IEEE Open Journal of the
Computer Society

Received 18 December 2023; accepted 14 January 2024. Date of publication 18 January 2024;
date of current version 16 February 2024. The review of this article was arranged by Associate Editor Amiya A Nayak.

Digital Object Identifier 10.1109/OJCS.2024.3355693

Low Area and Low Power FPGA Implementation of a DBSCAN-Based RF Modulation Classifier

BILL GAVIN , TIANTAI DENG (Member, IEEE), AND EDWARD BALL (Member, IEEE)

Department of Electrical and Electronic Engineering, The University of Sheffield, S1 3JD Sheffield, U.K.

CORRESPONDING AUTHOR: BILL GAVIN (e-mail: wcjgavin1@sheffield.ac.uk)

This work was supported by the U.K. Research and Innovation (UKRI) Future Leaders Fellowship under Grant MR/T043164/1.

ABSTRACT This paper presents a new low-area and low-power Field Programmable Gate Array (FPGA) implementation of a Radio Frequency (RF) modulation classifier based on the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, known as DBCLASS. The proposed architecture demonstrates a novel approach for the efficient hardware realisation of the DBSCAN algorithm by utilising parallelism, a bespoke sorting algorithm, and eliminating memory access. The design achieves 100% classification accuracy with lab-captured RF data above 8 dB signal-to-noise ratio(SNR) whilst exhibiting an improvement of latency in comparison to the next quickest design by a factor of 7.5, a reduction in terms of total FPGA resources used in comparison to the next smallest complete system by a factor of 3.65, and a reduction in power consumption over the next most efficient by a factor of 4.75. The proposed design is well suited for resource-constrained applications, such as mobile cognitive radios and spectrum monitoring systems.

INDEX TERMS RF classifier, cognitive radio, DBSCAN, FPGA, automatic modulation classification, AMC, beyond smart radio.

I. INTRODUCTION

The ever-increasing demand for wireless communication has led to the emergence of numerous communication standards and the need for efficient spectrum utilisation. Identifying and classifying the modulation schemes of radio signals is critical for dynamic spectrum access, cognitive radio systems, and the development of beyond smart radio systems for 6G [1]. Machine learning algorithms have proven to be effective in tackling such classification tasks. Among these algorithms, Convolutional neural networks(CNN) [2] and long-short-term memory (LSTM) [3] based systems have emerged as the most popular unsupervised learning method for detecting patterns in large datasets. While these models have shown strong performance [2], [3], their complex and generalized nature can be a limitation, particularly in mobile and low-power devices. Yingchun Wang et al. [5] detail the challenges with deploying deep learning systems in these scenarios. They conclude that to overcome the high power consumption and chip area requirements that machine learning models suffer from, engineers should either reduce model complexity or offload to the cloud for processing. Our work will attempt to solve this challenge by reducing complexity via introducing a bespoke clustering algorithm, specifically designed to address scenarios where CNNs and LSTMs fall short. The superiority of this approach is underscored by several critical factors:

- Neural Networks such as CNNs and LSTMs can be resource-heavy, requiring significant memory and processing power. This can be a limiting factor, especially when deploying models to mobile devices [5]. In contrast, our algorithm is optimized for energy efficiency, making it ideal for deployment in battery-operated or low-power devices.
- The streamlined design of our clustering algorithm allows for rapid data processing, resulting in lower latency compared to CNNs and LSTMs. This is particularly beneficial in applications requiring real-time data analysis, where the delay introduced by the computational complexity of CNNs and LSTMs can be prohibitive [5].

- Tailoring the algorithm to specific data scenarios not only enhances its efficiency but also reduces the computational overhead required for processing. This targeted approach allows the algorithm to bypass the extensive and often redundant calculations that CNNs and LSTMs perform, further contributing to lower power consumption and faster processing times [5].

In this work we propose a system called DBCLASS (Density-Based CLASSifier), based on the Density Based Spatial Clustering of Applications with Noise(DBSCAN) [4].

DBSCAN has not been applied thus far to tackle the problem of RF modulation classification. A traditional implementation of the algorithm in hardware would prove to be computationally slow due to its inherent sequential processing. FPGA implementations have the potential to address these challenges by exploiting parallelism and customization opportunities. This paper introduces a low area and low power FPGA custom implementation of DBSCAN that addresses these hardware limitations of the traditional algorithm.

The key achievements of this system are:

- A low-power design that reduces the overall power consumption of the FPGA implementation by a factor of 4.75 in comparison to the next most efficient [15].
- A highly optimised pre-processing system based upon DBSCAN and a minimally complex artificial intelligence(AI) model together achieves a factor of 3.65 reduction in total FPGA resources used in comparison to the next smallest complete system [19].
- A pipelined architecture that is designed to work on real-time data-streams which achieves a reduction in latency of by a factor of 7.5 compared to the next quickest system [19].
- Competitive classification performance which matches the accuracy of more complex CNN architectures at SNRs above 8 dB.

The remainder of this paper is organised as follows: Section II provides background on other work in this area from the literature. Section III gives an overview of the DBSCAN algorithm and its application to RF modulation classification. Section IV describes the proposed FPGA-based architecture in detail. Sections V and VI present and discuss the experimental results, and Section VII concludes the paper with a summary of the contributions.

II. RELATED WORKS

In the literature a number of approaches to modulation classification have shown their effectiveness in software. These can roughly be divided into three schools of thought, statistical wave feature extraction, automatic time series classification, and constellation diagram classification.

A. STATISTICAL CLASSIFICATION

The first of these approaches takes samples of waves from an incoming waveform and statistically determines features about the sample; examples of features which are used can be found in the article by A.K. Nandi and E.E. Azzouz [6]. Notable features include the kurtosis, entropy, standard deviation, skewness, and symmetry of a wave. A.K. Nandi and E.E. Azzouz create a system using these features and at 15 dB SNR the system correctly identifies Amplitude Shift Keying(ASK) and Frequency Shift Keying(FSK) at a minimum of 97% of the time, at 20 dB the results are 100% accurate. Boutte et al. [7] apply the same approach to modern modulation schemes, this approach combined with a Support Vector Machines (SVM) network is shown to be capable of achieving close to 100% accurate classification of Quadrature Phase Shift Keying(QPSK) above an SNR of 6 dB, as well as this Orthogonal Frequency Division Multiplexing(OFDM) BPSK is classified with 95% accuracy above an SNR of 15 dB. Both of these papers use a limited set of modulation types, a clearer picture of performance across a larger set of modulation schemes can be found in the work by D. Saharia et al. [8]. In this paper a large set of results is presented with a confusion matrix of 11 different schemes. While some modulation types are classified above 90% accuracy at an SNR of 16 dB it is clear that the technique struggles to deal with such a vast array of schemes, this is especially clear when differentiating between similar modulation types based upon Quadrature Amplitude Modulation(QAM), 16QAM and 64QAM. When attempting to differentiate between these two similar waves they are classified as each other at almost the same rate as themselves. Other modulation types are classified on average with a 70% accuracy. A downside to this approach is the intense pre-processing of signals which is required before classification can be performed, this will lead to a delay in obtaining a classification result as well as increase the size and complexity of any hardware implementation. Due to the poor performance at low SNRs and low throughput this approach has been supplanted by more modern approaches.

B. DIRECT WAVEFORM CLASSIFICATION

Rajendran et al. in [9] use a LSTM to automatically classify RF waveforms, this achieves two notable improvements over the statistical feature methods. Firstly the model exhibits an enormous improvement in classification accuracy, across exactly the same modulation types as in [8] there is an improvement in all but one. The system classifies most schemes with an accuracy of at least 90% at 0 dB SNR, there still exists some misclassification of similar waveforms such as 16QAM and 64QAM but the accuracy remains at 85% and above, a marked improvement over the 52% accuracy with high SNR data with the statistical feature classifiers. The second advantage of using the LSTM is that the model directly uses the incoming RF waveform, thereby avoiding the need for pre-processing, however these gains are mitigated due to requiring a larger sample of data for classification and the LSTM structure being larger than most model structures in general. This paper also gives a comparison of classification accuracy of various model structures across a range of SNRs. The LSTM is shown to be vastly superior to most model structures, achieving an average of 90% accuracy above SNRs of 0 dB, only the CNN comes close in terms of performance by achieving an accuracy of 80%. Similar results are obtained by Ke et al. in [10], a LSTM model is shown to have the greatest average accuracy across all SNRs with 90%, a confusion matrix of the same collection of modulation schemes shows strong differentiation between each type, however the reduced accuracy with similar waveforms remains. LSTM based models have therefore shown strong robustness to noise yet are unable to reach perfect classification accuracy of 100% at any SNR.

C. CONSTELLATION CLASSIFICATION

The final approach to modulation classification is to classify the data based upon the appearance of the constellation diagram. There are a few ways of approaching this problem, the first of which is to create images of the constellation diagram and use an image-recognition CNN to classify constellations based on learned appearances. This method is shown by Doan et al. [11], at an SNR of 5 dB and above the model correctly classifies all schemes with a 100% accuracy, by far the best performance at this low SNR. However there are drawbacks when using techniques such as this. Firstly the CNN, and especially image recognition CNNs will have a large implementation size on a FPGA. Secondly not only is the CNN structure large but an entire pre-processing system must be implemented to prepare the images, adding further complexity and resource utilisation. Finally, not only are large batches of data required for the creation of the constellation image but creating the image itself will add a significant delay to obtaining a classification result. So this technique of classification is capable of achieving the 100% accuracy but at the cost of requiring more pre-processing and a large deep learning model.

Yu Wang et al. [12] use a CNN to perform convolution on constellation diagrams to calculate the data densities, this is then used to train a second CNN model. The work again achieves 100% classification accuracy above an SNR of 5 dB and is capable of 90% accuracy at 0 dB. The key idea in this work is rather than treating the constellation as an image, the data is represented numerically and the densities of the data points are used for classification. Yet this work requires multiple CNNs connected in series and parallel, the input CNN will determine the broad modulation type such as M-PSK or M-QAM and then the data will pass to the model which is trained to differentiate between orders of modulation. This work shows that using data density for classification can allow for strong performance yet the complexity of the system makes it unsuitable for low-area and low-power embedded systems.

While software models have shown greater accuracy at low SNRs than FPGA models, owing to their use of floating point precision, FPGAs have the advantage of reduced delay and power consumption [2], [15], [16]. Thus FPGA and application-specific integrated circuit(ASIC) solutions are the optimal choice for low-power, low-area, and low-delay modulation classifiers in embedded systems.

D. HARDWARE COMPARISONS

The majority of hardware implementations found in the literature are based upon the CNN. This is to be expected as the CNN has shown the best accuracy in simulations [11], [12]. Just as in software the approaches can also broadly be characterised into either time-series or constellation demodulation. The papers which exhibit the highest classification accuracy are the ModNet system by Kumar et al. [22] and HistoSVM by Cardoso et al. [23], these works both achieve 100% classification accuracy above an SNR of 9 dB, at which point the accuracy of HistoSVM begins to decline and reaches 74% accuracy at 0 dB. The 100% classification accuracy of ModNet is maintained until 4 dB, below this SNR the performance degrades until 86% accuracy is reached at 0 dB. ModNet follows a similar approach to Doan [11] and creates images of constellations which a CNN classifies. HistoSVM introduces a wholly unique approach and creates histograms which are used in conjunction with a Support Vector Machines(SVM) classifier. The best performing time series hardware model is ResNet by J O'Shea et al. [25]. In this work the authors use a modified CNN known as a residual neural network and achieve an overall 96% accuracy, this performance is maintained until 10 dB SNR, although the authors do demonstrate that low order modulation classification accuracy reaches 100% accuracy. It is worth noting that the trend of lower order modulation scheme classification achieving higher classification accuracy is consistent across many papers [18], [22], [24], [25]. ResNet and ModNet are therefore the best performing examples of waveform and constellation classification in hardware respectively,

Out of these three best performing systems only HistoSVM provides data for the characteristics of the FPGA implementation, making a hardware comparison between each model difficult. A table of resource utilization of various designs can be found in Table 3 in Section VI, additionally Fig. 11 in the same section shows a comparison graph of accuracy against SNR. HistoSVM uses by far the least registers compared to other work, the majority of other designs are based upon the CNN and use tens to hundreds of thousands of registers. Conversely, HistoSVM uses an enormous amount of BRAM, the largest of any found in the literature, the latency of this work again is the largest which can be found. So while HistoSVM achieves 100% accuracy it comes at a cost of memory usage and latency. RUNet [19] again by Kumar et al. uses a similar residual neural network to ResNet and achieves very similar accuracy. This model uses the least registers, Look-Up-Tables(LUTs), Digital-signal-processors (DSP), and RAM of any deep learning based model bar Zhao et al. [2] which requires less registers and LUTs. Additionally RUNet has the least latency of any deep learning based system at 7.5 μs, narrowly beating S. Tridgell et al. [14], [16] by 0.5 μs.

In terms of area utilization and delay, RUNet [19] is the state-of-the-art in terms of implementation size, delay, and accuracy. The lowest power design found is that of Amad et al. [15] which uses 847 mW.

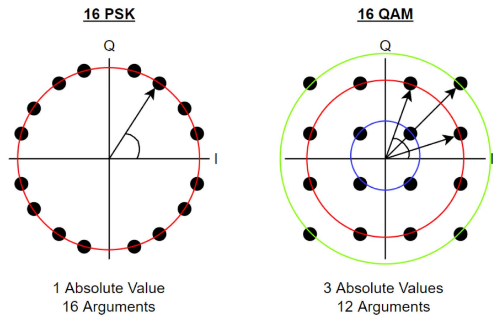

FIGURE 1. Difference between 16QAM and 16PSK.

Through efficient preprocessing in conjunction with a minimally complex machine learning classifier, similar to HistoSVM's approach, there is a possibility of creating a system that improves upon all work in terms of area, power consumption, and delay. The following sections will discuss the methodology in creating this system.

III. PROPOSED METHOD

In this work a new method of classification which minimises preprocessing, and does not require the use of a complex neural network model to achieve 100% accuracy is presented. The idea is to exploit the characteristics of the constellation diagram, which is essentially a set of clusters of points in 2D space, ideal for the application of a clustering algorithm. Most clustering algorithms such as K-Nearest Neighbours(KNN) will group points into a specified number of clusters [26], whereas the problem of this work is to solve the inverse. There are well defined clusters, if the number of them could be determined as well as their relative positions on the diagram, a minimally complex network could classify them based upon this information as each modulation type will have a unique number and arrangement of constellations. The clustering algorithm DBSCAN is suitable for this problem as it forms an arbitrary number of clusters, without a user specified parameter. This work will propose a novel method of using DBSCAN to extract the information about the clusters directly and use this information to achieve classification.

Time-series RF waves are decomposed and represented as two waves known as In-Phase(I) and Quadrature(Q) which respectively correspond to the instantaneous amplitude and phase of the original wave. The IQ point pairs can then be plotted in 2D space as a complex number Z. Modulation schemes which utilise changes in phase and amplitude will exhibit different clusters of points throughout the 2D plane as the I and Q values change to represent different data symbols, this forms a particular pattern known as the constellation diagram. A simple example of how this system will operate is by examining the examples of QPSK and 8PSK. Both of these modulation schemes can be distinguished as a human by recognising that the diagram with 4 constellations must represent the QPSK and likewise the 8 constellations the 8PSK. Similarly, the same process can be done with a computer through clustering in order to determine the number of constellations, therefore differentiating between QPSK and 8PSK.

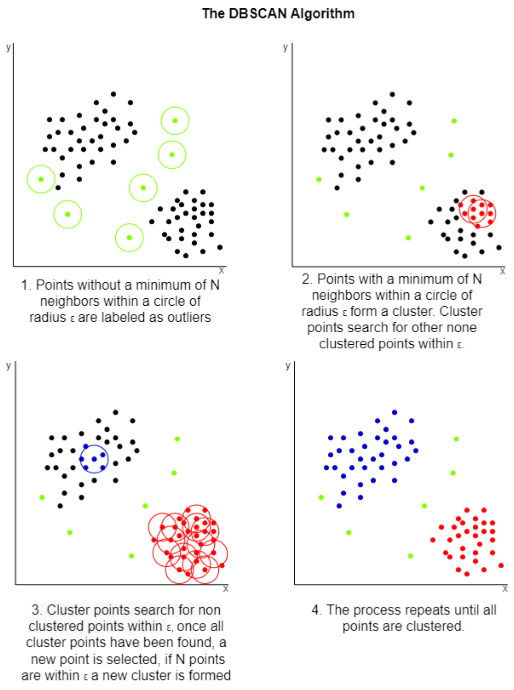
The DBSCAN Algorithm
y
x
1. Points without a minimum of N neighbors within a circle of radius ε are labeled as outliers
2. Points with a minimum of N neighbors within a circle of radius ε form a cluster. Cluster points search for other none clustered points within ε.
3. Cluster points search for non clustered points within ε, once all cluster points have been found, a new point is selected, if N points are within ε a new cluster is formed
4. The process repeats until all points are clustered.

FIGURE 2. Diagram of the operation of traditional DBSCAN.

Not all modulation types can be differentiated by the number of constellations, for example 16PSK and 16QAM both have 16 constellations but it is the positioning of the constellations which can be used to separate them. To achieve this a proxy for determining positioning is to calculate the absolute value and arguments of each constellation, an example is shown below in Fig. 1. The calculated absolute values and arguments can be clustered to sort them into groups. Once the clustering is finished, a final result is obtained which is the number of different arguments and absolute values of the constellations, with this data the modulation scheme can be determined with a machine learning classifier trained on similar data. In addition to the argument and absolute value data allowing for stronger differentiation between like constellations, the 1 dimensional nature of the data allows for a unidimensional DBSCAN to be executed on each set of data, which facilitates further efficiency gains which are outlined in Section IV-B.

A. DBSCAN

A diagram of the operation of DBSCAN can be found in Fig. 2. Two different parameters are required to achieve accurate clustering with DBSCAN. These parameters are the minimum number of spatially near points to constitute a cluster (minPts), and minimum distance between two points to be considered part of the same cluster ε. DBSCAN has a worst case computational complexity of $O(n^2)$ owing to the process of checking the distance to each point in the dataset from each point in the dataset. When working with 1 dimensional data as in this case, it is advantageous to sort the data and apply a modified algorithm. An example of unsorted and sorted data can be seen in Figs. 3 and 4 respectively.

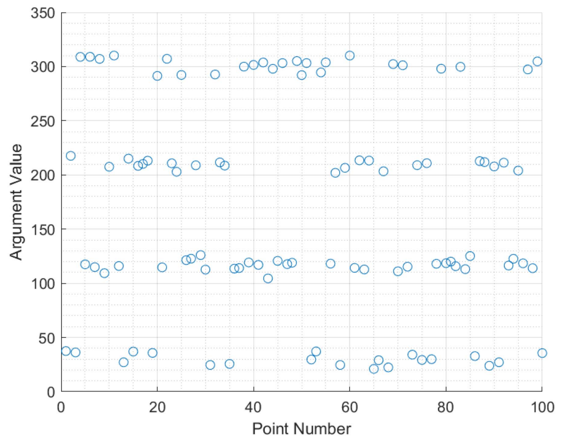

FIGURE 3. Unsorted QPSK argument data example sample.

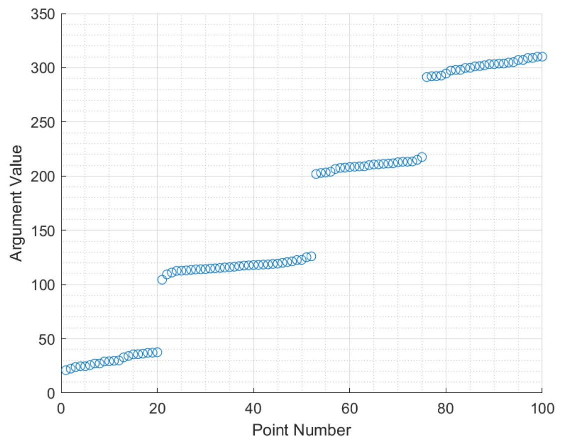

FIGURE 4. Sorted QPSK argument data example sample.

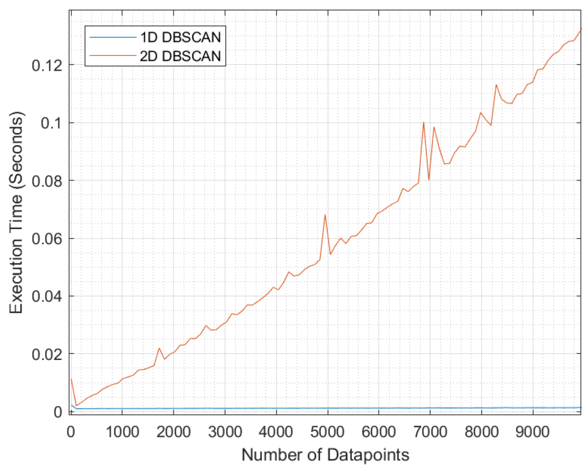

FIGURE 5. Speed-up comparison of sorted 1D DBSCAN and traditional DBSCAN in MATLAB.

By sorting the data only the distance to the next point in the array needs to be calculated to determine if the next point belongs to the same cluster. This results in a computational complexity reduction of $O(n^2)$ to $O(n)$. A graph of the speed-up difference in software can be found in Fig. 5.

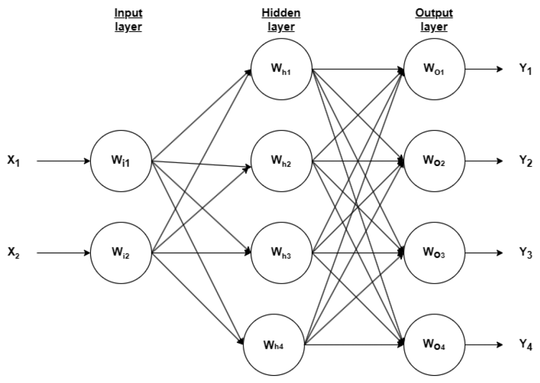

FIGURE 6. Diagram of the MLP structure.

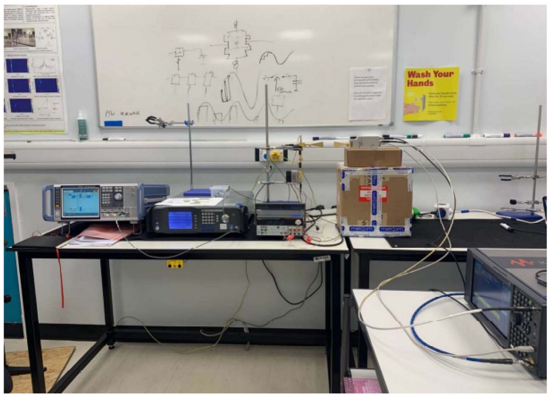

FIGURE 7. Photograph of lab setup for data capture.

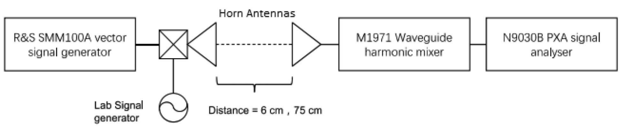
R&S SMM100A vector signal generator
Horn Antennas
M1971 Waveguide harmonic mixer
N9030B PXA signal analyser
Lab Signal generator
Distance = 6 cm , 75 cm

FIGURE 8. Block diagram of lab setup for data capture.

B. CLASSIFIER

The machine learning classifier was trained using the number of absolute value and argument clusters, which is output from DBSCAN. Testing of suitable model structures was performed using MATLAB R2021b. It was found that the data showed good separation and therefore a small 4 node hidden layer 4 node output layer Multilayer Perceptron (MLP) achieved as strong performance, a more complex models such as a CNN or RNN would lead to an unnecessary increase in FPGA utilization and power consumption. Its structure can be found in Fig. 6. Training was performed with data obtained from applying DBSCAN on arguments and absolute values of RF data, it was standardised between ±127 to mimic the 8-bit data in the implementation scenario, 5-fold cross validation and regularisation was employed to reduce overfitting.

C. DATA

The data capture setup can be seen in Figs. 7 and 8 which show a picture of the laboratory setup and its corresponding block diagram. All data used for testing of the system and training of

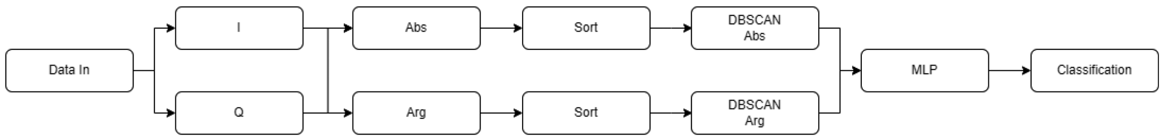

FIGURE 9. Full system diagram.

the MLP classifier was generated using the Rohde & Schwarz SMW100A [20] and captured with a Keysight N9030B PXA signal analyser [21], waves modulated with BPSK, QPSK, 8PSK, and 16QAM were created at SNRs which ranged from 30 dB to 3 dB. The signal analyser was configured to the same carrier frequency as the signal source but was not in carrier phase lock. Additional Gaussian noise was added to the 3 dB signals to generate 0 dB and −5 dB sets of data. RF samples of two frequencies were captured, 73 GHz and 28 GHz, in both cases the data rate was 50 Msymbols/s. The spectrum analyser sampled data at 200Msamples/s, with a 160 MHz intermediate frequency Bandwidth, and a 100µs capture duration. The 73 GHz horn antennas used were Eravant SAZ-2410-12-S1 with a gain of 24 dBi and the 28 GHz horn antennas were Quasar QWH21SB-URB-K-F-20 with a gain of 20 dBi, the horn antennas are represented as the triangles in Fig. 8. Data was radiated at a proximity of 6 cm between horn antennas. Our data can be downloaded from Github at https://github.com/billjgavin/28_and_75GHz_Capture_Files.

IV. HARDWARE IMPLEMENTATION

Following the confirmation of the performance of the system in a software simulation the process of implementing the algorithm in hardware began. The primary focus of the hardware implementation was to create a system which was capable of classifying real-time streams of RF data while maintaining the performance achieved in software simulations. The implementation is fully pipelined and designed in such a way that each module can operate continuously. A system diagram of the full algorithm can be seen below in Fig. 9.

The algorithm is split into 4 constituent blocks: The absolute and argument LUTs calculate the absolute values and arguments of the complex IQ pairs which represent the RF message. These values are split into two datapaths which operate simultaneously, the operations performed in each datapath are identical. The first step of the split data paths is a custom built sorting module which sorts data in real-time as it enters the system. Following this, the sorted data flows byte by byte into a custom DBSCAN module, a further explanation of these systems can be found in Section IV-B. The final block recombines both datapaths in an MLP classifier which outputs the predicted modulation scheme. The implementation of the design was written in Verilog but the place and routing of the implementation was handled by the Vivado 2021.2 tools. The implementation strategy was set to find the implementation with the strongest performance with the command performance_explore. Otherwise all settings remained in their default state.

A. ABSOLUTE AND ARGUMENT BLOCKS

Finding the absolute value and argument of a complex number can be done with (1a) and (1a).

$$arg = \tan^{-1}\left(\frac{q}{i}\right) \tag{1a}$$

$$abs = \sqrt{q^2 + i^2} \tag{1b}$$

Each of these equations require operations which are computationally slow to perform in hardware, finding the argument requires a division and an arctan, the absolute value requires multiple multiplications and a square root. The goal of this design is to handle a real time datastream, performing these operations would require too many clock cycles to facilitate this. Instead, a set of outputs for every combination of 8-bit I and Q inputs are precomputed. This required two large LUTs with 65536 entries each which used a significant amount of the available LUT slices on the FPGA. Despite this, performing the calculations in this way reduced the complex operations to a single clock cycle, enabling the rest of the design to function in real-time. Additionally, normalisation calculations were included in the output of the LUTs which eliminated a required step in the system, saving both time and resources. Incident data passed from the LUTs and into the sorting block.

B. SORTING AND DBSCAN

In this work a custom DBSCAN algorithm is employed which exploits the 1 dimensional nature of the absolute and argument data. This is achieved by pre-sorting data before the DBSCAN algorithm is applied. This sorting step allows for the minimum value to the next largest point ε to be calculated by simply taking the difference between point N and point N+1 in the data array, rather than taking the difference between point N and all other unclustered points. Overall algorithmic complexity is reduced from the traditional $O(n^2)$ for DBSCAN to the complexity of the sorting algorithm.

Further gains can be made to the calculation speed by sorting data as it enters the system. As shown in Fig. 10, an array of comparators lie between the input and an array of shift registers. An input datum X is compared to the currently held values in the shift register array, all previously stored data points are compared with the incoming datum and all stored data that is smaller than the new datum are shifted downwards, the new datum is placed into the empty register, between the values which are immediately larger and smaller than it. This method of sorting achieves an effective sorting time of 0 as by the time the final point of the sample for the DBSCAN operation enters the system the data is already sorted and can

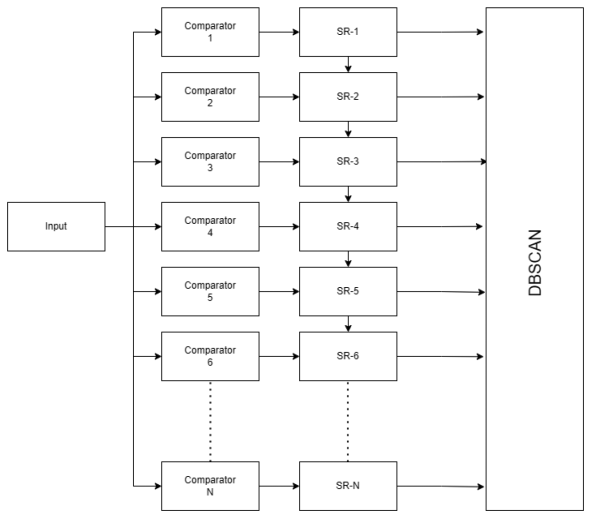

FIGURE 10. Real time sorting block.

Algorithm 1: Algorithm for optimized 1D DBSCAN.

```
$\varepsilon \leftarrow 8, minPts \leftarrow 3$
$ClusterCount \leftarrow 0, PointCount \leftarrow 0$
$Data[50] \leftarrow Input[50]$
for $i = 1$ to 50 do
  $Data[i] - Data[i-1] = Diff$
  if $Diff \leq \varepsilon$ then
    $PointCount{+}{+}$
  else
    if $PointCount \geq minPts$ then
      $ClusterCount{+}{+}$
      $PointCount \leftarrow 0$
    else
      $ClusterCount \leftarrow 0$
      $PointCount \leftarrow 0$
    end if
  end if
end for
$Output \leftarrow ClusterCount$
```

move on into the DBSCAN block, the sorting system can then begin sorting the next set of incoming data.

A major consideration of the DBSCAN algorithm is the values of the ε and MinPts hyper parameters. Optimal ε values vary between datasets and can have a large impact on classification performance. For instance, a choice of ε which is too high can allow outlier noise points to 'bridge' the gap between two constellation clusters which makes the algorithm combine the two clusters into one. Conversely, a ε which is too low can cause a single cluster to be counted as multiple or none at all. A case where this can cause an issue is that different SNR values introduce different values of separation between points as well as constellations themselves, meaning that an optimal ε value for 20 dB data will not be optimal for 5 dB. To counter this, the output of the absolute value and argument LUTs were scaled to between +-127 for all input values, this normalisation allowed a ε value of 5 to work optimally for all SNRs.

Similarly, the minPts optimal value can differ depending on the number of samples used per classification, the number of constellations expected in a modulation scheme, the ratio between these two values, and finally the SNR of the signal. Choosing too high of a minPts value leads to clusters potentially not being found, too low of a value can lead to randomly occurring noise clusters being treated as constellations. In testing the value of this hyperparameter was found to be less important than ε. As a small sample size of 50 datapoints was used to reduce latency and implementation size, it was found that noisy points were very unlikely to be classified as an extra constellation and minPts could be kept to small values such as 2 or 3.

DBSCAN is implemented as in Fig. 11. An algorithmic representation can be seen in Algorithm 1. Data is input serially from the sorting block, incident point N-1 is subtracted from the previous point N. The difference is compared with ε, should the difference be smaller than ε the point counter will increment, if not the point counter resets. When the point counter resets, its value is compared with minPts, if the count of points in the cluster is greater than minPts then the cluster count the will increment, otherwise the count remains the same. The system output is the cluster count after 50 operations.

This combination of real-time sorting and modified DBSCAN achieves an algorithmic complexity of $O(n)$ and allows for complete pipelining of the preprocessing system. This can be seen clearly in Algorithm 1, the DBSCAN algorithm has been reduced to 50 loops or 50 clock cycles. As soon as the sorting process completes the data is serially output and the empty registers filled with a new set of data. The time taken from when the first datum enters the system to achieving a DBSCAN result is $2N$ clock cycles, where N is the number of datapoints chosen for the DBSCAN calculation. This also achieves a significant reduction in implementation size and power consumption as the algorithm is reduced to a subtraction, 2 comparisons, and 2 counters.

C. MLP

The outputs of the DBSCAN algorithm enter the final stage of the system which is the MLP classifier. The MLP takes the number of different argument and absolute value clusters and predicts the modulation scheme. The number of nodes in the MLP is as follows: 2 in the input layer, 4 in the single hidden layer, and 4 at the output for the 4 different modulation schemes used in training, each output node is followed by a logistic outfunction that is calculated with a LUT. The largest value of the output nodes is taken as the classification result. Training of the MLP was performed off the FPGA in software using MATLAB, the weights and biases were exported from MATLAB and stored on the FPGA in ROM. The MLP features a 64-bit 2's complement datapath with a fixed point set

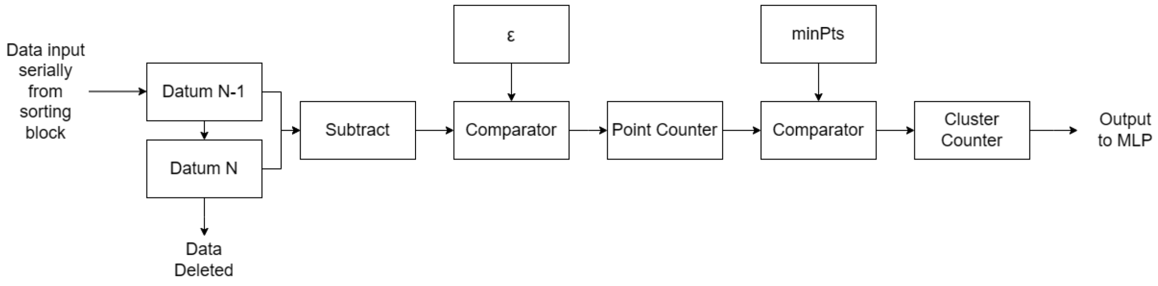

FIGURE 11. **1D DBSCAN optimised architecture.**

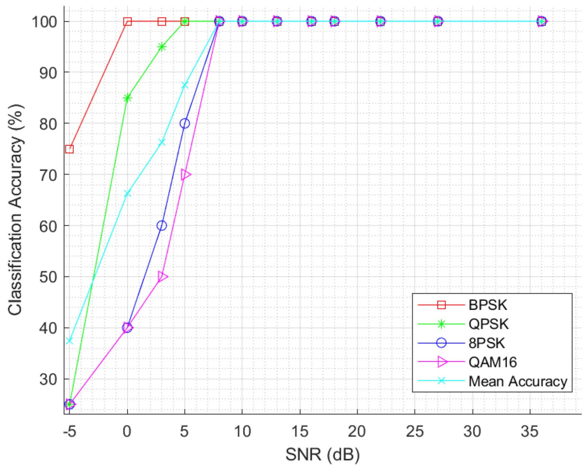

FIGURE 12. **Graph of classification accuracy against SNR of recorded signal in dB.**

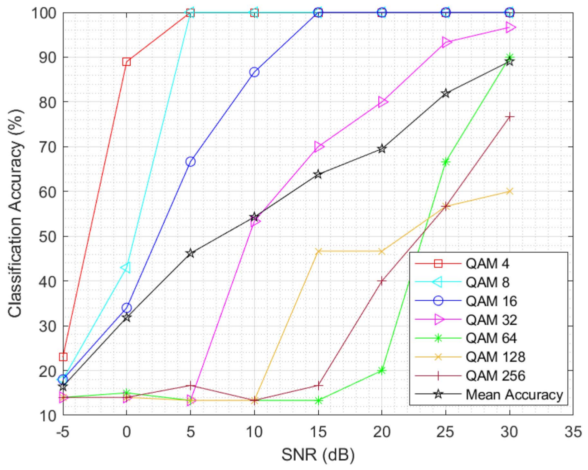

FIGURE 13. **Graph of classification accuracy against SNR of software generated signal in dB.**

after the 32nd bit. A datapath of at least this size was found to be a requirement to maintain the expected performance as it eliminated overflow issues, but more importantly, the precision of the weights and intermediate values needed to be as similar as possible to those in the software simulation. Weights and biases were stored to 16-bit precision. The output takes the 8 most significant bits of the 64-bit datapath results. Training of the model in software was performed using 30000 data points per modulation scheme per SNR value, totalling 720000 data points split into samples of 50, and therefore 14400 overall samples.

V. RESULTS

This section presents the accuracy and FPGA implementation characteristics. Section V-A provides the accuracy of the system across a range of SNRs, in Section V-B an overview of the hardware is found.

A. ACCURACY

The FPGA implementation of the proposed RF classifier was developed and evaluated using a Xilinx Zedboard. Fig. 12 presents the classification accuracy of the implemented RF classifier as a function of SNR. It can be observed that the classifier achieves 100% accuracy for all SNRs above 8 dB. At SNRs below 8 dB, the classification accuracy of 8PSK and 16QAM modulation schemes degrades severely, this is primarily due to the increasing effect of noise causing constellations to begin to overlap, the majority of 8PSK and 16QAM signals which were incorrectly classified were predicted to be QPSK signals. At 5 dB QPSK classification accuracy begins to decrease, likewise after 0 dB BPSK performance degrades. At -5 dB the accuracy of QPSK, 8PSK, and 16QAM becomes no better than a random guess while the performance of BPSK classification drops to 75%. Fig. 13 displays the classification accuracy against SNR for orders of QAM from 4 to 256. These results are obtained from using MATLAB generated waveforms and were included to illustrate how the performance of this system degrades as modulation complexity increases. The graph shows that the classification accuracy decreases as modulation order increases. From the graph it can be seen that the 4, 8, and 16QAM curves are similar but slightly less accurate than the results found for QPSK, 8PSK, and 16QAM in Fig. 12. This is attributed to the values of the ε and $minPts$ hyperparameters being slightly varied to 3 and 2 respectively for this test. This was required to tune the system for the higher order modulated data but came at a cost of slightly worse performance for the low order modulated data. The 32QAM curve shows the system has the ability to recognize and classify this modulation scheme with the

TABLE 1. Resource Utilization for FPGA Implementation

Resource	Utilization	Board Resources (%)
LUT Elements	12,963	24.37
Flip-Flops	2,350	2.21
DSP Units	38	17.27
BRAM	0	0

TABLE 2. Power Consumption of the FPGA Implementation

Element	Power Consumption (mW)
Clocks	5
Signals	12
Logic	11
DSP	11
Static	139
Processor	1,526
Dynamic Total	39
FPGA Total	178
Overall Total	1,704

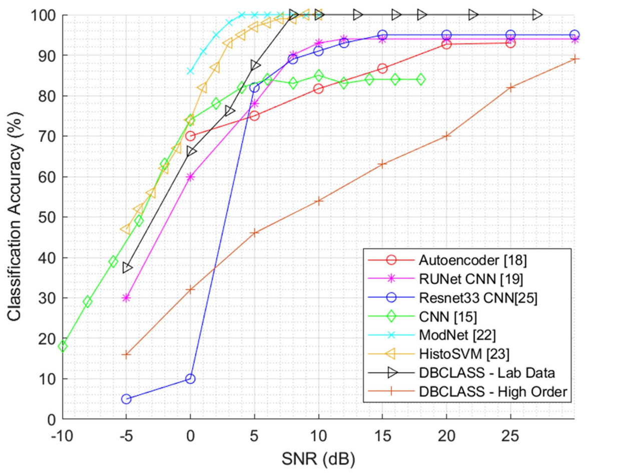

FIGURE 14. **Graph of comparison of classification accuracy against SNR in dB of this work and the state-of-the-art using recorded data.**

accuracy starting at 96% at 30 dB SNR. As the SNR decreases the 32QAM curve follows a similar trend to that of the lower order modulation's curves but reaches 14% accuracy at 5 dB rather than the −5 of that of 4, 8, and 16QAM, for these tests 14% is taken as being no better than a random guess between 7 classes. 64, 128, and 256QAM begin with strong classification accuracy at 30 dB SNR but performance quickly degrades as SNR decreases. Beyond this trend there is no other particular trend that can be observed from the three highest order modulated data's curves, the lines overlap and the strongest performer varies across SNRs. The weak performance shown by these curves is explained by the clustering system's inability to handle the densely spaced constellation diagrams of these modulation schemes, even at 30 dB there is overlap between constellations, at 20 dB and lower there is so much overlap that accurate clustering becomes difficult. Figs. 15 and 16 show the classification accuracy of each modulation scheme used in this work at 8 dB and −5 dB SNR. 8 dB is the lowest SNR at which 100% accuracy is achieved by the classifier and as can be seen in Fig. 14 each sample is correctly classified. Fig. 15 shows the classification accuracy at −5 dB, the system only correctly classifies each sample 25% of the time, as can be seen from the number of blue and red matrix elements, which is equal to a random guess, meaning that the system ceases to function at all at this SNR, apart from for BPSK which still maintains 82% accuracy.

This work has also shown to be carrier-frequency-offset (CFO) resistant, the lab recorded datasets featured significant CFO and there was no reduction in performance detected in comparison to the MATLAB generated data. This is primarily due to the system operating on small batches of data, so as long as CFO is not significant enough to cause distortion within a 50 sample window, the effect of CFO is negligible.

B. HARDWARE PERFORMANCE

In this section, the results of the FPGA implementation of the machine learning classifier using a ZedBoard with a Zynq-7000 SoC XC7Z020-CLG484-1 are presented. For testing a Zedboard was connected to a PC via UART, this connection was used to transmit and receive the recorded signals and classification outputs. Implementation statistics were obtained via the Vivado 2021.2 implementation reporting tools. The implementation utilized a total of 12,963 (24.37%) LUT elements, 2,350 (2.21%) flip-flops, and 38 (17.27%) DSP units. No BRAM usage is required for this system. The detailed resource utilization is summarized in Table 1.

The power consumption values for various components of the implemented classifier are summarized in Table 2. As shown in Table 2, the total power consumption of the implemented classifier is 1,704 mW. The power consumption is primarily dominated by the processor, consuming 1,526 mW. The other components, such as clocks, signals, logic, DSP, and static, exhibit a dynamic power consumption of 39 mW and a static power consumption of 139 mW.

VI. RESULTS COMPARISON

In this section the results from testing are compared to the state-of-the-art examples from the literature. Section VI-A begins by contextualising the hardware utilization. Section VI-B compares the accuracy of the system.

A. HARDWARE COMPARISON

Table 3 displays a comparison of the state of the art RF classifier implementations. In terms of total FPGA resources used there is no system of comparable size and efficiency to DBCLASS, column 5 shows that this work achieves a 3.65 times reduction in total resources used compared to the next smallest. Furthermore, this system uses the second least number of registers (although the design with the least number of registers has a non-traditional structure which mainly utilizes RAM [23]). Against the traditional CNN designs this work exhibits a 6.9 times reduction in registers required by the next smallest. Similarly, the number of LUT elements required also show a 2.6 times reduction to RUNET [19]. The lack

TABLE 3. Comparison of Resource Utilization for FPGA Implementation

Design	Registers	LUT	DSP	RAM	Total Resources Used	Frequency (MHz)	Latency (μs)	Power(W)	FPGA
FNN[13]	16222	158435	210	117380	292247	-	24	1.152	Ultrascale+
CNN[14]	217000	124000	1496	542	343038	-	8	11.11	XCZU28DR-FFVQ1517-2-E
CNN[15]	57726	74680	1116	14832	148354	70	26.78	0.847	Ultrascale+ ZCU104
CNN[16]	198000	121000	710	162	319872	250	8	-	Ultrascale+ RFSoC ZCU111
RUNet[19]	21357	34563	0	40	55960	-	7.5	-	Ultrascale+ RFSoC ZCU111
Histo-SVM[23]	462	-	-	61444	61906	50	4000	-	Altera Cyclone II EP2C20F484C
DBCLASS (This work)	**2350**	**12963**	**38**	**0**	**15351**	**50**	**1**	**0.178**	**XC7Z020-CLG484-1**

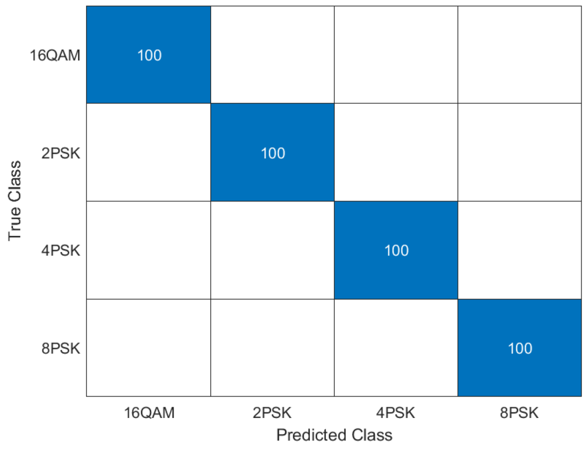

FIGURE 15. Confusion matrix of accuracy at 8 dB SNR.

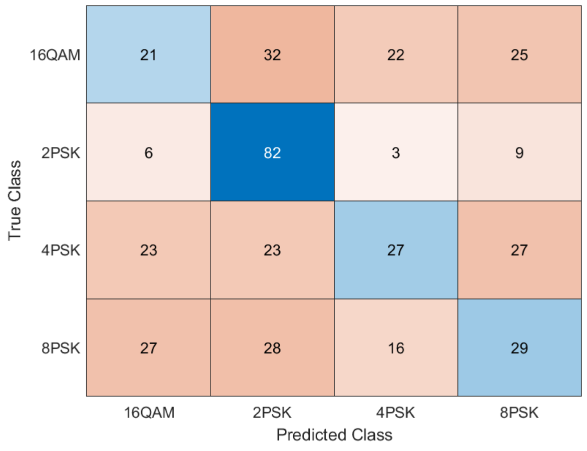

FIGURE 16. Confusion matrix of accuracy at −5 dB SNR.

of DSP usage of [19], [23] means that they have less DSP usage than in this work but of the designs which use DSP blocks DBCLASS is the lowest. Finally, DBCLASS requires no RAM. It is worth noting that some papers such as that of J.Zhao et al. [2] report comparable implementation sizes to this work but these results were discounted from Table 3 in order to maintain a fair comparison with the complete systems discussed here, this is due to the large amount of preprocessing which was done in software on a PC which will lead to a smaller implementation size. In summary, DBCLASS utilizes the least number of FPGA elements in 3 of 4 categories and the lowest total number of elements.

DBCLASS exhibits the quickest performance which is shown by a latency of 1 μs, which is 7.5 times quicker compared to the next quickest [19]. Similarly, the power consumption of this design is 4.75 times less than the next most efficient. This is to be expected due to the smaller implementation size and lack of requirements for memory accesses of this work. Further gains could be made to the latency as this design is limited to 50 MHz due to the longest critical path length. By further pipelining the sorting block, significant gains could be made to the longest path, therefore reducing latency via increasing clock speed. Conversely, using a larger sample size of data for classification will necessitate a larger implementation and latency, as the latency of the system is equal to $2N + 3$, where N equals the sample size. This work has thus shown that in a 50 sample DBSCAN configuration, it is the quickest, most efficient, and smallest design in comparison to all others found in the literature.

B. ACCURACY COMPARISON

Fig. 14 shows a comparison graph of the accuracy across a range of SNRs of this work and the state-of-the-art. The accuracy is taken as an average of each system's accuracy across a range of modulation schemes, it is important to note that each work uses a different combination of modulation schemes for testing. In general across all works, higher order modulation schemes show reduced performance in the presence of noise, due to the more densely spaced constellation diagrams featuring overlapping constellations more readily. Papers [19] and [25] use the largest number of modulation schemes for testing, consisting of a set of 24 different schemes including high order modulations of 256, 128, and 64QAM, utilizing these high order schemes in testing will naturally introduce a penalty to the average system classification accuracy due to the previously mentioned overlap in denser constellation diagrams, in this case 5 out of 24 total modulation schemes used in these papers are of order 64 and above. Conversely, [18], [23] use a maximum of 16QAM, [15], [22] use a maximum of 64QAM. Due to this, these works are expected to have a higher average accuracy due to the higher order datasets used.

Above 8 dB our work, HistoSVM [23], and ModNet [22] all exhibit 100% accuracy, beating the next most accurate designs RUNet [19] and Resnet33 [16]. Of all existing hardware models in the literature ModNet [22] achieves 100% accuracy at the lowest SNR, their work achieves perfect classification until 4 dB, at which point the accuracy begins to degrade. This work achieves an accuracy trend similar to that of HistoSVM, RUNet and ResNet, at 8 dB the performance of the DBSCAN system begins to decrease. This trend of reduced accuracy then continues until at -5 dB the accuracy becomes no better than a random guess for 3 of the 4 modulation schemes used for testing. This is shown in Fig. 12, in which the accuracy on BPSK and QPSK data remains at 100% yet the system cannot maintain perfect accuracy for the more densely spaced 8PSK and 16QAM modulation schemes. Although the DBSCAN system matches the trends seen in both ResNet and RUNet, it consistently achieves greater classification accuracy at all SNRs. At 0 dB the autoencoder accuracy is greater than the DBSCAN model in this work, but the performance remains comparable. Although this work is shown to have a greater accuracy than [19], [25], these two papers feature tests on high order modulation schemes.

A more apt comparison may be with the testing of this work on generated data which includes higher order modulated signals. However this comparison is still not 1-to-1 as the modulations of 64, 128, and 256QAM constitute 42% of this work's average and 21% of the average of [19] and [25]. We made the choice to not include the amplitude and frequency modulated schemes which feature in [19], [25], as DBCLASS had been designed to work on QAM and PSK only, this is due to the fact that these are the modulation types that modern communication systems predominantly use [1]. The accuracy curve for our work's data tests across all SNRs is lower than that of other work found in the literature. This is primarily due to the degradation of performance at very high modulation orders. Figs. 12 and 13 show that the performance on low order modulated data remains comparable regardless of whether the data is recorded or simulated owing to the similar curves across modulation orders included in both tests. Therefore DBCLASS remains competitive in terms of accuracy on low order modulated data but performance decreases at higher orders.

C. COMPARISON SUMMARY

The hardware comparisons discussed in Section VI-A conclusively show that this work is the smallest, quickest, and most efficient system for automatic modulation classification. The accuracy comparisons of Section VI-B demonstrated that when working with M-PSK and M-QAM modulation schemes where M is less than or equal to 16, the DBSCAN system of this work is competitive in terms of classification accuracy. When M is above 16 performance is seen to deteriorate. Therefore it can be concluded that this DBSCAN based modulation classifier is the optimal choice for a low-power, low-area, low-latency design working on low order modulated data.

VII. CONCLUSION

The paper presents a novel FPGA-based implementation of a machine learning classifier for RF modulation classification. An introduction to, and comparison of, the state of the art is presented and clustering is proposed as an improved method to achieve classification, DBSCAN was identified as the ideal algorithm. Additional optimisations to the DBSCAN algorithm lead to large improvements in the delay, size, and power consumption of the system. The latency was found to be 7.5 times lower than the next fastest work [19]. Similarly the design consumed 4.75 times less power than the most efficient system in the literature [15]. This work also required the second least number of registers [23], the second smallest number of LUTs [2] by 2.6 times, the second least number of DSP slices [19], and no RAM. The DBCLASS was found to have the smallest implementation by 3.65 times on aggregate in comparison to the next smallest work in the literature. Thus, to the best of the authors knowledge, this work's design has been shown to be the smallest, fastest, and most efficient, as well as being 100% accurate above 8 dB when using modulation schemes of orders below 16. The DBCLASS design is therefore the optimal choice for engineers working with low-power devices on real-time data-streams at noise levels above 8 dB.

ACKNOWLEDGMENTS

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

[1] C. Zhang, Y.-L. Ueng, C. Studer, and A. Burg, "Artificial intelligence for 5G and beyond 5G: Implementations, algorithms, and optimizations," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 2, pp. 149–163, Jun. 2020, doi: 10.1109/JETCAS.2020.3000103.

[2] J. Zhao, Y. Zhao, H. Li, Y. Zhang, and L. Wu, "HLS-Based FPGA implementation of convolutional deep belief network for signal modulation recognition," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2020, pp. 6985–6988, doi: 10.1109/IGARSS39084.2020.9324385.

[3] M. Bkassiny, "A deep learning-based signal classification approach for spectrum sensing using long short-term memory (LSTM) networks," in *Proc. 6th Int. Conf. Inf. Technol., Inf. Syst. Elect. Eng.*, 2022, pp. 667–672, doi: 10.1109/ICITISEE57756.2022.10057728.

[4] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander, "A distribution-based clustering algorithm for mining in large spatial databases," in *Proc. 14th Int. Conf. Data Eng.*, 1998, pp. 324–331, doi: 10.1109/ICDE.1998.655795.

[5] Y. Wang et al., "A survey on deploying mobile deep learning applications: A systemic and technical perspective," *Digit. Commun. Netw.*, vol. 8, pp. 1–17, 2022.

[6] A. K. Nandi and E. E. Azzouz, "Algorithms for automatic modulation recognition of communication signals," *IEEE Trans. Commun.*, vol. 46, no. 4, pp. 431–436, Apr. 1998, doi: 10.1109/26.664294.

[7] D. Boutte and B. Santhanam, "A feature weighted hybrid ICA-SVM approach to automatic modulation recognition," in *Proc. IEEE 13th Digit. Signal Process. Workshop 5th IEEE Signal Process. Educ. Workshop*, 2009, pp. 399–403, doi: 10.1109/DSP.2009.4785956.

[8] D. Saharia, M. R. Boruah, N. K. Pathak, and N. Sarma, "An ensemble based modulation recognition using feature extraction," in *Proc. Int. Conf. Intell. Technol.*, 2021, pp. 1–6, doi: 10.1109/CONIT51480.2021.9498547.

[9] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, Sep. 2018, doi: 10.1109/TCCN.2018.2835460.

[10] Z. Ke and H. Vikalo, "Real-time radio technology and modulation classification via an LSTM auto-encoder," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 370–382, Jan. 2022, doi: 10.1109/TWC.2021.3095855.

[11] V.-S. Doan, T. Huynh-The, C.-H. Hua, Q.-V. Pham, and D.-S. Kim, "Learning constellation map with deep CNN for accurate modulation recognition," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6, doi: 10.1109/GLOBECOM42002.2020.9348129.

[12] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4074–4077, Apr. 2019, doi: 10.1109/TVT.2019.2900460.

[13] 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,” in *Proc. IEEE Mil. Commun. Conf.*, 2019, pp. 1–6, doi: 10.1109/MILCOM47813.2019.9021098.

[14] S. Tridgell, D. Boland, P. H. W. Leong, and S. Siddhartha, “Real-time automatic modulation classification,” in *Proc. Int. Conf. Field- Program. Technol.*, 2019, pp. 299–302, doi: 10.1109/ICFPT47387.2019.00052.

[15] A. Emad et al., “Deep learning modulation recognition for RF spectrum monitoring,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2021, pp. 1–5, doi: 10.1109/ISCAS51556.2021.9401658.

[16] S. Tridgell, D. Boland, P. H. W. Leong, R. Kastner, A. Khodamoradi, and Siddhartha, “Real-time automatic modulation classification using RFSoC,” in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2020, pp. 82–89, doi: 10.1109/IPDPSW50202.2020.00021.

[17] G. J. Mendis, J. Wei-Kocsis, and A. Madanayake, “Deep learning based radio-signal identification with hardware design,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 55, no. 5, pp. 2516–2531, Oct. 2019, doi: 10.1109/TAES.2019.2891155.

[18] M. A. Azza, A. E. Moussati, and O. Moussaoui, “Implementation of an automatic modulation recognition system on a software defined radio platform,” in *Proc. Int. Symp. Adv. Elect. Commun. Technol.*, 2018, pp. 1–4, doi: 10.1109/ISAECT.2018.8618837.

[19] S. Kumar, R. Mahapatra, and A. Singh, “Automatic modulation recognition: An FPGA implementation,” *IEEE Commun. Lett.*, vol. 26, no. 9, pp. 2062–2066, Sep. 2022, doi: 10.1109/LCOMM.2022.3184771.

[20] Rohde & Schwarz, “R&SSMB100A Microwave Signal Generator,” Datasheet Version 3.0, Rohde & Schwarz Headquarters, Munich, Germany, Nov. 2023.

[21] Keysight, “X-series signal analyzer N9030B PXA signal analyzer,” Datasheet 5992-1317EN, Keysight Headquarters, Santa Rosa, CA, USA, Apr. 2023.

[22] S. Kumar, A. Singh, and R. Mahapatra, “Hardware implementation of automatic modulation classification with deep learning,” in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst.*, 2019, pp. 1–6, doi: 10.1109/ANTS47819.2019.9118057.

[23] C. Cardoso, A. R. Castro, and A. Klautau, “An efficient FPGA IP core for automatic modulation classification,” *IEEE Embedded Syst. Lett.*, vol. 5, no. 3, pp. 42–45, Sep. 2013, doi: 10.1109/LES.2013.2274793.

[24] A. F. D Castro, R. S. R. Milléo, L. H. A. Lolis, and A. A. Mariano, “Artificial neural network based automatic modulation classification system applied to FPGA,” in *Proc. 34th SBC/SBMicro/IEEE/ACM Symp. Integr. Circuits Syst. Des.*, 2021, pp. 1–6, doi: 10.1109/SBCCI53441.2021.9529976.

[25] T. J. O’Shea, T. Roy, and T. C. Clancy, “Over-the-air deep learning based radio signal classification,” *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 168–179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.

[26] K. M. A. Patel and P. Thakral, “The best clustering algorithms in data mining,” in *Proc. Int. Conf. Commun. Signal Process.*, 2016, pp. 2042–2046, doi: 10.1109/ICCSP.2016.7754534.

[27] “18650 3.7V 4400mAh product specification,” *Hunan Sounddon New Energy Co., Ltd.*, China, 2023. Accessed: Oct. 27, 2023. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/18650%204400mAh.pdf

BILL GAVIN recieved the M.Eng. (first class) in 2020 from The University of Sheffield, Sheffield, U.K., where he has been working toward the Ph.D. degree covering machine learning applied to the physical layer of communications systems, since 2021.

TIANTAI DENG (Member, IEEE) received and the B.Eng. from the Harbin Institute of Technologies, Harbin, China, in 2015, and the Ph.D. degree from Queen’s University Belfast, Belfast, Ireland, in 2019. He is currently a Lecturer with the University of Sheffield, Sheffield, U.K. Prior his career as an academic, he was a Senior Engineer with HiSilicon, Huawei. His main research interests include hardware acceleration for image processing, deep learning and high-level design environments.

EDWARD BALL (Member, IEEE) and was born in Blackpool, United Kingdom in November 1973. He received the Master of Engineering (with a 1st class) degree in electronic systems engineering from the University of York, York, U.K., in 1996. After graduating, he was with industry for 20 years, first spending 15 years working as Engineer, Senior RF Engineer and finally Principal RF Engineer with Cambridge Consultants Ltd in Cambridge, U.K. He then spent five years as Principal RF Engineer and Radio Systems Architect at Tunstall Healthcare Ltd in Whitley, U.K. In 2015, he joined the Department of Electronic and Electrical Engineering at the University of Sheffield, Sheffield, U.K., where he is currently a Reader in RF engineering. His research interests include radio technology, from RF system design, RF circuit design (sub-GHz to mm-wave) and the application of radio technology to real-world industrial and commercial problems. He has a particular passion for RF hardware design. Mr. Ball is a member of the IET and is a Chartered Engineer. He became a Member of IEEE in April 2008