HLS BASED SIGNAL CLASSIFICATION SYSTEM USING DEEP NEURAL NETWORKS

MINI PROJECT SUBMITTED IN
PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE
DEGREE OF **BACHELOR OF ENGINEERING** IN
ELECTRONICS AND COMMUNICATION ENGINEERING
OF THE ANNA UNIVERSITY

MINI PROJECT WORK

2023

Submitted by

CHANDRASEKAR S M
2014104
PRANAV S
2014126
SANDEEP SINGH
2014128
SURIYA N
2014134

Under the Guidance of

Dr. G. SUCHITRA
M.E., Ph.D.,
Associate Professor, ECE

ELECTRONICS AND COMMUNICATION ENGINEERING
GOVERNMENT COLLEGE OF TECHNOLOGY
(An Autonomous Institution affiliated to Anna University)
COIMBATORE - 641 013

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University)

COIMBATORE - 641 013

MINI PROJECT WORK

DECEMBER 2023

This is to certify that this mini project work entitled

HLS BASED SIGNAL CLASSIFICATION SYSTEM
USING DEEP NEURAL NETWORKS

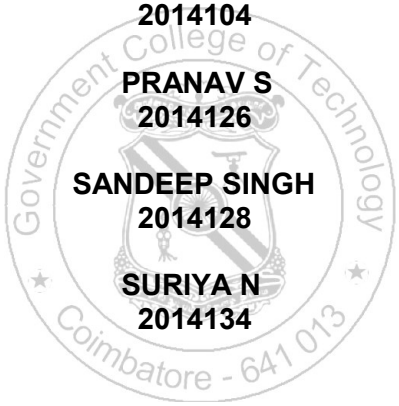
is the bonafide record of mini project work done by

CHANDRASEKAR S M
2014104

PRANAV S
2014126

SANDEEP SINGH
2014128

SURIYA N
2014134



of **B.E.** (ELECTRONICS AND COMMUNICATION ENGINEERING) during the year 2023 – 2024

PROJECT GUIDE	HEAD OF THE DEPARTMENT
DR. G. SUCHITRA M.E., Ph.D.,	**DR. C. SANTHI M.E., Ph.D.,**

Submitted for the mini project viva-voce examination held on _____

Internal Examiner	External Examiner

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We oblige ourselves to the Almighty who has given us all the strengths and knowledge to complete this Final Year Mini Project work.

We wish to thank **Dr. K. MANONMANI,** Principal, Government College of Technology, Coimbatore for providing us the necessary facilities for completing the Mini Project work.

We express our heartiest thanks to **Dr. C. SANTHI,** Head of the Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore for providing us the guidelines which are helpful in the Mini Project work.

We wish to enunciate our deep sense of gratitude and loyalty to our Project Guide **Dr. G. SUCHITHRA,** Associate Professor in Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore for being the beacon of guidance of our Mini Project work and motivating us in every possible manner, which led to the successful completion of the project work.

We wish to articulate our thanks to our project committee members **Dr. M. SANTHI,** Professor & HoD (PG) in Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore and **Prof. S. MATHIVANAN,** Assistant Professor in Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore for their ideas in improving this project work.

We also thank the technical and non-technical staff in the Department of Electronics and Communication Engineering, Government College of Technology, Coimbatore for their support.

Finally, we are also grateful to our parents and friends for their motivation, blessings, affection, and their loving co-operation from the day of this academic venture.

ABSTRACT

ABSTRACT

In the non-cooperative channel communication scenario, there comes a need to recognize and classify the modulation format of the received signals to identify the communication targets and to have better management over them. To solve problems such as high complexity, low accuracy and hard process of manual extraction of signal features we can go for a Deep Neural Network based modulation format identification block so that we can effectively classify received signals and apply specific demodulation technique to it. The proposed project will implement such a Neural Network to advanced 4G and 5G modulated signals and implement it on an FPGA board using High Level Synthesis for Machine Learning (hls4ml) concept.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

CHAPTER 1
INTRODUCTION

In the ever-evolving landscape of wireless communication, the need for robust and adaptive systems capable of accurately classifying modulated signals is paramount. The Signal Classification System proposed in this thesis represents a significant leap forward in artificial intelligence, leveraging deep neural networks to classify received signals based on their modulation techniques. With a focus on High-Level Synthesis (HLS), this system is designed to provide real-time, intelligent signal classification that extends beyond conventional approaches.

The primary objective of this project is to develop a Neural Network-based Artificial Intelligence system adept at discerning the modulation techniques employed during signal transmission. This capability holds immense importance, especially in scenarios where signals are subject to noise, interference, or other distortions. The proposed system aims to not only classify signals accurately but also to facilitate the application of appropriate demodulation techniques, thereby enhancing the processing of digital signals even in challenging communication environments.

Furthermore, the flexibility and efficiency of this Signal Classification System make it an ideal candidate for implementation on Field-Programmable Gate Array (FPGA) boards. By doing so, the system can seamlessly integrate as an Artificial Intelligent Block within communication systems, contributing to the creation of adaptive and intelligent communication networks. This integration holds the potential to revolutionize signal processing, paving the way for improved communication reliability and performance in diverse and dynamic environments.

As we delve into the intricacies of HLS-based signal classification, this thesis explores the design, implementation, and optimization aspects, shedding light on the novel approaches employed to achieve a high level of accuracy and efficiency. The significance of this research extends beyond theoretical advancements, as the outcomes have practical implications for the development of intelligent communication systems capable of adapting to the challenges posed by real-world signal propagation.

LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

[1] ***T. J. O'Shea, T. Roy and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," in IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 168-179, Feb. 2018, doi: 10.1109/JSTSP.2018.2797022.***

The paper titled "Over-the-Air Deep Learning Based Radio Signal Classification" by Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy serves as a valuable reference in understanding the performance aspects of deep learning-based radio signal classification. In this literature survey, we extract key insights obtained from this paper, particularly in the context of our project on HLS-Based Signal Classification System Using Deep Neural Networks.

1. Baseline Methodology:

- The paper introduces a rigorous baseline method that combines higher-order moments and strong boosted gradient tree classification for radio signal classification.
- This baseline method is utilized as a benchmark to evaluate the performance of deep learning-based approaches, providing a comparative analysis across various configurations and channel impairments.

2. Performance Evaluation:

- Performance evaluations are conducted considering the impact of carrier frequency offset, symbol rate, and multipath fading through simulation.
- Over-the-air measurements are performed in a laboratory setting using software radios, providing real-world insights into radio signal classification performance.
- The study explores the comparison of performance and training strategies between the baseline method and deep learning-based approaches.

3. Key Findings:

- The paper presents findings on the effectiveness of deep learning techniques in radio signal classification, highlighting their performance in comparison to traditional methods based on higher-order moments and boosted gradient trees.

- The effects of varying parameters such as carrier frequency offset, symbol rate, and multipath fading on the classification performance are systematically analyzed.

4. Remaining Challenges and Design Considerations:

- The paper concludes with a discussion on remaining challenges in the field of radio signal classification, indicating avenues for further research.

- Design considerations for the application of deep learning techniques in radio signal classification are addressed, providing valuable insights for researchers and practitioners.

[2] ***J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers and Z. Zhang, "High-Level Synthesis for FPGAs: From Prototyping to Deployment," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, pp. 473-491, April 2011, doi: 10.1109/TCAD.2011.2110592.***

The literature survey for our project draws insights from the paper titled "High-Level Synthesis for FPGAs: From Prototyping to Deployment." This paper explores the evolving landscape of high-level synthesis (HLS) methodologies and their applicability, with a specific focus on field-programmable gate array (FPGA) designs. The findings from this paper contribute valuable perspectives on the advancements in HLS tools, their effectiveness in FPGA designs, and the potential benefits in terms of resource utilization and design productivity.

The paper addresses the escalating complexity in system-on-chip (SoC) design and the increasing need to elevate the level of abstraction beyond the traditional register transfer level (RTL). Despite initial challenges faced by early generations of commercial HLS systems, the authors argue that a tipping point for transitioning to

HLS methodology is currently underway, particularly for FPGA designs.

Key insights obtained from the paper include:

1. Progress in HLS Tools:

- The latest generation of HLS tools, exemplified by AutoESL's AutoPilot HLS tool, has made significant strides in providing broad language coverage and robust compilation technology.
- These tools also incorporate platform-based modeling, core HLS algorithm enhancements, and a domain-specific approach to address the complexities of modern FPGA designs.

2. Demonstration of Effectiveness:

- The paper uses AutoPilot HLS tool in conjunction with domain-specific system-level implementation platforms developed by Xilinx to demonstrate the effectiveness of state-of-the-art C-to-FPGA synthesis solutions.
- Case studies involving complex industrial designs targeting Xilinx FPGAs are presented, showcasing the application of HLS solutions compared to optimized manual designs.

3. Resource Utilization and Productivity Improvement:

- A significant experiment on a sphere decoder reveals that the HLS solution can achieve an 11–31% reduction in FPGA resource usage.
- Design productivity is also enhanced compared to hand-coded designs, indicating the potential for streamlining the design process and accelerating time-to-market.

[3] ***J. Zhao, Y. Zhao, H. Li, Y. Zhang and L. Wu, "HLS-Based FPGA Implementation of Convolutional Deep Belief Network for Signal Modulation Recognition," IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, Waikoloa, HI, USA, 2020, pp. 6985-6988, doi: 10.1109/IGARSS39084.2020.9324385.***

The research paper titled "HLS-Based FPGA Implementation of Convolutional Deep

Belief Network for Signal Modulation Recognition" contributes significantly to the understanding and implementation of deep learning techniques, particularly in the context of Signal Modulation Recognition (SMR) using High-Level Synthesis (HLS) and FPGA technology. The key findings and insights obtained from this paper can be summarized as follows:

1. Application of Deep Learning in Signal Modulation Recognition:

- The paper emphasizes the widespread application of deep learning methods in modern artificial intelligence, particularly in Signal Modulation Recognition (SMR). Deep neural networks, such as Convolutional Deep Belief Networks (CDBN), have proven effective in extracting intricate features from modulated signals for accurate classification.

2. Energy-Efficient FPGA Implementation:

- Acknowledging the energy efficiency and low-latency streaming capabilities of Field-Programmable Gate Arrays (FPGAs), the paper proposes leveraging FPGAs for SMR applications. FPGAs are deemed more suitable for energy-sensitive and real-time machine learning projects compared to traditional Central Processing Units (CPUs) and Graphics Processing Units (GPUs).

3. Role of High-Level Synthesis (HLS):

- The paper introduces the concept of High-Level Synthesis (HLS) as a crucial tool for automatically converting high-level language descriptions into low-level abstraction language representations. HLS is highlighted as a key enabler for efficiently implementing deep learning models on FPGA platforms.

4. Optimization Techniques for FPGA Implementation:

- The authors propose a system that optimizes the Convolutional Deep Belief Network (CDBN) for Signal Modulation Recognition on an FPGA platform. The optimization techniques include loops pipelining and unrolling, memory buffering, and partitioning. These strategies aim to enhance the performance and efficiency of the FPGA-based accelerator.

5. Performance Metrics and Comparison with GPUs:

- The paper provides quantitative insights into the performance of the proposed HLS-based FPGA accelerator. It operates at a clock frequency of 150MHz and demonstrates a 28% higher throughput while consuming 80.5% less power compared to a GPU implementation. This indicates the superior energy efficiency of the FPGA-based approach.

6. Platform Specifics:

- The research focuses on the implementation of the proposed system using the Virtex-7 platform, showcasing the practicality and feasibility of deploying HLS-based FPGA accelerators for SMR on specific FPGA architectures.

[4] ***E. Mageiropoulos, N. Chrysos, N. Dimou and M. Katevenis, "Using hls4ml to Map Convolutional Neural Networks on Interconnected FPGA Devices," 2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Orlando, FL, USA, 2021, pp. 277-277, doi: 10.1109/FCCM51124.2021.00062.***

This paper serves as a foundational piece, contributing valuable insights into the realm of High-Level Synthesis (HLS) applied to signal classification through deep neural networks. Key findings and takeaways from this paper include:

1. HLS Architecture for Signal Classification:

- The paper introduces an HLS architecture for signal classification utilizing deep neural networks. The use of HLS emphasizes the importance of hardware acceleration for real-time signal processing, particularly in scenarios with varying modulation techniques and signal characteristics.

2. Partitioning of SqueezeNet CNN on Ultrascale + FPGAs:

- The authors demonstrate the effectiveness of their architecture by partitioning the SqueezeNet Convolutional Neural Network (CNN) across six Ultrascale+ FPGAs.

This highlights the scalability and parallelization potential of FPGA devices for handling complex neural network models.

3. Integration of Keras and hls4ml:

- The paper employs Keras for network definition, showcasing the compatibility of high-level neural network design frameworks with HLS. This integration streamlines the process of defining convolutional neural networks, making it accessible for practitioners with diverse backgrounds.

4. Optimization Challenges with hls4ml:

- An unexpected finding in the study is the suboptimal nature of the original HLS code generated by hls4ml for the intended purpose. This highlights a potential challenge in using off-the-shelf tools and emphasizes the need for optimization to achieve satisfactory results in signal classification tasks.

5. Optimization Strategies using Vivado HLS:

- To address the suboptimal HLS code, the authors apply optimization directives available in Vivado HLS. This strategy involves optimizing each kernel appropriately, demonstrating the importance of fine-tuning and customization in HLS-based implementations.

6. Adaptation of HLS Code for Specific Requirements:

- The paper underscores the necessity of adapting the original HLS code when needed to align with the specific requirements of signal classification. This adaptability is crucial for tailoring the system to handle diverse modulation techniques and signal characteristics effectively.

7. Simplifying the Path from Network Definition to HLS:

The study emphasizes the use of existing tools, such as Keras and hls4ml, to simplify the path from network definition to HLS. This approach facilitates a more user-friendly and accessible workflow for developing FPGA-based signal classification systems.

THEORITICAL FOUNDATIONS

CHAPTER 3
THEORETICAL FOUNDATIONS

3.1 Modulation Techniques and Signal Characteristics:

Introduction

In the realm of wireless communications, the efficient and reliable transmission of information relies heavily on modulation techniques. Modulation refers to the process of varying one or more properties of a carrier signal in accordance with the information being transmitted. This chapter delves into several key modulation techniques and their associated signal characteristics, providing a comprehensive understanding that forms the basis for the HLS (High-Level Synthesis) based signal classification system using Deep Neural Networks.

Modulation Techniques

1. Binary Phase Shift Keying (BPSK) - BPSK is a digital modulation technique where the phase of the carrier signal is altered to represent binary data. Let s(t) represent the BPSK signal, A as the amplitude, fc as the carrier frequency, and b(t) as the binary data. Mathematically, BPSK can be expressed as:

$$s(t)=A\cos(2\pi fct+\pi(1-b(t)))$$

2. Quadrature Phase Shift Keying (QPSK) - QPSK extends BPSK by modulating two independent signals in quadrature, enabling the transmission of two bits per symbol. The signal s(t) is represented as a sum of two BPSK signals with a phase difference of 90 degrees:

$$s(t)=A\cos(2\pi fct+\pi(1-b1(t)))\cos(2\pi fct+\pi(1-b2(t)))$$

3. 8-PSK - 8-PSK further increases data transmission rates by encoding three

bits per symbol. The phase is adjusted to one of eight possible values, enhancing spectral efficiency.

$$s(t)=A\cos(2\pi fct+\pi(1-b1(t)+2(1-b2(t)+4(1-b3(t))))$$

4. Quadrature Amplitude Modulation (QAM) - QAM combines amplitude and phase modulation, enabling the transmission of multiple bits per symbol. QAM16 and QAM64 are common variants with 16 and 64 possible signal states, respectively.

$$s(t)=A\cos(2\pi fct)+B\sin(2\pi fct)$$

5. Continuous Phase Frequency Shift Keying (CPFSK) - CPFSK is a frequency modulation technique with constant phase increments between symbols. The frequency deviation is proportional to the data, making it resistant to phase distortion.

6. Gaussian Frequency Shift Keying (GFSK) - GFSK is a form of frequency-shift keying where the modulating signal is passed through a Gaussian filter to minimize bandwidth. It offers improved spectral efficiency.

7. Pulse Amplitude Modulation (PAM4) - PAM4 is a multi-level signaling scheme that encodes two bits per symbol. The amplitude of the pulse is adjusted to represent different symbol states.

8. Amplitude Modulation (AM) - AM includes Double Sideband (AM-DSB), Single Sideband (AM-SSB), and Wide Band Frequency Modulation (WBFM).

AM-DSB: $s(t)=Ac[1+m(t)]\cos(2\pi fct)$

AM-SSB: $s(t)=Acm(t)\cos(2\pi fct)$

WBFM: $s(t)=Ac\cos(2\pi fct+\phi(t))$

Signal Characteristics

IQ Format: The In-phase (I) and Quadrature (Q) components of a signal, often

represented in a complex number form I+jQ, provide a convenient way to analyze and visualize modulation schemes.

Signal-to-Noise Ratio (SNR): SNR is a crucial parameter indicating the ratio of signal power to noise power. For a signal s(t) and additive white Gaussian noise n(t) , SNR is defined as:

SNR = Power of n(t)/Power of s(t)

Bit Error Rate (BER): BER is a metric quantifying the accuracy of the transmitted data. It is the ratio of incorrectly received bits to the total transmitted bits. For a binary system, it is given by:

BER = Total number of bits transmitted/Number of received errors

Conclusion:

Understanding modulation techniques and their associated signal characteristics is fundamental to the design of an HLS-based signal classification system using Deep Neural Networks. The mathematical premises and STEM details provided in this chapter lay the groundwork for the subsequent development and analysis of the classification system.

3.2: Neural Networks for Signal Classification

This chapter delves into the utilization of Convolutional Neural Networks (CNN), Gated Neural Networks (GATEDNN), and Recurrent Neural Networks (RNN) for the purpose of signal classification in the context of HLS-based systems.

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) have proven to be highly effective in image and signal processing tasks. In the context of signal classification for HLS-based systems, CNNs excel in capturing spatial hierarchies and patterns within

signals. The hierarchical feature extraction mechanism of CNNs allows them to automatically learn and identify intricate patterns, making them well-suited for the diverse range of signals encountered in HLS applications.

The CNN architecture comprises convolutional layers for feature extraction, pooling layers for spatial down-sampling, and fully connected layers for classification. The integration of CNNs in the HLS-based signal classification system enhances the model's ability to discern complex features and patterns, ultimately improving overall classification accuracy.

Gated Neural Networks (GATED DNN)

Gated Neural Networks (GATED DNN) represent a class of neural networks that incorporate gating mechanisms to control the flow of information within the network. These networks, including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are particularly well-suited for sequential data processing. In the context of HLS-based signal classification, GATED DNNs are adept at capturing temporal dependencies and long-range dependencies within the input signals.

The integration of GATED DNNs in the signal classification system allows for the modeling of intricate temporal relationships in HLS signals, which may be crucial for accurate classification. The gating mechanisms enable the network to selectively update and retain information over extended sequences, contributing to enhanced performance in handling time-series data.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) have long been recognized for their ability to model sequential information. In the context of HLS-based signal classification, RNNs are instrumental in capturing dependencies over time, making them suitable for signals with temporal characteristics.

The architecture of RNNs includes recurrent connections that enable the network to maintain a memory of previous inputs. This memory allows RNNs to effectively capture and utilize contextual information in the classification process. Integrating RNNs into the HLS-based signal classification system enhances the model's ability to discern patterns evolving over time, contributing to improved classification accuracy.

Conclusion

This chapter has explored the integration of Convolutional Neural Networks (CNN), Gated Neural Networks (GATEDNN), and Recurrent Neural Networks (RNN) into the HLS-based signal classification system. Each neural network architecture brings unique strengths to the table, addressing different aspects of signal processing and classification. The subsequent chapters will delve into the implementation details, training strategies, and performance evaluations of these networks within the context of the HLS-based signal classification system.

3.3 High-Level Synthesis: Concepts and Applications:

Introduction

High-Level Synthesis (HLS) has emerged as a pivotal technology in the realm of digital system design, offering a bridge between high-level programming languages and hardware description languages. This chapter delves into the fundamental concepts and applications of High-Level Synthesis, laying the groundwork for its integration into the design and implementation of a Signal Classification System using Deep Neural Networks (DNNs).

Overview of High-Level Synthesis

High-Level Synthesis represents a paradigm shift in digital system design by allowing designers to express their intent using high-level programming languages

such as C, C++, or SystemC. HLS tools then automatically translate this algorithmic-level description into an optimized hardware description, typically in the form of RTL (Register-Transfer Level) code. This abstraction enables designers to focus on algorithmic development without being burdened by low-level hardware details.

Key Concepts in High-Level Synthesis

Algorithmic Level Abstraction: HLS enables designers to describe algorithms and functionalities at a higher level of abstraction. This abstraction facilitates faster design iterations and promotes design exploration without the need for in-depth hardware knowledge.

Automatic Parallelization: HLS tools automatically explore parallelism within the algorithmic description, identifying opportunities for concurrent execution. This feature is crucial for accelerating the execution of computationally intensive tasks, such as those encountered in signal processing applications.

Optimizations and Trade-offs: HLS tools offer a range of optimizations, allowing designers to trade-off between factors like latency, throughput, and resource utilization. These optimizations are critical for tailoring the hardware implementation to meet specific application requirements.

Applications of High-Level Synthesis

Digital Signal Processing (DSP): HLS has found extensive applications in DSP, where complex algorithms for tasks like filtering, modulation, and signal classification can be efficiently implemented in hardware. The use of HLS in DSP contributes to faster development cycles and more flexible designs.

Deep Learning Acceleration: With the surge in demand for efficient

implementations of deep neural networks, HLS has become instrumental in accelerating the deployment of DNNs on custom hardware. This is particularly relevant to the development of Signal Classification Systems using DNNs, where the ability to process large datasets in real-time is paramount.

Communication Systems: HLS is widely employed in the design of communication systems, including error correction codes, modulation schemes, and channel encoding. The automatic translation from high-level specifications to hardware implementations enhance the efficiency of communication system designs.

HLS in Signal Classification Systems Using DNNs

Algorithmic Description: HLS facilitates the expression of DNN architectures in high-level languages, allowing designers to focus on the neural network's structure and parameters. This abstraction simplifies the integration of DNNs into signal classification systems.

Performance Optimization: HLS tools optimize the hardware implementation of DNNs, taking into account the parallelism inherent in neural network operations. This optimization contributes to achieving real-time processing capabilities, a crucial requirement for signal classification applications.

Flexibility and Adaptability: The flexibility provided by HLS allows for the Seamless adaptation of the DNN architecture to different signal classification tasks. Designers can experiment with various neural network configurations and easily explore the trade-offs between accuracy and hardware resource utilization.

Conclusion

This chapter provides an in-depth exploration of High-Level Synthesis, laying the foundation for its application in the design and implementation of a Signal

Classification System using Deep Neural Networks. The integration of HLS in this context holds the promise of accelerating the development of efficient and adaptable hardware solutions for signal processing applications. Subsequent chapters will delve into the specific methodologies employed in leveraging HLS for the proposed system, emphasizing the synergy between high-level abstraction hardware performance optimization.

3. 4 Integration of Neural Networks with High-Level Synthesis (HLS):

This chapter delves into the crucial aspect of integrating neural networks with High-Level Synthesis in the context of signal classification. Vivado HLS, a powerful tool in Xilinx's suite, is employed to bridge the gap between traditional hardware design methodologies and the complex requirements of deep neural networks.

1. Introduction

The integration of neural networks with High-Level Synthesis (HLS) represents a pivotal phase in our pursuit of a signal classification system using deep neural networks. In this chapter, we explore the rationale behind incorporating HLS into our project, discuss the challenges involved, and provide insights into the implementation using Vivado HLS.

2. Why High-Level Synthesis?

Traditional hardware design methodologies are often laborious and time-consuming, especially when dealing with complex algorithms like deep neural networks. High-Level Synthesis emerges as a game-changer by allowing designers to express their algorithms at a higher level of abstraction. HLS tools like Vivado HLS convert these high-level descriptions into hardware descriptions, significantly accelerating the design process.

3. Neural Network Integration

Model Selection

The choice of the neural network architecture significantly impacts the

integration process. In our project, we opt for a deep neural network, leveraging the expressive power of architectures like convolutional neural networks (CNNs) or recurrent neural networks (RNNs) based on the nature of the signal classification problem.

Network Quantization

To make the neural network amenable to hardware implementation, quantization becomes essential. Vivado HLS supports fixed-point arithmetic, allowing for the representation of neural network weights and activations in a fixed-point format. This step ensures a balance between accuracy and hardware resource utilization.

4. Vivado HLS Integration

Code Conversion

The neural network model, typically expressed in a high-level language like Python using frameworks such as TensorFlow or PyTorch, must be converted into a form compatible with Vivado HLS. This involves translating the neural network code into a hardware description language (HDL) like Verilog or VHDL.

HLS Directives

Vivado HLS provides directives that guide the synthesis process. These directives help optimize the hardware implementation by specifying aspects such as pipelining, loop unrolling, and dataflow. Careful consideration of these directives is crucial for achieving the desired balance between performance and resource utilization.

Verification and Simulation

Before moving to the hardware implementation, thorough verification and simulation are imperative. Vivado HLS includes simulation capabilities that allow designers to validate the functionality of the synthesized hardware description.

This step is crucial for identifying and rectifying any discrepancies between the original neural network model and the synthesized hardware description.

5. Performance Considerations

Efficient hardware implementation requires careful consideration of performance metrics. Throughput, latency, and resource utilization are critical factors that need to be optimized for the specific signal classification requirements. Vivado HLS provides performance estimates that guide the refinement of the hardware description to meet these criteria.

6. Conclusion

This chapter elucidates the significance of integrating neural networks with High-Level Synthesis in the context of signal classification. Vivado HLS emerges as a robust tool for bridging the gap between complex neural network models and efficient hardware implementation.

IMPLEMENTATION

CHAPTER 4
IMPLEMENTATION

4.1 System Architecture

4.1.1 Overview of Network Architecture

The network used in this project is a hybrid network called as CGRDNN which stands for Convoluted Gated Recurrent Deep Neural Network. The network starts with a zero padding Layer followed by a few convolutional layers and maximum pooling layers. A Gated Recurrent Unit (GRU) is used as one of the layer in the CGRDNN network.

Layers of Neural Network:

Zero padding - Zero padding is a technique that allows us to preserve the original input size. Zero padding occurs when we add a border of pixels all with value zero around the edges of the input images.

Convolution Layer - A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), parameters of which are to be learned throughout the training.

Max pooling Layer - The pooling operation involves sliding a two - dimensional filter over each channel of feature map and summarizing the features lying within the region covered by the filter.

Dropout Layer - The dropout layer is a layer used in the construction of neural networks to prevent overfitting. In this process, individual nodes are excluded in various training runs using a probability

GRU Layer - Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks The GRU is like a long short-term memory (LSTM) with a gating mechanism to input or forget certain features

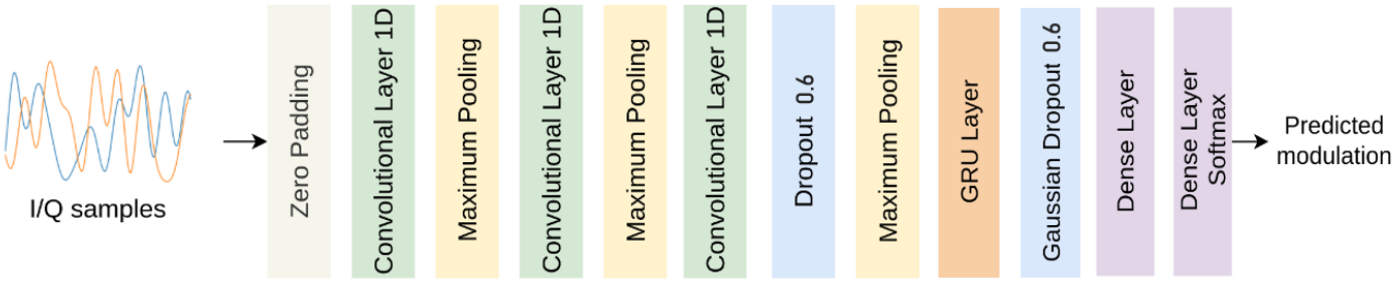


Figure 4.1.1 Architecture of the CGRDNN Neural Network

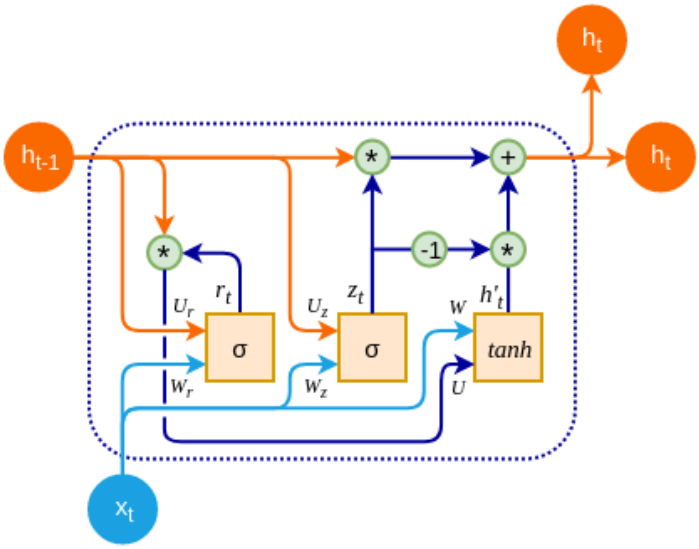


Figure 4.1.2 Internal Structure of GRU Layer

4.1.2 Integration with High - Level Synthesis

The Neural network that has been designed is integrated with high level synthesis for implementation. The HLS module starts processing and modelling the neural network and creates a HLS visualization of the network system.

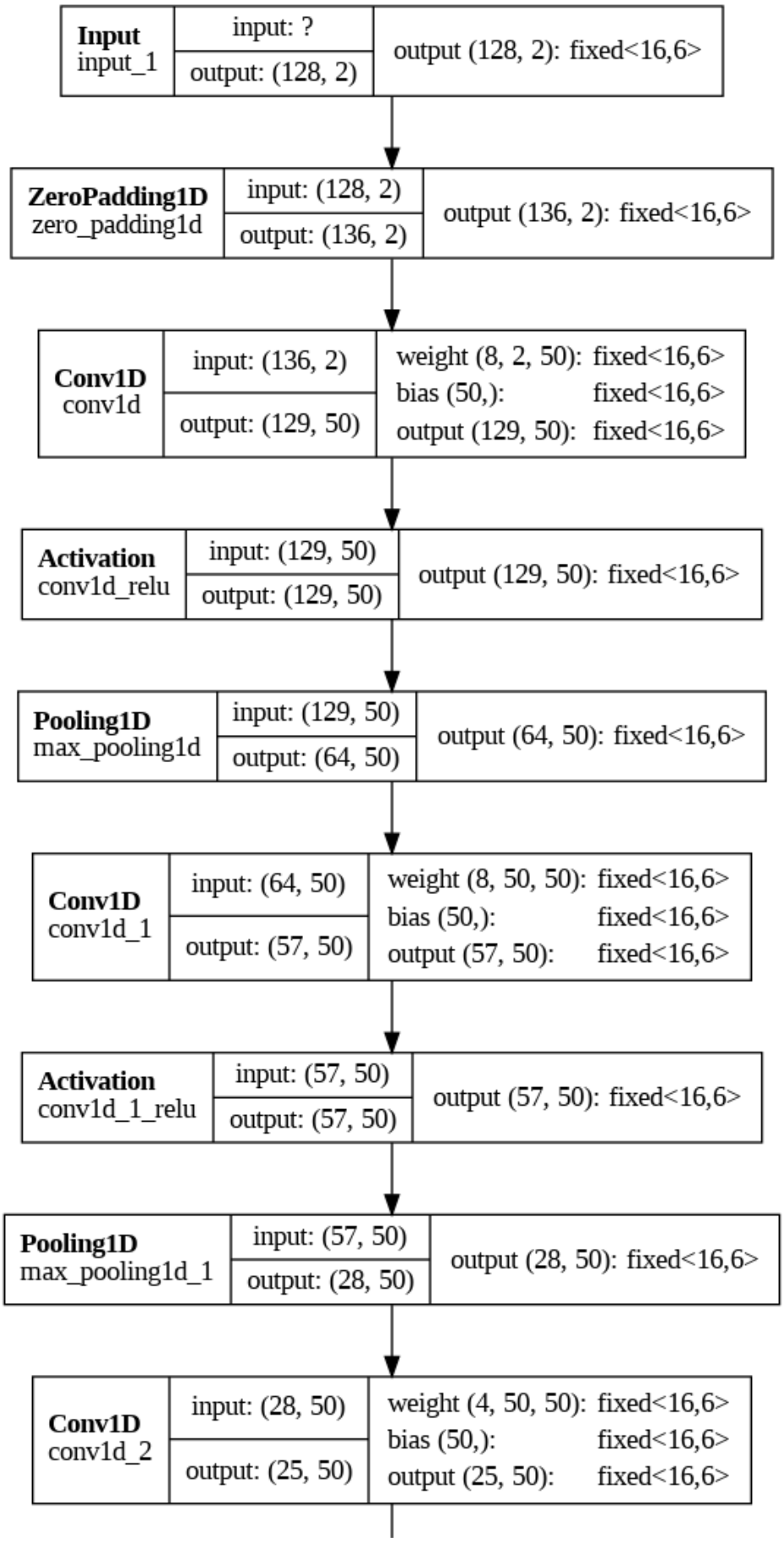


Figure 4.1.3 HLS - Layer Visualization 1

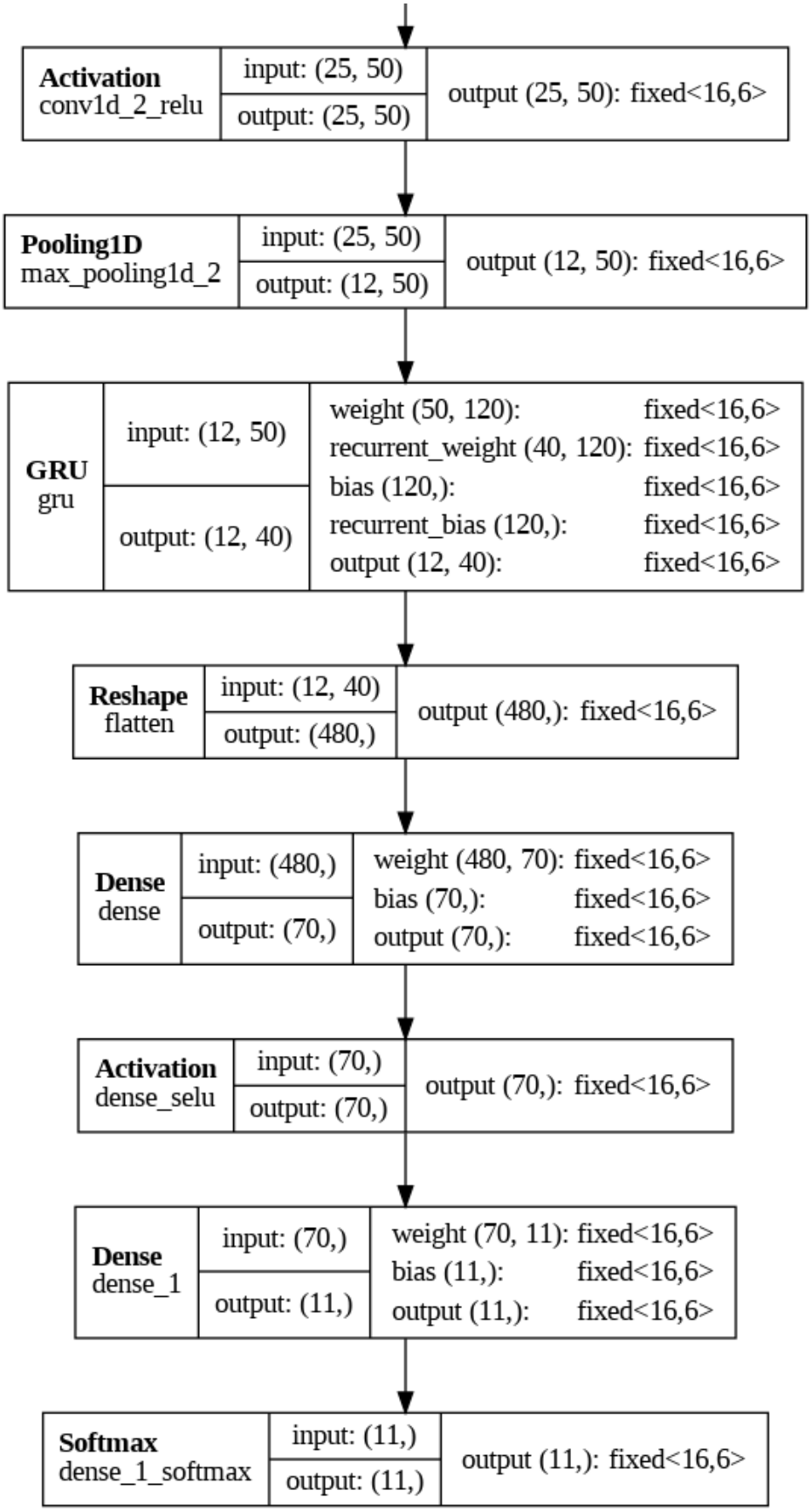


Figure 4.1.4 HLS - Layer Visualization 2

The Neural network layer sizes must be determined such that the efficiency is maximum and the loss is reduced so that the signals are classified in a much better way accurately. Such carefully selected layer sizes are helpful in reducing the complexity of the system.

Layer	Arguments	Output Shape
Input	Shape (128,2)	128 x 2
Zero Padding 1D	Padding 4	136 x 50
Convolution 1D	Filters 50, Kernels 8, ReLu	129 x 50
Max Pooling 1D	Pool Size 2	64 x 50
Convolution 1D	Filters 50, Kernels 8, ReLu	57 x 50
Max Pooling 1D	Pool Size 2	28 x 50
Convolution 1D	Filters 50, Kernels 4, ReLu	25 x 50
Dropout	Rate 0.6	25 x 50
Max Pooling 1D	Pool Size 2	12 x 50
GRU	Filters 64	58 x 40
Gaussian Dropout	Rate 0.6	58 x 40
Flatten	-	600
Dense	Units 70, SeLu	70
Dense	Units N, Softmax	N

Table 4.1.1 Layer sizes of Neural Network

4.2 Dataset Preparation

4.2.1 Data Collection

For the training and testing of the Neural Network, there are two different datasets that were used in this signal classification system. They are

RML2016.10a_dict. dat

RML2016.10b. dat

RML stands for Radio Machine Learning. It is an extensive and an elaborate collection of the different types of the modulated signals which are affected with different types of the channel noises. In 2016, the DeepSig company published

their first datasets determined for modulation classification. Signals in the datasets are synthetic and were generated with GNU Radio software. To simulate real-life scenarios, channel model blocks, consisting of sample rate offset, center frequency offset, selective fading, and additive white Gaussian noise, were added. As a final step, the data are scaled to unity energy as a prior step to further usage of the dataset in the machine learning field. The data are represented as 2x128 and 2x256 vectors of in-phase and quadrature signals (I/Q). They were generated for various signal-to-noise ratios (SNRs) in the range from -20 dB to 18 dB.

The signals in the dataset are in their original form with two different components in them. They are,

1) In-phase Components and
2) Quadrature Components

The signals in the dataset were modulated in 11 different modulation techniques so that the system can train and classify signals of different modulations. The following are the types of modulation techniques used on signals in the dataset.

BPSK, QPSK, 8PSK
QAM16, QAM64, PAM4
CPFSK (Continuous phase FSK)
GFSK (Gaussian FSK)
AM-DSB, AM-SSB, WBFM (wide band Frequency modulation)

Some of the channel characteristics considered for generating the dataset of modulated signals are as follows

FADING – Rician and Rayleigh Fading Channels with Selective Fading
NOISES – AWGN, WGN, Random noises

OFFSETS – Sample rate offset and center frequency offset

4.2.1 Data Preprocessing:

The two of the datasets used has 2,20,000 samples and 1,200,000 (1.2 million) samples of modulated signals within a SNR range of -22 to 18 dB which is usually a preferred range of signal to noise ratios that are used for real time communications.

As the first step of the signal classification process the neural network needs to get trained. The data preprocessing steps involves removing null values and feature extraction. After this step dataset is split for training, testing and validation.

4.3 Training the Neural Network

4.3.1 Selection of Neural Network parameters

Parameters are the coefficients of the model, and they are chosen by the model itself. It means that the algorithm, while learning, optimizes these coefficients (according to a given optimization strategy) and returns an array of parameters which minimize the error. It is a list of numbers that specifies the internal structure of the neural network. Each number in the list represents one hidden layer of the neural network, the value of which is the number of neurons in that layer.

When deciding the internal structure of the neural network, a good place to begin is using one hidden layer with a number of neurons between the number of inputs and the number of outputs creating a “funnel” shape.

4.3.2 Training Procedures

The dataset is first split as 70% for training, 15% for testing and 15% for

validation by using a function in the program. The training of neural network is an iterative process in which the calculations are carried out forward and backward through each layer in the network until the loss function is minimized.

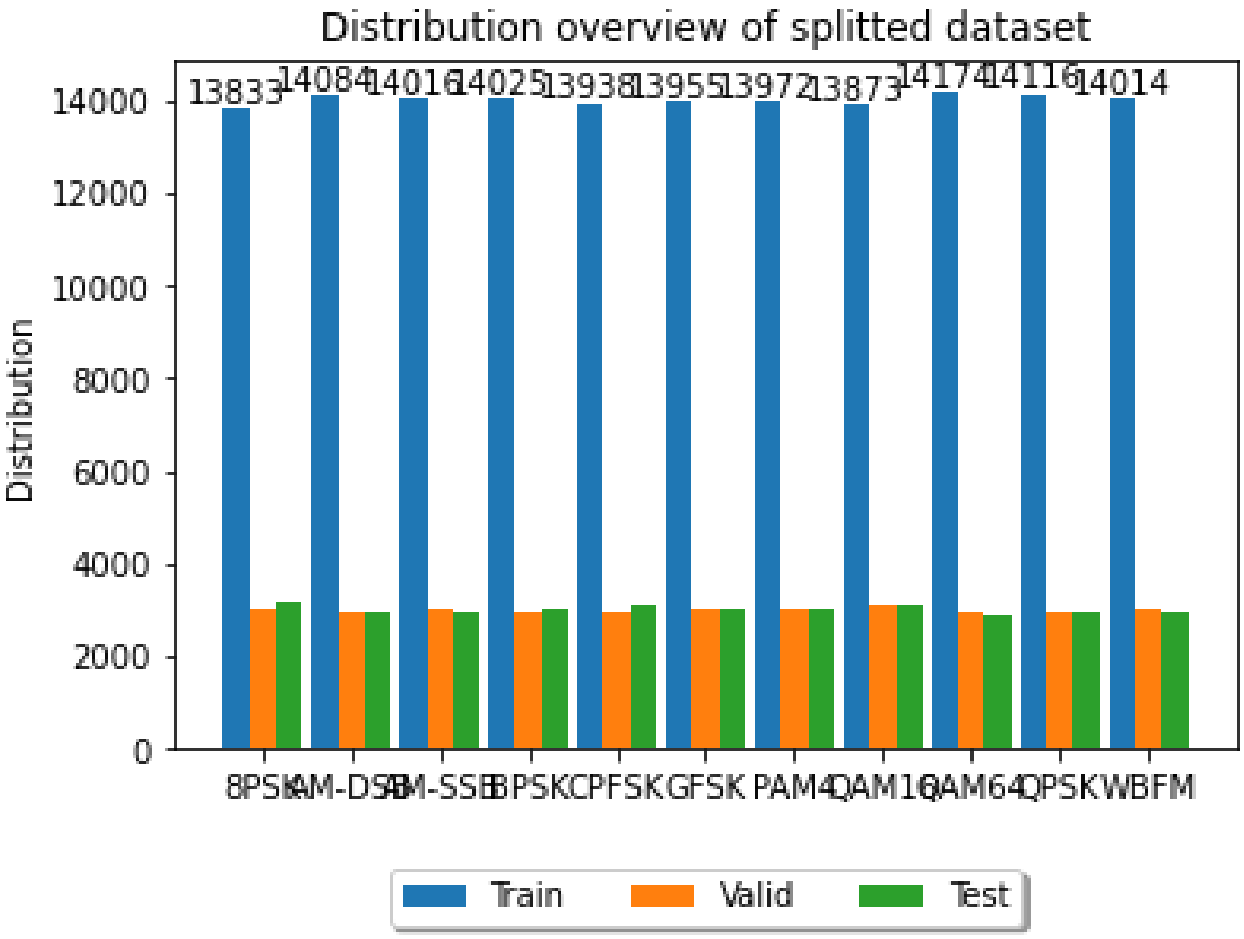


Figure 4.3.1 Splitting of the Training Dataset

4.4 Neural Network Layer Visualization

4.4.1 Activation Maps

Activation maps are a visual representation of the activation numbers at various layers of the network. It is necessary for us to understand the activation maps of each layer so that the filter strength in layer can be understood for selecting the most optimized path from input to the output within the neural network.

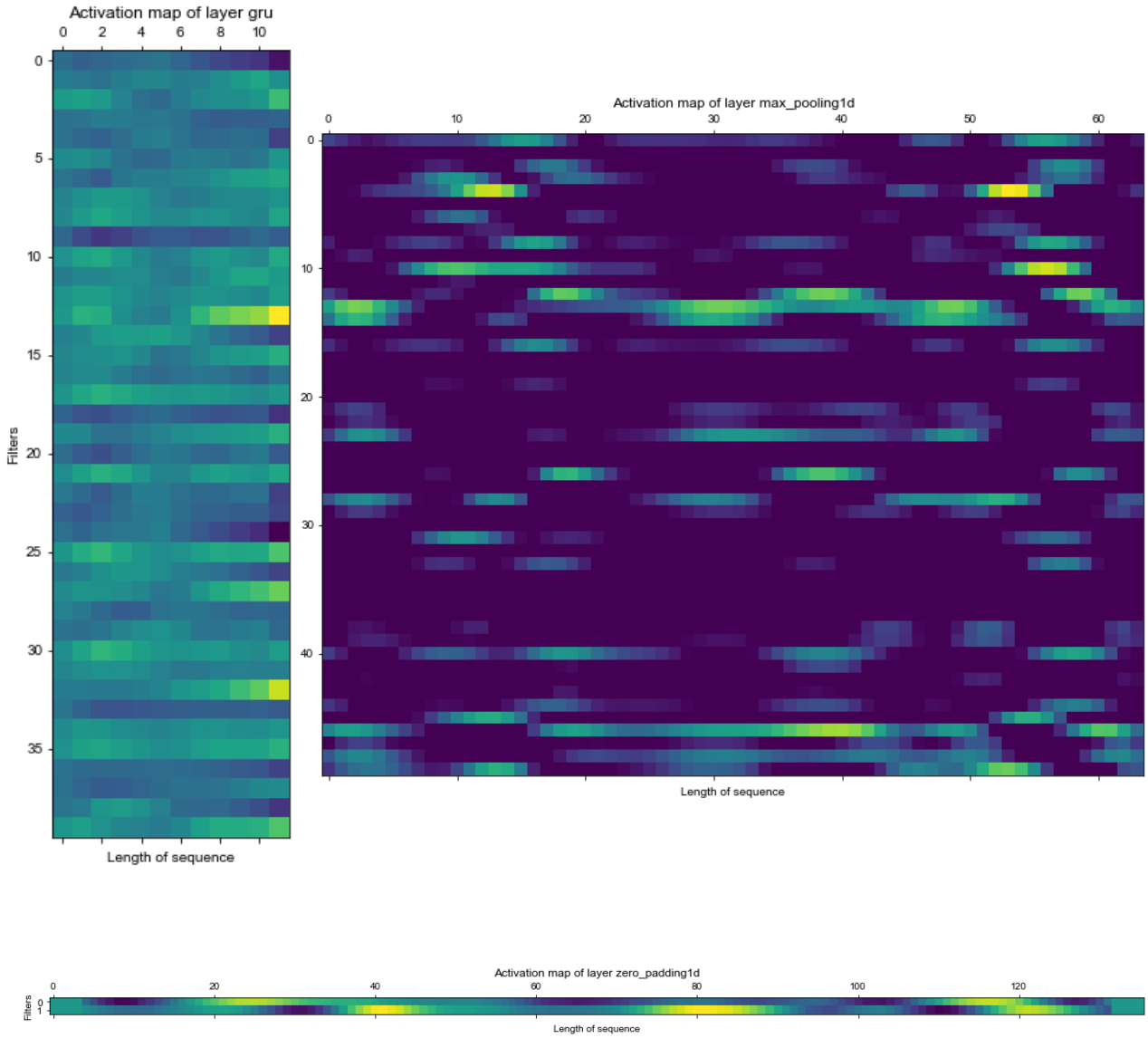


Figure 4.4.1 Activation Maps of NN Layers

4.4.2 Feature Maps

Each feature map corresponds to a specific filter and represents the response of that filter to the input image. Feature maps are generated by applying Filters or Feature detectors to the input image or the feature map output of the prior layers.

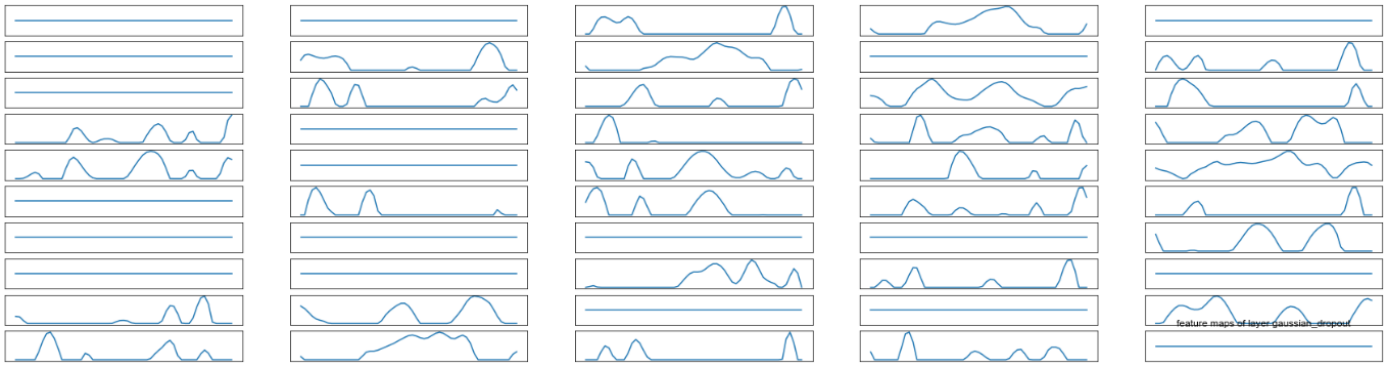


Figure 4.4.2 Feature Maps of NN Layers

RESULTS AND DISCUSSIONS

CHAPTER 5
RESULTS AND DISCUSSIONS

5.1 Evaluation Metrics

5.1.1 Accuracy

Accuracy measures how correctly the neural network predicted the positive and negative classes against all examples. Accuracy in neural networks is a measure of how well the model is able to predict the correct output for a given input. It is often represented as a percentage and is computed by taking the number of successful predictions made by the model and dividing it by the total number of predictions. This hybrid neural network has an overall accuracy of 91.89 % and as the SNR increases the accuracy stabilizes to a higher value indicating that the system is more effective in classifying signals in real time communication SNR ranges.

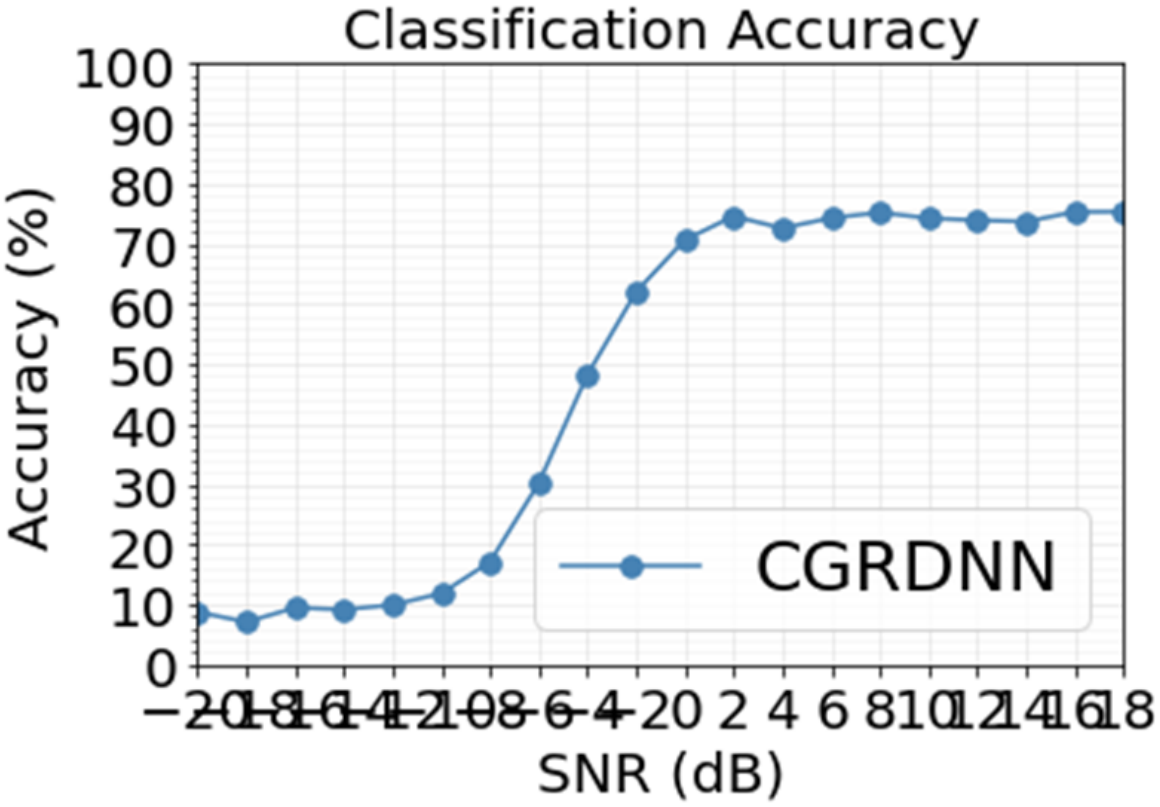


Figure 5.1.1 Classification Accuracy Plot

5.1.2 Precision and Confusion Matrix

The confusion matrix, also known as the error matrix, is mainly used for statistical classification. It is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents an instance in a predicted value while the column represents the actual value, or vice versa. The output matrix has four cells, true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP means the actual value and the predicted value are both positive, TN means the actual value is positive but the model predicted value is negative, FP means the actual value is negative but the model predicted value is positive, and, finally, FN means both the actual and predicted values are negative.

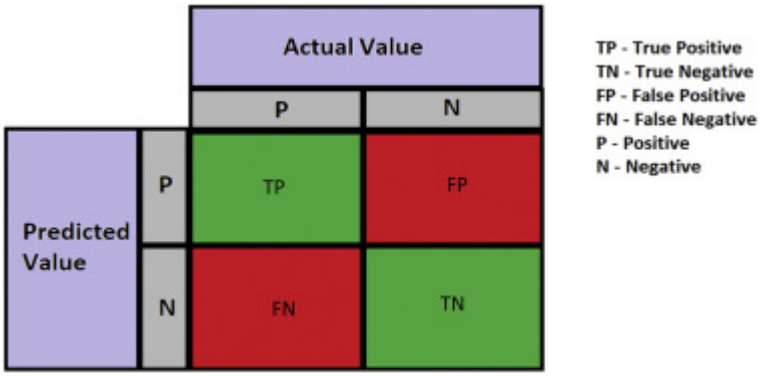


Figure 5.1.2 Model Confusion Matrix

TERM	***FORMULA***
Sensitivity	TP/(TP+FN)
Specificity	TN/(TN+FP)
Precision	TP/(TP+FP)
Recall	TP/(TP+FN)
Accuracy	(TP+TN)/(TP+TN+FP+FN)

Table 5.1.1 Performance Metrics Formula

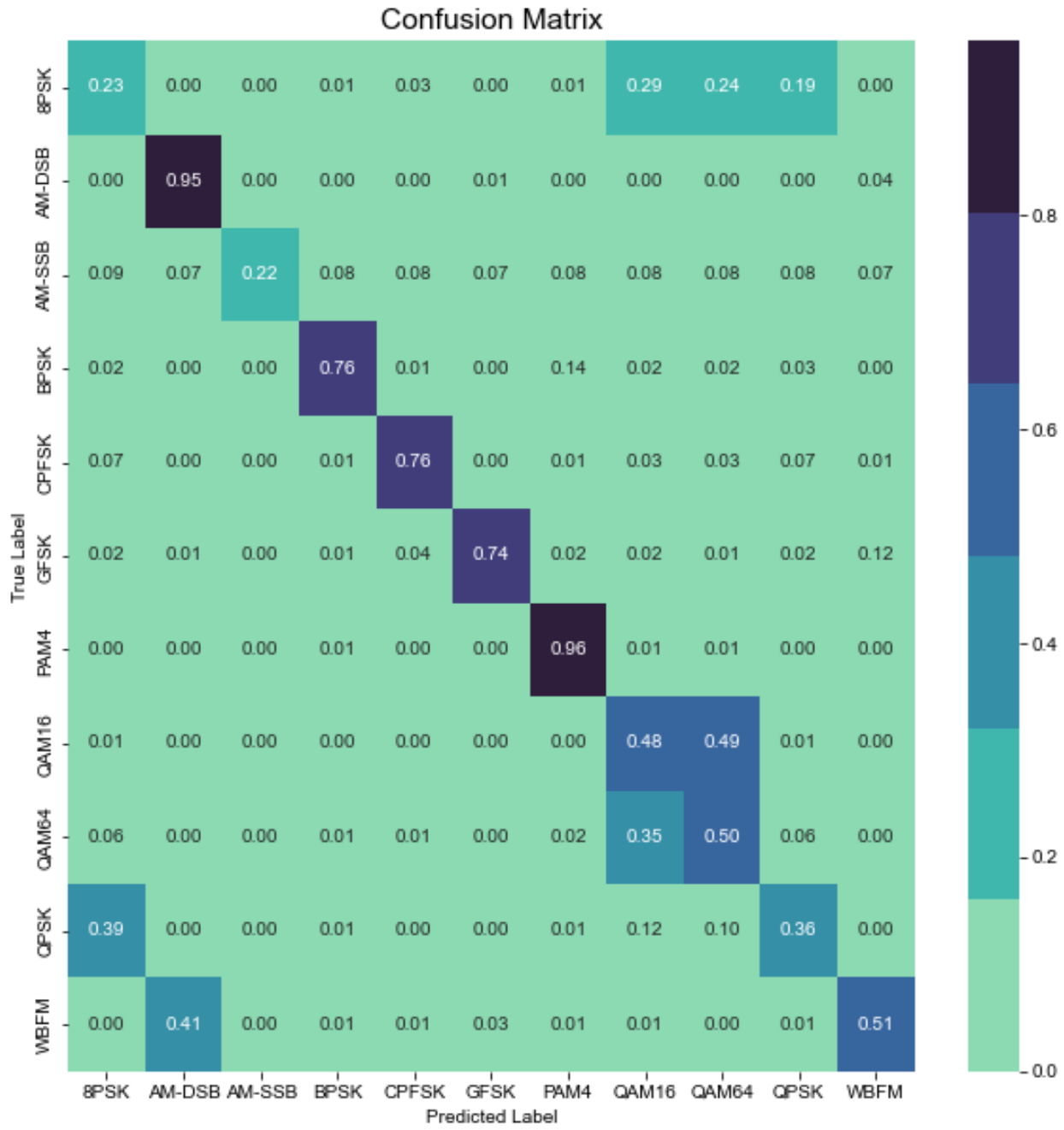


Figure 5.1.3 Confusion Matrix of Neural Network

5.2 Comparative Analysis with Existing Systems

There are other already existing signal classifications systems for classifying the signals according to the modulation techniques used in transmitting them over a communication channel. The already existing systems use simple Convoluted Neural Networks or CNN for classifying the signals.

One main disadvantage in this already existing system is the accuracy which is only about 75 % but our proposed CGRDNN Hybrid neural network reaches an accuracy level of nearly 92 % which means our model is far better in classifying the modulation techniques of the signals more precisely.

```
570000/570000 [==============================] - 58s 101us/step - loss: 1.0992 - acc: 0.5434 - val_loss: 1.0472 -
628
Epoch 47/80
570000/570000 [==============================] - 58s 101us/step - loss: 1.0962 - acc: 0.5445 - val_loss: 1.0626 -
588
Epoch 48/80
570000/570000 [==============================] - 58s 101us/step - loss: 1.0949 - acc: 0.5458 - val_loss: 1.0470 -
616
Epoch 49/80
570000/570000 [==============================] - 58s 101us/step - loss: 1.0920 - acc: 0.5461 - val_loss: 1.0541 -
651
Epoch 50/80
570000/570000 [==============================] - 58s 101us/step - loss: 1.0891 - acc: 0.5485 - val_loss: 1.0396 -
686
Epoch 51/80
570000/570000 [==============================] - 58s 101us/step - loss: 1.0890 - acc: 0.5483 - val_loss: 1.0395 -
679
Epoch 52/80
339968/570000 [================>.............] - ETA: 22s - loss: 1.0840 - acc: 0.5507
```

In [0]:
```
model.save("cnn.h5")
```

Figure 5.2.1 Accuracy of Existing CNN system

```
Saved model.json
6563/6563 - 513s - loss: 1.3204 - accuracy: 0.4482 - val_loss: 1.2051 - val_accuracy:
Test accuracy 0.9189343321011673
Test loss 0.27541012556201

SNR -20dB:
285/285 - 3s - loss: 2.3821 - accuracy: 0.0974 - 3s/epoch - 9ms/step

SNR -18dB:
282/282 - 2s - loss: 2.3698 - accuracy: 0.1028 - 2s/epoch - 7ms/step

SNR -16dB:
283/283 - 2s - loss: 2.3487 - accuracy: 0.1055 - 2s/epoch - 7ms/step

SNR -14dB:
282/282 - 2s - loss: 2.3014 - accuracy: 0.1055 - 2s/epoch - 6ms/step

SNR -12dB:
283/283 - 3s - loss: 2.2097 - accuracy: 0.1135 - 3s/epoch - 9ms/step

SNR -10dB:
281/281 - 3s - loss: 2.0393 - accuracy: 0.1584 - 3s/epoch - 11ms/step
```

Figure 5.2.2 Accuracy of proposed CGRDNN system

The number of trainable parameters is yet another reason for accuracy of the neural network. The number of trainable parameters for already existing system is nearly 2,830,170 while the number of trainable parameters of the proposed CGRDNN model is only 76,441. This increases the efficiency since only the optimized path from input to output in neural network are kept while the others are dropped down.

Layer (type)	Output Shape	Param #
reshape_66 (Reshape)	(None, 1, 2, 128)	0
zero_padding2d_76 (ZeroPaddi	(None, 1, 2, 132)	0
conv1 (Conv2D)	(None, 256, 2, 130)	1024
dropout_39 (Dropout)	(None, 256, 2, 130)	0
zero_padding2d_77 (ZeroPaddi	(None, 256, 2, 134)	0
conv3 (Conv2D)	(None, 80, 1, 132)	122960
dropout_40 (Dropout)	(None, 80, 1, 132)	0
flatten_16 (Flatten)	(None, 10560)	0
dense1 (Dense)	(None, 256)	2703616
dropout_41 (Dropout)	(None, 256)	0
dense3 (Dense)	(None, 10)	2570
reshape_67 (Reshape)	(None, 10)	0

Total params: 2,830,170
Trainable params: 2,830,170
Non-trainable params: 0

Table 5.2.1 Parameters of Existing CNN system

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 128, 2)]	0
zero_padding1d (ZeroPadding1D)	(None, 136, 2)	0
conv1d (Conv1D)	(None, 129, 50)	850
max_pooling1d (MaxPooling1D)	(None, 64, 50)	0
conv1d_1 (Conv1D)	(None, 57, 50)	20050
max_pooling1d_1 (MaxPooling1D)	(None, 28, 50)	0
conv1d_2 (Conv1D)	(None, 25, 50)	10050
dropout (Dropout)	(None, 25, 50)	0
max_pooling1d_2 (MaxPooling1D)	(None, 12, 50)	0
gru (GRU)	(None, 12, 40)	11040
gaussian_dropout (GaussianDropout)	(None, 12, 40)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 70)	33670
dense_1 (Dense)	(None, 11)	781

Total params: 76441 (298.60 KB)
Trainable params: 76441 (298.60 KB)

Table 5.2.2 Parameters of Proposed CGRDNN system

5.3 Real - Time Performance on FPGA

The program can be run in real time on an FPGA since it has been High Level Synthesized and the HLS model has been compiled and built. The HLS compiling visualizes each layer of the Neural Network as having different input and output sizes and the building reveals all the layers as a list for topology report.

```
Datatype of variable:  <class 'dict'>
class_name Functional
config {'name': 'model_8', 'trainable': True, 'layers': [{'module': 'keras.layers', 'class_name': 'InputLayer'
keras_version 2.14.0
backend tensorflow
{'OutputDir': '/content/drive/MyDrive/Colab Notebooks/out', 'ProjectName': 'hlstest', 'Backend': 'Vivado', 'Pa
Interpreting Model
Topology:
Layer name: input_1, layer type: InputLayer, input shapes: [[None, 128, 2]], output shape: [None, 128, 2]
Layer name: zero_padding1d, layer type: ZeroPadding1D, input shapes: [[None, 128, 2]], output shape: [None, 13
Layer name: conv1d, layer type: Conv1D, input shapes: [[None, 136, 2]], output shape: [None, 129, 50]
Layer name: max_pooling1d, layer type: MaxPooling1D, input shapes: [[None, 129, 50]], output shape: [None, 64,
Layer name: conv1d_1, layer type: Conv1D, input shapes: [[None, 64, 50]], output shape: [None, 57, 50]
Layer name: max_pooling1d_1, layer type: MaxPooling1D, input shapes: [[None, 57, 50]], output shape: [None, 28
Layer name: conv1d_2, layer type: Conv1D, input shapes: [[None, 28, 50]], output shape: [None, 25, 50]
Layer name: max_pooling1d_2, layer type: MaxPooling1D, input shapes: [[None, 25, 50]], output shape: [None, 12
Layer name: gru, layer type: GRU, input shapes: [[None, 12, 50]], output shape: [None, 12, 40]
Layer name: flatten, layer type: Reshape, input shapes: [[None, 12, 40]], output shape: [None, 480]
Layer name: dense, layer type: Dense, input shapes: [[None, 480]], output shape: [None, 70]
Layer name: dense_1, layer type: Dense, input shapes: [[None, 70]], output shape: [None, 11]
Creating HLS model
```

Figure 5.3.1 HLS Neural Network Topology Report

Thus this signal classification system can be implemented on an FPGA board so that it can act as an artificial intelligence block in a system.

CHALLENGES AND SOLUTIONS

CHAPTER 6
CHALLENGES AND SOLUTIONS

6.1 Noise and Interference Handling

As it is clear that any system must be reliable in any kind of adverse situations, our model is best in real time communication ranges. But all communication systems are always affected by a variety of noises. The main challenges faced by our CGRDNN neural network system is that when the signals are affected more by noise or if the communication channel of the signals impart and affect the signals with heavy distortions, then the signal classification system can reach only a certain low testing accuracy.

This is more important to note that the testing accuracy of a system in overall range of the SNR is not applicable if the SNR ratio is fixed. A very low SNR indicates that the noise strength or the power is very higher than the signal strength of the signal power.

Solution

The best solution to enhance this reduced efficiency in lower SNR conditions is to increase the training of the system on lower SNR signals. The most reliable and optimal solution to this problem could be noise reduction processes and filtering to be implemented before sending the signals to the neural network for classification process

6.2 Computational Efficiency in Real - Time

Any system must be accurate in real time applications. The natural environment is very complicated and the signals travelling through it undergoes lots of distortions. The noises considered in this system’s training are normal Additive

White Gaussian Noise, White Gaussian Noises and Random noises. But in real time the signals are affected by lots of different kinds of noises and most of the times these noises are complex functions. The system's reliability in such dynamic conditions is an important factor that must be considered while implementing it in real time.

Furthermore, the system's application includes classifying signals from non - cooperative communication channels in which the connection between the devices that are communicating are dynamic in nature and the signal strength varies accordingly.

One of the main factor that affects the reception of real time signals is the hindrances or reflectors faced by the signals while propagation. These reflectors can sometimes block the signals entirely.

So the accuracy and the system efficiency in the real time applications may vary due to several such factors. And so it is important to consider the physical environment while implementing this system in real time.

6.3 FPGA Resource Utilization and Constraints

Another problem that needs to be addressed arises when this system is implemented as an artificial intelligence block in an FPGA Board. This system is highly advanced and so the resources it utilizes when implemented is quite large.

Although the system is highly optimized by reducing the trainable parameters by using the dropout and Gaussian dropout layers, still the number of Flip Flops, memory and other digital blocks it requires can be huge. This will make other programs running on the same FPGA to work ineffectively.

Solution

The best solution for these constraints of FPGA implementation would be implementing this program in a separate FPGA board and having only the signal classification program to be running on the board. This is highly advantageous since the system is isolated it can be used as an AI block in any kind of communication system without any complications.

APPLICATIONS

CHAPTER 7
APPLICATIONS AND FUTURE DEVELOPMENTS

7.1 Integration into the Communication Systems

This signal classification system as a neural network is especially made to be implemented in an FPGA board so that it can act as an individual Artificial intelligence block in any communication system.

This system can be implemented at the receiver side of the communication system which uses non - cooperative communication channels for signal transmission. This is better suited for systems in which the transmitted signal modulation information is not known to the receiver prior to transmission.

Thus it is ideal to identify the modulation technique of the incoming signals to apply the specific demodulation technique to it.

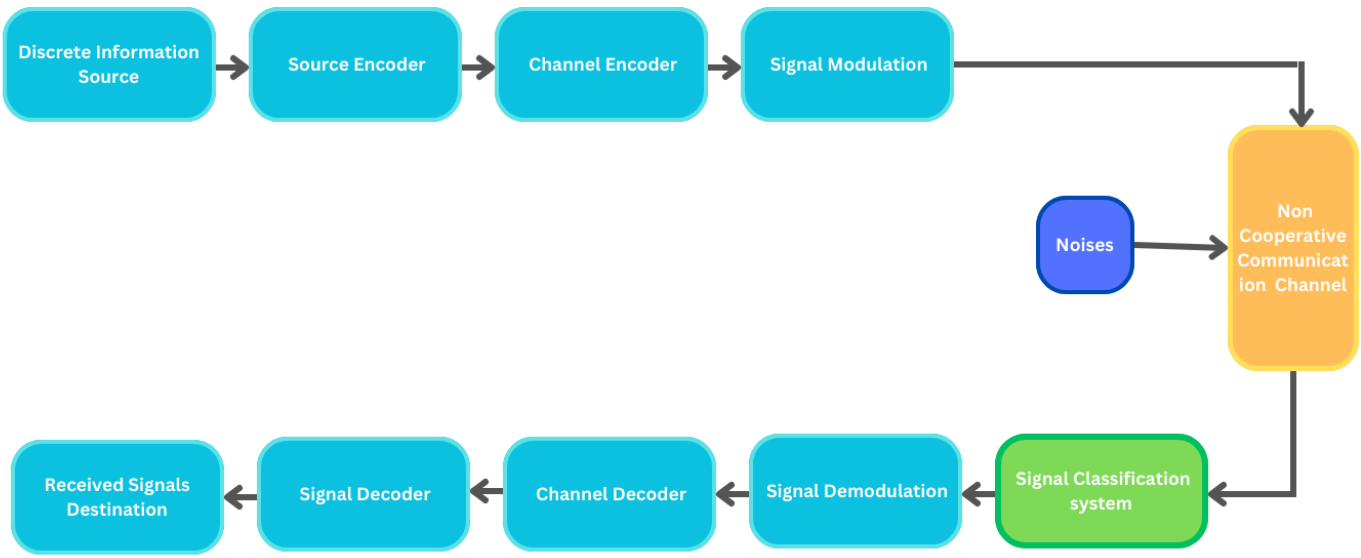


Figure 7.1.1 Communication system with signal classification system integrated at the Receiver side

7.2 Potential Use Cases

This neural network system can be used in non-cooperative channel communication scenario, when there comes a need to recognize and classify the modulation format of the received signals to identify the communication targets and to have better management over them.

A non - cooperative communication is the classical multiple access channel, where users send directly to a common destination, without repeating for one another. In such cases the modulation technique cannot be known prior to the receiver system.

To solve problems such as high complexity, low accuracy, hard process of manual extraction of signal features, we can go for a Deep Neural Network based modulation format identification block so that we can effectively classify received signals and apply specific demodulation technique to it.

This kind of signal classification system is more important in MANET and VANET networks.

MANET stands for Mobile Ad-hoc Network also called a wireless Ad-hoc network or Ad-hoc wireless network that usually has a routable networking environment on top of a Link Layer ad hoc network. They consist of a set of mobile nodes connected wirelessly in a self-configured, self-healing network without having a fixed infrastructure. MANET nodes are free to move randomly as the network topology changes frequently.

Vehicular ad hoc networks (VANETs) are created by applying the principles of mobile ad hoc networks (MANETs) – the spontaneous creation of a wireless network of mobile devices – to the domain of vehicles.

In these scenarios, different devices often establish a temporary connection between each other. When such connected devices use different modulation techniques to transmit signals then it is imperative that the signal modulation classification is necessary.

The signal classification system is an important part of SIGINT applications. Signals intelligence (SIGINT) is intelligence-gathering by interception of signals, whether communications between people (communications intelligence – abbreviated to COMINT) or from electronic signals not directly used in communication (electronic intelligence—abbreviated to ELINT).

Signal intelligence is a subset of intelligence collection management. As classified and sensitive information is usually encrypted, signals intelligence in turn involves the use of cryptanalysis to decipher the messages. This can be widely used for military purposes to track and intercept signals and messages.

Thus these are some of the potential use cases for signal modulation classification system.

7.3 Future Enhancements

Future work on this project can include specific FPGA implementation such as PYNQ - Z2 and analyzing the performance and possible integration of this AI block in any communication system. Futhermore, the system can be trained in all kinds of possible modulation techniques for achieving versatility. This system can be modified for 5G and 6G communication systems so that it meets the needs of recent advancements in the domain of digital communication.

CONCLUSIONS

CHAPTER 8
CONCLUSIONS

8.1 Summary of the Findings

The HLS based signal classification system on the whole is an effective and a reliable system for recent advancements in the field of digital communications. The artificial intelligence is a vast and ever expanding reality of future and so integration of AI in communication domain is necessary for a better understanding the future of human and machine communications. The implementation of hybrid neural networks is important so that the efficiency of any artificial machine learning system can be increased greatly that the systems can effectively replace all human interfaces in the domain of communication. This system could be a key for a new era in digital telecommunications.

8.2 Contributions to the Field

This HLS based signal classification system using Convoluted Gated Recurrent Deep Neural Network can serve as a base for future enhancements in integrating Artificial intelligence into the field of Digital Telecommunications. Futher explorations in the domain of machine learning and its applications in the rapidly progressing growth of AI integrations into already existing systems can be based firmly on this project. Thus this project has miraculous contributions to the field of Future Digital Telecommunications.

REFERENCES

CHAPTER 9

REFERENCES

[1] *A survey of modulation classification using deep learning: Signal representation and data preprocessing. S Peng, S Sun, YD Yao, IEEE 2017.*

[2] *Meng, P.F.; Chen, L.; Wu, X.W. Automatic modulation classification: A deep learning enabled approach. IEEE Trans. Veh. Technol. 2018, 67, 10760 –10772.*

[3] *Automatic modulation classification based on constellation density using deep learning Y Kumar, M Sheoran, G Jajoo , IEEE 2020.*

[4] *Li, Y.; Wang, B.; Shao, G.; Shao, S. Automatic modulation classification for short burst underwater acoustic communication signals based on hybrid neural networks. IEEE Access 2020, 8, 227793 – 227809.*

[5] *A Review of Research on Signal Modulation Recognition Based on Deep Learning by Wenshi Xiao, Zhongqiang Luo and Qian Hu*

[6] *HLS-Based FPGA Implementation of Convolutional Deep Belief Network for Signal Modulation Recognition by Jian Zhao, Yaqin Zhao, Hongbo Li, Yun Zhang and Longwen W, IEEE 20465005*

[7] *Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml by Elham E Khoda, Dylan Rankin, Rafael Teixeira de Lima*

[8] *A. Hazza, M. Shoaib, S. A. Alshebeili and A. Fahad, "An overview of feature-based methods for digital modulation classification," 2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA), Sharjah, United Arab Emirates, 2013, pp. 1-6, doi: 10.1109/ICCSPA.2013.6487244.*

[9] *S. Peng et al., "Modulation Classification Based on Signal Constellation Diagrams and Deep Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 3, pp. 718-727, March 2019, doi: 10.1109/TNNLS.2018.2850703.*

[10] *Hossen, Abdulnasir, Fakhri Al-Wadahi, and Joseph A. Jervase. "Classification of modulation signals using statistical signal characterization and artificial neural networks." Engineering Applications of Artificial Intelligence 20.4 (2007): 463-472.*

[11] *Hsue, S-Z., and Samir S. Soliman. "Automatic modulation classification using zero crossing." IEE Proceedings F (Radar and Signal Processing). Vol. 137. No. 6. IET Digital Library, 1990.*

[12] *Zhu, Zhechen, and Asoke K. Nandi. Automatic modulation classification: principles, algorithms and applications. John Wiley & Sons, 2015.*

[13] *Xu, Jefferson L., Wei Su, and Mengchu Zhou. "Likelihood-ratio approaches to automatic modulation classification." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 41.4 (2010): 455-469.*

[14] *Homsirikamol, Ekawat, and Kris Gaj George. "Toward a new HLS-based methodology for FPGA benchmarking of candidates in cryptographic competitions: The CAESAR contest case study." 2017 International Conference on Field Programmable Technology (ICFPT). IEEE, 2017.*

APPENDIX

CHAPTER 10
APPENDIX

10.1 Neural Network Architecture Details

A neural network is a software solution that leverages machine learning (ML) algorithms to ‘mimic’ the operations of a human brain. Neural networks process data more efficiently and feature improved pattern recognition and problem - solving capabilities when compared to traditional computers. Neural networks are also known as artificial neural networks (ANNs) or simulated neural networks (SNNs).

Neural networks are a subtype of machine learning and an essential element of deep learning algorithms. Just like its functionality, the architecture of a neural network is also based on the human brain. Its highly interlinked structure allows it to imitate the signaling processes of biological neurons.

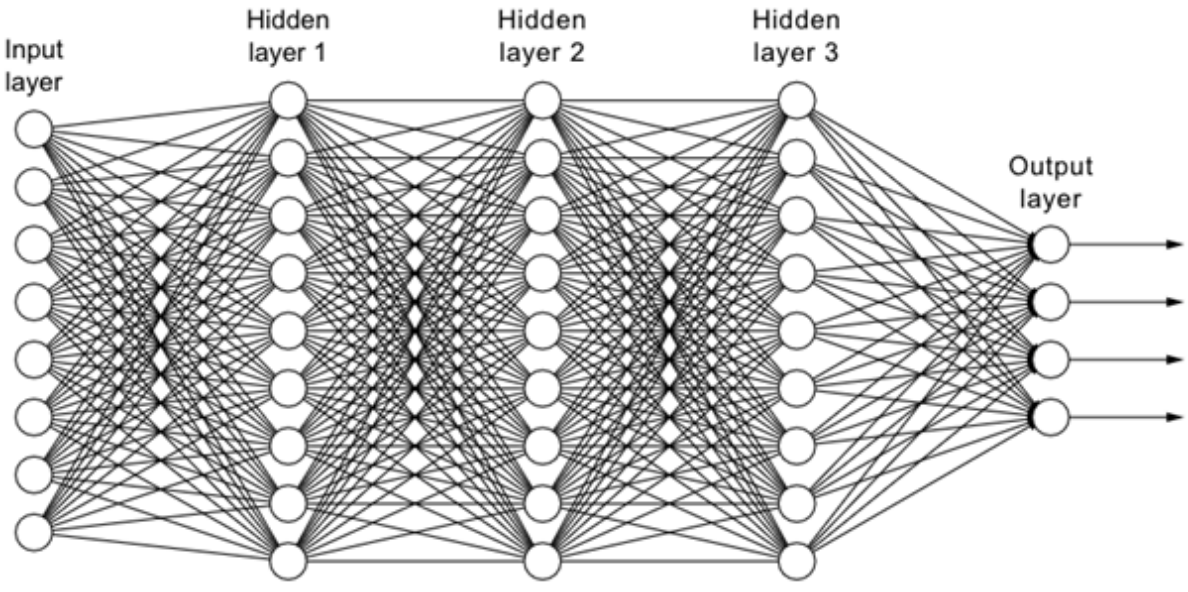


Figure 10.1.1 A simple DNN internal neuron connections

The architecture of a neural network comprises node layers that are distributed

across an input layer, single or multiple hidden layers, and an output layer. Nodes are ‘artificial neurons’ linked to each other and are associated with a particular weight and threshold. Once the output of a single node crosses its specified threshold, that particular node is activated, and its data is transmitted to the next layer in the network. If the threshold value of the node is not crossed, data is not transferred to the next network layer. Unlike traditional computers, which process data sequentially, neural networks can learn and multitask. In other words, while conventional computers only follow the instructions of their programming, neural networks continuously evolve through advanced algorithms. It can be said that neural computers ‘program themselves’ to derive solutions to previously unseen problems.

10.2 Use of Dropout Layers in Optimization

In the overfitting problem, the model learns the statistical noise. To be precise, the main motive of training is to decrease the loss function, given all the units (neurons). So in overfitting, a unit may change in a way that fixes up the mistakes of the other units. This leads to complex co-adaptations, which in turn leads to the overfitting problem because this complex co-adaptation fails to generalize on the unseen dataset. if we use dropout, it prevents these units to fix up the mistake of other units, thus preventing co-adaptation, as in every iteration the presence of a unit is highly unreliable. So by randomly dropping a few units (nodes), it forces the layers to take more or less responsibility for the input by taking a probabilistic approach. This ensures that the model is getting generalized and hence reducing the overfitting problem.

This can be undone by the use of dropout layers that optimize the path from input to output so that the efficiency is boosted.

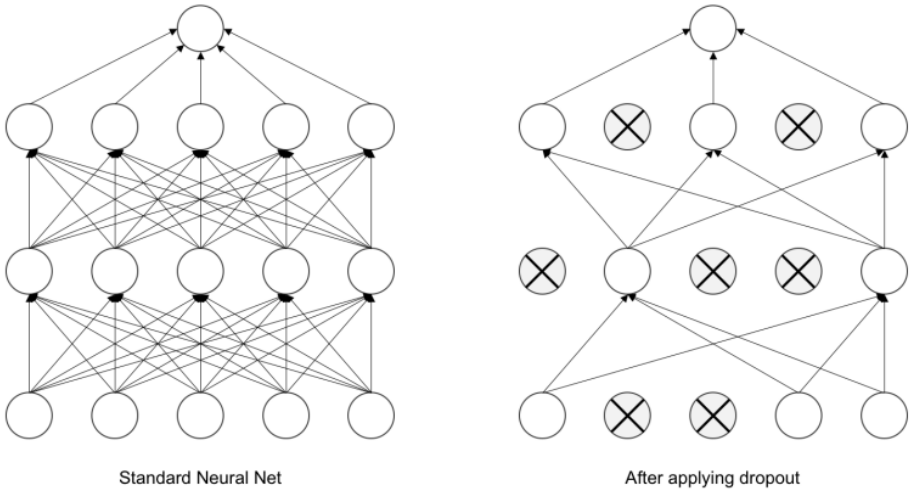


Fig. 3.2: Neural network vs. neural network with dropout[1]

Figure 10.2.1 Dropout Layer Comparison

Thus only the optimum paths are kept in the neural network and so the accuracy of the system greatly increases. This also reduces the number of trainable parameters so that the training time of the Neural Network is reduced.