Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

INVESTIGATE IMAGE RECONSTRUCTION BY USING CLASSIFIERS

Anoushka Piplai
anoushka.piplai@stud.fra-uas.de

Avradip Mazumdar
avradip.mazumdar@stud.fra-uas.de

Raka Sarkar
raka.sarkar@stud.fra-uas.de

Somava Ganguly
somava.ganguly@stud.fra-uas.de

Abstract- Input reconstruction is a key concept in machine learning and pattern recognition, involving the recovery of original inputs from transformed or encoded representations. This study explores the use of Sparse Distributed Representations (SDRs) for image reconstruction, utilizing models such as Hierarchical Temporal Memory (HTM) and K-Nearest Neighbors (KNN). By encoding input data and training classifiers on these representations, the approach aims to reconstruct original patterns and assess the quality of reconstruction using similarity metrics. The study provides insights into the potential of SDR-based models for effective input reconstruction in intelligent systems.

Keywords: *Hierarchical Temporal Memory (HTM), K-Nearest Neighbors (KNN), Input Reconstruction, Sparse Distributed Representations (SDRs), Pattern Recognition, Machine Learning*

I. INTRODUCTION

Sparse Distributed Representations (SDRs) are high-dimensional, binary encodings that represent information in a way that is both sparse and distributed, meaning that only a small fraction of bits is active at any time, yet the representation is spread across a wide vector space. This encoding scheme is biologically inspired and has been extensively explored in the context of Hierarchical Temporal Memory (HTM) systems [1]. SDRs offer several advantages for machine learning applications, including robustness to noise, fault tolerance, and the ability to generalize from limited data [5]. These properties make them particularly well-suited for complex tasks such as image recognition, signal processing, and anomaly detection.

HTM is a machine learning model grounded in the structure and functionality of the neocortex, particularly designed to handle temporal and spatial data streams. Unlike many conventional models that require static datasets and batch learning, HTM operates in an online, unsupervised manner, continuously adapting to new inputs [1]. A critical component of HTM is the Spatial Pooler, which transforms raw input into sparse and stable SDRs, effectively capturing the underlying structure of input patterns [3]. Because of this, HTM is particularly adept at learning structured representations and recognizing time-based patterns, which has been demonstrated in studies examining classifier performance and similarity-based evaluation methods [2].
In contrast, the k-Nearest Neighbors (KNN) algorithm is a traditional, instance-based classifier that categorizes input data by comparing it to the most similar examples in the training set. It is a non-parametric method that relies on the majority label among the *k* closest neighbors, determined using a predefined similarity metric [9]. KNN is especially useful for tasks where class boundaries are well defined and is often used as a baseline in machine learning experiments due to its interpretability and simplicity.

The reconstruction of input data from its encoded or classified form plays a critical role in evaluating the performance and fidelity of learning models. This process is especially relevant when working with SDRs or sparse representations, as it provides insight into how much of the original information is retained through the encoding, classification, and decoding pipeline. In visual tasks like image recognition, reconstruction allows researchers to visualize how accurately a model preserves spatial and structural features of an image [7]. High-quality reconstruction not only supports classification performance but also enhances model explainability, helping to identify which features or regions contribute most to the prediction.
To objectively assess reconstruction quality, similarity metrics are employed. Two commonly used metrics in this context are the Jaccard Index and Hamming Distance. The Jaccard Index measures the overlap between two binary sets, computed as the size of the intersection divided by the size of the union. It is particularly well-suited for sparse data, where most bits are inactive, and a few active bits carry the critical information [4]. The Hamming Distance, on the other hand, counts the number of differing bits between two binary vectors, making it a straightforward yet effective way to quantify exact mismatches [7]. These metrics are valuable not only for measuring classifier accuracy but also for analyzing how well encoded patterns maintain their semantic integrity after being processed and reconstructed.

Advanced models leveraging sparse representations, such as those applied in image super-resolution or compressed sensing, have shown that sparse coding and reconstruction can capture high-fidelity features with fewer active components [5]. Moreover, incorporating similarity metrics directly into model evaluation frameworks—as done in recent hybrid models—enhances the ability to fine-tune learning algorithms for better generalization and feature retention [2]. This capability is particularly useful when evaluating model robustness across datasets of varying complexity or noise levels.

As the field of artificial intelligence evolves, interest continues to grow in biologically inspired models like HTM and encoding strategies like SDRs due to their promise in achieving both interpretability and adaptability [1]. Their potential applications span across critical domains, including medical diagnostics, surveillance, and autonomous systems, where accurate pattern recognition and reliable reconstruction are essential [10]. These advancements suggest a shift toward models that do more than just classify— they provide insight into the structure of data and offer robust performance under real-world conditions.

II. Methods

This section describes the methods used to train and evaluate two classifiers, Hierarchical Temporal Memory (HTM) and k-Nearest Neighbors (KNN), for image recognition and reconstruction using Sparse Distributed Representations (SDRs). The image is first binarized then processed into SDRs by Spatial Pooler, and both classifiers are trained using these representations. Subsequently, the system performs predictions and image reconstructions, comparing the results to the original images. The overview is pictorially described in Figure 1.

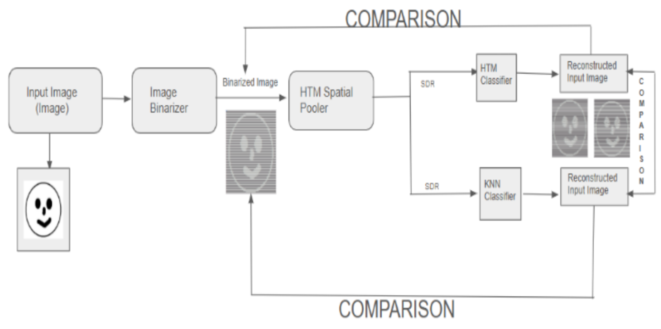


FIGURE 1: PROJECT OVERVIEW

A. Image Binarizer

Function:
Converts raw image data into a binarized format before generating Sparse Distributed Representations (SDRs) using the Spatial Pooler (SP).

Details: The Image Binarizer preprocesses input images by converting them into binary (black-and-white) representations, where pixel values are either 0 or 1 based on a predefined threshold [6]. This step is crucial for ensuring that the images are suitable for SDR generation and further processing by HTM (Hierarchical Temporal Memory) and KNN (K-Nearest Neighbors) classifiers. By simplifying the image structure while preserving key features, the binarization process enhances the ability of classifiers to recognize patterns effectively. The binarized images serve as input to the Spatial Pooler, which then produces the corresponding SDR representations for classification and reconstruction tasks [3].

B. Spatial Pooler (SP)

Function: Encodes the raw input data (image) using the Spatial Pooler (SP) to produce a Sparse Distributed Representation (SDR).

Details: This method activates specific columns based on the input image's pattern, turning on certain neurons (cells) in the pool. These active columns form the Sparse Distributed Representation (SDR), which is used for pattern recognition tasks. The SDR captures the essential features of the image while minimizing redundancy, allowing for more efficient processing. The SP serves as a feature extractor, transforming the input image into a compact and efficient binary representation [3]. The active array is then used to predict the class or label of the image in the subsequent steps as shown in Figure 2, ensuring accurate categorization and enhanced recognition performance.

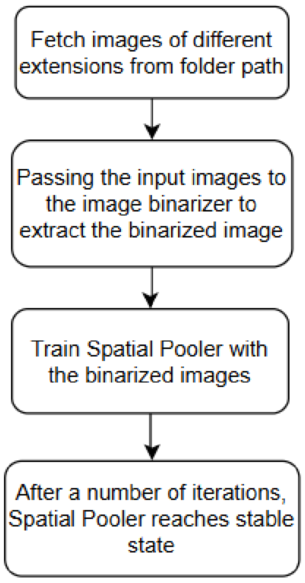

Fetch images of different extensions from folder path
Passing the input images to the image binarizer to extract the binarized image
Train Spatial Pooler with the binarized images
After a number of iterations, Spatial Pooler reaches stable state

FIGURE 2: SPATIAL POOLER LEARNING PHASE

C. SDR (Sparse Distributed Representation)

Function: Trains both the HTM and KNN classifiers using the Sparse Distributed Representations (SDRs) of training images.

Details: This method takes a dictionary of images and their corresponding SDRs as input, where each image is represented by an array of Cell objects (SDR). It then trains both HTM (Hierarchical Temporal Memory) and KNN

(K-Nearest Neighbors) classifiers using the provided SDRs. Each image's SDR is fed to the classifiers, which learn the patterns and features of the images for future predictions. The method logs the training process and tracks the images used for training [2]. Few values of the different HTM parameters are as shown in Table 1.

TABLE 1: VALUES OF HTM PARAMETERS

Parameters	**Values**
CellsPerColumn	10
InputDimensions	new int[]{64, 64}
NumInputs	64*64
ColumnDimensions	new int[] { 64, 64 }
MaxBoost	5.0
DutyCyclePeriod	100
MinPctOverlapDutyCycles	1.0
GlobalInhibition	False

D. CLASSIFIER TRAINING

Function: Trains both the HTM and KNN classifiers using binarized images represented as Sparse Distributed Representations (SDRs).

Details: The system processes training images by converting them into binary format and transforming them into Sparse Distributed Representations (SDRs), which efficiently capture the essential features of the images. The HTM classifier learns by adjusting synaptic permanence to recognize spatial patterns and associations over time, allowing for adaptive learning based on input data, while the KNN classifier stores SDR representations and associates them with labels for nearest-neighbor classification, providing a simple yet effective method for pattern recognition. The system logs the trained image names and records training time for performance evaluation, allowing for an assessment of both the model's accuracy and its efficiency during the training phase [3]. This comprehensive approach ensures that the system can effectively handle and classify large sets of image data.

E. PREDICTION & RECONSTRUCTION

Function: Performs image prediction and reconstruction using both HTM and KNN classifiers, comparing the reconstructed images to the original images.

Details: This method takes the encoded input (SDR) from the Spatial Pooler (SP) and performs prediction using both HTM and KNN classifiers, leveraging their complementary strengths to improve classification accuracy. After obtaining the predicted images, it uses an Image Reconstructor to reconstruct the predicted images and compares them with the original image for similarity, ensuring a detailed and robust evaluation of the model's performance. The reconstruction is evaluated using a similarity metric, which quantifies the closeness between the original and predicted images, and the method generates similarity graphs and plots to visually represent how well the classifiers have reconstructed the image [8]. This whole process is pictorially described in Figure 3, providing a clear visual representation of the entire workflow and its effectiveness in image prediction.

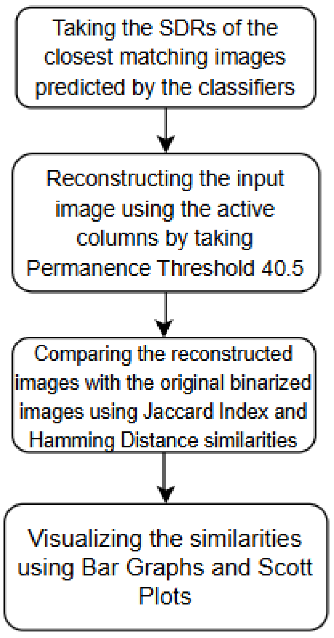

Taking the SDRs of the closest matching images predicted by the classifiers
Reconstructing the input image using the active columns by taking Permanence Threshold 40.5
Comparing the reconstructed images with the original binarized images using Jaccard Index and Hamming Distance similarities
Visualizing the similarities using Bar Graphs and Scott Plots

FIGURE 3: PREDICTION AND RECONSTRUCTION WORKFLOW

F. SIMILARITY CALCULATION OF SDRS

Function: The system calculates the similarity between reconstructed images and their original versions using Jaccard Index Similarity and Hamming Distance Similarity to assess reconstruction accuracy.

Details: The effectiveness of the classifiers is measured by comparing the reconstructed images with their original binarized versions. Jaccard Index Similarity evaluates how much overlap exists between the two images by calculating the ratio of common active pixels to the total unique pixels in both images. A higher Jaccard Index Similarity score indicates better reconstruction accuracy as shown in Figure 4. Hamming Distance Similarity, on the other hand, measures bitwise differences between the original and reconstructed images as shown in Figure 5. This metric provides insight into structural deviations and the level of distortion introduced during the reconstruction process. Both similarity measures are recorded and stored for further performance evaluation and analysis [4].

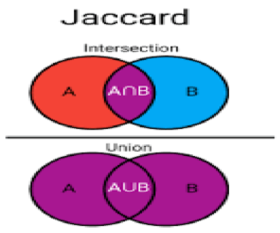


Figure 4: Visual Representation of Jaccard Similarity Index (reference: https://www.maartengrootendorst.com/blog/distances/)

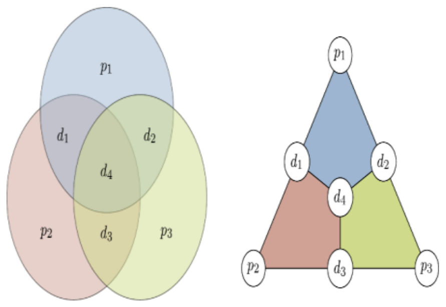


FIGURE 5: VISUAL REPRESENTATION OF HAMMING DISTANCE (REFERENCE: HTTPS://QUBIT.GUIDE/14.1-THE-HAMMING-CODE)

G. VISUALIZATION OF RESULTS

Function: The system generates graphical representations of the similarity results using Jaccard Similarity Graphs and Hamming Distance Similarity Plots to visualize classifier performance.

Details: The Jaccard Similarity Graph presents a side-by-side comparison of the accuracy scores of HTM and KNN classifiers for each test image. This allows for an easy assessment of how well each classifier preserves image structures. Additionally, a Hamming Distance Similarity Plot is created using ScottPlot, illustrating the variation in similarity scores across different test images. These plots visually highlight the strengths and weaknesses of each classifier, making it easier to interpret their effectiveness in reconstructing images. The visualizations are saved as images for reference and further analysis [4].

H. FINAL EVALUATION & CLASSIFIER RESET

Function: The system performs a final assessment to determine which classifier performed better in image prediction and reconstruction.

Details: After computing similarity scores and generating visualizations, the system evaluates the overall performance of HTM and KNN classifiers. It determines how often one classifier outperforms the other based on internal similarity metrics. By analyzing the results, the system identifies which classifier is more reliable in different scenarios. Once the evaluation is complete, the classifiers are reset, preparing them for future experiments [9].

III. RESULTS

A. IMAGE BINARIZATION

The image binarization process plays a fundamental role in preparing images for classification. The binarization algorithm, implemented in the binarizeImage function, processes input images, as shown in Figure 6, from a specified training folder by resizing them to 64x64 pixels and converts them into binary representations. This was achieved through the BinarizeImages function, which applies thresholding techniques to distinguish between black (0) and white (1) pixels. The dataset is randomly shuffled and divided into 80% training and 20% testing subsets to ensure a balanced learning process. Binarized images, as shown in Figure 7, are then stored in a designated folder named BiarizedImages, maintaining a mapping between actual images and their binarized versions for accurate reconstruction. The success of this step is evident in the structured SDR representations produced, ensuring robust feature extraction for the Spatial Pooler (SP) and subsequent classification tasks.

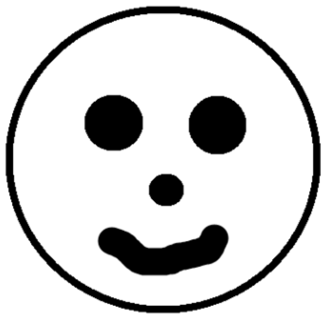


FIGURE 6: EXAMPLE OF AN INPUT IMAGE

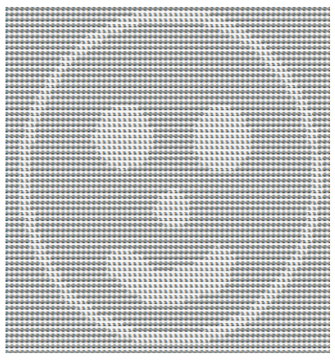
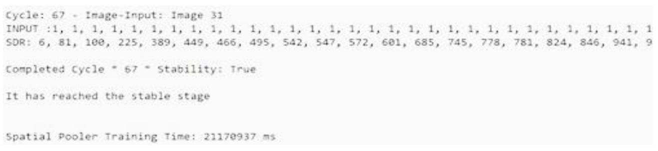


FIGURE 7: BINARIZED VERSION OF THE INPUT IMAGE

B. SPATIAL POOLER TRAINING

In the Spatial Pooler training phase, these binarized images are then encoded into sparse distributed representations (SDRs) by feeding them into the Spatial Pooler. As mentioned in Section II B, the Spatial Pooler learns spatial patterns by identifying active columns based on input overlap, updating synaptic connections, and adjusting permanence values over multiple training cycles. Stability is monitored using a Homeostatic Plasticity Controller, ensuring that the model reaches a stable state where the learned representations remain consistent as shown in Figure 8. Once the Spatial Pooler stabilizes, the SDRs generated for training images are stored and used for classification and reconstruction tasks.



```
Cycle: 67 - Image-Input: Image 31
INPUT :1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
SDR: 6, 81, 100, 225, 389, 449, 466, 495, 542, 547, 572, 601, 685, 745, 778, 781, 824, 846, 941, 9

Completed Cycle * 67 * Stability: True

It has reached the stable stage


Spatial Pooler Training Time: 21170937 ms
```

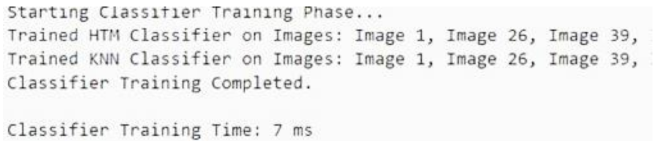
Figure 8: Output of Spatial Pooler Training

C. Classifier Training

The training phase involved utilizing two different classifiers: Hierarchical Temporal Memory (HTM) and K-Nearest Neighbors (KNN), both trained using Sparse Distributed Representations (SDRs) of binarized training images. A dictionary, training Images SDRs, was used where each key represents an image identifier (filename), and the value is the binarized SDR of the image.

For each training image, both classifiers processed the associated SDR to update their respective models, as discussed in Section II D. The training process for both classifiers was completed successfully without any errors as shown in Figure 9. Upon completion, the system logged the names of the images used for training.

Training Time: The total training time, as measured by the stopwatch, was recorded at approximately 7 ms.



```
Starting Classifier Training Phase...
Trained HTM Classifier on Images: Image 1, Image 26, Image 39,
Trained KNN Classifier on Images: Image 1, Image 26, Image 39,
Classifier Training Completed.

Classifier Training Time: 7 ms
```

FIGURE 9: OUTPUT OF CLASSIFIER TRAINING

D. Prediction and Reconstruction

The prediction phase was carried out using a set of binarized testing images. For each test image, the Spatial Pooler (SP) processed the input vector to identify active columns, which were then used as input to both the HTM and KNN classifiers for prediction, as mentioned in section II E. The classifiers then output predicted labels based on the similarity of the test image's SDR to those learned during training.

The similarity of the reconstructed images was evaluated by comparing them to the original binarized images. Both HTM and KNN classifiers were able to make predictions and reconstruction of images based on the input SDRs, producing reconstructed images for each test case, as described in Section II E**.** The images reconstructed by the HTM classifier, as shown in Figure 10, are stored in a folder named ReconstructedHTM. Similarly, the images reconstructed by the KNN classifier, as shown in Figure 11, are stored in the ReconstructedKNN folder.

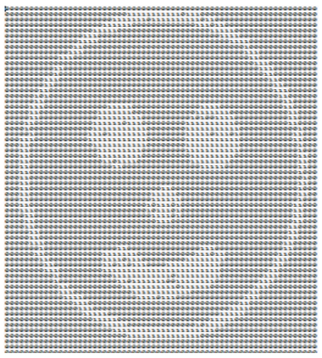


FIGURE 10: IMAGE RECONSTRUCTED BY HTM CLASSIFIER

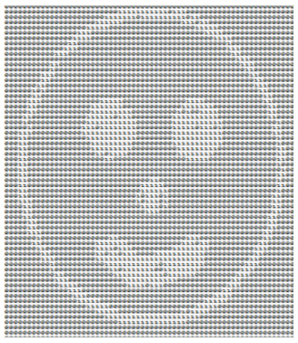


FIGURE 11: IMAGE RECONSTRUCTED BY KNN CLASSIFIER

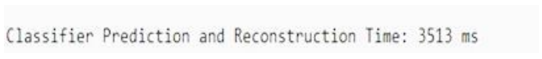
For each test image:

- HTM: The predicted label was compared, and the reconstructed image's similarity to the original was calculated.
- KNN: Similarly, the KNN predicted label was compared, and the reconstructed image's similarity was calculated.

Also, the performance of both classifiers is compared based on their internal similarity. The system tracked which classifier produced a higher similarity score for each test image, and results were logged for analysis, as mentioned in Section II F**.**

- Best Prediction: For each test image, the classifier that produced the higher similarity was flagged as the "better" classifier for that test case. In instances where both classifiers performed equally well, a message indicating equality was logged.

- Prediction and Reconstruction Time: The total time taken for prediction and reconstruction, as measured by the stopwatch, was 3513 ms as shown in Figure 12.



```
Classifier Prediction and Reconstruction Time: 3513 ms
```

FIGURE 12: OUTPUT OF CLASSIFIER PREDICTION AND RECONSTRUCTION TIME

E. Similarity Evaluation

The system generated similarity plots for both HTM and KNN classifiers. The similarity values were obtained from the reconstruction process, with higher values indicating a better match between the reconstructed image and the original binarized image.

- HTM Similarity Plot: A graph depicting the reconstruction similarity for HTM across all test images was generated using Jaccard Similarity Index and saved as HTM_Similarity_Graph.png in the designated folder. This plot illustrates the variation in HTM reconstruction performance across different test images as shown in Figure 13.

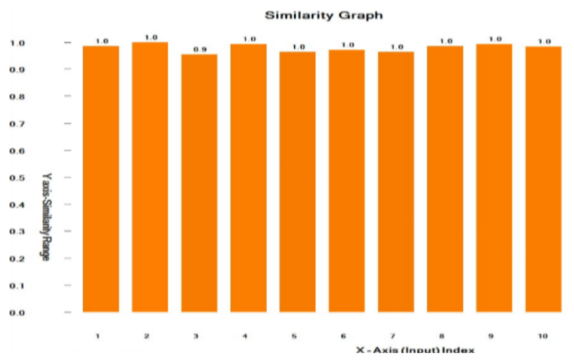


FIGURE 13: GRAPH DEPICTING THE SIMILARITY BETWEEN THE ORIGINAL BINARIZED IMAGE AND THE IMAGE RECONSTRUCTED BY HTM CLASSIFIER

- KNN Similarity Plot: Similarly, a graph for KNN reconstruction similarity was created and saved as KNN_Similarity_Graph.png in the designated folder. This plot provides a visual comparison of KNN's performance across all test images as shown in Figure 14.

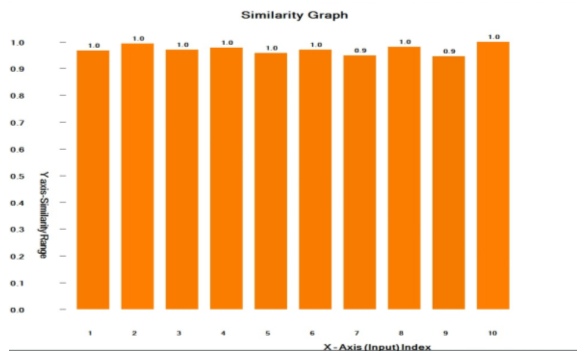


FIGURE 14: GRAPH DEPICTING THE SIMILARITY BETWEEN THE ORIGINAL BINARIZED IMAGE AND THE IMAGE RECONSTRUCTED BY KNN CLASSIFIER

Additionally, Scott Plots were generated for both HTM (as shown in Figure 15) and KNN similarity (as shown in Figure 16) results using Hamming Distance to further highlight the reconstruction similarity distribution. These plots were saved in their respective folders and can be used to assess the classifier's consistency in image reconstruction.

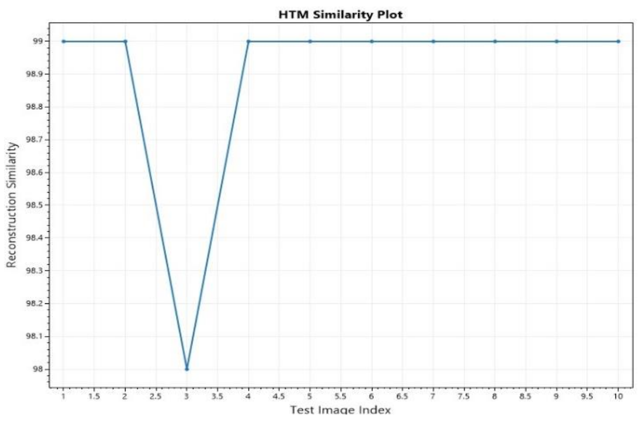


FIGURE 15: SCOTT PLOT DEPICTING THE SIMILARITY BETWEEN THE ORIGINAL BINARIZED IMAGE AND THE IMAGE RECONSTRUCTED BY HTM CLASSIFIER

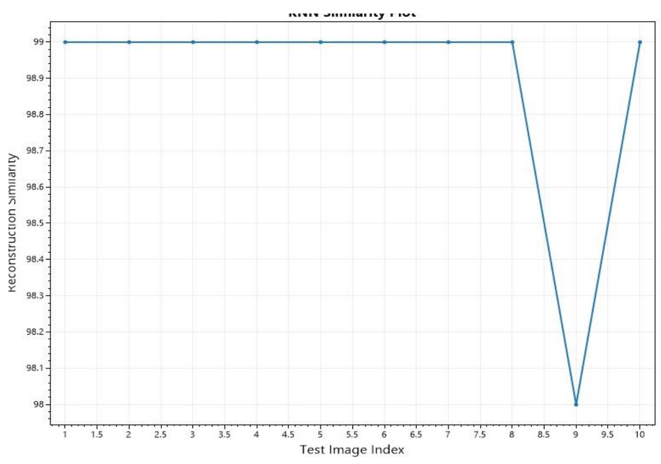


FIGURE 16: SCOTT PLOT DEPICTING THE SIMILARITY BETWEEN THE ORIGINAL BINARIZED IMAGE AND THE IMAGE RECONSTRUCTED BY KNN CLASSIFIER

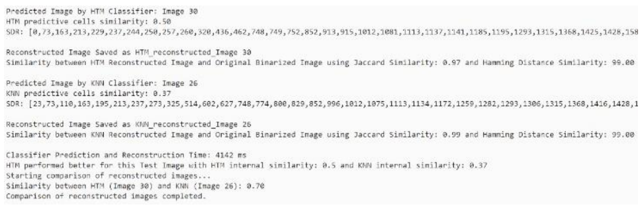
F. Comparison of Reconstructed Images

A direct comparison of the reconstructed images from HTM and KNN was carried out. The reconstruction process involves both classifiers generating reconstructed versions of the predicted images, which were then compared for similarity. This comparison is essential in determining the performance of each classifier in reconstructing the original images. For each test image, the reconstructed images from HTM and KNN were evaluated for their visual similarity and logged for further analysis.

G. HTM vs. KNN Performance:

There are two scenarios encountered in our project:

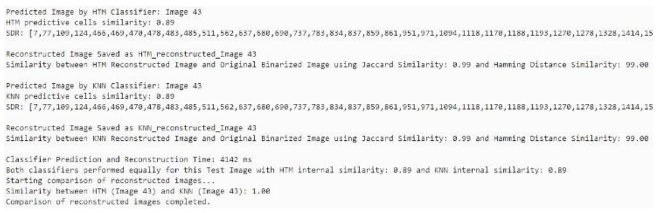
1. HTM Classifier performed better than KNN Classifier as shown in Figure 17.



```
Predicted Image by HTM Classifier: Image 30
HTM predictive cells similarity: 0.50
SDR: [0,73,163,213,229,237,244,250,257,260,320,436,462,748,749,752,852,913,915,1012,1081,1113,1137,1141,1185,1195,1293,1315,1368,1425,1428,158

Reconstructed Image Saved as HTM_reconstructed_Image 30
Similarity between HTM Reconstructed Image and Original Binarized Image using Jaccard Similarity: 0.97 and Hamming Distance Similarity: 99.00

Predicted Image by KNN Classifier: Image 26
KNN predictive cells similarity: 0.37
SDR: [23,73,110,163,195,213,237,273,325,514,602,627,748,774,800,829,852,996,1012,1075,1113,1134,1172,1259,1282,1293,1306,1315,1368,1416,1428,1

Reconstructed Image Saved as KNN_reconstructed_Image 26
Similarity between KNN Reconstructed Image and Original Binarized Image using Jaccard Similarity: 0.99 and Hamming Distance Similarity: 99.00

Classifier Prediction and Reconstruction Time: 4142 ms
HTM performed better for this Test Image with HTM internal similarity: 0.5 and KNN internal similarity: 0.37
Starting comparison of reconstructed images...
Similarity between HTM (Image 30) and KNN (Image 26): 0.70
Comparison of reconstructed images completed.
```

FIGURE 17. CONSOLE OUTPUT OF CLASSIFIER PERFORMANCE (CASE 1)

2. Both the Classifiers performed equally as shown in Figure 18.

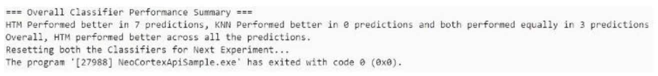


```
Predicted Image by HTM Classifier: Image 43
HTM predictive cells similarity: 0.89
SDR: [7,77,109,124,466,469,470,478,483,485,511,562,637,680,690,737,783,834,837,859,861,951,971,1094,1118,1170,1188,1193,1270,1278,1328,1414,15

Reconstructed Image Saved as HTM_reconstructed_Image 43
Similarity between HTM Reconstructed Image and Original Binarized Image using Jaccard Similarity: 0.99 and Hamming Distance Similarity: 99.00

Predicted Image by KNN Classifier: Image 43
KNN predictive cells similarity: 0.89
SDR: [7,77,109,124,466,469,470,478,483,485,511,562,637,680,690,737,783,834,837,859,861,951,971,1094,1118,1170,1188,1193,1270,1278,1328,1414,15

Reconstructed Image Saved as KNN_reconstructed_Image 43
Similarity between KNN Reconstructed Image and Original Binarized Image using Jaccard Similarity: 0.99 and Hamming Distance Similarity: 99.00

Classifier Prediction and Reconstruction Time: 4142 ms
Both classifiers performed equally for this Test Image with HTM internal similarity: 0.89 and KNN internal similarity: 0.89
Starting comparison of reconstructed images...
Similarity between HTM (Image 43) and KNN (Image 43): 1.00
Comparison of reconstructed images completed.
```

FIGURE 18 : CONSOLE OUTPUT OF CLASSIFIER PERFORMANCE (CASE 2)

H. FINAL SUMMARY AND RESET

After completing the predictions, reconstructions, and evaluations, the system logged the final results. For overall predictions and reconstructions which classifier performed better is flagged as better classifier. As shown in Figure 19 and visualized in Figure 20, the overall performance of HTM Classifier was better than KNN Classifier. The HTM and KNN classifiers were then reset, clearing any internal states, to prepare for any subsequent experiments or further testing.



```
=== Overall Classifier Performance Summary ===
HTM Performed better in 7 predictions, KNN Performed better in 0 predictions and both performed equally in 3 predictions
Overall, HTM performed better across all the predictions.
Resetting both the Classifiers for Next Experiment...
The program '[27988] NeoCortexApiSample.exe' has exited with code 0 (0x0).
```

FIGURE 19: CONSOLE OUTPUT OF OVERALL CLASSIFIER PERFORMANCE

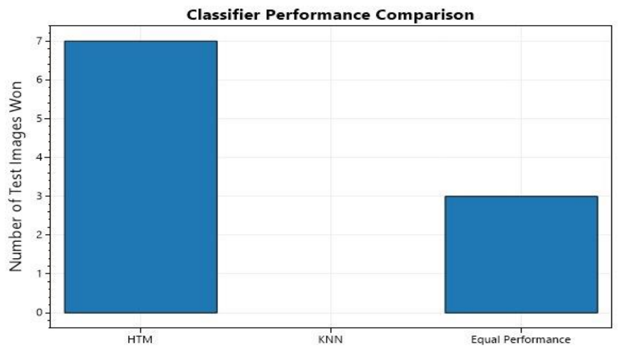


FIGURE 20: VISUALIZATION OF CLASSIFIER PERFORMANCE

IV. DISCUSSION

This project focused on implementing and analyzing two fundamentally different classification algorithms—Hierarchical Temporal Memory (HTM) and K-Nearest Neighbors (KNN)—within a framework centered around Sparse Distributed Representations (SDRs) for image recognition and reconstruction. SDRs, known for their biological plausibility and robustness, were used to encode binarized images in a format that mimics certain neural processing characteristics such as sparsity and distributed encoding. The core aim of the project was to investigate how well HTM and KNN could learn from and interpret these representations, and how effectively they could be used to reconstruct unseen image data.

The methodology was divided into two main components: prediction and reconstruction. In the prediction stage, the classifiers were trained on a dataset of binarized images transformed into SDRs, learning to associate input patterns with corresponding categories or outputs. HTM utilized its ability to learn temporal and spatial correlations through mechanisms inspired by the neocortex, while KNN classified inputs by comparing them to the most similar examples in the training set based on a chosen distance metric. Once classification was completed, the predicted SDRs were passed through a reconstruction process to regenerate visual approximations of the original input images.

To evaluate performance, the project employed two key metrics: the Jaccard Index and Hamming Distance. These measures provided insight into the similarity between the reconstructed and original images, capturing both overall overlap and bit-level differences. The results highlighted the complementary strengths of each approach. HTM demonstrated a strong ability to generalize in scenarios where pattern sequences or structural regularities existed, while KNN performed better in environments where direct similarity in feature space was a reliable predictor. Visual reconstruction added an additional layer of analysis, allowing the accuracy and nature of each prediction to be intuitively assessed. Similarity plots and Scott plots were also incorporated to visualize the spread and distribution of classification performance, further aiding in the comparative analysis of both models.

Overall, the findings emphasized that while HTM and KNN operate on fundamentally different principles, each has practical advantages depending on the structure and complexity of the input data. HTM's biologically inspired architecture made it especially robust in handling noisy or incomplete patterns, whereas KNN excelled in clearly defined, data-rich regions of the feature space. The project not only provided a technical comparison of these methods but also contributed a visual and metric-based approach for evaluating SDR-driven classification systems.

Looking forward, there are multiple avenues to enhance this work. Refining the image-to-SDR encoding process could improve the quality of input features and lead to better classification accuracy. Developing hybrid models that combine HTM’s contextual learning with KNN’s simplicity and adaptability could result in more balanced and efficient systems. Exploring alternative or adaptive similarity metrics may also improve prediction performance, especially in borderline cases. Furthermore, scaling the framework to handle higher-resolution images, more complex datasets, and real-time processing would open doors to practical applications in areas like computer vision, autonomous systems, and assistive AI technologies. This project thus sets a foundation for future research in leveraging SDRs for interpretable, robust, and biologically inspired machine learning systems.

V. References

[1] Ahmad, S., & Hawkins, J. (2015). Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory. *arXiv preprint arXiv: 1503.07469*. Available at: https://arxiv.org/abs/1503.07469. Accessed on: March 28, 2025.

[2] Balasubramaniam, J., Krishnaa, C. B. G., & Zhu, F. (2015). Enhancement of Classifiers in HTM-CLA Using Similarity Evaluation Methods. *Procedia Computer Science*, 60, 1516–1523. Retrieved on March 29, 2025, from https://www.sciencedirect.com/science/article/pii/S1877050915023881&#8203;:contentReference[oaicite:1]{index=1}

[3] Cui, Y., Ahmad, S., & Hawkins, J. (2017). The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding. *Frontiers in Computational Neuroscience*, 11, 111. Available at: https://www.frontiersin.org/articles/10.3389/fncom.2017.00111/full. Accessed on: March 28, 2025.

[4] Jaccard, P. (1912). *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*. *Bulletin de la Société vaudoise des sciences naturelles*, 47, 241–272. Retrieved on March 29, 2025, from: https://en.wikipedia.org/wiki/Jaccard_index

[5] Olshausen, B. A., & Field, D. J. (1997). Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?. *Vision Research*, 37(23), 3311–3325. Available at: https://www.sciencedirect.com/science/article/pii/S0042698997001697. Accessed on: March 28, 2025.

[6] Otsu, N. (1979). *A Threshold Selection Method from Gray-Level Histograms*. IEEE Transactions on Systems, Man, and Cybernetics. Retrieved on March 28,2025 from https://engineering.purdue.edu/kak/computervision/ECE661.08/OTSU_paper.pdf.

[7] Yang, J., Wright, J., Huang, T. S., & Ma, Y. (2010). Image Super-Resolution via Sparse Representation. *IEEE Transactions on Image Processing*, 19(11), 2861–2873. Available at: https://www.columbia.edu/~jw2966/papers/YWHM10-TIP.pdf
Accessed on: March 28, 2025.

[8] Yilmaz, E., & Gunal, S. (2020). Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Computer Science*, 1(3). Retrieved on March 29, 2025, from https://link.springer.com/article/10.1007/s42452-019-1356-9

[9] Yilmaz, E., Krishnaa, C. B. G., & Zhu, F. (2020). Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets using similarity measures. *Scientific Reports*, 10(1). Retrieved on March 29, 2025, from https://link.springer.com/article/10.1007/s42452-019-1356-9

[10] Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7), 3142–3155. Available at: https://ieeexplore.ieee.org/document/7839189. Accessed on: March 28, 2025.