### ML 24/25-02 Investigate Input reconstruction by using Classifiers

**Implementation Flow**

1. **Prepare Input:**
   * Load scalar inputs.
   * Encode using ScalarEncoder.
2. **Generate SDR:**
   * Use SpatialPooler to create SDRs.
3. **Reconstruct Input:**
   * Pass SDR to both HTMClassifier and KNNClassifier.
   * Reconstruct input from predictions.
4. **Compare Results:**
   * Use similarity metrics to compare original vs. reconstructed inputs.
   * Plot and analyze results.

**Sprint 1: Planning and Setup**

**Goals:**

* Research the project requirements and tools (NeoCortexAPI, NuPIC Legacy).
* Set up the project structure, development environment, and basic workflows.

**Tasks:**

1. **Research and Documentation:**
   * Review classifiers (HTMClassifier and KNNClassifier) and how they function.
   * Study ScalarEncoder and SpatialPooler implementations in NeoCortexAPI.
   * Understand the IClassifier<TIN, TOUT> interface.
   * Explore similarity functions in NeoCortexAPI.
2. **Setup Project Environment:**
   * Set up the repository, tools, and dependencies (e.g., .NET Core environment).
   * Create the basic structure of the application (input folder, core modules, test framework).
3. **Design Experiment Flow:**
   * Draft the overall workflow for the experiment:
     + Input encoding → SDR generation → Classification → Reconstruction → Evaluation.
4. **Write Initial Documentation:**
   * Prepare the project introduction, objectives, and architecture diagram.

**Sprint 2: Implementation of Input Encoding and SDR Generation**

**Goals:**

* Implement scalar input loading and encoding.
* Generate SDRs using the Spatial Pooler.

**Tasks:**

1. **Scalar Input Loader:**
   * Develop a module to load scalar inputs from a folder.
   * Ensure inputs are stored and ready for encoding.
2. **Encoding with ScalarEncoder:**
   * Implement the ScalarEncoder to convert scalar values to SDRs.
   * Test the encoding process for different inputs (e.g., range, resolution).
3. **SDR Generation with Spatial Pooler:**
   * Integrate the SpatialPooler to produce SDRs.
   * Test the Spatial Pooler output for accuracy and reliability.
4. **Unit Testing:**
   * Create unit tests for ScalarEncoder and Spatial Pooler functionalities.
   * Verify that the generated SDR matches expected results.
5. **Update Documentation:**
   * Document the encoding and SDR generation processes.

**Sprint 3: Implementation of Classifiers and Reconstruction**

**Goals:**

* Use HTMClassifier and KNNClassifier to reconstruct inputs.
* Integrate similarity comparison functions.

**Tasks:**

1. **HTMClassifier Integration:**
   * Implement functionality to use HTMClassifier for learning and prediction.
   * Reconstruct input from SDR using HTMClassifier.
2. **KNNClassifier Integration:**
   * Implement functionality to use KNNClassifier for learning and prediction.
   * Reconstruct input from SDR using KNNClassifier.
3. **Similarity Comparison:**
   * Compare original inputs with reconstructed inputs using similarity functions (e.g., cosine similarity).
   * Generate similarity scores for both classifiers.
4. **Unit Testing:**
   * Write unit tests for both classifiers to ensure accurate predictions.
   * Validate the similarity computation process.
5. **Update Documentation:**
   * Document the classification process and reconstruction method.
   * Include code snippets and diagrams.

**Sprint 4: Evaluation, Visualization, and Final Documentation**

**Goals:**

* Evaluate the performance of classifiers and visualize the results.
* Complete project documentation and prepare the paper.

**Tasks:**

1. **Performance Evaluation:**
   * Analyze the results of input reconstruction for both classifiers.
   * Compare results based on similarity scores, accuracy, and computational efficiency.
2. **Visualization:**
   * Create graphs:
     + Original Input vs. HTM Classifier Output (similarity graph).
     + Original Input vs. KNN Classifier Output (similarity graph).
   * Use a charting library like OxyPlot or Matplotlib for visualizations.
3. **Finalize Documentation:**
   * Complete project documentation, including:
     + Introduction, methodology, implementation details, and evaluation results.
     + Add diagrams and visualizations to enhance clarity.
4. **Write the Paper:**
   * Prepare a formal research paper with the following sections:
     + Abstract
     + Introduction and Problem Statement
     + Methodology
     + Implementation
     + Results and Analysis
     + Conclusion and Future Work
5. **Final Unit Testing:**
   * Conduct end-to-end testing of the complete workflow.
   * Ensure all components integrate seamlessly and perform as expected.
6. **Project Review:**
   * Conduct a final review of code, documentation, and paper.

**Sprint 5: Refinement, Optimization, and Deployment**

**Goals:**

* Optimize the performance of the system, address any identified issues, and prepare the project for deployment.

**Tasks:**

**Performance Optimization:**

* Review the performance bottlenecks in the current implementation (e.g., memory usage, processing speed).
* Optimize the ScalarEncoder, SpatialPooler, and classifier modules for better performance.
* Explore multi-threading or parallel processing where necessary to speed up encoding and classification.

**Error Handling and Edge Cases:**

* Add robust error handling and logging throughout the application.
* Test the system with edge cases and large input datasets to ensure stability.

**Refinement of Classifiers:**

* Fine-tune the HTMClassifier and KNNClassifier (e.g., parameter tuning) for improved accuracy or performance.
* Address any issues found during the evaluation phase (e.g., misclassifications, poor reconstruction accuracy).

**Testing**

1. **Unit Testing:**
   * Write unit tests for both classifiers to ensure accurate predictions.
   * Validate the similarity computation process.
2. **Update Documentation:**
   * Document the classification process and reconstruction method.
   * Include code snippets and diagrams.

**Final Testing:**

* Perform comprehensive tests to verify all optimizations, error handling, and edge cases.
* Ensure the system performs well under load and handles various input types smoothly.

**Update Documentation:**

* Update the documentation to include details of performance optimization, error handling, and deployment steps.

**Project Review:**

* + Conduct a final review of code, documentation, and paper.

**Deliverables for Each Sprint**

1. **Sprint 1 Deliverables:**
   * Project plan, architecture, and documentation outline.
   * Repository and environment setup.
2. **Sprint 2 Deliverables:**
   * Scalar input loader and SDR generation module.
   * Unit tests for ScalarEncoder and Spatial Pooler.
   * Documentation of encoding process.
3. **Sprint 3 Deliverables:**
   * Classification and reconstruction functionality for both classifiers.
   * Unit tests for classifiers and similarity comparisons.
   * Documentation of classification methods.
4. **Sprint 4 Deliverables:**
   * Performance evaluation results and visualizations.
   * Final project documentation.
   * Research paper ready for submission.
   * Comprehensive unit test results.
5. **Sprint 5 Deliverables:**
   * Robust error handling and test results from edge case testing.
   * Optimized code with performance improvements.
   * Deployment-ready application.
   * Updated documentation and deployment guide.

Final validation and testing results.

**Deliverables for Each Sprint**

1. **Sprint 1 Deliverables:**
   * Project plan, architecture, and documentation outline.
   * Repository and environment setup.
2. **Sprint 2 Deliverables:**
   * Scalar input loader and SDR generation module.
   * Unit tests for ScalarEncoder and Spatial Pooler.
   * Documentation of encoding process.
3. **Sprint 3 Deliverables:**
   * Classification and reconstruction functionality for both classifiers.
   * Unit tests for classifiers and similarity comparisons.
   * Documentation of classification methods.
4. **Sprint 4 Deliverables:**
   * Performance evaluation results and visualizations.
   * Final project documentation.
   * Research paper ready for submission.
   * Comprehensive unit test results.