Optimizing Proteus: Architectural Enhancements for Dynamic and Streaming Data.

Your Name

September 13, 2025

Abstract

This paper presents architectural and algorithmic augmentations that transform the foundational Proteus framework from a static modeling engine into a complete, high-performance, and adaptive data analysis system. We introduce three key contributions. First, we detail a high-performance query engine that operationalizes the generative model, enabling fast similarity search and complex logical queries. Second, we specify a new representation learning layer using self-scaling Elastic ANFIS Autoencoders, which produce the compact, dense feature vectors required for more advanced analytics. These vectors are consumed by a Recursive Tower of ANFIS-DNF-DNNs capable of automated, human-readable pattern discovery. Third, we introduce architectural enhancements, including Temporal Pyramids, for handling dynamic and streaming data, allowing the system to adapt to concept drift. Together, these augmentations provide a robust, end-to-end framework for real-world, evolving data analysis.

1 Introduction

The foundational Proteus framework, detailed in our prior work, provides a novel method for producing a high-fidelity, generative model of a static dataset. This model accurately captures the underlying data manifold's complex geometry and topology. However, producing the model is not an end in itself. Two critical questions remain: How can this rich model be operationalized for practical, high-performance machine learning tasks? And how can the framework be adapted to handle real-world data, which is rarely static and often arrives in dynamic, evolving streams?

This paper answers these questions by presenting a suite of architectural and algorithmic enhancements that augment the core Proteus engine. We transform it into a complete, queryable, and adaptive system. We detail three major contributions:

1. A high-performance query engine that uses a dual-index architecture to enable both fast geometric search and complex logical queries.

2. A new representation learning layer, built on self-scaling ANFIS autoencoders, that feeds a recursive tower of ANFIS-DNF-DNNs for automated, interpretable pattern discovery.

3. A novel streaming architecture, based on Temporal Pyramids, that allows the system to efficiently adapt to concept drift.

This paper will first briefly recap the foundational framework. We will then detail each of these three contributions in turn, followed by a holistic experimental analysis of the augmented system's capabilities.

2 The Proteus Generative Model: A Recap

For completeness, we briefly summarize the foundational Proteus framework, which is detailed in our companion paper. The framework is a two-stage, recursive pipeline designed to generate a high-fidelity model of a static data manifold.

- **Stage 1 (Scaffolding):** A fast, GNG-based exploratory stage that performs a scale-space analysis to discover the characteristic scales of the data and produce a coarse, piecewise-linear "skeleton" of the manifold.
- **Stage 2 (Refinement):** A high-fidelity stage that takes the scaffold and refines it into a generative, simplicial complex model. This stage's "dual flow" learning rule captures the local probability gradients, enabling generative sampling.

The final output is a hierarchical set of clusters, where each cluster is described by a rich geometric model (the simplicial complex) and a local probability distribution. This paper details how this output is leveraged for analysis.

3 The Proteus Query Engine

To operationalize the generative model, we introduce a query engine built on a dual-index architecture. This engine supports five core data interaction primitives.

3.1 From Model to Features

The engine first derives two key features from the Proteus model:

- **The 1D Key:** For fast similarity search, we use the integrated ρ-SFC framework (detailed in a companion work) to generate a manifold-aware, one-dimensional semantic key for each data point.
- **The Membership Vector:** For complex logical queries, we generate a fuzzy membership vector for each point, describing its partial membership to every cluster in the learned hierarchy. This vector is used to build a high-performance inverted index.

3.2 The Five Core Capabilities

These features enable a powerful, general-purpose foundation for tackling most machine learning tasks:

1. **k-Nearest Neighbor Search:** The 1D key enables extremely fast and accurate similarity search via simple range scans.
2. **Constrained Search:** The membership vector and inverted index allow the system to answer complex logical queries (e.g., "find items in cluster A but not B").
3. **Constrained Generation:** The system can synthesize new data points that conform to a set of fuzzy predicates by finding a valid starting point in the model and then performing an informed random walk using the learned generative model.
4. **Classification (Pattern Discovery):** In a "reverse query," the system can take a set of data points and analyze their common membership vectors to produce a human-readable, fuzzy predicate description of what makes them a coherent group.
5. **Stateful Feature Engineering:** The outputs themselves (the 1D key and membership vector) serve as powerful, semantically rich feature vectors for use in external models.

4 Sparse-to-Dense Representation with Elastic ANFIS Autoencoders

The raw feature set from Proteus—a high-dimensional, sparse fuzzy membership vector—is not optimal for direct use in complex analytical models. To address this, we introduce a foundational representation learning layer: a self-scaling autoencoder that produces compact, dense, and semantically rich latent vectors.

4.1 Architecture: The Elastic ANFIS-Autoencoder

The system uses a modular design where each input stream is handled by its own dedicated, deep ANFIS autoencoder. This allows for independent scaling and adaptation. The encoder itself is not a monolithic network but a deep, hierarchical structure with a tapering series of layers (e.g., $1024 \to 512 \to 256 \to 128$). This is implemented as a single ANFIS model with a hierarchy of shared fuzzy rule bases, where each layer learns concepts from the code vector produced by the layer above it, ensuring interpretability at every stage of compression.

4.2 Self-Scaling Mechanism

The "elastic" nature of the autoencoder comes from its ability to grow its capacity in response to new information without requiring a full retrain.

4.2.1 Growth Trigger and Strategy

The system continuously monitors the reconstruction loss of each autoencoder. A multi-rate bank of Exponential Moving Averages (EMAs) tracks both sharp and gradual increases in error. When the loss exceeds a quality threshold, a meta-learning module called the **Growth Factor Strategist** analyzes past growth events and their impact on performance to strategically determine the dimensionality of a new, secondary autoencoder to be added.

4.2.2 The Three-Stage Growth Cycle

When growth is triggered, the system follows a three-stage process:

1. **Additive Growth on the Residual:** The weights of the existing primary autoencoder (A) are temporarily frozen. A new, secondary autoencoder (B) is then trained *only on the residual error* of the first one ($input - output_A$). This allows the new module to rapidly learn to represent only the information that the original module was failing to capture.
2. **Joint Fine-Tuning:** The weights of both autoencoders are unfrozen, and a focused Annealing Schedule Controller manages a greedy, joint fine-tuning process to optimize the combined model until the reconstruction loss plateaus.
3. **Logical Fusion:** The two autoencoders (A and B) are now treated as a single, unified logical entity. The final code vector is the concatenation of the code from A and the code from B (which was trained on A's residual).

4.3 Periodic Consolidation and Refactoring

This additive growth model is effective but creates computational "debt." To address this, the system incorporates a **Periodic Consolidation** cycle as a background maintenance operation. To avoid introducing non-stationarity that would invalidate downstream models, this process uses a "teacher-student" approach. A new, clean, monolithic autoencoder (C) is trained to match the reconstruction output of the composite (A+B) model. Then, a small, fast **Mapper Network (M)** is trained to translate the code space of C back to the stable code space of A+B ($M(E_C(x)) \approx E_{A+B}(x)$). The system then performs a seamless "hot swap," replacing the slow A+B encoder with the highly efficient C+M chain. This provides a strict improvement in computational performance without sacrificing the semantic stability required by the rest of the system.

5 The Recursive Tower for Semantic Analysis

To move beyond simple queries to automated, human-readable insights, we introduce a Recursive Tower for semantic analysis. This component is a hierarchy of analytical models that mirrors the cluster hierarchy of the underlying Proteus instance, allowing it to discover patterns at multiple scales of abstraction.

5.1 Core Component: ANFIS-DNF-DNN for Pattern Discovery

The core analytical engine at each node of the tower is an ANFIS-DNF-DNN, a model designed for both high performance and interpretability.

- **Function:** The model takes the dense, latent vectors produced by the ANFIS autoencoders as input. It is then trained on a classification or analysis task (e.g., identifying anomalous data points within a given cluster).
- **Interpretability:** While the model is a Deep Neural Network, its ANFIS-based structure allows its learned knowledge to be extracted after training. The internal fuzzy rule base of the trained model can be traversed to generate a set of human-readable IF-THEN rules.
- **Rule Extraction to DNF:** We employ a procedure to convert this learned rule set into a formal logical statement in **Disjunctive Normal Form (DNF)**. The result is a precise, human-readable description of the conditions required for a given classification (e.g., "(feature_$AisHIGHandfeature_CisLOW$)or($featu

This capability allows the system not only to identify patterns, but to explain *why* a pattern was identified, transforming it from a black-box classifier into a transparent analysis tool.

6 Dynamic Enhancements via Temporal Pyramids

To adapt the static Proteus model to streaming data and evolving environments, we introduce an architecture for tracking model state and detecting concept drift over time.

6.1 Architecture: The Temporal Pyramid

The core of the streaming architecture is the **Temporal Pyramid**, a multi-resolution data structure that maintains snapshots of the Proteus manifold's state at different temporal granularities. For example, it might maintain summaries of the data distribution over the last minute, hour, day, and week.

6.2 Process: Drift Detection and Adaptation

Instead of operating on raw data, the Temporal Pyramid aggregates the compact, **dense latent vectors** produced by the ANFIS autoencoders. This provides a statistically stable and low-noise signal.

1. **Aggregation:** At each level of the pyramid, the system maintains a statistical profile (e.g., mean and covariance) of the latent vectors seen during that time window.

2. **Drift Detection:** Concept drift is detected by comparing the statistical profile of a recent time window to that of a more stable, longer-term window. A significant divergence indicates that the underlying data distribution has changed.

3. **Adaptation:** The detection of drift triggers an adaptation response. This can range from a minor, incremental update to the Proteus model for small drifts, to a full re-scaffolding (a targeted re-run of Stage 1) for major shifts in the data distribution.

This architecture allows the system to efficiently monitor for and react to changes in the data stream without requiring a costly full reprocessing at every timestep.

7 Performance Optimizations

To ensure the augmented Proteus system is scalable and suitable for high-performance applications, we incorporate several key optimizations.

7.1 ANN Self-Bootstrapping

Finding the closest model node for each new data point is a critical bottleneck. We use a self-bootstrapping process to accelerate this:

1. **Initial Seeding:** A standard Approximate Nearest-Neighbor (ANN) index (e.g., using HNSW) is built on an initial sample of the data to provide a coarse mapping.

2. **Internal Indexing:** As the Proteus model is constructed, it concurrently builds a high-performance ρ-index on its *own model nodes*.

3. **Index Hot-Swap:** Once the internal ρ-index is mature, it replaces the initial HNSW index as the primary mechanism for data-to-node mapping. The original index is discarded. This creates a feedback loop where the model's own structure is used to accelerate its future learning.

7.2 Parallelization and Training Efficiency

The architecture is designed with parallelization in mind. Key opportunities include:

- Embarrassingly parallel queries against the ensemble of recipe trees in the ρ-index.

- Parallel updates of the "dual flow" mechanism across thousands of simplexes simultaneously. - Efficient batch training and inference for the ANFIS autoencoders and the models in the Recursive Tower.

7.3 Model Compression

We use pruning and compression strategies to manage model size and memory, such as pruning insignificant nodes and links from the Proteus graph and using the "fast path" optimization in the ρ-index to compress uniform regions of space.

8 Experimental Analysis

We conduct a series of experiments to validate the performance and capabilities of the augmented Proteus system. The analysis is designed to test each of the major new architectural components.

8.1 Query Engine Performance

8.2 Representation Quality and Efficiency

8.3 Pattern Discovery Accuracy

8.4 Drift Adaptation

9 Conclusion

This work details a suite of augmentations that transform the foundational Proteus framework from a static modeling engine into a complete, adaptive, and high-performance data analysis system. We have demonstrated how the core generative model can be operationalized via a dual-index query engine. We have introduced a robust and self-scaling representation learning layer using Elastic ANFIS Autoencoders, which in turn feeds a Recursive Tower capable of automated, interpretable pattern discovery. Finally, we have presented an architecture based on Temporal Pyramids that enables the entire system to adapt to dynamic, streaming data.

By combining these contributions, the augmented Proteus framework provides a robust, end-to-end, and scalable solution for extracting insight from complex, real-world, and evolving data. Future work will focus on deploying this system as the core perceptual and reasoning engine within a larger cognitive architecture.

References