Grenze International Journal of Engineering and Technology, January Issue

FPGA based Hardware Implementation of k-Nearest Neighbors(k-NN) Algorithm using the Manhattan Distance Metric for Binary Classification Tasks

Vamsi SriKrishna Arumilli[1], Asish Krishna Ghatta[2], Gowshith Sabari[3] and Remya Ajai A. S[4]
[1-4]Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, Amritapuri, India
Email: arumillivamsi@gmail.com, asishkrishna13@gmail.com, gowshithsabari123@gmail.com, remya.amrita@gmail.com

***Abstract*—The k-Nearest Neighbor (k-NN) algorithm is widely utilized for classification tasks in machine learning due to its inherent simplicity and effective performance. However, achieving high accuracy in hardware implementations of k-NN poses significant challenges, particularly related to the computational demands of distance calculations across various datasets. This paper presents an implementation of the k-NN algorithm on a Field-Programmable Gate Array (FPGA) using Verilog Hardware Description Language (HDL), employing the Manhattan distance metric to enhance computational efficiency. The proposed design supports multiple classes and is validated through simulations using various datasets, ensuring precise classification by identifying the closest matches based on the input data attributes. Optimized for FPGA-based hardware acceleration, our solution enables real-time classification by efficiently processing absolute differences in coordinates while minimizing resource utilization. The resulting implementation of algorithm in Basys3 FPGA Board provides high-speed, efficient k-NN classification that is well-suited for real-time embedded systems applications, with a scalable design capable of accommodating datasets of varying complexities.**

***Index Terms*— Classification, k-Nearest Neighbors, FPGA, Machine Learning, Verilog HDL.**

I. INTRODUCTION

The evolution of machine learning algorithms has significantly transformed various fields, including healthcare and finance, by enabling more efficient data processing and decision-making. Among these algorithms, the k-Nearest Neighbors (k-NN) algorithm stands out due to its simplicity and effectiveness in classification tasks. The k-NN algorithm, which emerged in the 1970s, operates on the principle of classifying data points based on their proximity to other labelled points within the feature space as shown in Figure 1. As data volumes continue to increase, the demand for efficient processing solutions has led to greater interest in hardware accelerators, particularly Field Programmable Gate Arrays (FPGAs). These devices can significantly enhance the performance of computationally intensive algorithms like k-NN, achieving speeds up to 30,000 times faster than traditional CPUs through their parallel processing capabilities [1]. This reflects a broader trend toward integrating machine learning directly into hardware for real-time analytics and decision-making.

As applications become more sophisticated and datasets grow larger, the execution time for k-NN increases exponentially, making hardware acceleration essential. Techniques like DCT- k-NN improve accuracy through feature weighting, particularly beneficial for time-sensitive applications such as COVID-19 classification [2].

Grenze ID: 01.GIJET.11.1.541_1

In medical applications, k-NN has emerged as a promising method for cancer classification and seizure detection, with studies demonstrating that refining datasets through feature selection can significantly enhance classification accuracy [3]. Enhanced edge detection techniques facilitate effective segmentation of medical images, which in turn supports improved k-NN performance in classification tasks [4]. Research shows that FPGA-based k-NN implementations can achieve substantial performance gains, with dynamic partial reconfiguration providing flexibility and further speed improvements, significantly outperforming general-purpose processors [5]. The necessity for efficient implementations of k-NN is further underscored by its application in real-time systems, such as controlling motor direction through head movement recognition and interpreting EEG signals for navigation [6].

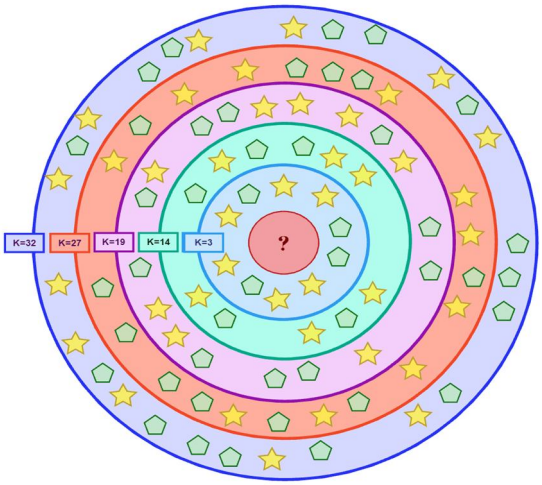

Fig.1 k-Nearest Neighbors (k-NN) algorithm Classification

The implementation of k-NN on FPGA platforms enables effective classification in environments that require speed and energy efficiency, making it a crucial component in the development of practical machine learning solutions. Moreover, k-NN's straightforward nature facilitates its integration into various applications, such as agriculture, where it can differentiate between healthy and diseased leaves, enabling real-time processing for quick detection [7]. Power dissipation remains a critical challenge in FPGA digital systems, especially for algorithms like k-NN, which require significant resources [8]. Optimizing k-NN for lower power consumption while maintaining high performance is vital for efficient implementation on FPGA platforms. Techniques to assess hardware utilization and processing speed emphasize k-NN’s capability for efficient classification in FPGA environments [9]. Furthermore, the Predetermined Range Search (PRS) method enhances k-NN’s efficiency by focusing on relevant data points, achieving higher performance while minimizing resource usage [10]. Fast classification algorithms are crucial for effective detection, and innovations such as transfer learning can inform improvements in k-NN's accuracy and performance for various applications [11].

In embedded systems, k-NN has diverse applications, including image recognition, medical diagnosis, and anomaly detection, where model interpretability and adaptability are crucial. Recent studies highlight how FPGA implementations of k-NN can improve classification accuracy, utilizing techniques such as the Reduced-Rank Local Distance Metric and high-level tools like OpenCL, which align with the goal of enhancing k-NN performance on the Basys3 FPGA [2]. Significantly, FPGA-based implementations can achieve a high throughput of 4.44 Gbit/sec while maintaining over 90% accuracy with minimal resource utilization [12]. The effectiveness of k-NN is further underscored by its performance in complex classifications, such as biomedical image analysis, where it consistently outperforms linear kernel SVMs due to its reliance on local data patterns [13]. Implementing classification algorithms on FPGA utilizes its parallel architecture for faster computations, which is crucial for real-time applications [14]. FPGA technology also plays a crucial role in automated systems, offering reconfigurability and parallel processing capabilities, enabling real-time adaptability and efficient management of complex operations [15].

This paper emphasizes a detailed exploration of the k-NN algorithm implemented as a binary classifier on an FPGA, specifically utilizing the Xilinx Vivado platform and designed for deployment on the Basys3 board. By employing Hardware Description Language (HDL), this implementation aims to achieve superior performance metrics compared to conventional software approaches. The proposed design demonstrates the feasibility of

FPGA technology for flexible, scalable, and efficient k-NN applications, validated through simulations on diverse datasets to ensure robustness across multiple classification tasks. Through simulation and evaluation, this study illustrates the feasibility and benefits of hardware-accelerated k-NN, facilitating further innovations in machine learning on embedded platforms. This work contributes significantly to the ongoing discussion on optimizing machine learning algorithms for hardware implementation, supporting advancements in electronic systems that require fast and reliable data processing

II. METHODOLOGY

The k-Nearest Neighbors (k-NN) algorithm is a well-established supervised classification technique that determines the class of an unknown data point by examining the closest known neighbors. In binary classification, the algorithm identifies the most common class among the k-Nearest Neighbors using a specific distance metric. This implementation utilizes Manhattan distance, noted for its computational efficiency on FPGA hardware [1].

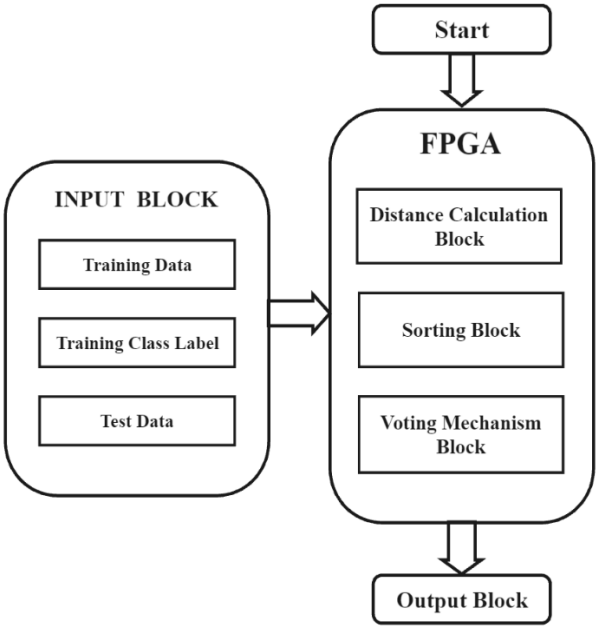

Fig.2 System Architecture of k-NN algorithm

A. Architecture

Input Logic Block: This block computes the Manhattan distance between the test point and each training point. By leveraging parallel processing, it quickly calculates the distances, which are essential for determining proximity in the k-NN algorithm.

Distance Calculation Block: This block sorts the calculated distances to identify the closest neighbors. Efficient sorting algorithms are implemented to minimize latency, ensuring that the nearest instances can be rapidly recovered for classification.

Sorting Block: This block determines the predicted class based on majority voting among the nearest neighbors. By counting the occurrences of each class label among the top K neighbors, it ensures that the prediction reflects the most relevant training instances.

Voting Mechanism Block: This block determines the predicted class based on majority voting among the nearest neighbours.

Output Block: This block provides the predicted class as the final output as shown in Figure 2. It formats the result for transmission or display, ensuring clarity and usability in the context of the broader application or system.

B. Loading Training Data

The initial step in the k-NN process involves loading the training data into the FPGA. This data is initialized as part of the module inputs, where each training point comprises its feature values and the corresponding class

label. The Input Logic Block facilitates efficient data handling, enabling rapid access during the classification process.

C. Distance Calculation

The similarity between test data and training data is evaluated using a distance function. While the Euclidean distance is a widely used metric, its requirement for a square root operation can be computationally intensive and complex to implement on FPGA platforms. Chebyshev distance is not typically included due to similar concerns. Consequently, Manhattan distance is employed for similarity detection. This distance is defined by Equation 1, where X_i represents the coordinates of the test point and Y_i denotes the coordinates of the training points. The calculation of this distance is executed within the distance calculation block in FPGA, utilizing combinational logic to compute distances for all training points in parallel. This approach enhances efficiency and optimizes performance in FPGA implementations

D. Sorting Distances

Sorting the computed distances is essential for identifying the k-Nearest Neighbors. While the basic design does not employ a specific sorting algorithm, it effectively simulates a sorting mechanism through a sequential search for the minimum distance within the array. By iterating through the distance array and marking the distances that have been processed, the design successfully identifies the nearest neighbors.

E. Classification of Unknown Data

After identifying the k-Nearest Neighbors, a majority voting mechanism is implemented to determine the predicted class. The classes associated with the k-Nearest Neighbors are counted, and the class with the highest frequency is selected as the predicted class. This process is executed using an iterative loop that traverses the K nearest distances, systematically updating the count for each respective class. The final output block then displays the predicted class, which is determined based on the majority vote.

III. RESULTS

The simulation of the k-Nearest Neighbors (k-NN) algorithm is performed on the Xilinx Vivado platform, followed by its implementation on the Basys 3 FPGA board, which is built on the Xilinx Artix-7 architecture. This board operates at a clock frequency of 100 MHz and contains approximately 20,800 Look-Up Tables (LUTs) and 41,600 flip-flops.

A. Simulation

To validate the simulation, a dataset comprising 307 samples was utilized, with 212 samples designated for training and 95 samples for testing. Each training instance includes two features and a corresponding class label, while a few test samples from the set of 95 are demonstrated in Figure 3.

The control algorithm operates by using the training samples, their corresponding class labels, and the test samples as inputs. The predicted class for each test sample is determined based on the similarities to the training instances and their class labels. The k value, which defines the number of nearest neighbors to be considered during the classification process, is optimized based on the training dataset. This tuning process ensures an appropriate balance between model complexity and generalization ability, enabling the model to achieve optimal performance in terms of accuracy and robustness when applied to the test dataset or unseen data.

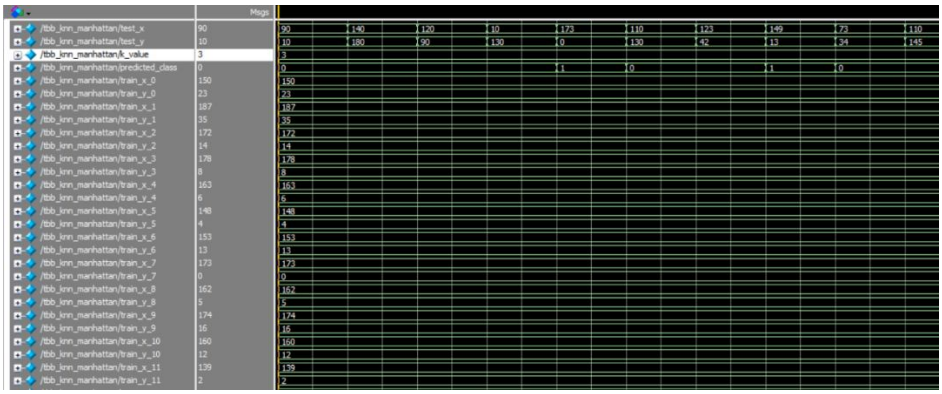

Fig.3 Simulation result of the Control Algorithm

B. Implementation on FPGA

The simulation was initially performed in Vivado using various datasets to test the control algorithm. The waveform in Figure 3 illustrates a simulation instance corresponding to one of the datasets. For the hardware implementation on the Basys3 FPGA board, the dataset was modified, while the control algorithm remains unchanged. The sample dataset used for implementation is provided in Table 1.

TABLE I. SAMPLE TRAINING DATASET

Index	Training Dataset		Class Label
	Feature 1 (x1)	Feature 2 (y1)	
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

C. Register Transfer Level Illustration

The RTL (Register Transfer Level) diagram is created using the Elaborated Design feature within the Vivado Design Suite. Figure 4 displays an RTL diagram that focuses on a segment of the k-NN (k-Nearest Neighbors) algorithm. This representation helps in visualizing the hardware implementation of the algorithm, making it easier to analyse the data flow and control mechanisms involved.

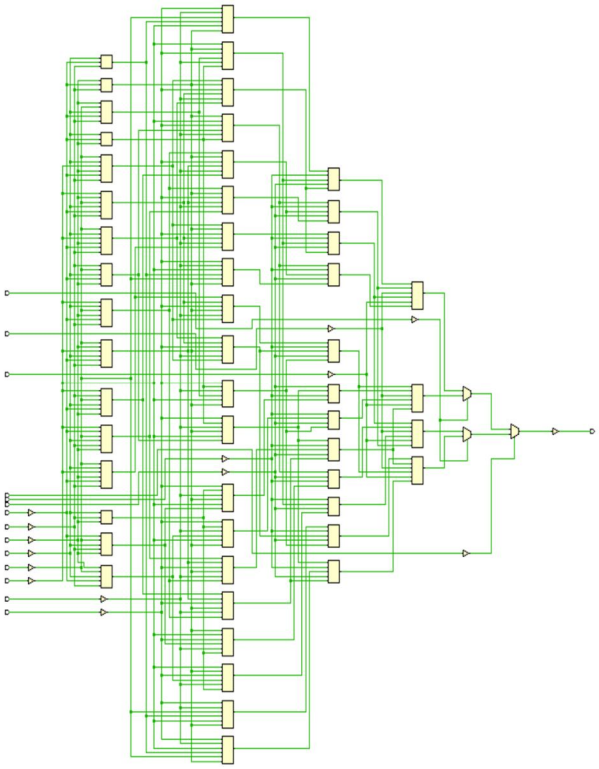

Fig.4 RTL Schematic of the k-NN Algorithm

D. Hardware Implementation

The implementation of the hardware has been successfully developed using Verilog Hardware Description Language (HDL), with synthesis and analysis conducted using the Vivado Design Suite. A few implementation results for the dataset shown in Table 1 are showcased in the figures below. When the LED corresponding to the predicted class is HIGH, it indicates classification into class 1; conversely, when the LED is LOW, it indicates classification into class 0.

Test data 1: (x2, y2) => (0,0)

In Table 2(a), the test point (0,0) is considered. The training data and the corresponding class labels including the calculated Manhattan distances are specified. The predicted class, i.e. 1, is shown is Figure 5(a).

TABLE II

Index	Training Dataset		Class Label	Distance
	Feature 1 (x1)	Feature 2 (y1)		
1	0	0	0	00
2	0	1	1	01
3	1	0	1	01
4	1	1	0	10

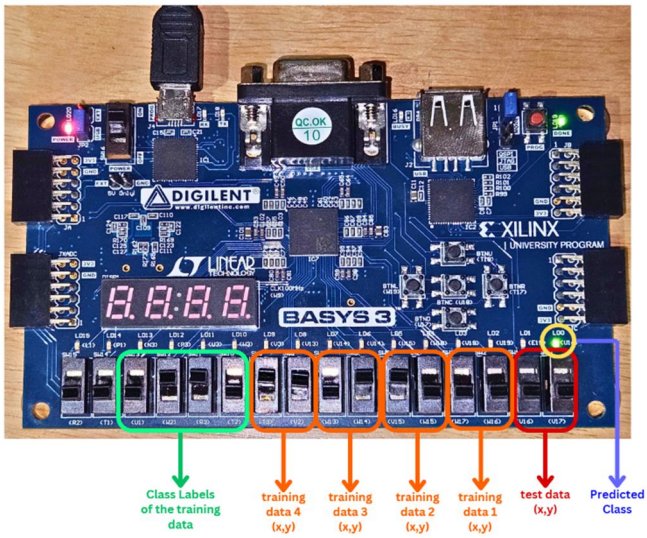

Fig.5 (a)

Test data 2: (x2, y2) => (0,1)
In Table 2(b), the test point (0,1) is analysed. The training data, corresponding class labels, and calculated Manhattan distances are provided. The predicted class, i.e. 0, is illustrated in Figure 5(b).

TABLE III

Index	Training Dataset		Class Label	Distance
	Feature 1 (x1)	Feature 2 (y1)		
1	0	0	0	01
2	0	1	1	00
3	1	0	1	10
4	1	1	0	01

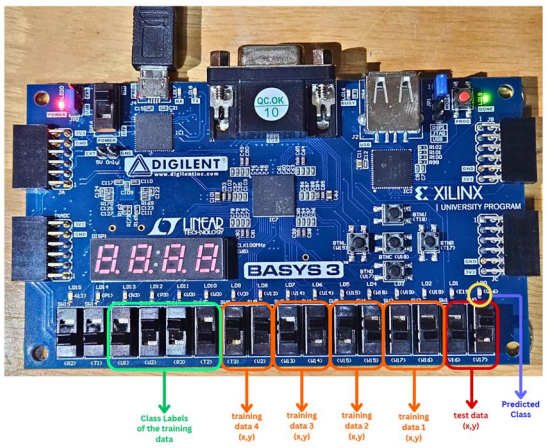

Fig.5 (b)

Test data 3: (x2, y2) => (1,0)
In Table 2(c), the test point (1,0) is considered. The training data and the corresponding class labels including the calculated Manhattan distances are specified. The predicted class, i.e. 0 is shown is Figure 5(c).

TABLE IV

Index	Training Dataset		Class Label	Distance
	Feature 1 (x1)	Feature 2 (y1)		
1	0	0	0	01
2	0	1	1	10
3	1	0	1	00
4	1	1	0	01

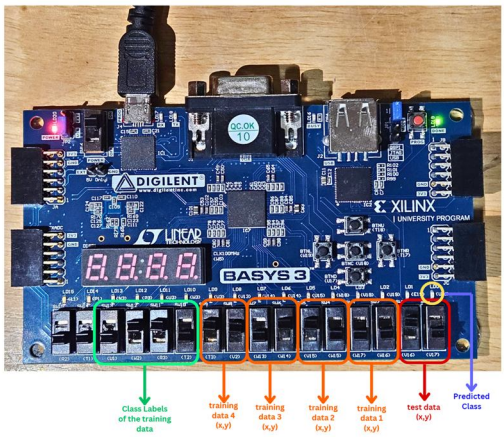

Fig.5 (c)

Test data 4: (x2, y2) => (1,1)
In Table 2(d), the test point (1,1) is analysed. The training data, corresponding class labels, and calculated Manhattan distances are provided. In Figure 5(d), the predicted class, i.e. 1, is illustrated.

TABLE V

Index	Training Dataset		Class Label	Distance
	Feature 1 (x1)	Feature 2 (y1)		
1	0	0	0	10
2	0	1	1	01
3	1	0	1	01
4	1	1	0	00

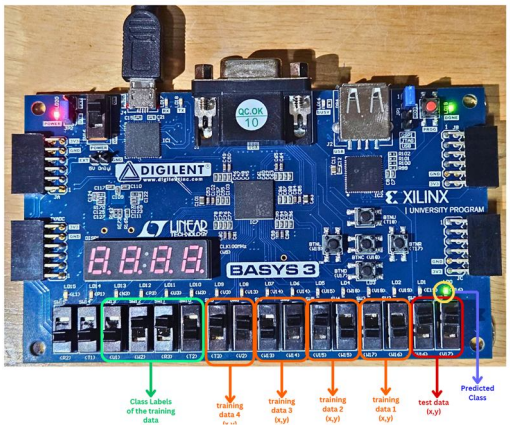

Fig.5 (d)

E. FPGA Resource Utilization

TABLE VI. RESOURCE UTILIZATION

Resource	Utilization	Available	Utilization Percentage
Slice LUTs	48	20800	0.23
F7 Muxes	2	16300	0.01
F8 Muxes	1	8150	0.01
Slice	15	8150	0.18
LUT as Logic	48	20800	0.23
LUT FF Pairs	48	20800	0.23
Bounded IOB	15	106	14.15

The analysis in Table 3 indicates that FPGA resources are utilized efficiently in the implemented design. Slice LUTs and Flip-Flops occupy only a small fraction of the available space, approximately 0.23%. The utilization of F7 and F8 Muxes is minimal, at 0.01% each. Overall Slice utilization is recorded at 0.18%. In contrast, the Bonded IOBs exhibit a higher utilization rate of 14.15%. This efficient resource allocation underscores the effectiveness of the design.

F. Power Consumption

A simulation of the k-Nearest Neighbors (k-NN) algorithm on an FPGA demonstrates a design characterized by efficiency and thermal stability, with a total on-chip power consumption of 2.851 W, primarily driven by dynamic power. A significant portion, specifically 82%, is attributed to input/output (I/O) operations, underscoring effective data handling crucial for high-speed classification tasks. The recorded junction temperature of 39.3°C, along with a thermal margin of 60.7°C, ensures that the design operates well within established limits, thereby minimizing the risk of overheating. Although the confidence level is low, indicating a potential need for improved activity data, the overall power profile remains favourable. This underscores the FPGA's capability to manage intensive computations in a controlled thermal environment. The stability and efficient I/O management affirm the FPGA as a suitable platform for the implementation of real-time, high-throughput algorithms such as k-NN.

IV. CONCLUSION

In conclusion, the FPGA-based implementation of the k-Nearest Neighbors (k-NN) algorithm effectively employs Manhattan distance for efficient classification. The architecture incorporates dedicated blocks for distance calculation, sorting, and voting, facilitating rapid processing and accurate predictions, as shown by the successful LED indicators on the Basys 3 FPGA board. Resource utilization remains low, indicating an efficient design, while the power consumption profile underscores the system's ability to manage intensive computations effectively. This implementation demonstrates the potential of FPGAs for real-time, high-throughput machine learning tasks. Future enhancements may focus on optimizing activity data for power profiling and expanding dataset capabilities, further reinforcing the suitability of FPGAs for advanced applications in machine learning.

FUTURE WORKS

The FPGA-based implementation of the k-Nearest Neighbors (k-NN) algorithm can be significantly enhanced through various strategies. One area for improvement involves algorithm optimization, including adaptive selection of K and the integration of additional distance metrics to enhance classification performance and efficiency. Hardware advancements may include the implementation of advanced parallel processing techniques and the optimization of resource utilization, particularly for larger datasets.

Expanding the dataset for testing with real-world applications would facilitate a more comprehensive assessment of scalability and accuracy. Additionally, efforts to reduce power consumption through dynamic voltage scaling and effective thermal management strategies would ensure reliability under varying loads. Integrating the FPGA implementation with established machine learning frameworks could streamline model deployment, while the development of user interfaces for real-time data interaction would improve usability. Exploration of ensemble methods and transfer learning could further enhance classification outcomes. Finally, addressing data security

concerns and providing comprehensive documentation would promote community contributions, thereby enhancing the robustness and applicability of the k-NN algorithm in practical scenarios

REFERENCES

[1] N. U. Sadad, A. Afrin and M. N. I. Mondal, "Binary Classification using K-Nearest Neighbor Algorithm on FPGA," 2021 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), Rajshahi, Bangladesh, 2021, pp. 1-4, doi: 10.1109/IC4ME253898.2021.9768439.

[2] Abedalmuhdi Almomany, Walaa R. Ayyad, Amin Jarrah, Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 6, Part B, 2022, Pages 3815-3827, ISSN 1319-1578, doi.org/10.1016/j.jksuci.2022.04.006.

[3] G. Jaffino, J. P. Jose, R. Raman, R. S. Venkatesan and V. P. M. B. Aarthi, "FPGA-Based System for Real-time Epileptic Seizure Detection using KNN Classifier," 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 2023, pp. 01-05, doi: 10.1109/ICEEICT56924.2023.10157075.

[4] Remya Ajai A S, Sundararaman Gopalan, "Comparative Analysis of Eight Direction Sobel Edge Detection Algorithm for Brain Tumor MRI Images", Procedia Computer Science, Volume 201, 2022, Pages 487-494, ISSN 1877-0509, doi.org/10.1016/j.procs.2022.03.063.

[5] H. M. Hussain, K. Benkrid and H. Seker, "An adaptive implementation of a dynamically reconfigurable K-nearest neighbour classifier on FPGA," 2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Erlangen, Germany, 2012, pp. 205-212, doi: 10.1109/AHS.2012.6268651.

[6] Y. Pu, J. Peng, L. Huang and J. Chen, "An Efficient KNN Algorithm Implemented on FPGA Based Heterogeneous Computing System Using OpenCL," 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines, Vancouver, BC, Canada, 2015, pp. 167-170, doi: 10.1109/FCCM.2015.7.

[7] J. N. Reddy, K. Vinod and A. S. Remya Ajai, "Analysis of Classification Algorithms for Plant Leaf Disease Detection," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2019, pp. 1-6, doi: 10.1109/ICECCT.2019.8869090.

[8] F. T. Naser, S. N. Hadi and I. A. Hashim, "Power Optimization of KNN Algorithm Based on FPGA," 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA), Najaf, Iraq, 2021, pp. 168-174, doi: 10.1109/IICETA51758.2021.9717930.

[9] "Data Classification with Different Sorting Techniques using Machine Learning Algorithm on FPGA," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-5, doi: 10.1109/ICCCNT56998.2023.10306748.

[10] Miren Tian, Xin'an Wang, Xing Zhang, Zhiqiang Yang, Jipan Huang and Hao Chen, "The implementation of a KNN classifier on FPGA with a parallel and pipelined architecture based on Predetermined Range Search," 2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Hangzhou, China, 2016, pp. 1491-1493, doi: 10.1109/ICSICT.2016.7998779.

[11] Archana R. Nair and Remya Ajai, A. S, "Analysis of COVID-19 Detection Algorithms Based on Convolutional Neural Network Models Using Chest X-ray Images", Advances in Computing and Data Sciences, 2022, Springer International Publishing, 52--63, doi.org/10.1007/978-3-031-12641-3_5

[12] M. H. Yacoub, S. M. Ismail, L. A. Said, A. H. Madian and A. G. Radwan, "Generic Hardware Realization of K Nearest Neighbors on FPGA," 2022 International Conference on Microelectronics (ICM), Casablanca, Morocco, 2022, pp. 169-172, doi: 10.1109/ICM56065.2022.10005537.

[13] A. S. Remya Ajai and S. Gopalan, "Analysis of Active Contours Without Edge-Based Segmentation Technique for Brain Tumor Classification Using SVM and KNN Classifiers" in , Singapore:Springer, pp. 1-10, 2020.

[14] J. Amrutha and A. S. Remya Ajai, "Performance analysis of Backpropagation Algorithm of Artificial Neural Networks in Verilog," 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 1547-1550, doi: 10.1109/RTEICT42901.2018.9012614.

[15] V. S. Arumilli, A. K. Ghatta and R. K. Megalingam, "FPGA – Controlled Automated Coffee Maker using Verilog," 2024 Third International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 2024, pp. 1-7, doi: 10.1109/ICEEICT61591.2024.10718631.