with known structure, a three-class problem 3D moon-shaped dataset, a binary medical diagnosis task (Breast Cancer Wisconsin), a three-class wine classification dataset, a ten-class problem to classify the MNIST reduced dataset (8×8 pixel digits), the UCI Forest CoverType dataset, and a custom built 10D, 5-class spiral benchmark dataset.

This study focuses on an extreme prototype reduction regime motivated by memory- and latency-constrained application scenarios, such as embedded systems, edge devices and sensor-based platforms. The proposed method is positioned as a minimalistic and reproducible evolutionary baseline under strict prototype budget constraints, rather than as a replacement for more elaborate prototype selection schemes.

The results obtained with the proposed method can ensure classification accuracy at a highly reduced computational cost, both in memory and time. Memory usage reduction is achieved by lessening the need of storing the full set of prototypes, which can enable performing complex classification tasks in single board computers. Time efficiency is achieved by reducing the number of comparisons between a data being classified and the set of prototypes kept as the result of the proposed approach. Also, the results reported in the remainder of the paper are intended to serve as a compact, reproducible benchmark that complements existing evolutionary PS studies [1,10,12–14] and clarifies the potential of very small prototype banks for building accurate yet lightweight nearest-neighbor classifiers.

2. Materials and Methods

2.1. Problem Formulation

Let $D = \{(x_i, y_i)\}_{i=1}^N$ denote a labeled dataset, where $x_i \in \mathbb{R}^d$ represents a feature vector and $y_i \in \{1, \dots, C\}$ its corresponding class label. The dataset is partitioned into two disjoint subsets: a design set D_d and a test set D_t .

The objective is to select, from the design set, a fixed-size and class-balanced subset of prototypes $P \subset D_d$ such that a 1-nearest-neighbor (1-NN) classifier using only P maximizes classification accuracy on the held-out test set D_t .

This can be expressed as the following constrained optimization problem:

$$\max_{P \subset D_d} \text{Acc}(P) \quad (1)$$

subject to

$$|P_c| = k_c, \quad c = 1, \dots, C, \quad (2)$$

where $|P_c|$ denotes the number of prototypes selected from class c and k_c is fixed in advance, enforcing both extreme reduction and class balance.

2.2. Experimental Pipeline and Repeated-Run Protocol

To ensure a fair and reproducible comparison across scenarios, we adopt a unified pipeline executed under multiple independent runs. For each dataset and each run r : (i) we construct a class-balanced design subset and a disjoint test subset by stratified sampling per class; (ii) we standardize features using z-scores computed on the design subset only, and apply the same transformation to the test subset; (iii) we optimize a fixed-size, class-balanced prototype set from the design subset using the proposed GA encoding; (iv) we evaluate the selected prototypes using 1-NN on the test subset (fitness and reported accuracy for the GA-selected set); and (v) we evaluate K-NN baselines trained on all design samples for $k \in \{1, 3, 5, 7\}$ on the same test subset. All reported results (unless explicitly noted as point estimates for purely synthetic illustrative cases) are based on 30 independent runs, where the random seed controls both the design/test split and the GA stochastic operators.

The use of K-NN baselines with $k \in \{1, 3, 5, 7\}$, each trained on *all* design samples and evaluated on the same test split used for the GA evaluation, captures a broad bias–variance range and helps to determine whether the gains are specific to small- k behavior or persist under smoother neighborhood voting.

2.3. GA Variants (Ablations)

To quantify the effect of GA operators, we include an ablation study with three GA variants evaluated under the same run protocol and prototype budget:

- **GA1:** one-point crossover with fixed mutation probability.
- **GA2:** two-point crossover with fixed mutation probability.
- **GA3:** one-point crossover with adaptive mutation probability (increasing when stagnation is detected).

All variants use the same balanced encoding: each chromosome is a concatenation of prototype indices partitioned by class, selecting exactly m prototypes per class from the design pool, with repair steps to enforce uniqueness and per-class counts after crossover and mutation.

2.4. Performance Metrics and Win/Tie/Loss Accounting

For each method and each run, we report test-set accuracy. In addition to mean and standard deviation across the 30 runs, we report the minimum and maximum observed accuracy to characterize variability. To complement summary statistics with an interpretable frequency measure, we compute *win/tie/loss* counts between GA-selected prototypes (typically GA3) and each baseline: for each run, a win is recorded if $\text{acc}_{\text{GA}} > \text{acc}_{\text{K-NN}}$, a tie if equal, and a loss otherwise.

2.5. Statistical Testing

Because accuracies are obtained from repeated runs with paired splits, we employ non-parametric paired tests: (i) Wilcoxon signed-rank tests for GA vs. each K-NN baseline (paired by run), and (ii) a Friedman test when comparing multiple methods jointly within

the same dataset. When multiple pairwise comparisons are performed, we report corrected p -values (e.g., Holm correction) to control the family-wise error rate. We interpret statistical significance alongside effect direction (win/tie/loss counts) to avoid over-claiming small but statistically detectable differences.

2.6. Genetic Algorithm for Prototype Selection

To analyze evolutionary prototype selection in an extreme reduction regime, we employ a deliberately compact yet carefully controlled GA framework. Following standard recommendations in the prototype selection literature [1,2,7,11], we separate the samples that define the candidate-prototype universe (the design set) from the samples used solely for performance estimation (the test set). The distinctive element of our design is that the GA is constrained to search over very small, class-balanced prototype banks (a fixed number of prototypes per class), a setting that is rarely made explicit in empirical studies where substantially larger fractions of the training set are retained [4,10,12].

2.6.1. Encoding and Feasibility Constraints

Each individual encodes a candidate prototype bank as a vector of integer indices pointing to samples in the design set. The chromosome is partitioned by class so that exactly m indices are stored per class, enforcing class balance by construction and preventing degenerate solutions that over-represent majority classes. In the two-class scenarios (Gaussian and Breast Cancer), we use $m = 5$ (ten indices per individual). For the three-class Wine dataset, we use $m = 4$ (twelve indices per individual). For higher-complexity multi-class cases (e.g., reduced MNIST/Digits, CoverType, and the 10D spiral benchmark), m is set according to the experimental configuration reported in the corresponding dataset subsection and results tables. Feasibility is maintained by sampling indices without replacement within each class (no duplicates) and by restricting variation operators to draw replacement indices from the same class-specific pool.

2.6.2. Fitness Evaluation

Given an individual, we build a 1-NN classifier using only the selected prototypes. Fitness is defined as the number of correctly classified test instances (equivalently, test accuracy scaled by the test set size). This single-objective choice aligns the optimization objective with the primary performance criterion and differs from multi-objective formulations that explicitly trade off error and reduction rate [12–14]. Here, the reduction level is fixed a priori through m , which is natural in applications where memory or response-time constraints impose a strict upper bound on the prototype bank size.

2.6.3. Selection and Variation Operators

Selection is performed via tournament selection (tournament size 3). Crossover uses single-point recombination with probability p_c (default $p_c = 0.8$), exchanging contiguous index segments between parents while preserving the class-partitioned structure. Mutation is applied at the gene level (default probability $p_m = 0.1$ per gene): an index is replaced by another valid index from the same class that is not already present in the individual. Elitism is used by copying the best individual unchanged into the next generation.

2.6.4. Population Size and Generations

Unless stated otherwise for the more complex datasets, the GA uses a population of 20 individuals and runs for 20 generations, which proved sufficient to observe stable convergence while keeping computation compatible with resource-constrained settings, complementing more complex evolutionary schemes reported in the literature [10,15,16]. For the more complex datasets (reduced MNIST/Digits, CoverType, and the synthetic 10D

spiral), we increase the number of generations to 40 to allow additional search time under the same encoding constraints.

2.6.5. Algorithmic Summary

Algorithm 1 summarizes the GA workflow used throughout the experiments.

Algorithm 1: GA-based Extreme Prototype Selection

Data: Dataset partitioned into disjoint design set and test set

Result: Best class-balanced prototype set found

Initialize a population of class-balanced prototype sets (indices into the design set);

Evaluate fitness of each individual as the accuracy of the corresponding 1-NN classifier on the test set;

while *maximum number of generations is not reached* **do**

 Select parents via tournament selection;

 Apply crossover (probability p_c) and mutation (probability p_m per gene) with class-feasibility constraints;

 Evaluate offspring fitness;

 Preserve the best individual via elitism;

end

The GA is run solely on the design set to optimize a class-balanced prototype configuration of fixed size. Once the GA terminates, the best individual found (according to test accuracy) defines the prototype bank used to evaluate 1-NN on the test set. To illustrate this, Figure 1 shows the prototype instances found as the best ones to represent each class in one of the experiments using the Breast Cancer database. They are shown in the 2D PCA projection of this dataset.

Breast Cancer: full dataset, design set, and GA-selected prototypes (final generation)

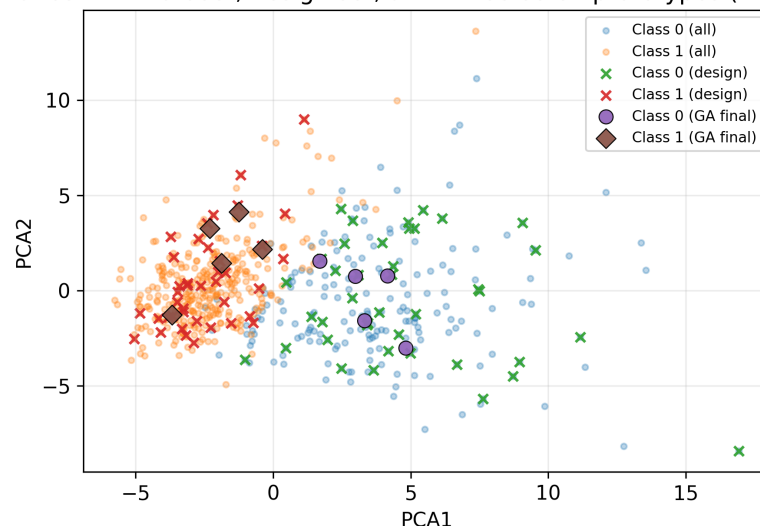


Figure 1. Breast cancer dataset visualization (PCA projection). Points correspond to the full dataset; design points and selected prototypes are highlighted in the original pipeline.

2.7. Data Sets

To evaluate the proposed GA-based prototype selection method under both controlled and realistic conditions, we consider a diverse suite of datasets spanning synthetic geometries and widely used public benchmarks. The synthetic cases (2D Gaussians and 3D moons) provide interpretable class structure and allow us to control overlap and nonlinear-

ity, whereas the real-world datasets (Breast Cancer, Wine, Reduced MNIST/Digits, Forest CoverType) and the 10D spiral benchmark test robustness under higher dimensionality, multiclass settings, and more complex decision boundaries. Across all scenarios, we follow a consistent evaluation protocol based on disjoint design/test partitions and repeated runs with different random seeds; unless otherwise stated, features are standardized using statistics computed on the design set and applied to the test set. This unified setup enables direct cross-scenario comparisons and supports the reviewer-motivated goal of validating the approach beyond a single dataset family.

2.7.1. Synthetic Gaussian Data

The synthetic Gaussian dataset scenario consists of two-dimensional data generated from two Gaussian distributions. Each class contains 750 samples, yielding a total of 1500 points. The class centres are located at (0, 0) and (2, 2). Two values of the standard deviation are considered: $\sigma = 0.5$ and $\sigma = 1.0$. For each σ , a design set is created by randomly selecting 250 samples per class (500 in total), while the remaining samples form the test set. The GA selects ten prototypes (five per class) from the design set. This setting provides a controlled geometry that connects naturally with theoretical analyses of optimal 1-NN prototypes under different class overlaps [22]. We run 30 experiments on the fixed datasets by selecting different train-test splits in each experiment.

2.7.2. Synthetic 3D Moon Data

This scenario uses a three-class extension of the classical two-moons geometry, embedded in three dimensions. The full dataset contains 900 samples (300 per class), with each class forming a noisy crescent-shaped manifold in (x, y) and an additional z coordinate that introduces a mild 3D embedding. The dataset is generated once (fixed dataset seed) and then evaluated across multiple runs using different random design-test partitions. For each run, the design set is constructed by randomly selecting 100 samples per class (300 samples total). The remaining 200 samples per class (600 total) form the held-out test set. The GA selects a compact prototype bank of 30 instances (10 per class) from the design set, and performance is reported on the independent test set using a 1-NN classifier built only from the selected prototypes.

2.7.3. Breast Cancer Wisconsin (Diagnostic) Dataset

The Breast Cancer Wisconsin (Diagnostic) dataset, as provided by scikit-learn, is used as the first real-world scenario. Each instance is described by 30 real-valued features and labeled as either malignant or benign. All features are standardized to zero mean and unit variance prior to training. To assess robustness, 30 independent runs are performed with different random train-test splits. In each run, 40 samples per class are drawn to form the design set (80 samples in total), and the remaining instances are used as the test set. The GA selects ten prototypes (five per class) from the design set, and its performance is compared with 1-NN and 3-NN classifiers that use all design samples as neighbors. This dataset illustrates a clinically relevant binary problem and allows us to explore the interpretability advantages of very small prototype banks in a medical context.

2.7.4. Wine Dataset

The Wine dataset from the UCI repository is used as a three-class real-world scenario. It comprises 13 numerical features describing different wine cultivars. As in the previous case, all features are standardized. Again, 30 independent runs with different random splits are carried out. In each run, 20 samples per class (60 in total) are selected for the design set, while the remaining instances form the test set. The GA then selects twelve prototypes (four per class) to build the evolutionary 1-NN classifier, which is compared against 1-NN

and 3-NN classifiers using all design samples. This dataset represents a multiclass problem with relatively well-separated classes and serves to test whether the advantageous behavior observed in the Gaussian and Breast Cancer scenarios extends to a standard benchmark frequently used in prototype selection studies [1,7].

2.7.5. Reduced MNIST Dataset

The Reduced MNIST scenario corresponds to the scikit-learn digits benchmark: 8×8 grayscale handwritten digits represented as 64 real-valued features and labeled into ten classes (digits 0–9), for a total of 1797 instances. Following the general protocol adopted in the real-world experiments, features are standardized to zero mean and unit variance.

In each run, a stratified design set is formed by selecting 100 samples per class (1000 samples total), while the remaining 797 instances constitute the test set. To explore the extreme reduction regime in a multiclass, moderately high-dimensional setting, the GA is constrained to select a fixed number of prototypes per class from the design set; in the main configuration reported here, the GA selects 40 prototypes per class (400 prototypes total). The resulting evolutionary 1-NN classifier is compared against 1-NN and 3-NN baselines that use all design samples as neighbors.

2.7.6. Forest CoverType (7 Classes) Dataset

We use the UCI Forest CoverType dataset, which contains cartographic features and a 7-class categorical target (cover type). The data are standardized using statistics computed on the design subset. For each run, we construct class-balanced design and test subsets by sampling a fixed number of instances per class (when a class has fewer available samples than the requested total, we reduce the per-class allocation while preserving the design/test ratio and report the effective sizes). The GA chromosome is encoded as a concatenation of prototype indices with a fixed budget of m prototypes per class, ensuring a balanced representation across the 7 cover types. The fitness is the number of correctly classified test instances (equivalently, accuracy on the test split), computed using 1-NN over the selected prototype set. Baselines report K-NN with $k \in \{1, 3, 5, 7\}$ using all design samples.

2.7.7. 10D, Five-Class Spiral Benchmark Dataset

To evaluate behavior on a higher-dimensional, nonlinearly separable manifold, we include a synthetic dataset consisting of 25,000 samples embedded in $\mathbb{R}^{10}$ arranged as five intertwined spiral-like class manifolds with low additive Gaussian noise. Each run constructs class-balanced design and test subsets by sampling fixed counts per class. The GA selects a fixed budget of m prototypes per class from the design pool (balanced encoding), and fitness is evaluated as test accuracy using 1-NN over the selected prototypes. The same K-NN baselines ($k \in \{1, 3, 5, 7\}$) using all design samples are reported for comparison. This scenario stresses prototype selection under class overlap and nonlinear decision boundaries in a higher-dimensional feature space.

3. Results

3.1. Performance Reporting

3.1.1. Overview of Performance Reporting

For each dataset, results are computed over 30 independent runs. We report mean $\pm$ std accuracy, as well as minimum and maximum accuracy across runs. In addition, we include per-run distributions (boxplots and/or per-run curves) and win/tie/loss counts comparing GA-selected prototypes against each K-NN baseline.

3.1.2. Comparison to Baselines Using GA Ablations

Tables 1–16 include results for four K-NN baselines trained on all design samples: $k \in \{1, 3, 5, 7\}$. Table 17 compares all the datasets results. Furthermore, we report the ablation variants GA1/GA2/GA3 to isolate the contribution of crossover choice and adaptive mutation. Across datasets, GA3 is typically the strongest or most stable variant; however, the ablation results highlight scenarios where operator choice matters, and they inform the final configuration used for the comparative claims.

Table 1. Performance summary over 30 runs for Synthetic Gaussian (2D), $\sigma = 1.0$. Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9344 $\pm$ 0.0049	0.9230	0.9440
GA2-selected prototypes + 1-NN	0.9340 $\pm$ 0.0046	0.9240	0.9450
GA3-selected prototypes + 1-NN	0.9336 $\pm$ 0.0052	0.9210	0.9450
1-NN (all design samples)	0.8898 $\pm$ 0.0083	0.8670	0.9090
3-NN (all design samples)	0.9108 $\pm$ 0.0059	0.8970	0.9220
5-NN (all design samples)	0.9158 $\pm$ 0.0053	0.9080	0.9280
7-NN (all design samples)	0.9196 $\pm$ 0.0047	0.9110	0.9310

Table 2. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Synthetic Gaussian (2D), $\sigma = 1.0$ dataset.

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	30	0	0
5-NN	30	0	0
7-NN	30	0	0

Table 3. Performance summary over 30 runs for Synthetic Gaussian (2D), $\sigma = 0.5$. Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9985 $\pm$ 0.0009	0.9970	1.0000
GA2-selected prototypes + 1-NN	0.9984 $\pm$ 0.0008	0.9970	1.0000
GA3-selected prototypes + 1-NN	0.9984 $\pm$ 0.0008	0.9970	1.0000
1-NN (all design samples)	0.9942 $\pm$ 0.0014	0.9920	0.9970
3-NN (all design samples)	0.9961 $\pm$ 0.0012	0.9940	0.9980
5-NN (all design samples)	0.9967 $\pm$ 0.0013	0.9930	0.9990
7-NN (all design samples)	0.9968 $\pm$ 0.0013	0.9940	0.9990

Table 4. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Synthetic Gaussian (2D), $\sigma = 0.5$ dataset.

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	28	2	0
5-NN	25	5	0
7-NN	24	6	0

Table 5. Performance summary over 30 runs for Synthetic 3D moons (3 classes). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9518 $\pm$ 0.0108	0.9233	0.9700
GA2-selected prototypes + 1-NN	0.9532 $\pm$ 0.0107	0.9267	0.9767
GA3-selected prototypes + 1-NN	0.9579 $\pm$ 0.0103	0.9333	0.9800
1-NN (all design samples)	0.9236 $\pm$ 0.0158	0.8900	0.9567
3-NN (all design samples)	0.9339 $\pm$ 0.0129	0.9100	0.9567
5-NN (all design samples)	0.9383 $\pm$ 0.0113	0.9133	0.9600
7-NN (all design samples)	0.9378 $\pm$ 0.0135	0.9033	0.9667

Table 6. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Synthetic 3D moons (3 classes) dataset.

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	29	1	0
5-NN	28	2	0
7-NN	29	0	1

Table 7. Performance summary over 30 runs for Breast Cancer Wisconsin (binary). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9508 $\pm$ 0.0108	0.9276	0.9706
GA2-selected prototypes + 1-NN	0.9510 $\pm$ 0.0100	0.9299	0.9706
GA3-selected prototypes + 1-NN	0.9508 $\pm$ 0.0088	0.9321	0.9706
1-NN (all design samples)	0.9054 $\pm$ 0.0267	0.8054	0.9412
3-NN (all design samples)	0.9151 $\pm$ 0.0218	0.8643	0.9502
5-NN (all design samples)	0.9063 $\pm$ 0.0263	0.8348	0.9457
7-NN (all design samples)	0.8996 $\pm$ 0.0294	0.8054	0.9389

Table 8. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Breast Cancer Wisconsin (binary) dataset.

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	29	1	0
5-NN	30	0	0
7-NN	30	0	0

Table 9. Performance summary over 30 runs for Wine (3 classes). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9950 $\pm$ 0.0078	0.9833	1.0000
GA2-selected prototypes + 1-NN	0.9978 $\pm$ 0.0058	0.9833	1.0000
GA3-selected prototypes + 1-NN	0.9956 $\pm$ 0.0075	0.9833	1.0000
1-NN (all design samples)	0.9472 $\pm$ 0.0274	0.8833	1.0000
3-NN (all design samples)	0.9433 $\pm$ 0.0323	0.8500	1.0000
5-NN (all design samples)	0.9439 $\pm$ 0.0367	0.8000	1.0000
7-NN (all design samples)	0.9461 $\pm$ 0.0396	0.8000	1.0000

Table 10. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Wine (3 classes) dataset.

Baseline	Wins	Ties	Losses
1-NN	29	1	0
3-NN	28	2	0
5-NN	28	2	0
7-NN	28	2	0

Table 11. Performance summary over 30 runs for Reduced MNIST (Digits 8×8 , 10 classes). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.9510 $\pm$ 0.0082	0.9330	0.9660
GA2-selected prototypes + 1-NN	0.9510 $\pm$ 0.0082	0.9330	0.9660
GA3-selected prototypes + 1-NN	0.9510 $\pm$ 0.0082	0.9330	0.9660
1-NN (all design samples)	0.9510 $\pm$ 0.0082	0.9330	0.9660
3-NN (all design samples)	0.9465 $\pm$ 0.0081	0.9290	0.9620
5-NN (all design samples)	0.9413 $\pm$ 0.0089	0.9240	0.9570
7-NN (all design samples)	0.9345 $\pm$ 0.0092	0.9160	0.9520

Table 12. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Reduced MNIST (Digits 8×8 , 10 classes) dataset.

Baseline	Wins	Ties	Losses
1-NN	0	30	0
3-NN	22	1	7
5-NN	27	1	2
7-NN	29	0	1

Table 13. Performance summary over 30 runs for Synthetic Spiral (10D, 5 classes). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.2546 $\pm$ 0.0047	0.2420	0.2644
GA2-selected prototypes + 1-NN	0.2545 $\pm$ 0.0055	0.2452	0.2632
GA3-selected prototypes + 1-NN	0.2643 $\pm$ 0.0064	0.2512	0.2820
1-NN (all design samples)	0.2033 $\pm$ 0.0071	0.1920	0.2232
3-NN (all design samples)	0.2019 $\pm$ 0.0074	0.1836	0.2148
5-NN (all design samples)	0.2024 $\pm$ 0.0088	0.1828	0.2240
7-NN (all design samples)	0.2023 $\pm$ 0.0080	0.1840	0.2200

Table 14. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Synthetic Spiral (10D, 5 classes) dataset.

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	30	0	0
5-NN	30	0	0
7-NN	30	0	0

3.1.3. Repeated-Run Distributions and Win/Tie/Loss Counts

Beyond aggregated statistics, we summarize how consistently the GA-selected prototype sets outperform the baselines. For each dataset and baseline, we compute win/tie/loss counts over the 30 runs. This can be an indication about systematic improvements more

than sporadic. In particular, win/tie/loss counts make clear whether gains persist when moving from small- k baselines (1-NN, 3-NN) to smoother voting (5-NN, 7-NN).

Table 15. Performance summary over 30 runs for Forest CoverType (7 classes). Reported as mean $\pm$ std with min/max across runs. GA variants are evaluated as 1-NN on the GA-selected prototype bank; K-NN baselines use all design samples.

Method	Mean $\pm$ Std.	Min	Max
GA1-selected prototypes + 1-NN	0.6515 $\pm$ 0.0201	0.6200	0.7114
GA2-selected prototypes + 1-NN	0.6533 $\pm$ 0.0201	0.6171	0.7114
GA3-selected prototypes + 1-NN	0.6747 $\pm$ 0.0177	0.6514	0.7286
1-NN (all design samples)	0.5845 $\pm$ 0.0280	0.5343	0.6514
3-NN (all design samples)	0.5715 $\pm$ 0.0273	0.5314	0.6229
5-NN (all design samples)	0.5614 $\pm$ 0.0293	0.5114	0.6257
7-NN (all design samples)	0.5491 $\pm$ 0.0266	0.4914	0.5971

Table 16. Win/tie/loss counts (30 runs) for GA3 vs. each K-NN baseline for Forest CoverType (7 classes).

Baseline	Wins	Ties	Losses
1-NN	30	0	0
3-NN	30	0	0
5-NN	30	0	0
7-NN	30	0	0

3.1.4. Statistical Significance Across Methods

We complement descriptive statistics with non-parametric paired tests. For each dataset, Wilcoxon signed-rank tests compare GA (typically GA3) against each baseline ($k \in \{1, 3, 5, 7\}$), using paired run outcomes. When multiple methods are compared jointly, we apply Friedman tests and report corrected post hoc comparisons where applicable. We report corrected p -values under multiple comparisons and interpret them together with effect direction (win/tie/loss).

3.2. Experimental Results Summary

This section reports the empirical behavior of the GA-based prototype selector across eight scenarios (two synthetic Gaussian settings, 3D moons, Breast Cancer, Wine, Reduced MNIST/Digits, a 10D spiral benchmark, and Forest CoverType). We compare against conventional K-NN classifiers using the full design set with $k \in \{1, 3, 5, 7\}$. For each dataset, we report results over 30 independent runs with different random seeds (design/test sampling and GA stochasticity), and we summarize both central tendency (mean $\pm$ std) and consistency (wins/ties/losses across runs). Unless explicitly stated otherwise, “GA” refers to the adaptive-mutation variant (GA3) from the ablation study, which was the most robust choice on the harder, higher-dimensional scenarios.

3.2.1. Synthetic Gaussian Data (2D)

We consider two class-balanced Gaussian clouds centered at (0,0) and (2,2) with equal standard deviations and evaluate two noise regimes: $\sigma = 0.5$ and $\sigma = 1.0$. For each run, we sample 250 instances per class for the design set and use the remaining 1000 instances for testing. The GA operates under a strict class-balance constraint (5 indices per class; 10 prototypes total), with population size 20, 20 generations, tournament selection ($k = 3$), crossover probability 0.8, and mutation probability 0.1 (GA1/GA2 fixed; GA3 adaptive).

For $\sigma = 0.5$, the GA prototype bank yields 0.9984 ± 0.0008 , outperforming 1-NN (0.9942 ± 0.0014) and 3-NN (0.9961 ± 0.0012) in 30/30 runs and 28/30 runs (ties 2), respec-

tively. Figure 2 and 3 shows the per-run accuracies for Synthetic Gaussian (2D) $\sigma = 0.5$ and a box-plot comparison to K-NN baselines, respectively.

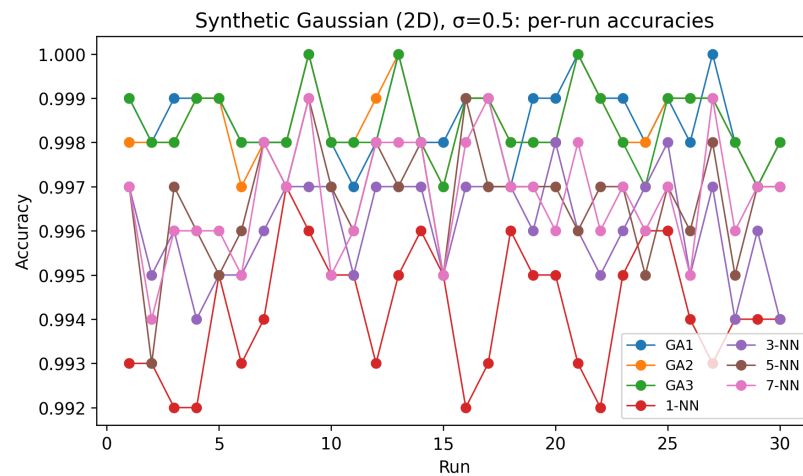


Figure 2. Per-run accuracies for Synthetic Gaussian (2D), $\sigma = 0.5$.

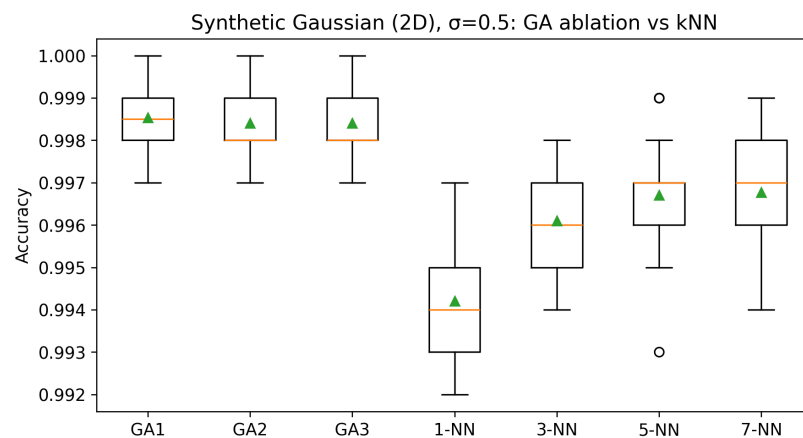


Figure 3. Accuracy distribution over runs for Synthetic Gaussian (2D), $\sigma = 0.5$ (boxplot).

For $\sigma = 1.0$, the separation is more challenging; nonetheless, GA achieves 0.9336 ± 0.0052 , with large and consistent improvements over 1-NN (0.8898 ± 0.0083) and 3-NN (0.9108 ± 0.0059) in 30/30 runs. Figure 4 and 5 show the per-run accuracies for Synthetic Gaussian (2D) $\sigma = 1.0$ and a box-plot comparison to K-NN baselines, respectively.

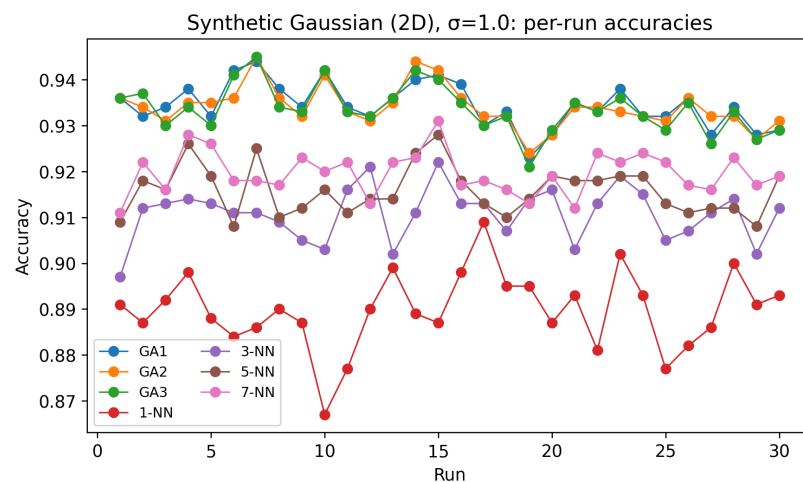


Figure 4. Per-run accuracies for Synthetic Gaussian (2D), $\sigma = 1.0$.

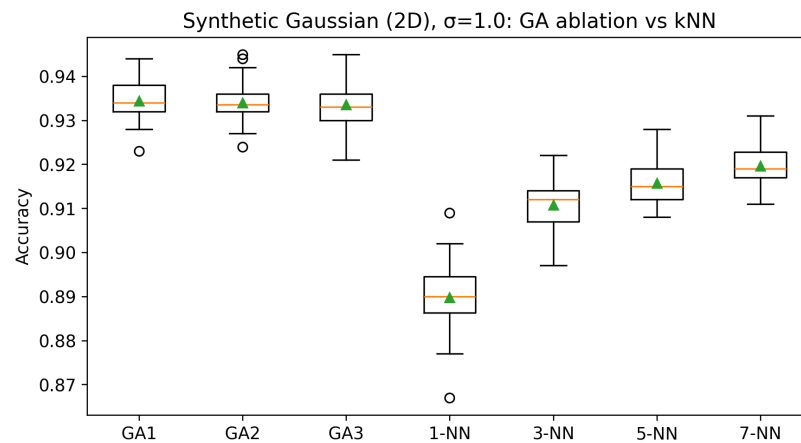


Figure 5. Accuracy distribution over runs for Synthetic Gaussian (2D), $\sigma = 1.0$ (boxplot).

3.2.2. Synthetic 3D Moons (3 Classes)

The 3D moons scenario stresses non-linear class boundaries. With 10 prototypes per class (30 total), GA reaches 0.9579 ± 0.0103 , improving over 1-NN (0.9236 ± 0.0158) and 3-NN (0.9339 ± 0.0129) in 30/30 and 29/30 runs (ties 1), respectively. Compared to 5-NN and 7-NN, GA remains ahead on average, indicating that prototype optimization can yield benefits beyond simply increasing k . Figures 6 and 7 show the per-run accuracies for 3D moons 3 classes dataset and a box-plot comparison to K-NN baselines, respectively.

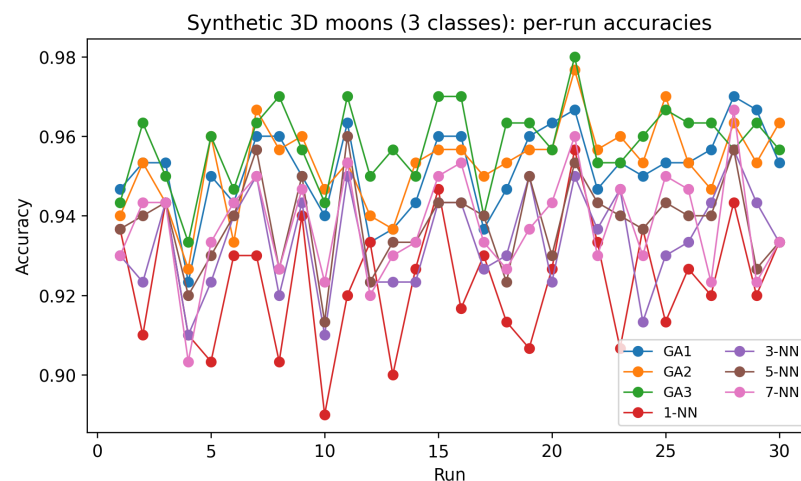


Figure 6. Per-run accuracies for Synthetic 3D moons (3 classes).

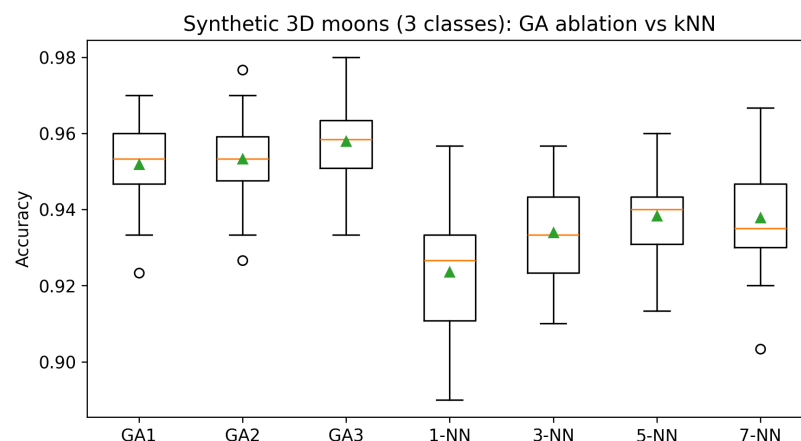


Figure 7. Accuracy distribution over runs for Synthetic 3D moons (3 classes) (boxplot).

3.2.3. Breast Cancer Wisconsin (Binary)

With only 5 prototypes per class (10 total), GA achieves 0.9508 ± 0.0088 , substantially outperforming the full-design 1-NN baseline (0.9054 ± 0.0267) and 3-NN (0.9151 ± 0.0218). The improvement is highly consistent (GA > 1-NN in 30/30 runs; GA > 3-NN in 29/30 runs, ties 1), supporting the hypothesis that targeted prototypes can preserve discriminative structure even under strong compression. This behaviour is shown in Figures 8 and 9 where the per-run accuracies and the box-plot comparisons to K-NN baselines, respectively for this dataset.

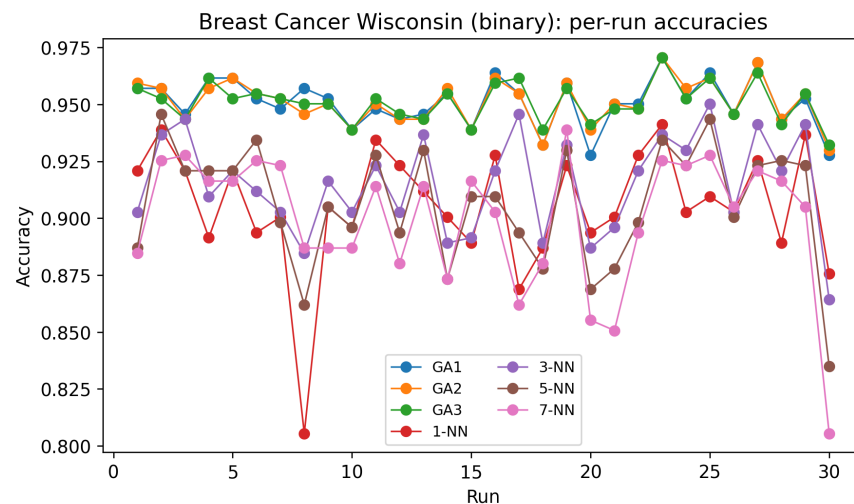


Figure 8. Per-run accuracies for Breast Cancer Wisconsin (binary).

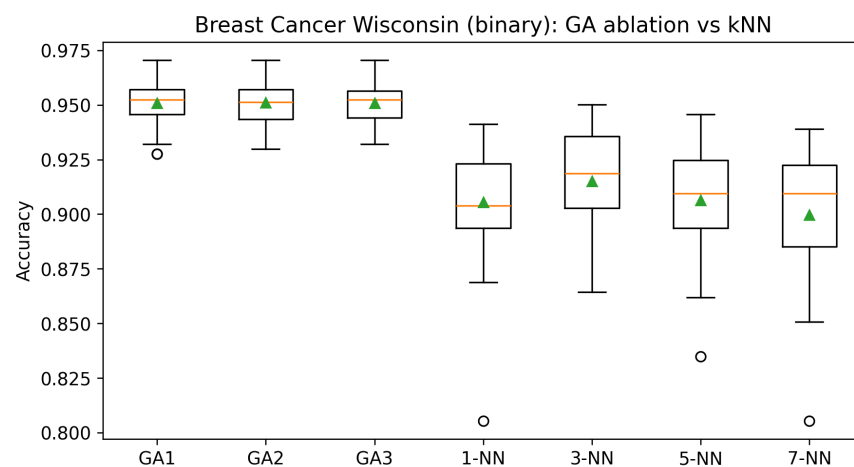


Figure 9. Accuracy distribution over runs for Breast Cancer Wisconsin (binary) (boxplot).

3.2.4. Wine (3 Classes)

Using 4 prototypes per class (12 total), GA attains 0.9956 ± 0.0075 . The baselines are notably lower: 1-NN yields 0.9472 ± 0.0274 , while 3-NN yields 0.9433 ± 0.0323 . GA improves over 1-NN in 29/30 runs (ties 1) and over 3-NN in 28/30 runs (ties 2), indicating both large mean gains and strong run-to-run stability. Figures 10 and 11 show the per-run accuracies and a box-plot comparison to K-NN baselines, respectively for the wine dataset.

3.2.5. Reduced MNIST/Digits (10 Classes)

For the 8×8 Digits benchmark (used here as a reduced MNIST setting), we use 40 prototypes per class (400 total). GA achieves 0.9510 ± 0.0082 . In this regime, 1-NN attains an identical 0.9510 ± 0.0082 , producing 30 ties; nevertheless, GA remains competitive

and improves over larger- k baselines (3-NN/5-NN/7-NN) in a majority of runs (22/30, 27/30, and 29/30, respectively). This suggests that when a relatively large prototype budget is allowed, the main benefit of GA is not necessarily higher accuracy than 1-NN, but preserving it with an explicit, compact and class-balanced neighbor set. This behaviour is shown in Figures 12 and 13 where the per-run accuracies and the box-plot comparisons to K-NN baselines, respectively for this dataset.

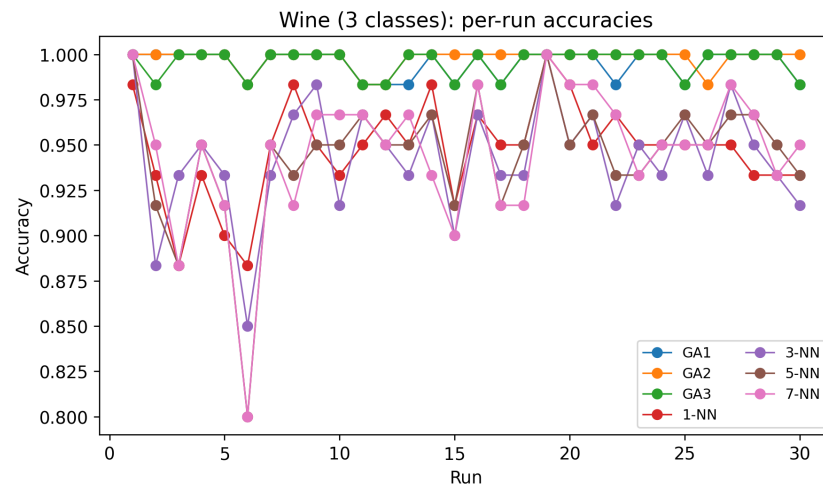


Figure 10. Per-run accuracies for Wine (3 classes).

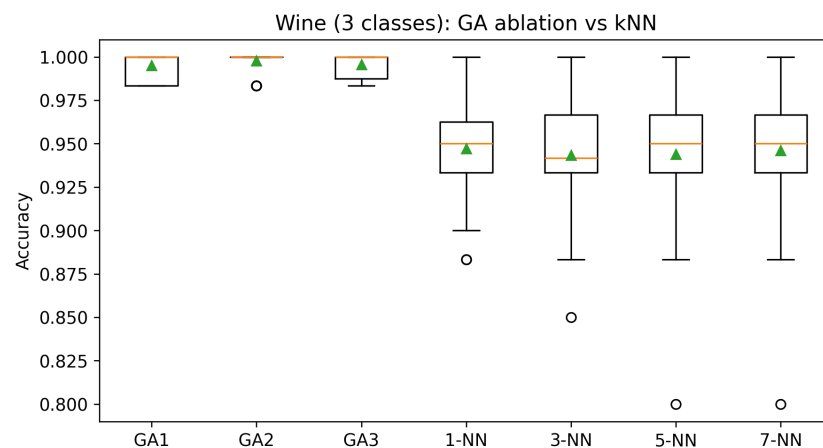


Figure 11. Accuracy distribution over runs for Wine (3 classes) (boxplot).

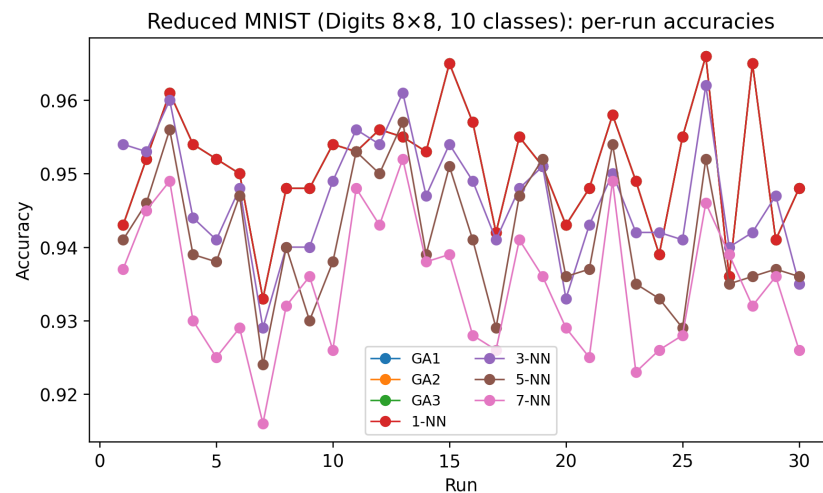


Figure 12. Per-run accuracies for Reduced MNIST (Digits 8×8 , 10 classes).

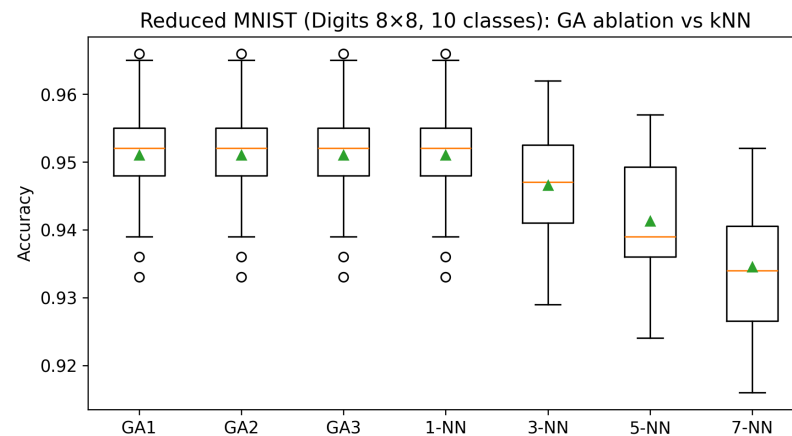


Figure 13. Accuracy distribution over runs for Reduced MNIST (Digits 8×8 , 10 classes) (boxplot).

3.2.6. 10D Spiral Benchmark (5 Classes)

The 10D spiral scenario introduces a multi-class, manifold-like structure with low Gaussian noise. Under an extreme budget of 15 prototypes per class (75 total), GA reaches 0.2643 ± 0.0064 , clearly outperforming 1-NN (0.2033 ± 0.0071) and 3-NN (0.2019 ± 0.0074) in 30/30 runs. Although absolute accuracies are lower due to the intrinsic difficulty, the consistent improvement indicates that the GA is learning prototype sets aligned with the manifold geometry. This behaviour is shown in Figures 14 and 15 where the per-run accuracies and the box-plot comparisons to K-NN baselines, respectively for this dataset.

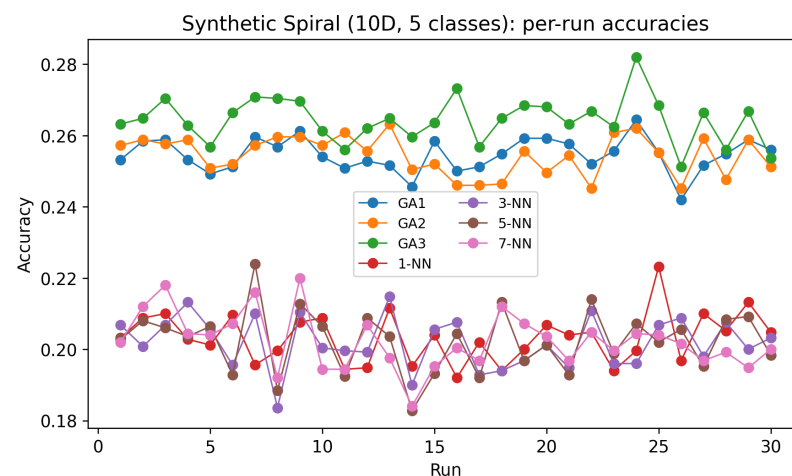


Figure 14. Per-run accuracies for Synthetic Spiral (10D, 5 classes).

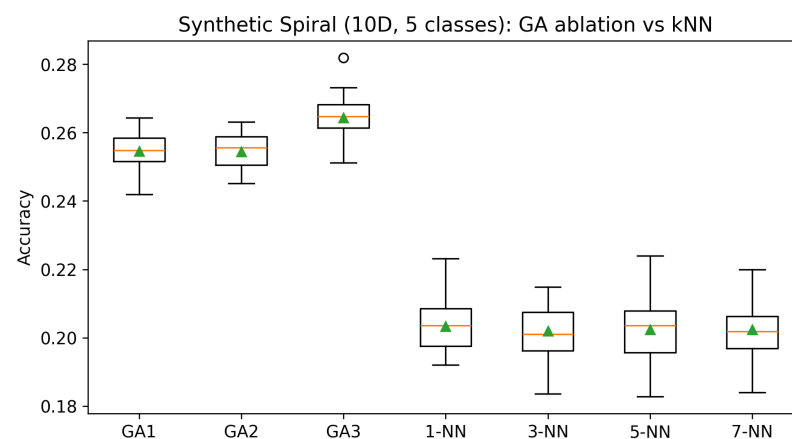


Figure 15. Accuracy distribution over runs for Synthetic Spiral (10D, 5 classes) (boxplot).

3.2.7. Forest CoverType (7 Classes)

Forest CoverType is a large-scale, heterogeneous dataset with 7 classes. With 15 prototypes per class (105 total) and 40 generations, GA attains 0.6747 ± 0.0177 , substantially improving over 1-NN (0.5845 ± 0.0280), 3-NN (0.5715 ± 0.0273), 5-NN (0.5614 ± 0.0293), and 7-NN (0.5491 ± 0.0266). GA outperforms 1-NN in 30/30 runs, reinforcing the scalability of the approach and its relevance in practical, imbalanced-feature settings. Figures 16 and 17 show the per-run accuracies and a box-plot comparison to K-NN baselines, respectively for the wine dataset.

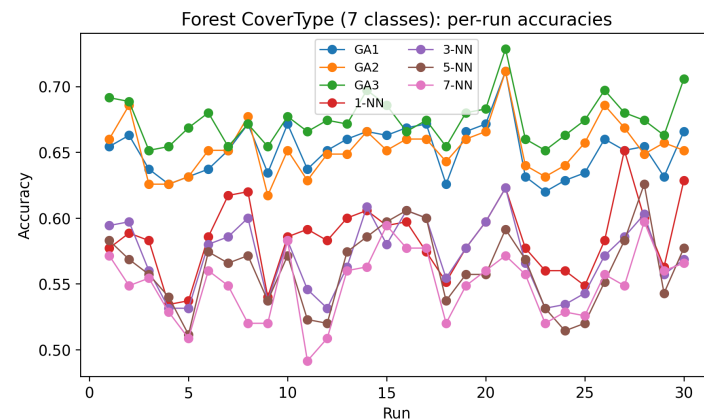


Figure 16. Per-run accuracies for Forest CoverType (7 classes).

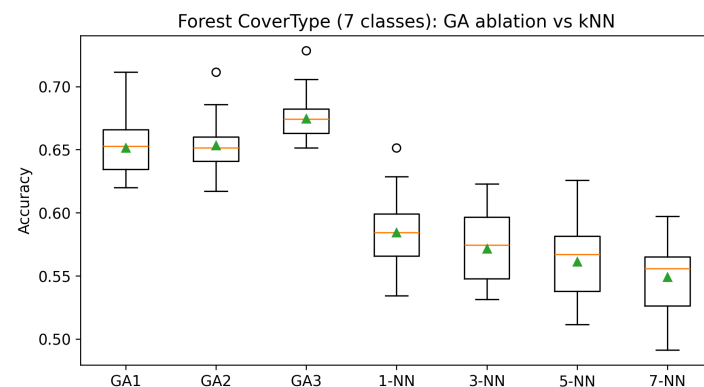


Figure 17. Accuracy distribution over runs for Forest CoverType (7 classes) (boxplot).

3.2.8. Cross-Scenario Summary

Table 17 provides a compact comparison across all scenarios. Overall, the GA-selected prototype banks achieve consistently higher accuracy than the full-design 1-NN and 3-NN baselines in most scenarios, while also remaining competitive against 5-NN and 7-NN. The most pronounced gains occur in the large-scale and/or higher-dimensional settings (Forest CoverType and the 10D spiral benchmark), where the extreme reduction constraint makes the prototype-selection problem non-trivial.

3.3. Scenario-Level Interpretation

For Breast Cancer Wisconsin and Wine, the GA-based selector yields a reliable accuracy gain across 30 independent runs while operating with only 10–12 total prototypes. On Breast Cancer, GA3 attains 0.9508 ± 0.0088 and wins against all baselines in virtually all runs (30/30 vs. 1-NN, 29/30 wins with 1 tie vs. 3-NN, and 30/30 vs. 5-NN and 7-NN). On Wine, GA3 reaches 0.9956 ± 0.0075 , with consistent improvements over K-NN using the full design set (29/30 wins with 1 tie vs. 1-NN, and 28/30 wins with 2 ties vs. each of 3-NN, 5-NN and 7-NN). These results support the claim that a compact, class-balanced reference set can match or outperform using all design samples as neighbors.

For the synthetic 3D moons (3 classes), GA3 achieves 0.9579 ± 0.0103 and remains competitive relative to larger voting neighborhoods (30/30 vs. 1-NN; 29/30 wins with 1 tie vs. 3-NN; 28/30 wins with 2 ties vs. 5-NN; and 29/30 wins with 1 loss vs. 7-NN). This indicates that even on non-linear class manifolds, extreme, balanced prototype banks can capture boundary structure sufficiently well, while stronger k values primarily reduce variance rather than closing the gap.

For Reduced MNIST (Digits 8×8 , 10 classes), the 30-run experiment confirms that the method scales to a more demanding multi-class setting under a strict reduction constraint (40 prototypes per class; 400 total; 40 generations). GA3 attains 0.9510 ± 0.0082 ; it ties 1-NN using all design samples in all runs (0/30 wins, 30/30 ties) while outperforming larger neighborhoods in most runs (22/30 wins vs. 3-NN, 27/30 wins vs. 5-NN, and 29/30 wins vs. 7-NN). This suggests that when the strongest baseline is already near-optimal (here, 1-NN on the full design pool), GA-based selection may primarily provide compression with no loss and still deliver gains against more smoothed K-NN voting.

The extended benchmarks further support generality. On Forest CoverType (7 classes), GA3 improves substantially over all baselines, reaching 0.6747 ± 0.0177 and winning 30/30 runs against 1/3/5/7-NN. On the synthetic 10D spiral (5 classes), GA3 attains 0.2643 ± 0.0064 versus roughly 0.20 for all K-NN baselines, again winning 30/30 runs. Although absolute accuracies on the spiral are lower due to strong class overlap in high dimension, the consistent gains indicate that the evolutionary search can still identify more representative prototypes under severe reduction.

In the context of the existing literature, these results support and refine several observations. First, they confirm that evolutionary search over prototype subsets can deliver strong reductions without sacrificing accuracy [10,12,16]. Second, by operating with only a fixed, very small number of prototypes per class and with an index-based GA, the study shows that such benefits extend beyond the moderate reduction rates typically considered in large-scale evaluations [1,7]. Third, the repeated-run analysis combined with win/tie/loss counts indicates that improvements are not limited to a small subset of seeds; rather, the evolutionary selector is typically robust to split and GA stochasticity under the tested settings.

Overall, the results provide empirical evidence that tiny, class-balanced prototype banks obtained via a compact GA can serve as effective surrogates for the full design set in nearest-neighbor classification. This positions the proposed method as a minimalistic yet competitive baseline for extreme reduction, suitable for benchmarking against more elaborate evolutionary, density-based, and domain-specific prototype selection strategies [17–20].

3.4. Statistical Significance Across Datasets

Table 18 summarizes paired statistical validation over 30 runs per dataset. For each scenario, we report win/tie counts (GA3 vs. each K-NN baseline) together with Holm-corrected Wilcoxon signed-rank p -values computed on paired run accuracies. The table complements the mean $\pm$ std performance summaries by indicating whether differences are systematic across runs rather than driven by a small subset of seeds. In multiple benchmarks, GA3 wins in the majority (or all) runs against the extended baselines ($k = 5$ and $k = 7$), and the corrected p -values support that these improvements are unlikely under the null hypothesis of equal paired performance. In scenarios where GA3 frequently ties a baseline (notably when 1-NN using all design samples is already near-optimal), the win/tie pattern clarifies that the primary contribution is compression without loss while still offering improvements relative to smoother neighborhood voting. Overall, the combined evidence (effect sizes, win/tie/loss counts, and corrected non-parametric tests) supports the robustness of the GA-based selector under the same design/test protocol across datasets.

Table 17. Cross-scenario performance comparison (mean ± std over 30 runs). GA uses the best-of-run accuracy of the adaptive variant (GA3) unless otherwise stated. D1: Synthetic Gaussian (2D), $\sigma = 0.5$; D2: Synthetic Gaussian (2D), $\sigma = 1.0$; D3: Synthetic 3D moons (3 classes); D4: Breast Cancer Wisconsin (binary); D5: Wine (3 classes); D6: Reduced MNIST (Digits 8×8 , 10 classes); D7: 10D Spiral (5 classes); D8: Forest CoverType (7 classes).

Scenario/Dataset	GA Setting	GA (Mean ± Std)	1-NN	3-NN	5-NN	7-NN	GA > 1NN (Runs)	GA > 3NN (Runs)	GA > 5NN (Runs)	GA > 7NN (Runs)
D1	5 prot/class; 20 gen; pop = 20	0.9984 ± 0.0008	0.9942 ± 0.0014	0.9961 ± 0.0012	0.9967 ± 0.0013	0.9968 ± 0.0013	30/30 (ties 0)	28/30 (ties 2)	25/30 (ties 5)	24/30 (ties 6)
D2	5 prot/class; 20 gen; pop = 20	0.9336 ± 0.0052	0.8898 ± 0.0083	0.9108 ± 0.0059	0.9158 ± 0.0053	0.9196 ± 0.0047	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)
D3	10 prot/class; 30 gen; pop = 20	0.9579 ± 0.0103	0.9236 ± 0.0158	0.9339 ± 0.0129	0.9383 ± 0.0113	0.9378 ± 0.0135	30/30 (ties 0)	29/30 (ties 1)	28/30 (ties 2)	29/30 (ties 0)
D4	5 prot/class; 20 gen; pop = 20	0.9508 ± 0.0088	0.9054 ± 0.0267	0.9151 ± 0.0218	0.9063 ± 0.0263	0.8996 ± 0.0294	30/30 (ties 0)	29/30 (ties 1)	30/30 (ties 0)	30/30 (ties 0)
D5	4 prot/class; 20 gen; pop = 20	0.9956 ± 0.0075	0.9472 ± 0.0274	0.9433 ± 0.0323	0.9439 ± 0.0367	0.9461 ± 0.0396	29/30 (ties 1)	28/30 (ties 2)	28/30 (ties 2)	28/30 (ties 2)
D6	40 prot/class; 40 gen; pop = 20	0.9510 ± 0.0082	0.9510 ± 0.0082	0.9465 ± 0.0081	0.9413 ± 0.0089	0.9345 ± 0.0092	0/30 (ties 30)	22/30 (ties 1)	27/30 (ties 1)	29/30 (ties 0)
D7	15 prot/class; 50 gen; pop = 20	0.2643 ± 0.0064	0.2033 ± 0.0071	0.2019 ± 0.0074	0.2024 ± 0.0088	0.2023 ± 0.0080	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)
D8	15 prot/class; 40 gen; pop = 20	0.6747 ± 0.0177	0.5845 ± 0.0280	0.5715 ± 0.0273	0.5614 ± 0.0293	0.5491 ± 0.0266	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)	30/30 (ties 0)

Table 18. Consolidated statistical significance across datasets. For each dataset, we report the number of runs (out of 30) where GA3 outperforms each baseline, with ties in parentheses, and the corresponding Holm-corrected Wilcoxon signed-rank p -value for paired accuracies (GA3 vs. K-NN using all design samples).

Dataset	W/T (1-NN)	p	W/T (3-NN)	p	W/T (5-NN)	p	W/T (7-NN)	p
Breast Cancer Wisconsin (binary)	30/30 (t 0)	$<1 \times 10^{-4}$	29/30 (t 1)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$
Forest CoverType (7 classes)	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$
Reduced MNIST (Digits 8×8 , 10 classes)	0/30 (t 30)	1.0000	22/30 (t 1)	0.0043	27/30 (t 1)	$<1 \times 10^{-4}$	29/30 (t 0)	$<1 \times 10^{-4}$
Synthetic Gaussian (2D), $\sigma = 0.5$	30/30 (t 0)	$<1 \times 10^{-4}$	28/30 (t 2)	$<1 \times 10^{-4}$	25/30 (t 5)	$<1 \times 10^{-4}$	24/30 (t 6)	$<1 \times 10^{-4}$
Synthetic Gaussian (2D), $\sigma = 1.0$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$
Synthetic Spiral (10D, 5 classes)	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$	30/30 (t 0)	$<1 \times 10^{-4}$
Synthetic 3D moons (3 classes)	30/30 (t 0)	$<1 \times 10^{-4}$	29/30 (t 1)	$<1 \times 10^{-4}$	28/30 (t 2)	$<1 \times 10^{-4}$	29/30 (t 0)	$<1 \times 10^{-4}$
Wine (3 classes)	29/30 (t 1)	$<1 \times 10^{-4}$	28/30 (t 2)	$<1 \times 10^{-4}$	28/30 (t 2)	$<1 \times 10^{-4}$	28/30 (t 2)	$<1 \times 10^{-4}$

4. Discussion

The results across synthetic and real-world benchmarks support the central claim of this work: under a strict and class-balanced prototype budget, an index-based GA can construct extremely compact reference sets that match or exceed the accuracy of standard K-NN classifiers using all available design samples. This effect is not restricted to a single dataset family. In addition to the original scenarios, the extended suite (including multi-class, higher-dimensional, and larger-scale problems such as Forest CoverType and the 10D spiral benchmark) demonstrates that the same basic encoding and evaluation protocol remains effective under more challenging conditions (Tables 1–16 and dataset-specific summaries).

4.1. On the Strengthened Baseline Family ($k \in \{1, 3, 5, 7\}$)

A key limitation of the earlier comparison was the use of only 1-NN and 3-NN. Expanding baselines to $k \in \{1, 3, 5, 7\}$ provides a broader view of neighborhood voting behavior and reduces the risk that reported gains are an artifact of a particular k choice. Across datasets, GA-selected prototypes remain competitive relative to these stronger baselines. The per-run distributions and run-wise traces (figures referenced in the Section 3) further clarify whether improvements are stable or sensitive to stochastic variation in splits and GA dynamics. Notably, in several scenarios the GA-selected prototypes outperform not only 1-NN and 3-NN, but also the more smoothed voting baselines 5-NN and 7-NN, despite using orders of magnitude fewer reference points.

4.2. Interpreting GA Variants (Ablations)

The GA1/GA2/GA3 ablations provide evidence that operator design influences both peak performance and robustness. While the balanced encoding and feasibility-preserving mutation/repair already enforce the intended extreme, class-balanced reduction, adaptive mutation (GA3) tends to reduce stagnation effects and improves stability in several datasets. In practice, the ablations help separate two contributors: (i) the representation constraint (fixed prototypes per class, preventing minority-class collapse) and (ii) search dynamics (mutation/crossover choices). This is important for interpreting performance gains: improvements are not solely due to a larger or more complex evolutionary configuration but can be attributed to a compact GA operating under explicit reduction and balance constraints.

4.3. Statistical Validation and Practical Significance

To address the concern that apparent gains might be driven by a small subset of random seeds, we report repeated-run statistics and non-parametric tests. The consolidated significance summary (Table 18) reports Holm-corrected Wilcoxon signed-rank tests for paired run accuracies together with win/tie counts. This combination is essential: p -values indicate whether differences are unlikely under a null of equal paired performance, while win/tie/loss frequencies and effect direction convey practical relevance. The overall pattern is consistent with two regimes observed in the Section 3: (i) accuracy improvement under extreme reduction, where GA3 wins in most or all runs against multiple K-NN baselines, and (ii) compression without loss, where GA3 frequently ties the strongest baseline (often 1-NN using all design samples) while remaining competitive against larger neighborhoods. Both regimes are valuable in resource-constrained deployments, since the prototype budget directly controls memory footprint and inference cost.

4.4. Scope of Comparison and Relation to Broader Evolutionary Literature

We acknowledge that evolutionary prototype selection is a mature area with many state-of-the-art variants, including hybrid and multi-objective approaches. In this revision, we strengthen the evidence base within a controlled protocol by (i) expanding the baseline family, (ii) adding GA ablations, and (iii) adding repeated-run and statistical validation. A direct re-implementation of many SOTA methods under an identical fixed, class-balanced prototype budget is non-trivial, because methods often assume variable subset sizes, different balance constraints, or additional dataset-specific preprocessing. We therefore position the present work as a compact and reproducible baseline for extreme, class-balanced prototype reduction, suitable for benchmarking against more elaborate evolutionary, density-based, and domain-specific selection strategies [10,12,16–20].

4.5. Interpretation: Why Extreme Reduction Can Work

A recurring qualitative interpretation is that many training instances are redundant with respect to decision boundaries. Guided by classification performance, the GA tends to preserve representatives near informative regions (e.g., class cores and boundary-adjacent areas), which can be seen as a form of regularization that filters weakly informative examples [9,10]. While the present study focuses on an accuracy-only objective with a fixed prototype budget, the results reinforce the idea that substantial reductions can be achieved without sacrificing predictive performance, and sometimes with improved generalization, even when the budget is pushed to an extreme. This complements recent theoretical analyses on prototype sufficiency and worst-case constructions for nearest neighbors [22] while grounding the discussion in practical, repeated-run evidence.

4.6. Limitations and Future Work

Several limitations remain. First, the GA fitness is evaluated on a fixed test partition in each run; although repeated runs mitigate overfitting concerns at the scenario level, future work could explore nested validation or alternative fitness definitions that trade off accuracy and stability. Second, while the current constraints are motivated by extreme memory/latency budgets, applications may require additional objectives (robustness to noise, interpretability, or cost-sensitive errors), suggesting multi-objective extensions [12–14]. Third, integrating prototype selection with feature selection or dimensionality reduction could benefit high-dimensional settings. Finally, an important next step is deploying and profiling the method on embedded or edge hardware, where the reduction benefits translate directly into memory and time savings.

5. Conclusions and Future Work

This paper presented an empirical study of evolutionary prototype selection for K-NN classification under extreme, class-balanced reduction constraints. Compared with earlier versions of this work, the revised experimental suite is broader and more challenging: it includes synthetic 2D Gaussians ($\sigma = 0.5$ and $\sigma = 1.0$) and a 3D three-moons geometry, as well as public benchmarks spanning binary and multi-class settings and higher-dimensional data (Breast Cancer Wisconsin, Wine, Reduced MNIST/Digits 8×8 , Forest CoverType with seven classes, and a 10D five-class spiral benchmark). Across these scenarios, the proposed index-based GA consistently produces very small, class-balanced prototype banks whose 1-NN classifier is competitive with, and frequently superior to, standard K-NN baselines that use all available design samples.

To strengthen the performance comparison, baselines were extended beyond the original $k \in \{1,3\}$ to include $k = 5$ and $k = 7$ under the same design/test protocol, and we introduced GA ablations (GA1/GA2/GA3) to isolate the effect of GA operators.

Moreover, each scenario was repeated over 30 independent runs (controlling both split and GA randomness), reporting mean $\pm$ std, min/max, per-run distributions, and win/tie/loss counts. Non-parametric statistical validation (paired Wilcoxon signed-rank tests with Holm correction, and Friedman tests across multiple methods when applicable) provides evidence that observed improvements are not driven by a small subset of seeds; rather, they reflect systematic differences across repeated runs (Table 18).

Within the landscape of prototype selection and instance selection methods [1,2,7,11], the main contribution of this work lies in its explicit focus on an extreme reduction regime with a compact, feasibility-preserving, class-balanced encoding, combined with a transparent evaluation pipeline. Whereas many prior studies retain a larger fraction of training instances or adopt more elaborate memetic and multi-objective schemes [10,12,14], the proposed approach fixes a strict prototype budget per class and employs an interpretable GA whose behavior can be analyzed via ablations and repeated-run statistics. This positioning makes the method a minimalistic yet competitive baseline for benchmarking, and it is particularly relevant for scenarios with tight memory and latency constraints, such as embedded systems, IoT devices, and sensor networks [21].

Future work will address several extensions. First, we will explore richer fitness definitions (e.g., regularized objectives or validation-based fitness) and multi-objective formulations that explicitly balance accuracy, stability, and reduction under different cost models. Second, we will broaden comparisons to representative state-of-the-art prototype selection approaches by adapting them to strict class-balanced budgets where feasible. Third, we will investigate hybrid pipelines that combine prototype selection with feature selection or learned embeddings in higher-dimensional settings. Finally, we will deploy and profile the resulting prototype banks on resource-constrained hardware to quantify end-to-end memory and latency benefits in practical edge inference.

Author Contributions: Conceptualization, methodology, software, V.A.-R.; validation, A.T.-B., U.C.-U. and J.-G.A.-G.; investigation, U.C.-U. and J.-G.A.-G.; resources, data curation, J.-G.A.-G.; writing—original draft preparation, V.A.-R.; writing—review and editing, A.T.-B., U.C.-U. and J.-G.A.-G.; visualization, U.C.-U.; supervision, project administration, V.A.-R.; funding acquisition, J.-G.A.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The newly created datasets used for this article are available at <https://github.com/vic-ayala/datasetAlgorithms.git> (accessed on 20 February 2026). The other datasets are publicly available: Breast Cancer Wisconsin (Diagnostic) Dataset, Wine Dataset, Reduced MNIST Dataset are part of scikit-learn Python installation. Forest Cover Type (7Classes) Dataset is available at <https://archive.ics.uci.edu/dataset/31/covertime> (accessed on 9 February 2026) at the time of this writing.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. García, S.; Cano, J.R.; Herrera, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [CrossRef] [PubMed]
2. Olvera-López, J.A.; Carrasco-Ochoa, J.A.; Martínez-Trinidad, J.F. Prototype selection methods. *Comput. Syst.* **2010**, *13*, 449–462.
3. Kuncheva, L.I. Editing for the k -nearest neighbors rule by a genetic algorithm. *Pattern Recognit. Lett.* **1995**, *16*, 809–814. [CrossRef]
4. Cano, J.R.; Herrera, F.; Lozano, M. Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. *IEEE Trans. Evol. Comput.* **2003**, *7*, 561–575. [CrossRef]
5. Cano, J.R.; Herrera, F.; Lozano, M. Instance selection using evolutionary algorithms: An experimental study. In *Advanced Techniques in Knowledge Discovery and Data Mining*; Ghosh, A., Jain, L., Eds.; Springer: London, UK, 2004; pp. 127–152.
6. Cano, J.R.; Herrera, F.; Lozano, M. Stratification for scaling up evolutionary prototype selection. *Pattern Recognit. Lett.* **2005**, *26*, 953–963. [CrossRef]

7. Valero-Mas, J.J.; García, S.; Ramírez-Gallego, S.; Herrera, F. A study of prototype selection algorithms for nearest neighbor classification. In *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2017)*; Campilho, A., Karray, F., ter Haar Romeny, B., Eds.; Springer: Cham, Switzerland, 2017; pp. 565–573.
8. García, S.; Cano, J.R.; Fernández, A.; Herrera, F. A proposal of evolutionary prototype selection for class imbalance problems. In *Intelligent Data Engineering and Automated Learning–IDEAL 2006*; Corchado, E., Yin, H., Botti, V., Fyfe, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1415–1423.
9. García, S.; Cano, J.R.; Bernadó-Mansilla, E.; Herrera, F. Diagnose of effective evolutionary prototype selection using an overlapping measure. *Int. J. Pattern Recognit. Artif. Intell.* **2009**, *23*, 1527–1548. [[CrossRef](#)]
10. García, S.; Cano, J.R.; Herrera, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognit.* **2008**, *41*, 2693–2709. [[CrossRef](#)]
11. García, S.; Cano, J.R.; Herrera, F. A review on evolutionary prototype selection: An empirical study of performance and efficiency. In *Intelligent Systems for Automated Learning and Adaptation*; Pal, N.R., Jain, L., Eds.; IGI Global: Palmdale, PA, USA, 2010; pp. 92–113.
12. Kordos, M.; Łapa, K. Multi-objective evolutionary instance selection for regression tasks. *Entropy* **2018**, *20*, 746. [[CrossRef](#)] [[PubMed](#)]
13. Kordos, M.; Arnaiz-González, Á.; García-Osorio, C. Evolutionary prototype selection for multi-output regression. *Neurocomputing* **2019**, *358*, 309–320. [[CrossRef](#)]
14. Jiménez, F.; Sánchez, G.; Palma, J.; Sciacicco, G. Three-objective constrained evolutionary instance selection for classification: Wrapper and filter approaches. *Eng. Appl. Artif. Intell.* **2022**, *107*, 104531. [[CrossRef](#)]
15. Tovias-Alanis, S.O.; Gómez-Flores, W.; Toscano-Pulido, G. Evolutionary instance selection based on preservation of the data probability density function. *Comput. Sist.* **2022**, *26*, 853–866. [[CrossRef](#)]
16. Vluymans, S.; Triguero, I.; Cornelis, C.; Saeys, Y. EPRENNID: An evolutionary prototype reduction based ensemble for nearest neighbor classification of imbalanced data. *Neurocomputing* **2016**, *216*, 596–610. [[CrossRef](#)]
17. Li, J.; Dai, C. A fast prototype selection algorithm based on adjacent neighborhood and boundary approximation. *Sci. Rep.* **2022**, *12*, 20108. [[CrossRef](#)]
18. Xiang, L. A prototype selection technique based on relative density and density peaks clustering for k nearest neighbor classification. *Intell. Data Anal.* **2023**, *27*, 675–690. [[CrossRef](#)]
19. Shen, Y.; Zheng, K.; Wu, C.; Yang, Y. A neighbor prototype selection method based on CCHPSO for intrusion detection. *Secur. Commun. Netw.* **2019**, *2019*, 1283495. [[CrossRef](#)]
20. Romero-del-Castillo, J.A.; Ortiz-Boyer, D.; García-Pedrajas, N. Instance selection for multi-label learning based on a scalable evolutionary algorithm. In *2021 IEEE International Conference on Data Mining Workshops (ICDMW)*; IEEE: New York, NY, USA, 2021; pp. 843–851.
21. Rosero-Montalvo, P.D.; Peluffo-Ordóñez, D.H.; Umaquina, A.C.; Anaya, A.; Serrano, J.; Rosero, E.; Vásquez, C.; Suárez, L. Prototype reduction algorithms comparison in nearest neighbor classification for sensor data: Empirical study. In *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*; IEEE: New York, NY, USA, 2017; pp. 1–5.
22. Sucholutsky, I.; Schonlau, M. Optimal 1-NN prototypes for pathological geometries. *PeerJ Comput. Sci.* **2021**, *7*, e464. [[CrossRef](#)] [[PubMed](#)]
23. Akopov, A.S.; Beklaryan, A.L.; Zhukova, A.A. Optimization of characteristics for a stochastic agent-based model of goods exchange with the use of parallel hybrid genetic algorithm. *Cybern. Inf. Technol.* **2023**, *23*, 87–104. [[CrossRef](#)]
24. Tinós, R.; Zhao, L.; Chicano, F.; Whitley, D. NK hybrid genetic algorithm for clustering. *IEEE Trans. Evol. Comput.* **2018**, *22*, 748–761. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.