IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

FPGA-BASED IMPLEMENTATION AND PERFORMANCE ANALYSIS OF K-NEAREST NEIGHBOR ALGORITHM

Marneni Sravya Treasa[1], DR. D. Karunakar Reddy[2]

[1]Post Graduate Student, M. Tech (Embedded Systems), [2]Associate Professor, Department of Electronics & Communication Engineering, JNTUH University College of Engineering, Science & Technology Hyderabad

Abstract: ***K-Nearest Neighbors (KNN) algorithm puts the data points into categories that belong to the class most represented among their closest neighbors, measured by a distance metric, such as Euclidean distance. It is a popular method because of its simplicity and effectiveness but can be very time-consuming when dealing with large and high-dimensional datasets. This paper is about speeding up KNN with FPGAs to facilitate processing of high-dimensional data for classification. Among various datasets, the MNIST handwritten digit dataset is chosen for the study reasoned by its benchmark status in classification tasks. The KNN implementation done in software serves as a baseline and sets the bar for accuracy. A parallel hardware architecture is created and its model is generated in hardware description language to enable faster distance computation and neighbor selection. Based on simulation, the projections indicate a considerable reduction of latency and increase of speed compared to software while accuracy of classification is also maintained.***

Keywords **- K-Nearest Neighbors, KNN, FPGA, Hardware Acceleration, Parallel Processing, VHDL, High-Dimensional Data, MNIST Dataset.**

I. INTRODUCTION

K-Nearest Neighbors (KNN) is one of the simplest but still effective classification algorithms which is broadly applied in machine learning domain. The class of the test sample is determined by KNN based on the majority class of the nearest neighbors that it finds by using a distance metric such as Euclidean distance. However, the prediction of KNN is a heavy computational process that necessitates distance calculations and sorting over large, high-dimensional datasets and, therefore, it is not suitable for real-time applications. Nevertheless, the Field-Programmable Gate Arrays (FPGAs) can serve as an excellent hardware platform for such computation, where the parallelism and pipelining can be exploited for accelerating the process. An FPGA-accelerated KNN architecture has been put forward for the identification of high-dimensional classification tasks, taking the MNIST handwritten digits dataset as a test case. First of all, a software KNN implementation is done as a baseline, then the core algorithm is hardware designed in VHDL to speed up the basic operations. The results of the simulation for classification latency and throughput indicate the performance gains of hardware over software methods, thus, the acceleration via FPGA for energy-efficient KNN classification can be realized.

II. BACKGROUND AND RELATED WORK

2.1 K-Nearest Neighbors (KNN) Algorithm

K-Nearest Neighbors (KNN) algorithm is a type of supervised machine learning which can do both classification and regression. In classification, the new piece of data gets its class by asking the neighbors what class it should be. In other words, it will belong to the one among K closest neighbors which has the highest number of common classes. Usually, Euclidean distance is used for "distance" measurement, which is a standard straight-line distance between two points in Euclidean space. The setting of K is very important and normally the value of K is found out through cross-validation. Although it takes up less space and is fairly reliable, the KNN

computational cost at the time of inference could be quite large for big data, since it is necessary to keep the entire training set and do long distance calculations for every new prediction.

2.2 MNIST Dataset

MNIST (Modified National Institute of Standards and Technology) dataset is a collection of images of digits written by hand and is one of the standard benchmark tasks for various image processing systems. The dataset is 70,000 grayscale images (60,000 for training, 10,000 for testing) of handwritten numbers from 0 to 9. Every image with a size of 28x28 pixels is turned into a 784-dimensional vector, which makes it a high-dimensional dataset and thus very suitable for benchmarking classification algorithms and their hardware implementations.

2.3 FPGA Architecture and Advantages

Field-Programmable Gate Arrays (FPGAs) are semiconductor devices that are made up of a matrix of configurable logic blocks (CLBs) that are linked together by programmable interconnects. Any digital circuit can be implemented if users configure these blocks and interconnects according to the required digital circuit. Coupled with the ability of achieving high levels of parallelism and pipelining, this reconfigurability makes FPGAs the most suitable devices to accelerate algorithms that are computationally intensive. Unlike general-purpose CPUs that carry out instructions one after another, FPGAs can execute several operations at the same time which results in higher throughput and lower latency to specific tasks to a great extent. They also are more power efficient than GPUs for a lot of embedded applications because of their customization.

2.4 Related work

Various researchers have worked on the implementation of K-Nearest Neighbors (KNN) on FPGA to speed up the classification process. Their main goal has been to manage the KNN's computational complexity effectively. Through the paper, “Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study” the authors have shown how KNN can be accelerated in real situations using Intel FPGAs. The key change in this research was the adoption of both memory management and parallel processing, which allowed them to have a successful real-time application. The work “Optimized k-Nearest neighbors search implementation on resource-constrained FPGA” reflects the authors' idea of introducing clever ways of cutting the memory needs and simplifying distance calculations and selection parts to make the logic easier for FPGAs of limited resources. This paper can be considered a stepping stone toward designing the hardware that efficiently accelerates KNN. Moreover, the paper “FPGA-Based Acceleration of K-Nearest Neighbor Algorithm on Fully Connected Neural Networks” reveals the combined use of pipelined architectures and fixed-point arithmetic as the hybrid method to facilitate one of the highest throughputs on FPGA platforms without consuming a large amount of resources.

III. METHODOLOGY

3.1 Software Implementation

3.1.1 Dataset: The MNIST dataset with 70,000 gray-level images of handwritten digits, each being 28x28 pixels in size and compressed into a 784-dimensional vector is used. This dataset is a real-world challenge for classification algorithms by virtue of its high dimensionality and variability in handwriting styles.

3.1.2 Preprocessing: Input images are normalized by rescaling pixel intensities from the native 0-255 range to a floating-point range between 0 and 1. The dataset is split into training (75%) and testing (25%) subsets by a randomized split to prevent bias.

3.1.3 KNN Model: The K-Nearest Neighbors classifier is applied with the help of the scikit-learn library in Python. Euclidean distance measure is utilized to calculate similarity between test samples and training points. A neighbor number

K=3

K=3 is optimal for balancing overfitting and underfitting in this classification problem.

3.1.4 Performance Metrics: Core metrics captured are classification accuracy, train time, and prediction time on the test set of 17,500 samples. These offer quantitative metrics to evaluate the advantage of hardware acceleration.

3.2 FPGA Hardware Design:

3.2.1 Architectural Overview:

The accelerator design has the following:

Distance Calculation Units (DCUs): Several parallel DCUs do Euclidean distance computations between a single test image vector and multiple training samples concurrently. To save FPGA resources, calculations are done in fixed-point arithmetic instead of floating-point.

K-Nearest Neighbor Selector (KNS): Comparator network or min-k finder module is responsible for tracking and updating the list of the K smallest distances and their respective training labels at all times. It facilitates efficient updating without full sorting to minimize computational delay.

On-Chip Memory (BRAM): Block RAMs contain the training set, with low-latency simultaneous access supported by DCUs. Memory organization and dual-port BRAM use support fast data reading.

Control Logic: A finite state machine (FSM) controls data flow synchronization between modules, command sequencing, timing control, and interface handshaking with the processing system.

3.2.2 Data Transfer and Interface: Communication between the Programmable Logic (FPGA fabric) and Processing System (host CPU) exploits the AXI4 bus protocol, with Direct Memory Access (DMA) engines providing support for bulk data transfer. Processor overhead during data transfer is minimized through this architecture, enabling easy integration into embedded systems.

3.2.3 Fixed-Point Arithmetic Considerations: In order to reduce the area and latency overheads of floating-point operations on FPGAs, distances are calculated based on 16 to 18-bit fixed-point numbers. This preserves classification accuracy but reduces resource usage considerably.

3.3 Performance Assessment:

Verification: Functional testbenches for distance calculation, neighbor find, and overall integration emulate the accelerator's functionality based on representative MNIST examples. Correctness is checked by comparing hardware-predicted distances and class results with software outputs.

3.3.1Metrics Evaluated:

Classification Accuracy: Comparison of predicted labels with ground truth on test samples.

Latency: One-image classification time measured in clock cycles and converted based on assumptions of synthesized clock frequency.

Throughput: Number of images classified per second, derived from latency and pipelining level.

Resource Utilization: Estimated use of FPGA resources such as LUTs, flip-flops, block RAMs, and DSP slices using design synthesis reports.

Comparison to Software Baseline: Software benchmarks give a reference point, noting speed and latency gains due to hardware acceleration. Predicted speed improvements and latency savings measure the advantage of FPGA implementation for the KNN classifier.

IV. ALGORITHM

4. K-Nearest Neighbors Algorithm

The K-Nearest Neighbors (KNN) algorithm is a method of supervised classification that is applied to a situation where it is needed to figure out which class a new data sample belongs to by looking up the closest instances to the new sample in the training set.
KNN algorithm main steps:

4.1 Overview of the Algorithm:

1. **Input:** The testing sample is x and the training set is {(xi,yi)} where xi denotes the feature vector of the ith training sample and yi denotes its class label.
2. **Distance Calculation:** Calculate the distances between the testing point and each training point. A very common distance metric is the Euclidean distance which can be represented as

$$d(x,x_i) = \sqrt{\sum_{j=1}^{D}(x_j - x_{ij})^2}$$

   where D is the number of features.
3. **Nearest Neighbors Selection:** Select the K training samples which are closest to the test sample based on the computed distances.
4. **Classification:** The test sample is assigned the class label that occurs most frequently among the k nearest neighbors.

V. RESULTS AND DISCUSSION

5.1 Software Implementation Results

The baseline software KNN classifier was evaluated using the MNIST dataset with 784 features per sample. Key performance metrics are summarized in Table 1.

Metric	Without PCA	With PCA (64 features)
Classification Accuracy	97.23%	96.05%
Training Time (seconds)	0.85	0.92
Prediction Time (seconds)	321.45	45.21
Average Prediction Time (ms)	18.37	2.58

5.1.1 Discussion: The KNN classifier software accomplished a remarkable 97.23% accuracy on the MNIST test set, thus proving the effectiveness of the algorithm. Nevertheless, the prediction time of 321.45 seconds for 17,500 test images is quite alarming, which means that the average time for one image is 18.37 milliseconds. Therefore, such a long prediction time period basically illuminates the major computational bottleneck of KNN, particularly for regular general-purpose processors, and this is due to the fact that it results from distance calculations (17,500 test images * 52,500 training images * 784 pixels per computation) and sorting for the K-nearest neighbors. When PCA was used for dimensionality reduction during the prediction process (down to 64 components), the time was significantly reduced to 45.21 seconds; however, the accuracy dropped to 96.05%. The point of this example is that there is always a trade-off between accuracy and available hardware resources. It is planned in the FPGA to keep the 784-pixel input to achieve the highest accuracy possible, and will only use PCM in the areas where the resources are few.

5.2 FPGA Hardware Simulation Results

In order to check the functionality and measure the performance, the FPGA design has been simulated with hardware description language testbenches. The following were obtained from the simulations:

1. Matching software outputs as accurate Euclidean distance computation.
2. Comparator networks efficient K-nearest neighbor selection.
3. Microseconds classification latency of the order.
4. Parallelism and pipelining high throughput.
5. Resource estimates compatible with the capabilities of mid-range FPGA devices.

5.3 Comparative analysis

Metric	Software Baseline	FPGA Acceleration	Speed-up Factor
Classification Accuracy	97.23%	~97.2%	Comparable
Latency per Classification	~18.37 ms	10 - 50 microseconds	360x - 1800x
Throughput	~54 images per second	20,000 - 100,000 images/sec	370x - 1850x

Metric	Software Baseline	FPGA Acceleration	Speed-up Factor
Power Efficiency	Lower	Higher	Significant
Flexibility	High	Low	-

5.3.1 Discussion: The comparative assessment distinctly marks the trade-offs. The software application presents great flexibility and simplicity in development but sequential processing is basically a constraint that limits performance for large KNN. On the other hand, the FPGA layout, even though taking a long development time and being less flexible, assures drastic enhancement in classification time and throughput over many times. Thus, FPGAs are a good option for real-time, low-latency applications where the algorithm is already set. The resource usage anticipated is under the limit of the resource requirements of commercial FPGAs, which is positive for the design's practical feasibility.

VI. CONCLUSION

This paper introduced the plan and assessment of a K-Nearest Neighbors classifier, which was realized on an FPGA through simulation and directed towards high-dimensional data classification. A trustworthy Python-based software implementation with the MNIST dataset served as the performance baseline. A parallel, pipelined hardware architecture was devised in VHDL to speed up the distance calculations and neighbor selection that are very resource-demanding. The simulation results indicated that the FPGA design reaches the same accuracy as that of software but with a reduction in time for classification of up to 1000× and consequently, a hundred-fold increase in throughput, thus, it is ready for real-time applications. The design makes good use of fixed-point arithmetic and on-chip memory which is an indication of its suitability for mid-range FPGAs.

VI. REFERENCES

[1] M. Kadiyala, A. N., & K. M. (2022). Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study. *Computers and Electrical Engineering*, *103*, 108343.
[2] N. H. S. Ahmed, S. M. I., & E. S. (2024). Optimized k-Nearest neighbors search implementation on resource-constrained FPGA. *Microprocessors and Microsystems*, *112*, 1050