ISSN: 2584-0495 Vol. 2, Issue 4, pp.714- 721

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

Design of an Efficient Selective Stochastic Model for Delay-Aware Digital Classification using Nexys A7 FPGA

Sachin D. Kohale, Trapti Sharma

Cite as: Kohale, S. D., & Sharma, T. (2024). Design of an Efficient Selective Stochastic Model for Delay-Aware Digital Classification using Nexys A7 FPGA. International Journal of Microsystems and IoT, 2(4), 714-721. https://doi.org/10.5281/zenodo.11221335

Published online: 22 April 2024

DOI: https://doi.org/10.5281/zenodo.11221335

International Journal of Microsystems and IoT
Vol. 2, Issue 4, pp. 714-721; DOI: https://doi.org/10.5281/zenodo.11221335

Design of an Efficient Selective Stochastic Model for Delay-Aware Digital Classification using Nexys A7 FPGA

Sachin D. Kohale[1], Trapti Sharma[2]

[1]Department of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan
[2]School of Computing science and Engineering, VIT Bhopal University, Bhopal-Indore Highway, Kothrikalan, India

ABSTRACT
As the dimension of the speech recognition dataset increases while performing the classification of the same, results in an increment of several features as well as its classification delay. Therefore, to reduce the delay required during the classification, this paper proposes the Selective Stochastic Model for delay-aware Digital Classification (SSMDC) which consists of a Gold Code-based Intelligent Pseudo Stochastic Number Generator (GCI PSNG) followed by Message Digest 5 (MD5) model. The role of the PSNG model is to generate efficient sample indices for classification into different categories followed by a low-complexity truncated MD5 model for the generation of sample-selection indices. These indices are sequentially given to feature memory blocks for the selection of samples, which are classified by the underlying classifiers. The SSMDC model performance is tested on auditory Mel-Frequency Cepstral Coefficients (MFCC) component features with k-Nearest Neighbors (k-NN) classifier but can be extended to any other application with minimal configurations, thereby making it useful for a wide variety of real-time scenarios. The proposed model is implemented using Verilog HDL on Xilinx Vivado Design Suite 2023.1, and simulation results are obtained considering the device xc7a100tcsg324-1 Nexys A7 FPGA. Also, with SSMDC, the proposed model reduces the delay of classification by 27.35% when compared with other classification models.

KEYWORDS
Message Digest 5 (MD5); Mel Frequency Cepstral Coefficients (MFCC); Pseudo Stochastic Number Generator (PSNG); Speech Classification.

1. INTRODUCTION

In a microarray dataset, the rows and columns indicate the number of entries and the dimensions to be evaluated. The work required to determine what is most important grows proportionally with the number of dimensions in a dataset. The hardest part of big feature space datasets (i.e., microarray datasets) is projecting a vast feature space into a smaller one while preserving as much information as feasible [1]-[5]. Preprocessing such massive volumes of data may be costly since each record in a microarray dataset may include up to 450,000 attributes. Furthermore, data sparsity and the quantity of relevant data diminish as the dimensionality of a dataset rises exponentially. Datasets with a wide feature space and few records or observations tend to exhibit this more frequently. Because of the rapid change in available data, a model built using overfitted data runs the risk of producing significant classification mistakes. Problems might be further exacerbated by the presence of noise. Noisy data is information that either contains errors or varies significantly from the expected value. It is also worth noticing that the machine learning algorithm's performance may hampered if fed with inaccurate or otherwise flawed data [6]. To reduce the model's complexity and improve its machine-learning performance, noisy input must be filtered out. The reasons to select suitable feature selection methods include huge dataset dimensions, high cost of computing, the presence of irrelevant data, the risk of overfitting, and manual feature extraction. The accessible data is preserved while a vast feature space is narrowed down to its most crucial characteristics using the feature selection method. By focusing on the features that matter most, we may get accurate classifications in less time [7]. Task-relevant data is significant in determining the efficiency of the data mining approach. Thus, there is a linear relationship established between the input and output. Learning across a large feature space would be difficult, if a data mining approach is applied to irrelevant data, such as data containing redundant information or noise. As data mining grows in popularity, more researchers are focusing on feature selection because of its importance [8]. Feature selection is the process of narrowing down a large feature space, such as a dataset, to a manageable subset that may be used in model building. By filtering out irrelevant information about the data, feature selection helps in the fast and efficient functioning of the data mining process [9]. Filters, wrappers, and embedding techniques are a few examples of feature-selection methods.

As the filtering method does not use any pre-defined pattern it selects the most distinguishing and relevant features of a product. The filter assessment methods include relief, information gain, and chi-square tests. The "wrapper approach" is a method for selecting features that consider the classification algorithm used to make the ultimate decision [10]-[14].

The application of the wrapper method for the classification algorithm may generate a complete and well-tuned set of features to be used for several families of algorithms, such as Principal Component Analysis (PCA), Particle Swarm Optimization (PSO), etc. The model fitting in embedded techniques needs feature selection, with the resulting methodology choice depending on the embedded approach's target model. Some of the examples of embedded algorithms include Random forests, singular value decompositions, and accurate Bayes approximations. Since external methods do not consider all possible combinations of characteristics, it has been shown that filter and wrapper techniques are insufficient as a solution [11]. Selecting many candidate feature subsets allows a feature selection method to cover more ground and learn more about the feature space. There is no connection between search terms and metrics used for assessment. Searches may be conducted in one of three ways: exhaustively, sequentially, or at random [15]

.

An in-depth search, often called an exhaustive search provides more accurate results. Nonetheless, when applied to a big feature space, W(2n) (W stands for exhaustive search) becomes computationally expensive (n). Since the best answer or subset relies on the rankings generated by previous methods, it cannot be found using sequential search. The computational cost is still lower than that of W. Also, the best results are not always guaranteed by random search, which uses a sequential search technique and picks a random part of the feature space, to begin with. After picking an option, the process begins again, creating a new random group [16]-[20]. Most often researchers use feature selection methods when there are many features but few samples to work with (or data points). The two classic examples of feature selection, which frequently involve thousands of features and tens to hundreds of samples, are text analysis and DNA microarray data [14]. The capability to limit the available features to those that are most relevant for the present task may improve the classification's effectiveness and efficiency. Restricting model overfitting to a confined feature space also enhances the generalization capacity of classifiers. To prevent it from becoming distracted by unrelated information, the model or classifier is trained using the most pertinent features of the data. Many of them could be useful in treating various artifacts. The disease profiling may improve in terms of classification and sickness prediction accuracy by using even a small gene microarray dataset based on a feature set. To reduce the massive amounts of information different methods have been developed often seen in gene microarray datasets. Improving illness classification and forecasting is the main goal. Feature selection can be used to limit the feature space to a more manageable subset, which will further enhance the responsiveness of the model. Currently, images, radiological reports, prescription profiles, signals, patient histories, pathology reports, and treatment records are all contained in the organized and semi-structured databases of healthcare institutions. This dataset has a wide feature space, heterogeneity, complexity, ambiguity, and noise. As a result, data mining offers a wide range of approaches that offer appropriate and correct feature selection methods and classification algorithms, which may help unearth previously buried patterns and information in such datasets. The clinical researchers and practitioners may benefit widely from these patterns and data as well as more wise decisions for early disease detection can be taken.

This establishes a formal connection between the number of feature sets and computational complexity, and it explains why state-of-the-art classification and feature selection models like the Support Vector Machine (SVM), k-nearest Neighbor (k-NN), and others need N-clock cycles. When there are more attributes to consider, more time and resources are needed to train these classifiers. To address these issues, the researchers have reported a variety of feature reduction strategies [26], but their implementation is costly and time-consuming preventing them from using moderately advanced Field Programmable Gate Arrays (FPGAs). Furthermore, feature reduction is quite energy-intensive, which restricts the scalability of such methods. To overcome these limitations, a further part of this paper proposed a powerful stochastic model for the delay-aware digital classification of FPGAs. Also, we analyze the model's efficacy in a variety of scenarios by comparing in terms of computing metrics such as latency(delay), energy(power), and area (Look Up Tables – LUTs) to those of existing techniques in [1,5,10,27]. The paper is organized as follows: Section II describes the details of existing works. The proposed SSMDC model which is an integration of PSNG [28-29] and MD5 details are covered in Section III. Section IV covers the results and comparison with the existing work. Finally, the article is summarized.

2. PROPOSED MODEL

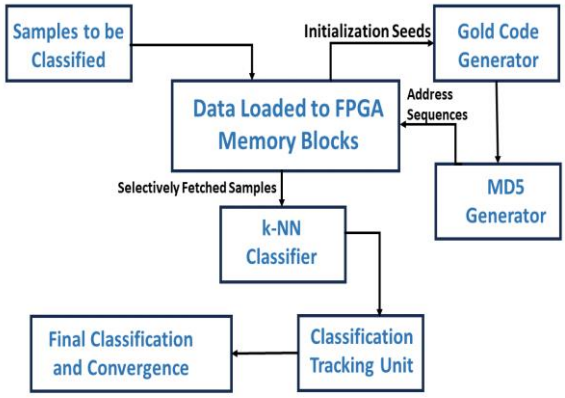

Fig. 1 Proposed Intelligent Stochastic Classification Process

Figure 1 proposes the design of a Gold Code-based Intelligent Pseudo Stochastic Number Generator (GCI PSNG). It generates sample-selection indices using a Message-Digest 5 (MD5) Model with reduced complexity. These indices are sequentially provided

to feature memory blocks to select samples for classification by the underlying classifiers. To ensure non-repetitive features, an in-memory cache of computed numbers is maintained, which helps to reduce index calculations. To maintain track of sample-to-class mapping counts for individual classes, the proposed design converges when counts for any class exceed fifty percent of the total number of samples. To perform this task, the data is initially loaded into the memory, which consists of selected feature samples, and their individual classes. This data is segregated into training, testing & validation samples. The training samples are used to identify classes of testing and validation samples. For each of the testing and validation samples, an intelligent and stochastic set of indices is initially generated via gold sequences. The Gold sequence generator is a type of linear feedback shift register (LFSR) that uses two shift registers and an XOR operation to generate a sequence of pseudo-random bits. The sequence generator output is a binary sequence with good statistical properties and can be used to generate stochastic numbers. A binary fraction expansion method is used to convert binary output sequence into decimal value lies in the range (0,255) which is employed in stochastic number generation and utilized in the gold sequence generator.

Alternatively, we use the output sequence directly as a stochastic sequence by treating each 0 as a "heads" outcome and each 1 as a "tails" outcome via the following operations,

- Initialize two seed values seed_1 ,and seed_2 *as*

$$seed_1 = N_t/2 \quad (1)$$

$$seed_2 = N_t/4 \quad (2)$$

where N_t represents the total number of samples available in the training sets.

- Now, define two binary polynomials P_1 and P_2 as

$$P_1 = x_1^4 + x_1 + 1 \quad (3)$$

$$P_2 = x_2^8 + x_2^4 + x_2^2 + 1 \quad (4)$$

In both these equations, the output will be obtained via low-complexity shifting operations.

- Extract the Least Significant Bits (LSBs) from both seeds using

$$LSB_i = seed_i >> 7 \quad (5)$$

where, $i \in (1,2)$ for the two seed value sets.

- Generate an output bit $Output_i$ as

$$Output_i = LSB_1 \; XOR \; LSB_2 \quad (6)$$

Shift the registers by one bit using the generator polynomials, using following;

$$seed(new)_i = P_i \quad (7)$$

to calculate the new value sets.

- Update the shift registers with the new values using

$$seed(new)_i = (seed(new)_i + seed(old)_i) >> 1 \quad (8)$$

Based on these operations, the $Output$ value is returned for an individual set of bits. A set of 8 bits is aggregated to form a stochastic byte, which is used by the Message-Digest 5 (MD5) [3] Model for the calculation of next & consecutive memory index sets. An input message is fed into the cryptographic hash function MD5, which outputs a fixed-size, 128-bit (16-byte) hash result. The 16 bytes are often non-repetitive and can be used as classification indices.

To perform this task, initially, a set of auxiliary functions are defined using the following

$$F(X,Y,Z) = (X \,\&\, Y) \,|\, (\bar{X} \,\&\, Z) \quad (9)$$

$$G(X,Y,Z) = (X \,\&\, Z) \,|\, (Y \,\&\, \bar{Z}) \quad (10)$$

$$H(X,Y,Z) = X \,\hat{}\, Y \,\hat{}\, Z \quad (11)$$

$$I(X,Y,Z) = Y \,\hat{}\, (X \,\&\, \bar{Z}) \quad (12)$$

Using these functions, a set of rounds $Round_i(A,B,C,D,X,S,T)$ are defined as

$$Round_i(A,B,C,D,X,S,T) = B + LR((A + Funi(B,C,D) + X + T), S) \quad (13)$$

where, LR represents left rotation operations, while $i \in (1,4)$, and $Fun_i \in (F, G, H, I)$ for individual set of rounds. To setup the MD5 constants of 32-bit words each denoted as A, B, C, and D

$$A = N_t * 2 \quad (14)$$

$$B = N_t * 4 \quad (15)$$

$$C = N_t * 8 \quad (16)$$

$$D = N_t * 16 \quad (17)$$

Pre-process the generated stochastic number by appending a single "1" bit and padding it with zeros. Now generate updated constant values *A(New), B(New), C(New), and D(New)* as

$$A(New) = Round_i(A,B,C,D,SI[j],7,N_{test}) \quad (18)$$

$$B(New) = Round_i(A,B,C,D,SI[j+1],12,N_{test}) \quad (19)$$

$$C(New) = Round_i(A,B,C,D,SI[j+2],17,N_{test}) \quad (20)$$

$$D(New) = Round_i(A,B,C,D,SI[j+3],22,N_{test}) \quad (21)$$

where N_{test} are total number of testing samples

j is the input bit number

SI is the stochastically generated number

i ∈ (1,4), representing 4 different rounds.

This process is repeated for each round, and new values of constants *A(New), B(New), C(New), and D(New)* are estimated as

$$A(New) = A(New) + A(Old) \% N(Prime) \quad (22)$$

where, *N(Prime)* is a large prime number, used to limit the value of *A* to 32 bits.

$$B(New) = B(New) + B(Old) \% N(Prime) \quad (23)$$

$$C(New) = C(New) + C(Old) \% N(Prime) \quad (24)$$

$$D(New) = D(New) + D(Old) \% N(Prime) \quad (25)$$

Hash is formed after processing each bit of stochastically generated input numbers by concatenating the constants *A(New), B(New), C(New), and D(New)* as

$$Hash = A(New), B(New), C(New), D(New) \quad (26)$$

Individual bytes (or words) are extracted from this hash value and checked for uniqueness. This is done as per the following process,

- Create an N_t bit long register, and set all its bits to '0'
- Bits are set to '1', depending upon which set of training samples have been used for the classification operations.
- If a bit is found to be '0', then its corresponding hash-byte is given to the k-Nearest Neighbors (k-NN) classifier.
- This ensures that there are a minimum number of comparisons to obtain the final class.

Each of the pseudo stochastic number generator operations, MD5 operations, and selective indices operations are simple to deploy on FPGAs as they contain only shifting & logical operations. Extending to this simplicity, the k-nearest Neighbors (k-NN) Model is used here for the classification task.

3. COMPARATIVE PERFORMANCE ANALYSIS

In this section, the proposed SSMDC model is implemented using Verilog HDL, and all the simulations are performed using Xilinx Vivado Design Suite 2023.1. For the result analysis, a set of 1300 different samples were recorded using audacity at 16 kHz sampling frequency and stored in the dataset. These speech keyword samples are categorized as 665 samples of Indian-Male, 127 of Indian-Female, 127 of Taiwanese-Male, 127 of Uganda-Male, 127 of Indonesian Female, and 127 of Indonesian Male respectively. Mel Frequency Cepstral Component (MFCC) features were extracted using MATLAB for "Forward", "Reverse", "Left", "Right", "Start", and "Stop" speech keywords and the converted samples were classified using k-NN classifier to evaluate its performance [35]-[40]. Based on this, the average delay needed for classification was estimated and compared with different models by normalizing the results via. the following equation

$$P(Norm) = P(Obtained) / Maximum\_Samples \quad (27)$$

where, *P(Norm)* and *P(Obtained)* are the values of the normalized and obtained parameters, and *Maximum_Samples* represents the maximum number of samples for which the model is evaluated in the reference text for different input sets. In a linear classifier, the normalized values are calculated by performing the division operation of the obtained parameters with the maximum number of samples. Based on this strategy, the area needed for deploying the design on standard FPGA is evaluated and compared with existing models in Table 1.

Table. 1 Comparison of Normalized Area Utilization for Different Models

Classification Models	Look Up Table (LUTs)
Deep Neural Network (DNN) [1]	14964
Convolutional Neural Network (CNN) [5]	63968
Convolutional Neural Network (CNN) [10]	43675
RF Signal Classification [27]	158435
SSMDC Proposed	**13.98**

According to the Table 1 results, the proposed model shows an improvement in terms of LUTs as compared to other models such as Deep Neural Network [1], Convolutional Neural Network [5,10], and RF Signal Classification [27]. This is mainly due to the sample-by-sample processing of speech inputs processed by the integration of PSNG [28-29] and MD5 components in the proposed SSMDC model.

The comparison of normalized area utilization in terms of LUTs required for the proposed SSMDC model with and without optimization for the k-NN classifier has been illustrated in Table 2.

Table. 2 Comparison of Normalized Area Utilization with and without Optimization

No. of Test Samples (NTS)	LUT required without SSMDC	LUT required with SSMDC
100	1351	1474
200	1829	2500
300	4100	3778
400	3456	14732
500	6754	6734
600	8127	7907
700	6827	8353
800	7975	10573
900	10603	10267
1000	10132	9165
1100	9358	13415
1200	13893	13186
1300	10610	12248
Average LUT required per sample	**10.91**	**13.98**

Similarly, the power consumed while simulating the proposed SSMDC model, the following results are obtained as shown in Table 3. As it is clearly observed from the tabular results, the power consumption improvement for the proposed model is 96% as compared to the least reported values of the DNN [1] model. Likewise, the computation results for the power consumption with and without optimization are demonstrated in Table 4.

Table. 3 Comparison of Power Consumption for Different Models

Classification Models	Power (in watts)
Deep Neural Network (DNN) [1]	0.74
Convolutional Neural Network (CNN) [5]	2.41
Convolutional Neural Network (CNN) [10]	2.41
RF Signal Classification [27]	1.152
SSMDC Proposed	**0.025**

Table. 4 Comparison of Power Consumption with and without Optimization

No. of Test Samples (NTS)	Power Consumed (in watts) without SSMDC	Power Consumed (in watts) with SSMDC
100	0.316	1.564
200	0.304	1.730
300	11.727	15.554
400	0.268	22.399
500	31.470	16.273
600	18.569	1.625
700	8.880	20.597
800	6.495	37.346
900	12.147	21.034
1000	25.860	18.068
1100	19.022	37.853
1200	19.112	9.150
1300	28.588	2.078

Average LUT required per sample	**0.019**	**0.025**

Additionally, the suggested SSMDC model's delay computation results are computed, and Table 5 displays a comparison with previous studies. According to the findings, the suggested model classifies speech keyword signals more quickly than other models that are currently in use [1,5,10,27].

Table. 5 Comparison of Normalized Delay for Different Models

Classification Models	Delay (In microseconds)
Deep Neural Network (DNN) [1]	15.3
Convolutional Neural Network (CNN) [5]	70,000
Convolutional Neural Network (CNN) [10]	71,000
RF Signal Classification [27]	24
SSMDC Proposed	**0.03648**

The proposed SSMDC model can be deployed for multiple real-time scenarios and can be used to update the performance of existing models [1,5,10,27] that perform comparisons of full-length datasets and samples for the purpose of classifications. This analysis is illustrated in Table 6, where the delay is compared with and without optimization for a standard k-NN classifier.

Table. 6 Comparison of Delay with and without Optimization

No. of Test Samples (NTS)	Delay (in microseconds) without SSMDC	Delay (in microseconds) with SSMDC
100	0.99	0.82
200	7.56	3.72
300	10.36	9.82
400	16.78	13.54
500	25.45	16.52
600	29.67	21.25
700	40.50	27.97
800	43.40	31.30
900	50.50	36.97
1000	60.40	45.43
1100	75.90	53.77
1200	85.40	58.63
1300	89.50	63.79
Average Delay required per sample	**0.050936**	**0.03648**

The analysis of Table 6 results indicates that the proposed SSMDC model shows a classification delay improvement of 27.35% as compared to without SSMDC (without optimization) model. Further, this optimized classification (SSMDC model) delay results can assist in deploying these models for different real-time high-speed use cases.

Figure 2 represents the comparative analysis of various performance parameters such as Area Utilization (in 2(A), Power Consumption (in 2(B)), and Classification delay (in 2(C)) of the proposed SSMDC model with other models [1,5,10,27].

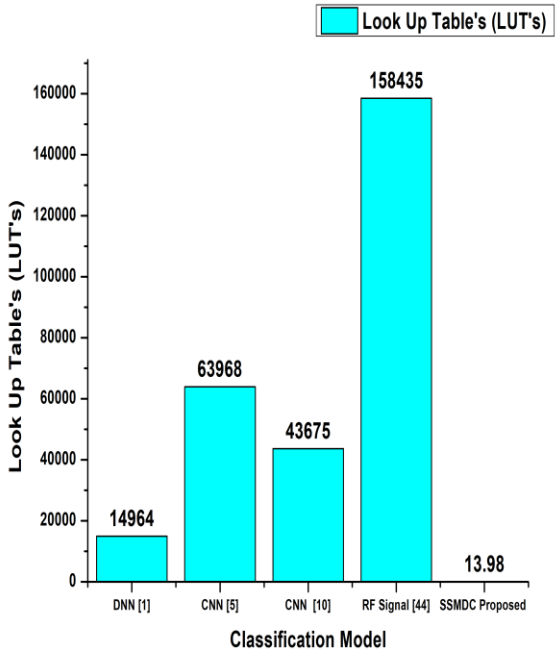

Fig. 2 (A) Comparison of Area Utilization for various models.

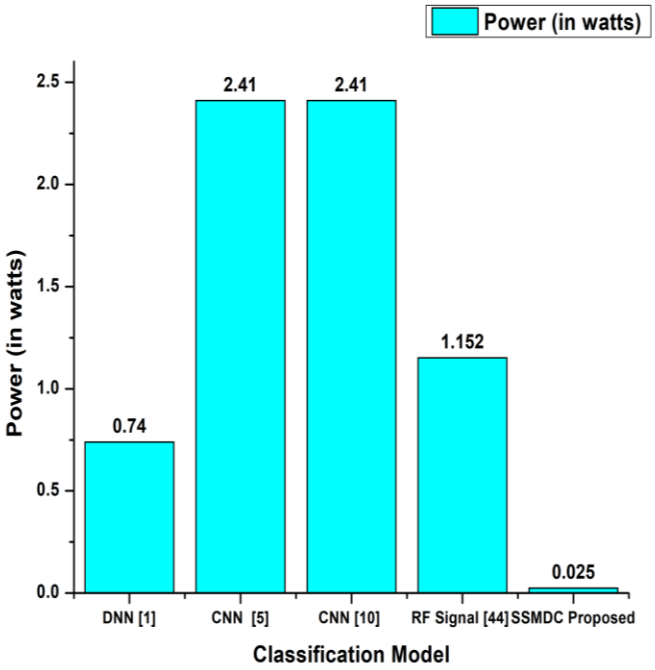

Fig. 2 (B) Comparison of Power consumption for various models.

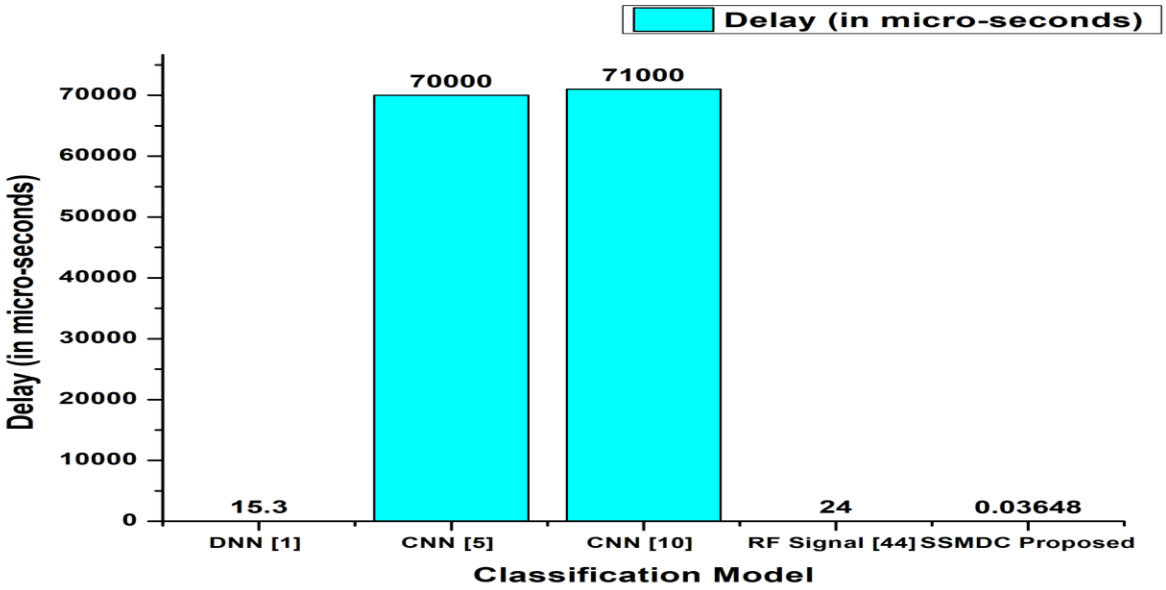

Fig. 2 (C) Comparison of Classification Delay for various models.

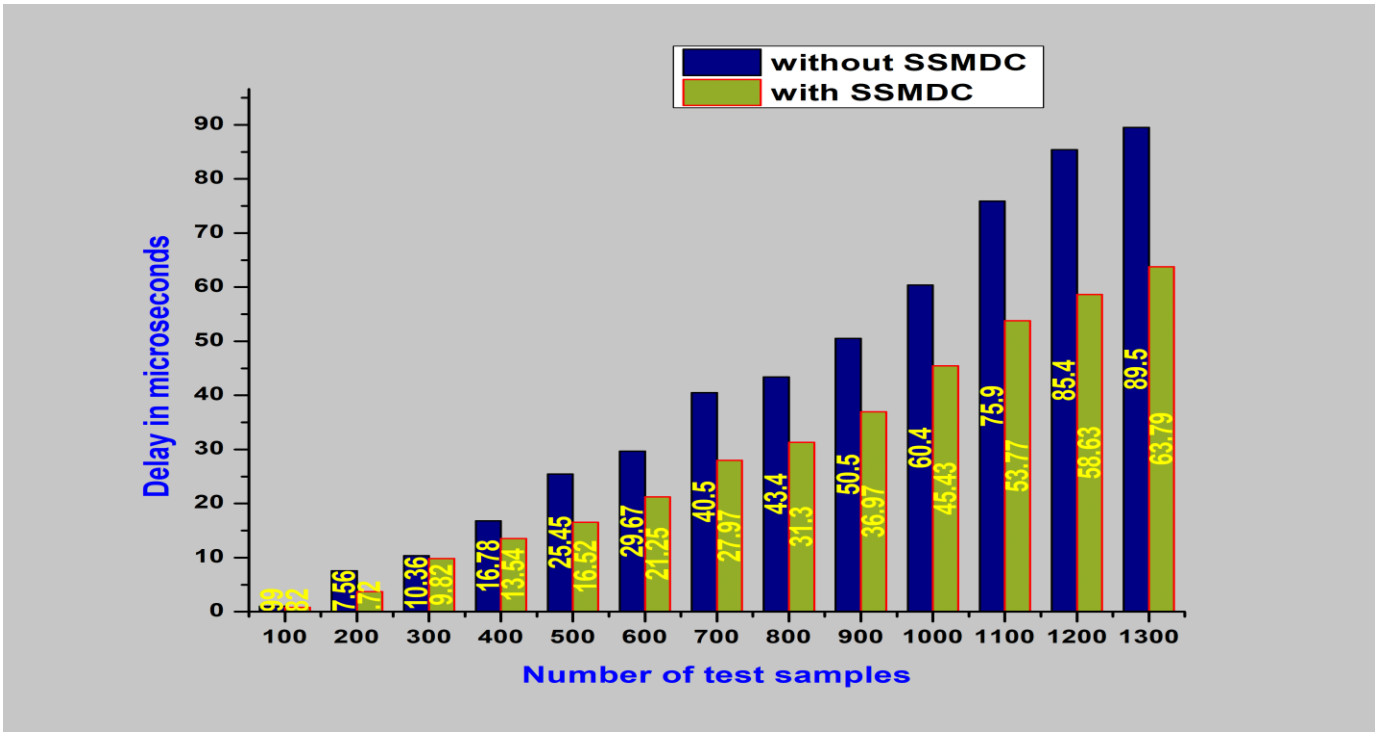

Fig. 3 Comparative Analysis of Classification Delay with and without optimization of SSMDC Model.

Likewise, the comparative estimation of classification delay with and without the SSMDC model for the speech keyword dataset is represented in Figure 3. The figure indicates that the classification delay with the SSMDC model provides superior results in all the test samples in contrast to those without SSMDC model. Hence, the proposed SSMDC model outperforms in terms of classification delay, power consumption, and LUT as compared to other state-of-the-art models [1,5,10,27]. Therefore, the proposed optimized SSMDC model which provides a minimum classification delay can be a suitable classifier for dense speech processing FPGA-based datasets.

4. CONCLUSION

This work presents the unique SSMDC model, which is ideal for delay-aware classification of the limited datasets related to

recorded speech processing. It is generated by integrating the PSNG and MD5 models. In order to help create useful sample indices that fall into a number of categories, the paper describes the creation of an Intelligent Pseudo Stochastic Number Generator (GCI PSNG) based on the Gold Code. The low-complexity truncated Message Digest 5 (MD5) model aids in the development of sample-selection indices in the GCI PSNG model. The suggested SSMDC model lowers the classification time by roughly 27.35% when compared to alternative feature classification models. Our suggested model performed better in delay-aware deployments in a variety of applications, including wireless communication, industrial automation, speech, and picture recognition, and so forth, then other current models, such as DNN, Convolutional Neural Networks, and RF Classifiers. In conclusion, the suggested approach is a noteworthy advancement in the field of FPGA-based digital categorization. It provides an effective and innovative way to boost the functionality of FPGA-based systems, make digital classification quicker and more accurate, and make it easier to design new applications across a range of industries and their deployment use cases.

REFERENCES

1. Maedeh Nobari, and Hadi Jahanirad, “FPGA-based implementation of deep neural network using stochastic computing”, in Applied Soft Computing, vol. 137, April 2023, doi: https://doi.org/10.1016/j.asoc.2023.110166.

2. Matheus M. de A. Kotaki et al., “FPGA implementation of a Pseudorandom Number Generator Based on k-Logistic Map”, in 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS), Feb. 2020, doi: https://doi.org/10.1109/LASCAS45839.2020.9068999.

3. J. Deepakumara et al., “FPGA implementation of MD5 hash algorithm”, in Canadian Conference on Electrical and Computer Engineering 2001, pp. 919-924, May 2001, doi: https://doi.org/10.1109/CCECE.2001.933564.

4. K. -Y. Chou, and Y. P. Chen, "Real-Time and Low-Memory Multi-Faces Detection System Design with Naive Bayes Classifier Implemented on FPGA," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 4380-4389, Nov. 2020, doi: https://doi.org/10.1109/TCSVT.2019.2955926.

5. Lunyi Guo, Shining Mu, Yijie Deng, Chaofan Shi, Bo Yan, and Zhuoling Xiao, “Efficient Binary Weight Convolutional Network Accelerator for Speech Recognition”, Sensors 2023, vol.23, no. 3, Jan. 2023, doi: https://doi.org/10.3390/s23031530.

6. J. Arias-Garcia et al., "Enhancing Performance of Gabriel Graph-Based Classifiers by a Hardware Co-Processor for Embedded System Applications," in IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 1186-1196, Feb. 2021, doi: https://doi.org/10.1109/TII.2020.2987329.

7. N. Attarmoghaddam, and K. F. Li, "An Area-Efficient FPGA Implementation of a Real-Time Multi-Class Classifier for Binary Images," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 4, pp. 2306-2310, April 2022, doi: https://doi.org/10.1109/TCSII.2022.3148228.

8. M. Elnawawy, A. Sagahyroon and T. Shanableh, "FPGA-Based Network Traffic Classification Using Machine Learning," in IEEE Access, vol. 8, pp. 175637-175650, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3026831.

9. D. Peng and J. Sha, "Efficient HLS Implementation of Fast Linear Discriminant Analysis Classifier," in IEEE Embedded Systems Letters, vol. 13, no. 4, pp. 214-217, Dec. 2021, doi: https://doi.org/10.1109/LES.2021.3078180.

10. Yuexuan Luo, Xiang Cai, Jiandong Qi, Dongdong Guo, and Wenqing Che, “FPGA-accelerated CNN for real-time plant disease identification”, Computers and Electronics in Agriculture, vol. 207, April 2023, doi: https://doi.org/10.1016/j.compag.2023.107715.

11. M. Sahani, and P. K. Dash, "FPGA-Based Deep Convolutional Neural Network of Process Adaptive VMD Data with Online Sequential RVFLN for Power Quality Events Recognition," in IEEE Transactions on Power Electronics, vol. 36, no. 4, pp. 4006-4015, April 2021, doi: https://doi.org/10.1109/TPEL.2020.3023770.

12. R. Li, Q. Yang, Y. Li, X. Gu, W. Xiao, and K. Li, "HeteroYARN: A Heterogeneous FPGA-Accelerated Architecture Based on YARN," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 12, pp. 2968-2980, 1 Dec. 2020, doi: https://doi.org/10.1109/TPDS.2019.2905201.

13. X. Wang et al., "Design of a Real-Time Movement Decomposition-Based Rodent Tracker and Behavioral Analyzer Based on FPGA," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 9, pp. 1133-1143, Sept. 2022, doi: https://doi.org/10.1109/TVLSI.2022.3168783.

14. Z. Zheng, Y. Zhong, A. Ma, and L. Zhang, "FPGA: Fast Patch-Free Global Learning Framework for Fully End-to-End Hyperspectral Image Classification," in IEEE Transactions on Geoscience and Remote Sensing, vol. 58, no. 8, pp. 5612-5626, Aug. 2020, doi: https://doi.org/10.1109/TGRS.2020.2967821.

15. O. Mujahid, and Z. Ullah, "High Speed Partial Pattern Classification System Using a CAM-Based LBP Histogram on FPGA," in IEEE Embedded Systems Letters, vol. 12, no. 3, pp. 87-90, Sept. 2020, doi: https://doi.org/10.1109/LES.2019.2956154.

16. N. Rezaei, M. N. Uddin, I. K. Amin, M. L. Othman, M. B. Marsadek, and M. M. Hasan, "A Novel Hybrid Machine Learning Classifier-Based Digital Differential Protection Scheme for Intertie Zone of Large-Scale Centralized DFIG-Based Wind Farms," in IEEE Transactions on Industry Applications, vol. 56, no. 4, pp. 3453-3465, July-Aug. 2020, doi: https://doi.org/10.1109/TIA.2020.2990584.

17. L. Andrade Maciel, M. Alcântara Souza, and H. Cota de Freitas, "Reconfigurable FPGA-Based K-Means/K-Modes Architecture for Network Intrusion Detection," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 8, pp. 1459-1463, Aug. 2020, doi: https://doi.org/10.1109/TCSII.2019.2939826.

18. A. K. Jameil, and H. Al-Raweshidy, "Efficient CNN Architecture on FPGA Using High Level Module for Healthcare Devices," in IEEE Access, vol. 10, pp. 60486-60495, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3180829.

19. H. Liu, A. Panahi, D. Andrews, and A. Nelson, "An FPGA-Based Upper-Limb Rehabilitation Device for Gesture Recognition and Motion Evaluation Using Multi-Task Recurrent Neural Networks," in IEEE Sensors Journal, vol. 22, no. 4, pp. 3605-3615, 15 Feb.15, 2022, doi: https://doi.org/10.1109/JSEN.2022.3141659.

20. J. C. Fabero et al., "Single Event Upsets Under 14-MeV Neutrons in a 28-nm SRAM-Based FPGA in Static Mode," in IEEE Transactions on Nuclear Science, vol. 67, no. 7, pp. 1461-1469, July 2020, doi: https://doi.org/10.1109/TNS.2020.2977874.

21. A. J. A. El-Maksoud, M. Ebbed, A. H. Khalil, and H. Mostafa, "Power Efficient Design of High-Performance Convolutional Neural Networks Hardware Accelerator on FPGA: A Case Study With GoogLeNet," in IEEE Access, vol. 9, pp. 151897-151911, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3126838.

22. S. K. Mousavikia, E. Gholizadehazari, M. Mousazadeh, and S. B. O. Yalcin, "Instruction Set Extension of a RiscV Based SoC for Driver Drowsiness Detection," in IEEE Access, vol. 10, pp. 58151-58162, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3177743.

23. X. Wang, L. Gu, and Z. Wang, "Computer Medical Image Segmentation Based on Neural Network," in IEEE Access, vol. 8, pp. 158778-158786, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3015541.

24. M. Zeghid et al., "Modified Optical Burst Switching (OBS) Based Edge Node Architecture Using Real-Time Scheduling Techniques," in IEEE Access, vol. 9, pp. 167305-167321, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3132578.

25. Y. Cao, S. Jiang, S. Guo, W. Ling, X. Zhou, and Z. Yu, "Real-Time SAR Imaging Based on Reconfigurable Computing," in IEEE Access, vol. 9, pp. 93684-93690, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3093299.

26. Ayat Naji Hussain, et. al., "Impact of feature reduction techniques on classification accuracy of machine learning techniques in leg rehabilitation", in Measurement: Sensors , vol. 25, Feb. 2023, doi: https://doi.org/10.1016/j.measen.2022.100527.

27. S. Soltani, Y. E. Sagduyu, R. Hasan, K. Davaslioglu, H. Deng and T. Erpek, "Real-Time and Embedded Deep Learning on FPGA for RF Signal Classification," MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM), Norfolk, VA, USA, 2019, pp. 1-6, doi: https://doi.org/10.1109/MILCOM47813.2019.9021098.

28. Mohamed Gafsi, et. al., "Hardware implementation of digital pseudo-random number generators for real-time applications", signal, image and video processing, 2024, doi: https://doi.org/10.1007/s11760-024-03082-8.

29. Feali, M.S. Realization of a pseudo-random number generator utilizing two coupled Izhikevich neurons on an FPGA platform. Analog Integr Circ Sig Process 119, 57–68 (2024). https://doi.org/10.1007/s10470-023-02223-2.

30. Fei Yu, et. al., "Design and FPGA implementation of a Pseudorandom Number Generator Based on a Four-Wing Memristive Hyperchaotic System and Bernoulli Map", IEEE access, vol. 7, 2019, doi: https://doi.org/10.1109/ACCESS.2019.2956573.

31. B. Deng, Y. Fan, J. Wang, and S. Yang, "Reconstruction of a Fully Paralleled Auditory Spiking Neural Network and FPGA Implementation," in IEEE Transactions on Biomedical Circuits and Systems, vol. 15, no. 6, pp. 1320-1331, Dec. 2021, doi: https://doi.org/10.1109/TBCAS.2021.3122549.

32. R. Sayed, H. Azmi, A. M. Nassar, and H. Shawkey, "Design Automation and Implementation of Machine Learning Classifier Chips," in IEEE Access, vol. 8, pp. 192155-192164, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3032658.

33. Z. Xue, J. Wei, and W. Guo, "A Real-Time Naive Bayes Classifier Accelerator on FPGA," in IEEE Access, vol. 8, pp. 40755-40766, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2976879.

34. N. Kishor, R. Singh H, S. R. Mohanty, and O. Yadav, "Evolving Disturbances Detection and Classification in Real-time for Grid-Connected System," in IEEE Transactions on Industrial Electronics, vol. 68, no. 9, pp. 8265-8273, Sept. 2021, doi: https://doi.org/10.1109/TIE.2020.3013739.

35. A. Luiz Barbosa, G. Loureiro, S. Manea, J. Marcelo Lima Duarte, and G. Paulineli Garbi, "Ranking of Fault Mitigation Techniques for Spatial Radiation in Commercial Off-the-Shelf Field Programmable Gate Array," in IEEE Latin America Transactions, vol. 18, no. 04, pp. 736-743, April 2020, doi: https://doi.org/10.1109/TLA.2020.9082217.

36. T. Liang, Z. Huang, and V. Dinavahi, "Adaptive Real-Time Hybrid Neural Network-Based Device-Level Modeling for DC Traction HIL Application," in IEEE Access, vol. 8, pp. 69543-69556, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2986298.

37. D. Lee, and V. Bertacco, "Bypassing Multicore Memory Bugs With Coarse-Grained Reconfigurable Logic," in IEEE Transactions on Computers, vol. 71, no. 9, pp. 2191-2204, 1 Sept. 2022, doi: https://doi.org/10.1109/TC.2021.3125188.

38. H. Fan, M. Ferianc, Z. Que, X. Niu, M. Rodrigues, and W. Luk, "Accelerating Bayesian Neural Networks via Algorithmic and Hardware Optimizations," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 12, pp. 3387-3399, 1 Dec. 2022, doi: https://doi.org/10.1109/TPDS.2022.3153682.

39. E. Youssef, H. A. Elsimary, M. A. El-Moursy, H. Mostafa, and A. Khattab, "Energy-Efficient Precision-Scaled CNN Implementation With Dynamic Partial Reconfiguration," in IEEE Access, vol. 10, pp. 95571-95584, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3204704.

40. Xinlong Wang, et. al., "A Low-Resource-Cost FPGA implementation of Population Threshold Coding for Spiking Neural Networks ," 2024 4th International Conference on Neural Networks, Information and Communication Engineering (NNICE), pp., Information and Communication Engineering (NNICE), pp.-79, doi: https://doi.org/0.1109/NNICE61279.2024.10498425.

Authors

Sachin D. Kohale received BE degree in electronics engineering from R.T.M. Nagpur University, Nagpur, India in 2008 and ME degree in embedded systems and computing from G.H. Raisoni College of Engineering, Nagpur, India in 2013. He is currently pursuing PhD at the Department of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan in speech classifications, and applications of speech based robotic control systems.

E-mail: sdk26091986@gmail.com

Trapti Sharma received PhD degree from Maulana Azad National Institute of Technology, Bhopal, India. She is currently an Assistant Professor at VIT Bhopal University, Bhopal, India. Her research interests include computer arithmetic circuits, multi-valued logic design, nano-electronics, speech processing, and low-power VLSI design.

E-mail: trapti16sharma@gmail.com