BIO-INFORMATICS DATA CLASSIFICATION USING KNN ON FPGA BOARD

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTERS OF TECHNOLOGY
IN
VLSI DESIGN AND EMBEDDED SYSTEMS

Submitted by:

MAYANK MANGLA
(2K21/VLS/13)

Under the supervision of

Dr. O. P. VERMA, PROFESSOR (ECE)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Bhawana Road, Rohini, Delhi- 110042

MAY, 2023

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bhawana Road, Rohini, Delhi- 110042

CANDIDATE'S DECLARATION

I, MAYANK MANGLA (2K21/VLS/13), student of MTech (VLSI Design and EMBEDDED SYSTEM), hereby declare that the MAJOR Project report "**BIO-INFORMATICS DATA CLASSIFICATION USING KNN ON FPGA BOARD"** which is submitted by me to the department of Electronics and Communication Engineering, DELHI TECHNOLOGICAL UNIVERSITY, DELHI in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associate ship, fellowship or other similar title or recognition.

MAYANK MANGLA (2K21/VLS/13)
MTech (VLSI Design and EMBEDDED SYSTEM)
Department of Electronics & Communication Engineering

Date: 30 - 05 - 2023
Place: Delhi

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bhawana Road, Rohini, Delhi- 110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "BIO-INFORMATICS DATA CLASSIFICATION USING KNN ON FPGA BOARD" which is submitted by MAYANK MANGLA, Roll No. 2K21/VLS/13 to the department of Electronics and Communication Engineering, Delhi Technological University, Delhi in the partial fulfilment of the requirement for the award of the degree of Master of Technology, is record work of the report work carried out by student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Prof. O. P. Verma

(Project Supervisor)

Department of Electronics & Communication Engineering

Date: 30 - 05 - 2023

Place: Delhi

ABSTRACT

Bioinformatics data is treated as high-dimensional data by nature, which requires great computational demands. Scientists around the world have proposed many computing solutions such as Field-Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs). In order to meet these high computational demands, a feature wherein we can alter some specific areas in the chip proves to be very helpful and resourceful, from which FPGAs are benefiting. FPGAs enable flexible, reconfigurable computing, as most enable the user to reprogram the hardware circuit with different logic functions. Applying classification machine learning algorithms to the bioinformatics data set on a conventional PC to get the desired result proves to be a very time-consuming task. This is where the FPGAs come into the picture and help in drastically reducing this computational time. In this project, we have implemented the above approach using an FPGA board and executed its software-based implementation on a CPU to compare them on the grounds of timing. The hardware implementation of the algorithm is done using Verilog and for the software-based implementation we have used Python. Furthermore, in this project we have done a comparative analysis by adopting different sorting technique which plays a vital role in the KNN classification algorithm implemented. These algorithms also are one of the pivotal factors for the speed and hardware utilization requirement for implementing the algorithm on the FPGA board.

ELECTRONICS AND COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bhawana Road, Rohini, Delhi- 110042

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to my highly respected and esteemed guide **Prof. O. P. Verma** for his useful guidance, encouragement and contribution offered throughout all phases of this project. He has been very encouraging and motivating and the intensity of encouragement has always increased with time. Without his constant support and guidance, I would not have been able to complete this work. I extend my sincere thanks to all my friends who have patiently helped us directly or indirectly in accomplishing this work successfully.

MAYANK MANGLA (2K21/VLS/13)
MTech (VLSI Design and EMBEDDED SYSTEM)
Department of Electronics & Communication Engineering

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

Terabytes of data are processed daily by massive data centres in the information technology age. These data centres heavily rely on machine learning algorithms for classifications, forecasts, recognition, recommendations, etc. Processing big amounts of data using these techniques is quite computationally intensive. It has motivated attempts in recent years to use hardware accelerations to address this issue, which are accomplished by utilising various architectures like GPUs and FPGAs.

An important architecture for attaining great efficiency is heterogeneous computing system architecture, which combines CPUs with FPGAs. The FPGA architecture may be utilised to implement machine learning algorithms like K-NN since it is ideal for parallelization and reconfiguration [4]. FPGAs are a good contender since many businesses are looking for effective solutions to attain high performance and low energy expenses. The K-NN method is frequently used in applications such as data mining, text classification, prediction analysis, pattern recognition, etc.

In the field of data mining, it is regarded as one of the most significant algorithms of the 20th century [7]. Hardware-accelerated K-NN is a preferable option because the K-NN algorithm's execution time grows exponentially with the quantity of the training data and requires a lot of computing resources. A new era with a plethora of innovative biomedical technology is quickly approaching. The use of computer and analytic tools to collect and analyse biological data is what bioinformatics is all about.

In this project we have also demonstrated the supremacy of one sorting algorithm over another. As, the sorting algorithm plays a vital role in defining the speed and the hardware utilization of the machine learning algorithm implemented, we have compared the 2 adapted sorting algorithms on the grounds of speed and hardware utilization.

1.2 Literature Review

Firstly, B. B. C, A. Deshmukh, and A. V. Narasimhadhan in their paper “Modulation and signal class labelling with active learning and classification using machine learning” (2022 IEEE International Conference on Electronics, Computing and Communication Technologies), explores the application of active learning and classification techniques using machine learning for modulation and signal class labelling [1]. They propose a methodology that combines active learning and classification algorithms to improve the accuracy and efficiency of modulation and signal class labelling. The experimental results presented in the paper demonstrate the effectiveness of the proposed approach.

A comparison of different classification techniques in the context of vocational guidance data is presented in the paper “Comparison of Classification Techniques used in Machine Learning as Applied on Vocational Guidance Data” authored by H. I. Bulbul and O. Unsal [2]. The authors evaluate and compare the performance of various machine learning algorithms for classifying vocational guidance data. The results and analysis provided in the paper offer insights into the effectiveness of different classification techniques in this specific domain. Addressing the problem of multi-class classification of Turkish texts, the paper “Multi-Class Classification of Turkish Texts with Machine Learning Algorithms” by F. G¨urcan, explores the use of machine learning algorithms. The authors evaluate the performance of various machine learning techniques in classifying Turkish texts into multiple classes [3]. The paper discusses the challenges specific to Turkish language text classification and provides insights into the effectiveness of different machine learning algorithms in this context.

S. Gandhare and B. Karthikeyan in “Survey on FPGA Architecture and Recent Applications” (ISMSIT) presents a survey on FPGA architecture and its recent applications [4]. The paper discusses the basics of FPGA architecture, its advantages for implementing various applications, and reviews recent research and developments in FPGA technology. It provides a comprehensive overview of FPGA applications in different domains. An optimized FPGA architecture for machine learning applications using Posit multipliers is proposed by K. Elsaid, M. Safar and M. W. El-Kharashi in “Optimized FPGA Architecture for Machine Learning Applications using Posit Multipliers” [5].

To accelerate the training process of Support Vector Machine (SVM) machine learning algorithms, C. Kardaris, C. Kachris and D. Soudris in paper "A high-performance FPGA architecture for Acceleration of SVM Machine Learning Training" (PACET) presents a high-performance FPGA architecture [6]. The authors propose an efficient FPGA design that optimizes computation and memory access patterns. Experimental results and comparisons with CPU-based implementations highlight the superior performance of the FPGA architecture. The research and application of an intersection similarity algorithm based on the K-Nearest Neighbor (KNN) classification model are discussed in "Research and Application of Intersection Similarity Algorithm Based on KNN Classification Model" by W. Lv, H. Huang, W. Tang and T. Chen [7]. The authors propose a novel approach to calculate intersection similarity for KNN classification, aiming to improve accuracy and efficiency. The paper presents experimental results and discusses practical applications of the proposed algorithm.

The implementation of binary classification using the K-Nearest Neighbor (KNN) algorithm on FPGA is presented in "Binary Classification using K-Nearest Neighbor Algorithm on FPGA" [8]. The authors N. U. Sadad, A. Afrin and M. N. I. Mondal, propose an FPGA-based approach for efficient binary classification tasks. They provide details of the FPGA implementation and evaluate its performance through experimental results and comparisons with other approaches.

R. S. Latha et al. in the paper "Stock Movement Prediction using KNN Machine Learning Algorithm" (ICCCI), the application of the K-Nearest Neighbor (KNN) algorithm for stock movement prediction is explored [9]. The authors propose a KNN-based approach to predict the movement of stocks based on historical data. The paper discusses the methodology, features used, and experimental results to demonstrate the effectiveness of the KNN algorithm in stock prediction. For the classification of Electromyography (EMG) signals for human hand rehabilitation, S. Briouza, H. Gritli, N. Khraief, S. Belghith and D. Singh in paper "EMG Signal Classification for Human Hand Rehabilitation via Two Machine Learning Techniques: KNN and SVM" compares the performance of two machine learning techniques: KNN and SVM. The authors conduct a comparative study between KNN and SVM algorithms and evaluate their accuracy in classifying EMG signals [10]. The paper presents the experimental setup, feature extraction methods, and performance evaluation of both algorithms.

Majumder et al. in paper “Machine learning approach for argument extraction of bio-molecular events” presents a machine learning approach for argument extraction of bio-molecular events [11]. The authors propose a methodology that leverages machine learning techniques to automatically extract arguments related to bio-molecular events from scientific literature. The paper describes the dataset used, feature extraction techniques, and the evaluation of the proposed approach.

In the paper “FPGA Architecture To Enhance Hardware Acceleration for Machine Learning Applications” by Itagi et al., an FPGA architecture is presented with the objective of enhancing hardware acceleration for machine learning applications [12]. The authors propose an optimized FPGA design that focuses on improving the performance and efficiency of machine learning algorithms. The paper discusses the detailed architecture design and provides insights into the implementation process. Additionally, the performance gains achieved through FPGA-based acceleration are evaluated, showcasing the effectiveness of the proposed approach. Furthermore, Y. Zou and M. Lin in “GridGAS: An I/O-Efficient Heterogeneous FPGA+CPU Computing Platform for Very Large-Scale Graph Analytics” introduces the GridGas platform, which combines FPGA and CPU to create an I/O-efficient computing platform specifically designed for large-scale graph analytics [13]. This platform demonstrates promising results in terms of performance and scalability, making a significant contribution to the field of graph analytics.

1.3 What is Machine Learning

Machine learning is a field within AI that seeks to develop systems capable of learning from past data, identifying patterns, and drawing logical inferences with minimal human intervention. It involves employing data analysis techniques on various types of digital information, including text, numbers, clicks, and images, to automatically construct analytical models.

Machine learning applications utilize automated optimization techniques to continuously improve the accuracy of their outputs by learning from input data. The effectiveness of a machine learning model is influenced by two key factors:

Data quality: The saying "Garbage in, garbage out" holds true in the development of machine learning algorithms. If the input data is disorganized or of poor quality, the model's results will likely be significantly flawed.

Model selection: Different machine learning techniques have specific applications that data scientists can choose from. It is essential to select the appropriate algorithm for each use case. Neural networks, for example, have gained considerable attention due to their impressive accuracy and versatility. However, simpler models often yield better results when dealing with limited amounts of data

1.4 Need for Machine Learning

Machine learning is becoming increasingly significant because of the expanding quantity and variety of data, along with the enhanced availability and affordability of computing power and high-speed internet. These factors associated with digital transformation allow for the swift and automated development of models capable of analysing extensive and complex datasets with accuracy and efficiency. Machine learning is utilized in diverse fields, including suggesting products and services, identifying cybersecurity breaches, and enabling autonomous vehicles.

Machine learning has become crucial due to the exponential growth of data, as traditional data analysis and processing methods are insufficient in handling large volumes of data, including unstructured and complex data types like text, images, and videos. Machine learning algorithms excel in processing such data and enable computers to recognize patterns and make predictions based on historical data. This capability has proven invaluable in fields like finance, healthcare, marketing, and cybersecurity, where the ability to identify trends, anomalies, and potential risks can provide significant insights for informed decision-making.

Another advantage of machine learning is its ability to automate time-consuming and repetitive tasks. By leveraging existing data, machines can automate processes, make intelligent decisions, and optimize workflows, resulting in enhanced efficiency and productivity across various industries.

1.5 Introduction to FPGA

The hardware design engineer has the capability to program custom Digital Logic using Field Programmable Gate Arrays (FPGAs), which are digital Integrated Circuits (ICs) [2]. FPGAs are referred to as "Field Programmable" because the end-user or designer can program the Digital Logic of the IC, as opposed to it being fixed during the manufacturing process.

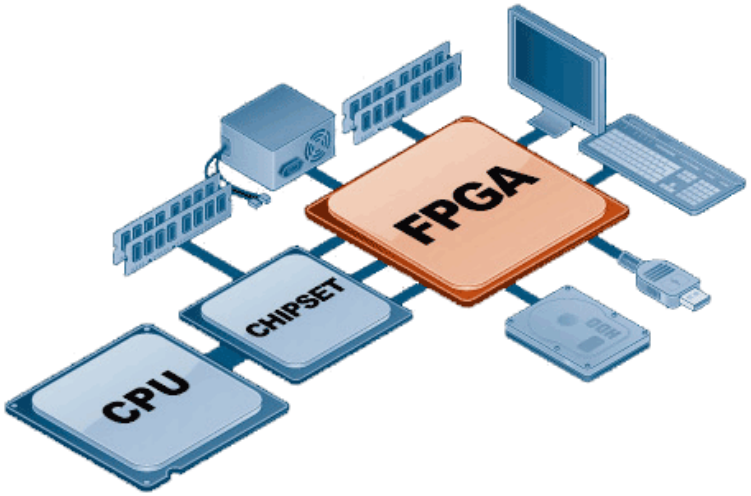

Figure 1.5.1: FPGA

The FPGAs constitutes of programmable logic blocks which enables then to be versatile in nature and gives them the ability of handle complex data with a very small computational time. The logic which we implement in the FPGAs are hardwired unlike in CPUs because of which we see the supremacy of FPGA over CPUs on the grounds of speed. But, this enhancement in speed comes with a trade-off in power consumption, as FPGAs consume more power as compared to our conventional CPUs.

FPGAs have been available in the market for over three decades, and during this time, they have undergone significant technological advancements and gained increasing popularity [3]. Also, as compared to the conventional CPUs, FPGAs are less complex as they do not have fixed set of processes and logic blocks designed by the manufacturers.

CHAPTER 2: BACKGROUND

2.1 Introduction to Classification & its Types

Depending on the nature of the dependent variable, various machine learning classification techniques can be applied. Among the different classification techniques, the following are the most common ones:

2.1.1 Binary Classification:

Binary classification is the most fundamental and widely used type of classification. It involves a dependent variable that consists of two distinct categories represented by the numbers 1 and 0. Typically, 1 represents "True" and 0 represents "False." For example, if we are predicting whether a bank member will default on a loan, the dependent variable "Loan Defaulter" would be either 1 (True) or 0 (False). Binary classification serves as the foundation for many classification algorithms and is the most well-known type of classification method.

2.1.2 Binomial Classification:

Binomial classification is like binary classification, but the dependent variable has two categories instead of just one. These categories may not necessarily be in the form of "True" or "False." For example, a dataset may have features indicating pixel density, and the dependent variable could have categories like "Car" or "Bike." These categories can still be encoded as 0 and 1, making it like binary classification, especially in the context of machine learning.

2.1.3 Multi-class Classification:

When the target variable has more than two categories or groups, multi-class classification is used. The classification method seeks to ascertain the correlation between the input variables and the data points, and each data point is given a distinct class assignment. Each piece of data is categorised into a single, distinct class during prediction. A multi-class classification challenge would be one in which data points must be categorised as "Safe," "At-Risk," or "Unsafe," for example. A data point cannot simultaneously belong to numerous classes in this situation.

2.1.4 Multi-label Classification:

Multi-label classification is like multi-class classification, but here the dependent variable can have more than two categories, and a single observation can be assigned multiple categories. The classification algorithm needs to understand the potential classes for each observation and interpret the patterns accordingly. This type of classification is commonly used in text mining, where an observation (e.g., a newspaper article) can have multiple categories in the dependent variable (e.g., "Politics," "Names of Politicians Involved," "Important Geographical Locations").

In summary, classification can take different forms depending on the business problem and the nature of the dependent variable. It is important to identify the appropriate classification technique and algorithms based on the specific problem at hand.

2.2 Machine Learning Classification Algorithm

We are aware that the bulk of algorithms used to oversee machine learning are classification and regression techniques. Utilising training data, the classification process uses supervised learning to categorise subsequent observations [4]. It is a programme that classifies fresh observations and divides them into several classes or groups after learning from the supplied data set or observations.

In classification algorithms, the output is classified into a certain category, such as "Diabetic" or "Non-Diabetic," "Class A" or "Class B," "Black" or "White," etc. Regression algorithms, however, produce a final output that takes the form of a value. Since the K-NN classification technique is a supervised learning algorithm, it uses labelled input data, which implies that it has input and the related output.

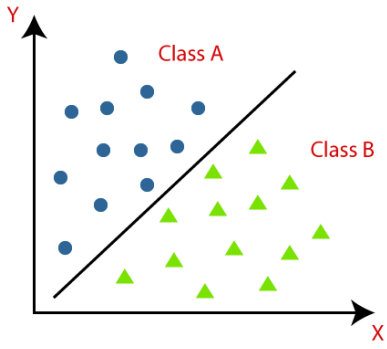

Figure 2.1.1: Classification Algorithm

The classification algorithm, however, can be classify into two subcategories, i.e., binary classifiers and multi-class classifiers. This project uses the binary classifier approach, which states that the problem will only have two possible outcomes.

2.3 KNN – Introduction

This paper's primary contribution is a comparison of the processing rates of traditional CPUs and FPGAs, as well as an application of the K-NN classification method on an FPGA. In this paper, we will demonstrate why FPGAs are a far better option for demanding high-level processing and large-scale applications. According to the K-NN classification technique, all computations are suspended pending the local approximation and evaluation of the function. More diverse training data is necessary in order to improve the effectiveness and accuracy of this classification algorithm, which will significantly improve accuracy [1]. This algorithm assigns weights to the nearest neighbours, which proves to be a beneficial method for not only classification but also regression by making the closer neighbour more contributively as compared to the farther one.

In order to represent non-numerical data, preprocessing and feature engineering may be necessary to create feature vectors. A feature vector is a mathematical representation of data, where each entry in the vector corresponds to a specific feature of the data point. For data with N distinct features, the feature vector would be a vector of length N, and each entry represents the value of the corresponding feature. Essentially, each feature vector can be viewed as a point in an N-dimensional space (R^N).

K-nearest neighbours (KNN) is a type of lazy learning algorithm, which means it does not have an explicit training phase before classification, unlike many other classification methods. Instead, any abstractions or generalizations of the data are based on the classification itself. While this allows for immediate classification once data is available, there are some inherent challenges with this approach. The computational cost of classification can be high because the algorithm needs to compare the data point with all other data points for each classification. This means that the complete training set must be stored in memory, unless some form of data reduction is applied. Consequently, KNN often performs better on smaller datasets with fewer features due to these considerations.

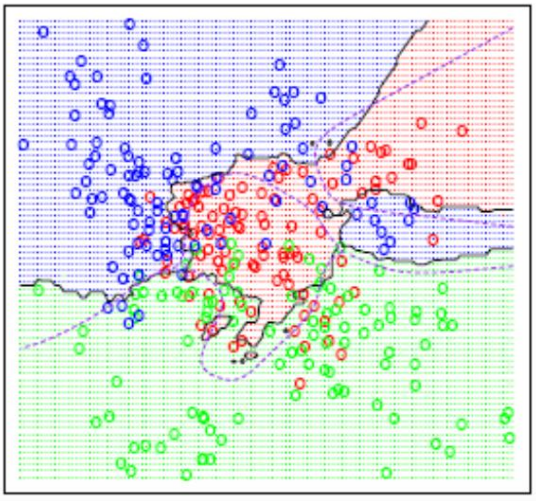

Figure 2.3.1: KNN Classification

Unless you have some prior understanding that clearly leads to selecting one over the other, it may be advisable to just utilise cross-validation to decide when choosing a measure because it can be difficult sometimes. The direction of a word is more significant than the magnitude of the component values, therefore you might want to utilise Cosine similarity for things like word vectors. Both approaches will typically take about the same amount of time to complete and will struggle with large dimensional data.

The outcome of the KNN algorithm is a decision boundary that divides RN into parts after carrying out all of the aforementioned steps and selecting a measure. A class is represented by each part (clearly coloured below) in the classification issue. The boundaries are determined using the distance measure and the available training points rather than having to be created with real training instances. We can determine the most likely class for a hypothetical data point in an area by taking R^N in (small) chunks, and we then colour that chunk as being in the region for that class.

2.4 KNN – Algorithm

There is a designated algorithm flow for the KNN algorithm where we make the below assumptions:

- A data set 'D.'
- A Distance Set metric is required in order to store and measure the distance between the test and training data set.

- The most crucial and important parameter of the K-NN algorithm is ‘K’, which denotes the number of nearest neighbours to be considered. The following steps need to be taken in order to get the output based on our input values: Here we assume the output to be denoted by “Y” and the input as “X.”

STEP 1: Calculation of the total distance between the test data input ’X’ and the training data points.

STEP 2: Now, we recognize the nearest ’K’ values between the test data and the training data set.

STEP 3: If it is a regression problem, we will take the mean of the ’Y’ outputs we have obtained considering nearest ’K’ neighbours. Whereas, if it turns out to be an classification problem case we will take the Mode of the same ’Y’ output observations.

STEP 4: The value calculated in the step 3 will be the final prediction.

CHAPTER 3: METHODOLOGY

3.1 Algorithm Flow

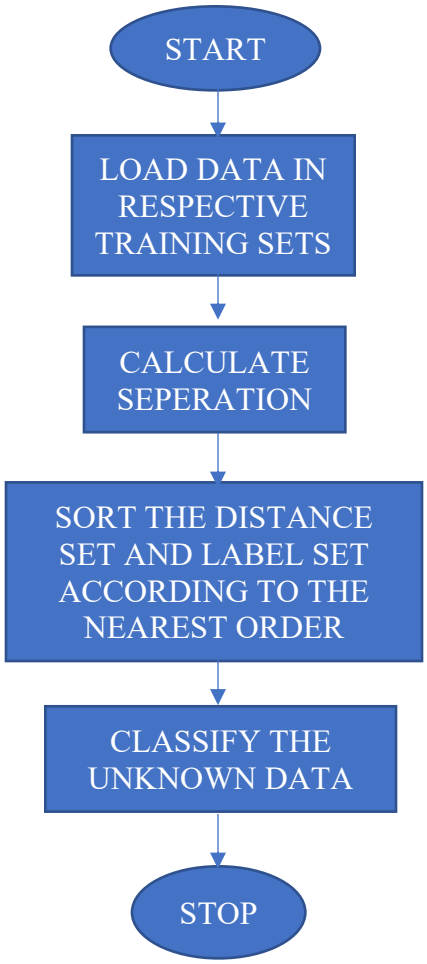

Figure 3.1.1: Algorithm Flow

As depicted in Figure 2, the very first thing the program does is load all the training data set values in to the respective sets. After this, the calculation of each test data with its respective training data set takes place and gets stored into another distance set which is mapped to respective test input. After calculating the separation, sorting algorithm starts running and sorts the distance as well as the label set in accordance to the nearest distance. After this step we reach the final stage of execution where we after implemented a majority function to classify our test input to a particular class.

3.2 Hardware Implementation

A sequential, clock driven circuit implements the KNN algorithm. It consists of four steps: loading data, comparing data, sorting training data, and categorising unclassified data. After the information has been fed into the registers, the calculation for the closest distance begins.

3.3 Similarity Calculation

We need to find the most appropriate distance metric in accordance with the dataset in order for the algorithm to perform optimally. We will just discuss a few frequently used distance measurements out of the many various distance metrics that are accessible. The Manhattan distance function is the most often used of them all since it is the default setting in the Python scalar KNN classifier package.

Some of the Distance methods are mentioned below for the reference:

Minkowski Distance: It is a metric specific to real-valued vector spaces. It is used to calculate distances between vectors in a normed vector space, where distances are represented by non-negative vector lengths. The Minkowski distance satisfies several conditions as a distance metric, including:

- Non-negativity: There is never a negative distance between any two places.
- Identity of indiscernibles: If and only if two points are identical, the distance between them is 0.
- Symmetry: The separation between points A and B is equal to the separation between points B and A.
- Triangle inequality: The distance between two points A and C is never greater than or equal to the sum of the distances between A and B and B and C. This is known as the triangle inequality.

These conditions ensure that the Minkowski distance is a valid distance metric that satisfies fundamental properties.

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

(1)

Manhattan Distance: It is also referred to as taxicab distance or city block distance, is a measure of distance between two points. It is named as such because it reflects the distance a taxicab would need to travel to navigate the city blocks or streets.

The Manhattan Distance is calculated by taking the sum of the absolute differences between the Cartesian coordinates of the two points. In other words, it measures the total distance a taxicab would need to travel horizontally and vertically to move from one point to another, without considering diagonal movement.

The name "Manhattan" is derived from the layout of streets in Manhattan, New York, where the streets form a grid-like pattern, making the taxicab distance a suitable measure for calculating travel distances.

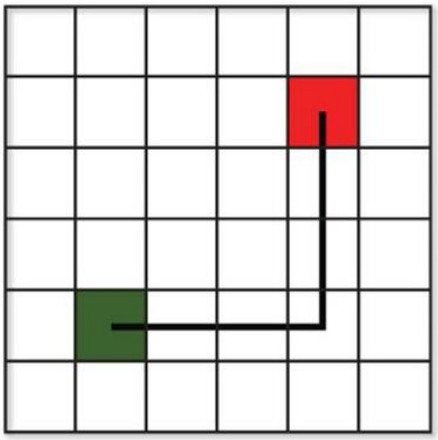

Figure 3.3.1: Manhattan Distance

Euclidean Distance – Euclidean Distance is a widely used distance metric, particularly in machine learning algorithms such as K-Nearest Neighbors (KNN). It is the default metric used by the Python SKlearn package for KNN.

The Euclidean Distance is the linear separation between two points in Euclidean space. The total of the squared differences between the corresponding coordinates of the two points is used to calculate it.

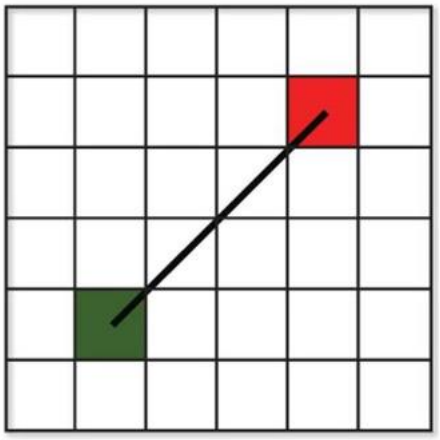

Figure 3.3.2: Euclidian Distance

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

(2)

Cosine Distance – Cosine Distance is a distance measure commonly used to assess the similarity between two vectors. It determines whether two vectors are pointing in the same direction by calculating the cosine of the angle between them.

In text analysis, Cosine Distance is often utilized to measure document similarity. By considering the frequency of terms in a document, it helps determine how similar or related two documents are to each other. When combined with K-Nearest Neighbours (KNN), Cosine Distance can provide new insights into business problems and uncover hidden information that may not be evident when using other distance measures.

By analysing the occurrence of specific terms in a document, text analytics can leverage Cosine Distance to compare and quantify the similarity between two documents. This measure proves valuable in various text-related tasks, such as document clustering, information retrieval, and text classification.

Jaccard Distance - Jaccard Distance is a comparison technique that is similar to Cosine Similarity in the sense that both methods evaluate a single type of attribute distributed across all data points.

The Jaccard coefficient measures the similarity between two sets of data by identifying the instances where both values are 1 (or present). By calculating the proportion of 1:1 matches to all data points, it provides a measure of how closely the two sets align. This concept is closely related to what the Cosine Similarity algorithm aims to achieve.

However, it's important to note that Jaccard Distance can be highly sensitive to small sample sizes and may yield erroneous results, particularly with very small data sets or in the presence of missing observations. Care should be taken when applying Jaccard Distance in such cases to avoid potential inaccuracies or misleading interpretations.

Hamming Distance - A metric for contrasting two binary data strings is the hamming distance. The number of bit places in which two binary strings of equal length are different from one another is known as the hamming distance. The Hamming distance approach examines the entire set of data to determine whether individual data points are comparable or dissimilar. The Hamming distance reveals the number of various qualities. This is primarily utilised while one-hot encoding data and determining the separation between two binary vectors.

3.4 Data Sorting Algorithm

One of the most computationally demanding activities on the entire hardware architecture is sorting training data in accordance with a similarity function. We used the faster and easier selection sorting algorithm to accomplish the classification process. In the selection sort algorithm, every element is selected repeatedly and then compared across all the unselected values. If the

selected value is found to be smaller than the unselected one, the values will be swapped and the method will be repeated. This carries on until we get our sorted array.

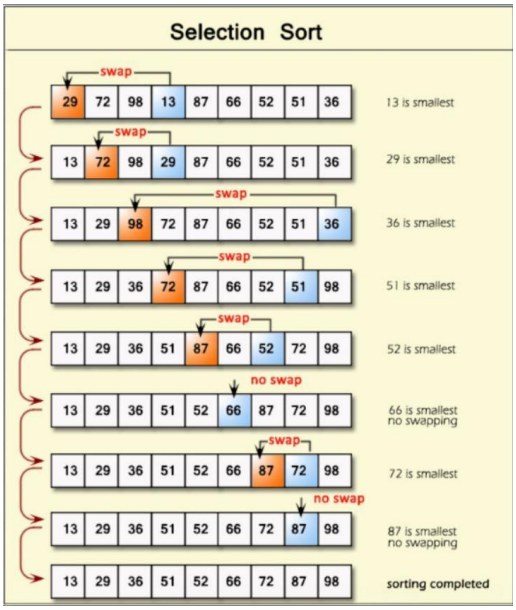

Figure 3.4.1: Selection Sort Algorithm

Each array is maintained by the method with two sub-arrays.

• The sub-array which has already been sorted.

• The final sub-array was unsorted.

The minimum element from the unsorted sub-array is selected and moved to the sorted sub-array in each iteration of the selection sort.

3.4.1 Introduction to Data Sorting

Data sorting is a process of arranging data in a meaningful order to simplify comprehension, analysis, and visualization. It is commonly used when working with research data to present it in a way that facilitates understanding the underlying story conveyed by the data. Sorting can be applied to both raw data, encompassing all individual records, and aggregated data presented in tables, graphics, or other summarized outputs.

Typically, data sorting involves arranging the data in either ascending or descending order based on actual numeric values, counts, or percentages. Additionally, data can be sorted based on the labels of variable values. Some software programs allow researchers to assign labels to each value choice of a categorical variable, enabling sorting based on these labels. Furthermore, sorting can be performed based on multiple factors, prioritizing specific variables. For instance, a dataset with fields for country and region can be sorted first by region as the primary sort, followed by a secondary sort based on country within each region.

By organizing data in a systematic order, data sorting facilitates data analysis, pattern recognition, and drawing meaningful insights from the information at hand

3.4.2 Standard Application of Data Sorting

When working with data, there are several common applications of sorting that are widely used. One of these applications is data cleaning, which involves sorting through the data to identify any irregularities or patterns that require attention. For example, monthly sales data can be sorted by month to detect any fluctuations or changes in sales volume over time.

Sorting is also frequently used for ranking or prioritizing records based on specific criteria. This can involve sorting data according to a rank, computed score, or other prioritization values. For instance, sorting customer data based on their purchase volume or ranking accounts based on their importance can aid in decision-making and resource allocation.

In order to facilitate effective data interpretation, visualizations such as tables and charts need to be properly sorted. It is common practice in market research to arrange the results of a single response question in descending order by column percentage, from the most answered option to the least answered option. This allows for a clear and meaningful representation of the data, particularly when analysing brand preferences or other similar questions.

3.4.3 Classification of Sorting Algorithms

Sorting algorithms can be classified based on various parameters, which include:

1. Number of swaps or inversions required: This parameter indicates the number of times elements need to be swapped to sort the input. For example, selection sort requires the minimum number of swaps among sorting algorithms.

2. Number of comparisons: This parameter measures the number of comparisons performed by the algorithm before sorting the input. Most sorting algorithms, such as the ones mentioned earlier, require at least O($nlogn$) comparisons in the best case and O(N^2) comparisons in the worst case, according to Big-O notation.

3. Use of recursion: Some sorting algorithms employ recursive approaches, like quicksort, to sort the input. On the other hand, non-recursive methods are used by other algorithms like selection sort and insertion sort. Additionally, certain algorithms, such as merge sort, utilize both recursive and non-recursive techniques.

4. Stability: Stability refers to whether the relative order of items with equal values or keys is preserved during sorting. Stable sorting algorithms maintain the relative arrangement of such items, while unstable sorting algorithms do not guarantee this preservation.

These parameters help categorize sorting algorithms and provide insights into their performance characteristics behaviour.

CHAPTER 4: SORTING TECHIQUES UTILIZED

4.1 Introduction

In this project we have successfully demonstrated a machine learning classification algorithm and ran it over a 3-dimensional complex bio-medical database. In this we were successfully able achieve our goal of demonstrating the supremacy of FPGAs over the conventional CPUs. For this, we took 30 random bio-medical samples and test through our implemented algorithm. For convergence we also took the help of F-measure which is a key notifying factor on how much accurate the implemented algorithm is.

Upon this conclusion, we were very inclined to improve our machine learning algorithm much further on the FPGA board. The one factor we took into consideration on improving the efficiency and the speed of the algorithm is to enhance the sorting algorithm embedded in the implementation of the same. Earlier we used the "Selection Sort" algorithm to get the sorted set of data and label. In this improved version we have incorporated the "Insertion Sort" algorithm which proves to be much faster and more stable version of the existing sorting algorithm incorporated.

Not only our aim was to improve speed by using this sorting algorithm, but also to reduce the utilization of the hardware components. As we know in the biomedical fields, the time and cost are the two constraints that are very crucial. This new advancement not only proves to be passing the time criteria of the field but also is a more cost-effective algorithm as it utilizes less components of hardware.

4.2 Technique Utilized

4.2.1 Introduction to Insertion Sorting

The simple sorting technique known as insertion sort is like holding a deck of cards in your hand. The sorted section and the unsorted portion of the array are separated. Elements are chosen from the unsorted component and positioned where they belong in the sorted portion. The process of insertion sort is like sorting playing cards. Before selecting an unsorted card in a card game, it is assumed that the previous cards have already been sorted. If the chosen unsorted card is larger than

the first card, it is placed to the right; otherwise, it is placed to the left. This process is repeated for all unsorted cards, collecting them, and placing them in their correct positions. This approach is also followed in insertion sort.

Insertion sort works by iterating over the sorted array one element at a time. Although it is easy to implement, insertion sort is not efficient for large data sets due to its time complexity, which is O(n^2) in both the best and worst cases.

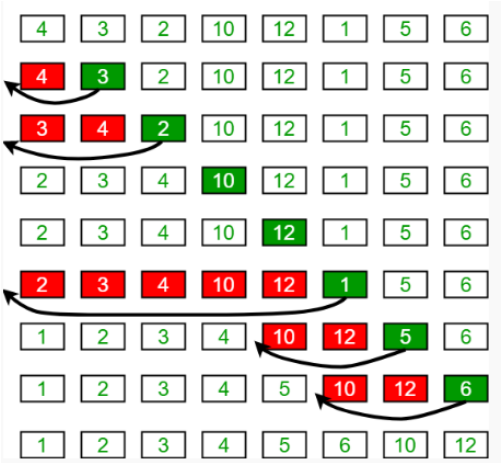

Figure 4.2.1: Insertion Sort Algorithm

4.2.2 Working of Insertion Sort

Suppose we need to sort the following array.

Figure 4.2.2.1: Sample Unsorted Array

1. The In the insertion sort algorithm, the first element in the array is considered already sorted. The second element is then taken and stored separately as a key.

The key is compared with the first element. If the first element is greater than the key, the key is placed in front of the first element.

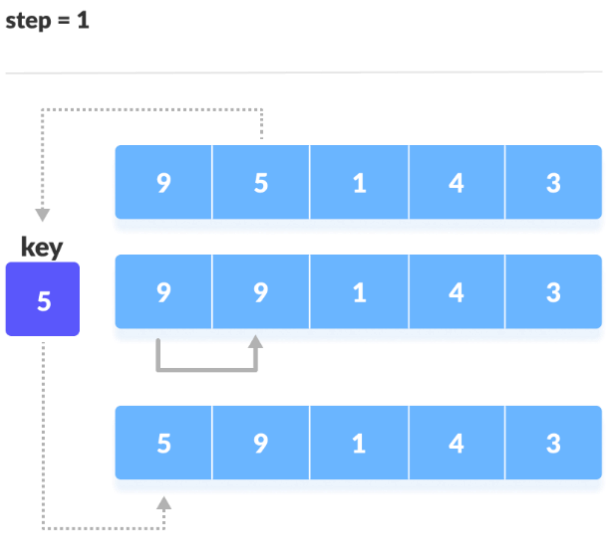

Figure 4.2.2.2: Insertion Sort Step 1

1. The first two components have now been sorted. Compare the third element to the ones to its left to see how they compare. Put it behind the component that was smaller than it in front of it. If there is not an element smaller than it, put it at the beginning of the array.

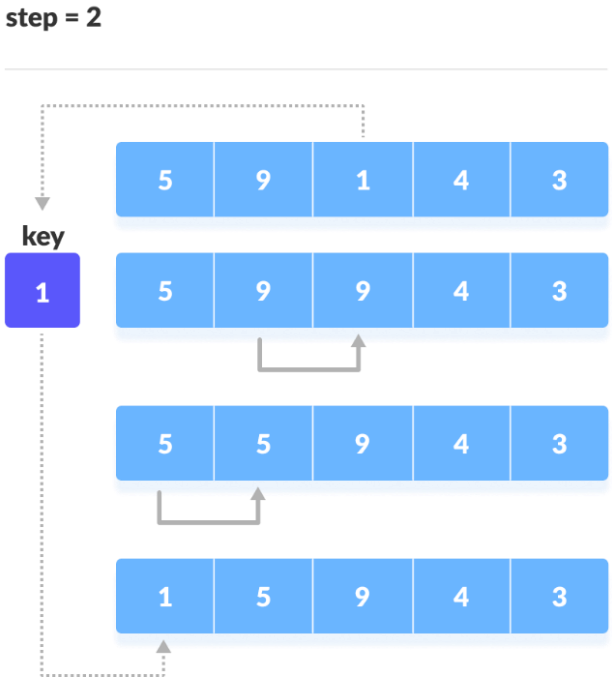

Figure 4.2.2.3: Insertion Sort Step 2

2. In a similar manner, put each unsorted component in its proper location.

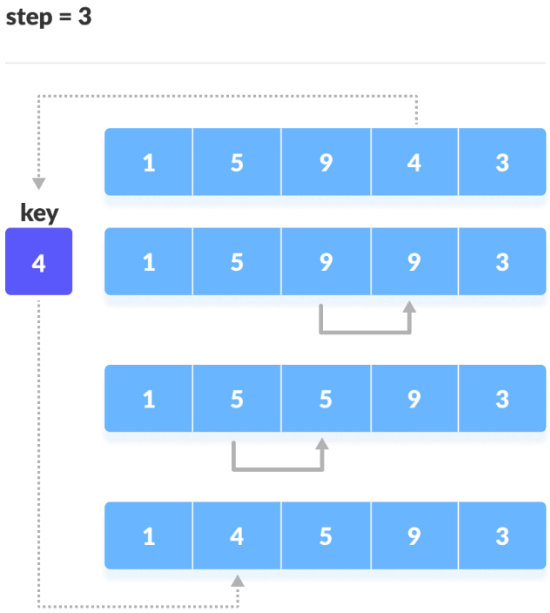

Figure 4.2.2.4: Insertion Sort Step 3

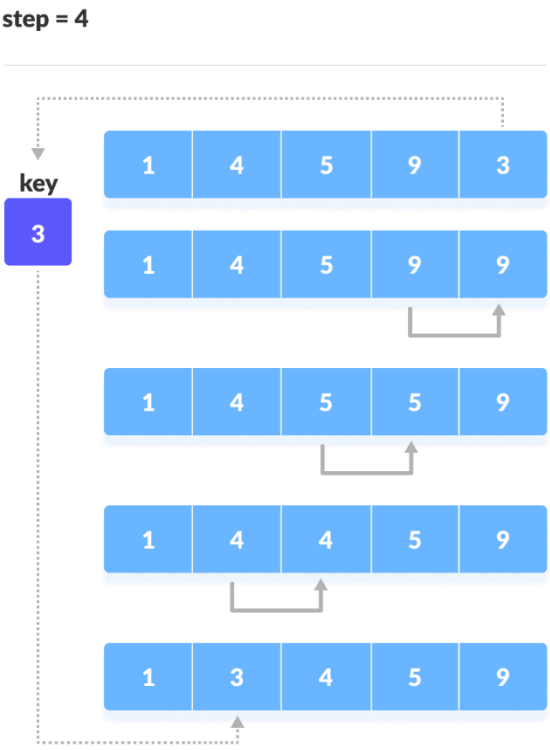

Figure 4.2.2.5: Insertion Sort Step 4

4.3 Comparison between Selection Sort and Insertion Sort technique

Two well-liked sorting algorithms, insertion sort and selection sort, vary primarily in how they choose and arrange elements in a sorted sequence.

- While insertion sort scans the sorted section to identify the ideal position to insert the element, selection sort scans the unsorted part to find the element that is the minimum.
- Compared to insertion sort, selection sort necessitates fewer swaps but more comparisons.
- When the input array is just slightly sorted or almost sorted, insertion sort is more effective than selection sort; selection sort is more effective when the array is significantly unsorted.
- The insertion sort sorts a set of values by inserting the values into a prestored file. The selection sort, on the other hand, determines the least number from the list.

In conclusion, the temporal complexity of the two algorithms is comparable, but the selection and placement strategies are different. The decision between them is based on the qualities of the input data and the particular specifications of the current situation.

4.4 Results comparison on the grounds of Timing and Hardware Utilization

The two main areas of comparison which will determine the supremacy of one technique over the other would-be Timing and Hardware Utilization.

4.4.1 Timing & Hardware Utilization Comparison

For this comparison we have used our 3-Dimensional bio-medical dataset with 40 sample values.

TABLE I: Timing & Hardware Utilisation

Technique	Timing (nsec)	Utilization			
		LUT	FF	IO	BUFG
Selection Sort	8.773	5272	3835	1338	1
Insertion Sort	5.85	3589	4046	1338	1

CHAPTER 5: RESULTS

On a Windows 10 PC with a 2.5 GHz 8th Gen Intel Core i7 CPU, Python is used to build algorithms for software-based K-NN implementation on the CPU. On the Zynq 7000 series FPGA board, a K-NN implementation built on an FPGA is employed. This board's video codec supports a wide range of popular peripherals and interfaces for embedded vision use cases and is based on the FPGA architecture from Xilinx. All the scripts are written in HDL, and the hardware designs are synthesised and analysed using the Vivado Design Suite.

5.1 Sample Dataset

We used a simple data set to verify our FPGA implementation. We used up to 30 pieces of data for training purposes. Our training data has two attributes and one label. Sample training data is shown in Table I. The label 1 represents a diabetic category, and conversely, 0 represents a non-diabetic category. The below table is just a part of the training data set that we have used and implemented our project upon. This is a 3-dimensional data set that helped us with the accuracy of the result and to validate the same in all the three factors which is Glucose level, Sugar level and Age.

TABLE II: Sample Data Set

Training Data No.	Data Set 1 (Glucose Level)	Data Set 2 (BP-Diastolic)	Data Set 3 (Age)	Label
1	148	72	50	1
2	85	66	31	0
3	183	64	32	1
4	89	66	21	0
5	137	40	33	1

5.2 Hardware Simulation

The simulation feature of Vivado Design Suite generates the timing diagram. Figure 3 and 4 depict the timing diagrams pertaining to the design run of the project. The circuit includes clk, K (a parameter), Tran data 1, Tran data 2, Tran data 3, and Train data 4. Label the input pins as “input” and the output pins as “output” and “result set”. When the circuit has completed its calclation, it will set the Result bit as the label value and classify it accordingly. Here, we have considered that if the label value is 1, the person is classified as diabetic and otherwise non-diabetic.

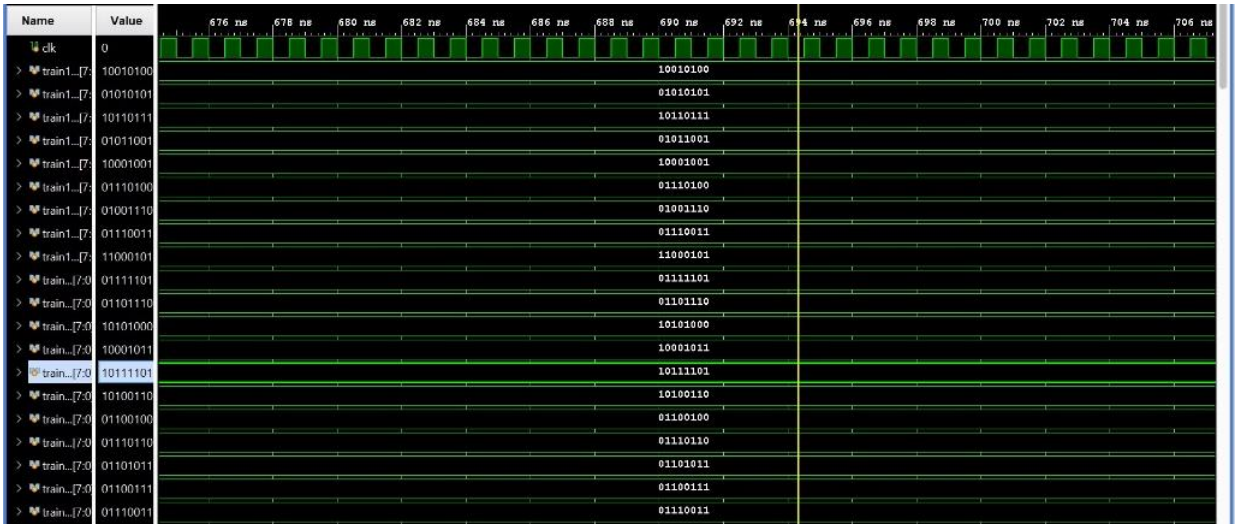

Figure 5.2.1: Timing Diagram of KNN Algorithm depicting all input sets.

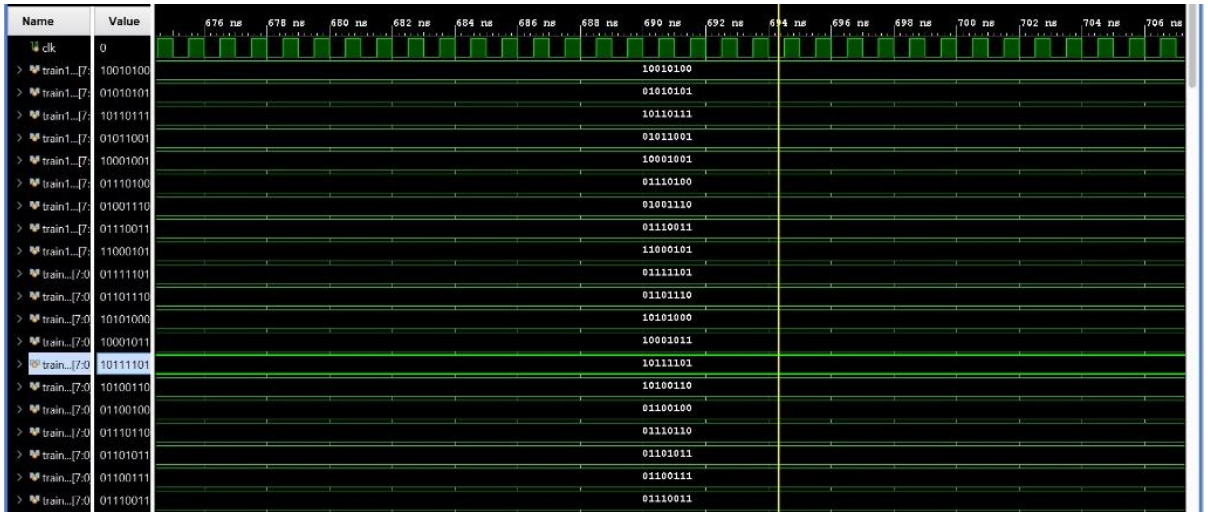

Figure 5.2.2: Timing Diagram depicting the result categorised in the result set by the algorithm.

5.3 Synthesized Design

The Elaborated Design feature of Vivado Design Suite generates the synthesised design. In simple terms, synthesis is the most crucial and essential part of the design methodology. Its focus is to convert our high-level description code into a gate-level netlist, which is an optimised gate-level representation. Its focus is to exhibit meaningful connections between the various strands of data in order to get the desired and expected behaviour during the research phase. Figure 5 depicts the synthesised diagram of the K-NN algorithm.

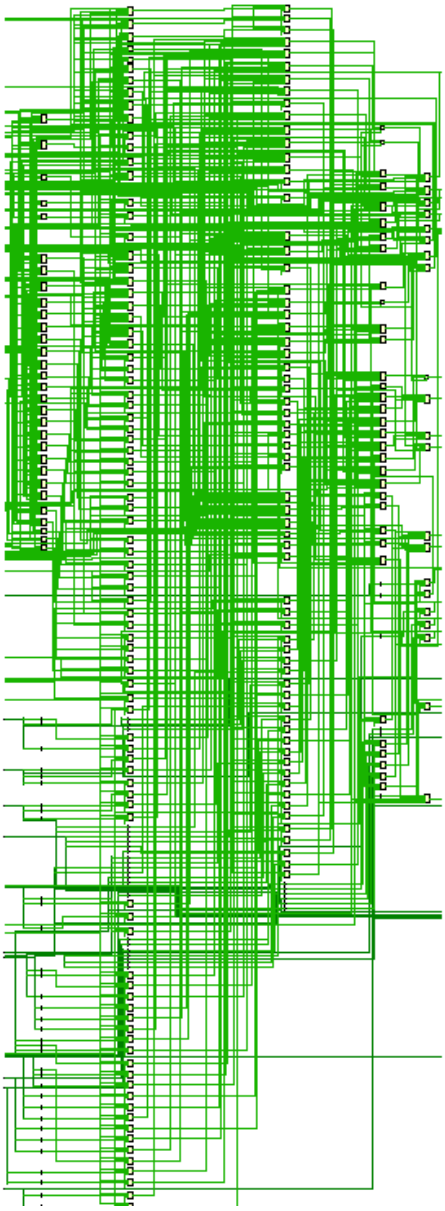

Figure 5.3.1: Synthesized design of the implemented algorithm.

5.4 Comparison and Analysis

As shown below in Table II, the CPU execution Time increases rapidly as the size of inputs increases. But FPGA simulation time increases very little as it is better at parallel computation and its implementation. To prove this parallel computation result we implemented the CPU based model in Python language and consequently the FPGA based model was implemented in Verilog on Xilinx Vivado. Clearly, FPGA-based K-NN is much faster than CPU-based K-NN by a high margin of average of 300,000 times. This result turns out to be a very promising, as it dictates that implanting algorithms on FPGA can save a fortune of time as compared to CPU.

TABLE III: CPU vs FPGA Timing Comparison

No. of Training Data	CPU	FPGA
30	**0.997 msec**	**2.823 nsec**

5.5 Utilization Report

We also examined how our synthesised project's implementation used its resources. Look Up Table (LUT) and Flip Flop (FF) implementation is effective since it uses less space in FPGA, as shown in Table III. The utilisation table below demonstrates how little infrastructure is needed to implement such a challenging algorithm.

TABLE IV: FPGA Utilization Report

Resource	Estimation	Available	Utilization %
LUT	**3934**	**230400**	**1.71**
FF	**2849**	**460800**	**0.62**
BUFG	**1**	**544**	**0.18**

5.6 Accuracy Achieved

In classification problems, accuracy is a metric used to measure the proportion of correct predictions made by a model. It is a statistical measure that compares the number of accurate predictions to the total number of predictions made by the model.

TABLE V: Result Data Set for F-Measure Calculation

Test Data No.	Glucose Level	BP-Diastolic	Age	Result
1	88	58	22	PASS
2	90	68	27	FAIL
3	109	75	60	FAIL
4	111	72	56	PASS
5	180	64	26	PASS
6	62	78	41	FAIL
7	102	76	46	PASS
8	131	0	26	PASS
9	126	88	27	FAIL
10	145	82	57	FAIL
11	136	70	43	PASS
12	117	92	38	PASS
13	158	76	28	PASS
14	102	76	46	FAIL
15	171	110	54	PASS

We took 15 random test samples as shown in Table IV, in order to test the accuracy of the bio informatics classification system implemented. It was found out that our system is approximately 60% accurate which is very promising given the data set.

5.7 F-Measure Calculation

Precision and recall are two very crucial and important factors that are useful when weighing in on the reliability and quality of a machine learning algorithm. This is where F-Measure comes into the picture, where it considers both factors in its calculation and gives us a promising result as to whether a certain algorithm is up to the mark or not. In cases where we have poor precision, it can be balanced off with a very good recall value, and vice versa. Below is an equation that clearly depicts how the F-measure takes both factors into account. The traditional F measure is calculated as follows:

$$FMeasure = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)}$$

Here, the harmonic mean of the two factors has been depicted. This is also known as the F-Score or the F1-Score and is conceivably the most used metric for imbalanced classification problems. An F-Measure value of 0.0 is considered bad, whereas an F-Measure value of 1.0 is considered a perfect score.

Precision calculation based on our result, let precision be denoted as 'P':

$$P = TruePositives \,/\, (TruePositives + FalsePositives)$$

$$P = 9 / (9 + 6)$$

P = 0.60

Recall calculation based on our result, let recall be denoted as 'R':

$$R = TruePositives \,/\, (TruePositives + FalseNegatives)$$

$$R = 9 / (9 + 2)$$

R = 0.818

Now as we have obtained the value of precision and recall, we can calculate the value of F-Measure. Let F-Measure be denoted as 'F-Score':

$$F - Score = (2 * Precision * Recall) \,/\, (Precision + Recall)$$

$$F\text{-}Score = (2 * 0.60 * 0.818) / (0.60 + 0.818)$$

$$F\text{-}Score = (2 * 0.4908) / 1.418$$

F-Score = 0.69224

Now, we have obtained an F-Measure score of ≈ 0.7 which is quite good value given our multi-dimensional data set.

CHAPTER 6: CONCLUSION

In this project, we have utilized HDL to implement the K-NN algorithm on an FPGA, harnessing the raw computational power of the hardware. Our focus was to showcase the applicability of the K-NN classification algorithm on bioinformatics data, specifically in segregating patients into diabetic and non-diabetic categories based on their sugar level, glucose level, and age information. The results demonstrated the remarkable accuracy and efficiency of the method. Notably, the FPGA implementation of the K-NN algorithm exhibited a speed improvement of 300,000 times compared to its CPU counterpart. This significant difference in computational performance can be attributed to the fundamental distinction in how logic is executed between the CPU and FPGA. While the CPU utilizes Fetch/Decode instructions, in FPGA, the logic is hardwired. Furthermore, we observed that the FPGA implementation required less space, in addition to reduced execution time and utilization, while achieving an acceptable F-measure value. Moving forward, this FPGA-based K-NN implementation holds promise for multi-class classification tasks on large datasets, addressing the critical time constraints in the medical industry. By adopting FPGA technology, we can bridge the gap and achieve faster processing. Additionally, we explored the utilization of different sorting techniques, specifically insertion sort, in implementing the same algorithm on the FPGA board. The results showcased the advancements achieved in terms of hardware resource utilization, where we were able to save approximately 1472 hardware components by sparing around 3 nanoseconds, leading to cost reduction.

REFERENCES

[1] B. B. C, A. Deshmukh and A. V. Narasimhadhan, ”Modulation and signal class labelling with active learning and classification using machine learning,” 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/CONECCT55679.2022.9865826.

[2] H. I. Bulbul and O¨ . Unsal, ”Comparison of Classification Techniques used in Machine Learning as Applied on Vocational Guidance Data,” 2011 10th International Conference on Machine Learning and Applications and Workshops, Honolulu, HI, USA, 2011, pp. 298-301, doi: 10.1109/ICMLA.2011.49.

[3] F. G¨urcan, ”Multi-Class Classification of Turkish Texts with Machine Learning Algorithms,” 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2018, pp. 1-5, doi: 10.1109/ISMSIT.2018.8567307.

[4] S. Gandhare and B. Karthikeyan, ”Survey on FPGA Architecture and Recent Applications,” 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019, pp. 1-4, doi: 10.1109/ViTECoN.2019.8899550.

[5] K. Elsaid, M. Safar and M. W. El-Kharashi, ”Optimized FPGA Architecture for Machine Learning Applications using Posit Multipliers,” 2022 International Conference on Microelectronics (ICM), Casablanca, Morocco, 2022, pp. 50-53, doi: 10.1109/ICM56065.2022.10005431.

[6] C. Kardaris, C. Kachris and D. Soudris, ”A high-performance FPGA architecture for Acceleration of SVM Machine Learning Training,” 2022 Panhellenic Conference on Electronics Telecommunications (PACET), Tripolis, Greece, 2022, pp. 1-6, doi: 10.1109/PACET56979.2022.9976358.

[7] W. Lv, H. Huang, W. Tang and T. Chen, ”Research and Application of Intersection Similarity Algorithm Based on KNN Classification Model,” 2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA), Xi’an, China, 2021, pp. 141-144, doi: 10.1109/CAIBDA53561.2021.00037.

[8] 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.

[9] R. S. Latha et al., ”Stock Movement Prediction using KNN Machine Learning Algorithm,” 2022 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2022, pp. 1-5, doi: 10.1109/ICCCI54379.2022.9740781.

[10] S. Briouza, H. Gritli, N. Khraief, S. Belghith and D. Singh, ”EMG Signal Classification for Human Hand Rehabilitation via Two Machine Learning Techniques: kNN and SVM,” 2022 5th International Conference on Advanced Systems and Emergent Technologies (IC ASET), Hammamet, Tunisia, 2022, pp. 412-417, doi: 10.1109/IC ASET53395.2022.9765856.

[11] Majumder, M. Hasanuzzaman, A. Ekbal and S. Saha, ”Machine learning approach for argument extraction of bio-molecular events,” 2012 NATIONAL CONFERENCE ON COMPUTING AND COMMUNICATION SYSTEMS, Durgapur, India, 2012, pp. 1-5, doi: 10.1109/NCCCS. 2012.6413017.

[12] Itagi, S. Krishvadana, K. P. Bharath and M. Rajesh Kumar, ”FPGA Architecture To Enhance Hardware Acceleration for Machine Learning Applications,” 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2021, pp. 1716-1722, doi: 10.1109/ICCMC51019.2021.9418015.

[13] Y. Zou and M. Lin, ”GridGAS: An I/O-Efficient Heterogeneous FPGA+CPU Computing Platform for Very Large-Scale Graph Analytics,” 2018 International Conference on Field-Programmable Technology (FPT), Naha, Japan, 2018, pp. 246-249, doi: 10.1109/FPT.2018.00045.