DEEP LEARNING FOR HF RADIO SIGNAL TYPE CLASSIFICATION

A PROJECT REPORT

submitted by

CB.EN.U4ELC20020	**G. KAVYA SUDHA**
CB.EN.U4ELC20023	**H. KESAVA SRAVAN**
CB.EN.U4ELC20036	**L. MODITHA CHOWDARY**
CB.EN.U4ELC20048	**P. LAXMI GANESH**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ELECTRICAL AND COMPUTER ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

MAY 2024

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING,COIMBATORE -641 112

BONAFIDE CERTIFICATE

This is to certify that this project entitled **"DEEP LEARNING FOR HF RADIO SIGNAL TYPE CLASSIFICATION"** submitted by

CB.EN.U4ELC20020	**G. KAVYA SUDHA**
CB.EN.U4ELC20023	**H. KESAVA SRAVAN**
CB.EN.U4ELC20036	**L. MODITHA CHOWDARY**
CB.EN.U4ELC20048	**P. LAXMI GANESH**

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology** in **ELECTRICAL & COMPUTER ENGINEERING** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering.

04/05/24

Amit Agarwal, PhD
Professor
Department of Electrical and Electronics Engineering / Cyber Security
Amrita School of Engineering
Coimbatore- 641112

Balamurugan S
Chairperson
Professor
Department of Electrical and Electronics Engineering
Amrita School of Engineering
Coimbatore- 64111

This project report was evaluated by us on……

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEGEMENT

We offer our sincere pranams to the lotus feet of her holiness Mata Amritanandamayi Devi, the chancellor of our college Amrita School of Engineering, whom we lovingly call as AMMA. We like to express heartfelt gratitude to Dr. Sasangan Ramanathan, Professor and Dean, Amrita School of Engineering, Coimbatore for continuous support and guidance throughout the project.

We would like to express grateful thanks to Dr. Balamurugan S, Professor and Chairperson, Department of Electrical and Electronics Engineering, Amrita School of Engineering, Coimbatore for continuous support and guidance throughout.

We express our sincere grateful acknowledgement to our guide Dr. Amit Agarwal, Professor, Department of Electrical and Electronics Engineering / Cyber Secuity, Amrita School of Engineering, Coimbatore, who has been instrumental in guiding us towards the successful completion of the project.

We would like to express grateful thanks to all the faculty and staff of Department of EEE and CSE, Amrita School of Engineering, Coimbatore for their continuous support and guidance throughout. We would like to express our sincere and special thanks to our friends for sharing their knowledge and co-operation throughout the project. We would also like to extend our greatest gratitude to our parents for their eternal love and constant support.

ABSTRACT

The categorization of radio signals holds significant importance in fields like signal intelligence and surveillance, as well as emerging applications such as cognitive radio and dynamic spectrum access. Traditionally, signal classification relied on features derived from probabilistic methods, statistics, or cyclostationarity, which required expertise in designing them.

In contrast to traditional methods that rely on manually designed features, neural networks offer a more flexible and adaptable approach to signal classification. By learning directly from the raw data, these networks can uncover intricate patterns and relationships that might be challenging for human experts to identify. Moreover, their ability to generalize across diverse signal types and adapt to new scenarios makes them particularly well-suited for dynamic environments such as those encountered in radio frequency (RF) communications. Additionally, the scalability of neural networks allows them to handle large datasets efficiently, enabling robust performance even in complex signal environments. As a result, the adoption of advanced machine learning techniques represents a significant leap forward in the field of signal classification, offering enhanced accuracy, reliability, and versatility for a wide range of applications.

However, recent advancements in machine learning, particularly deep neural networks, have garnered considerable attention for their exceptional performance in various classification tasks. In signal classification, these networks are trained on vast amounts of raw data, such as IQ data samples from radio signals, enabling them to differentiate between different signal classes.

This study explores the application of neural networks like Deep CNN and RESNET for directly classifying signals based on their transmission modes, rather than just their modulation types. By embracing a data-driven approach, these networks map input IQ data to output modes. The investigation highlights the remarkable capability of neural networks in accurately classifying signals into their transmission modes, even when they share very similar characteristics.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

BLOCK DIAGRAM

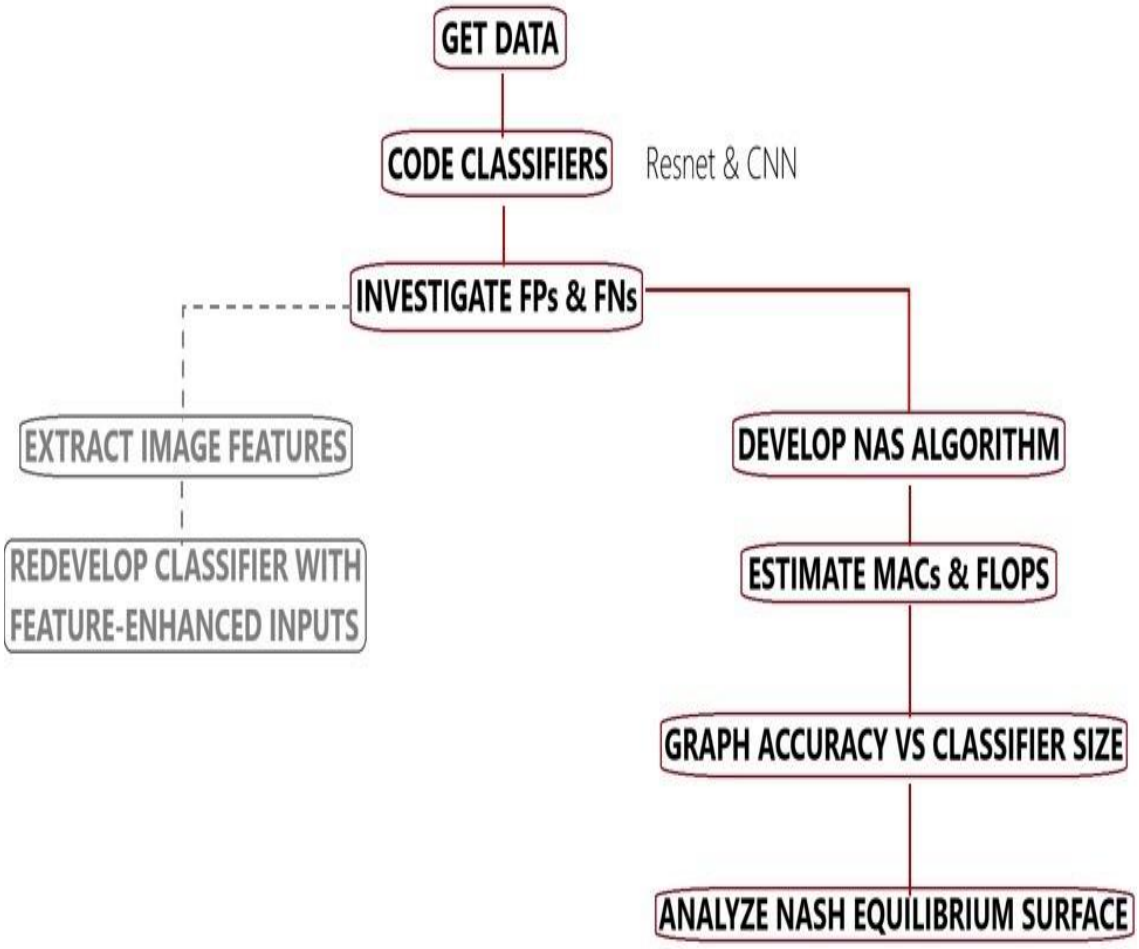

Figure 1: Complete Block Diagram

Chapter 1

DATASET DETAILS

1.1 Introduction to Dataset

The dataset has been created for training and experimenting with machine learning systems, particularly for tasks such as signal and modulation categorization. It aims to facilitate the utilization of modern techniques such as deep learning with neural networks. [1]

The dataset contains artificially generated signals, created using standard software by altering text, music, and speech. This generation method mimics real-world scenarios where signals undergo modifications and are then segmented into smaller parts. To emulate real reception signals, various impairments are introduced into the generated signals. These impairments encompass random frequency and phase shifts, Gaussian noise, and Watterson fading (to simulate ionospheric propagation). This approach aims to enhance the dataset's similarity to real signal reception conditions.

Properties of the dataset:

- Data stored as 2-D numpy array with shape=(172800, 2048)
- There are 172,800 signal vectors in the dataset.
- Every signal vector has 2048 complex IQ samples in it.
- The signal lasts for 340 ms and has a sampling frequency (fs) of 6 kHz.
- The signals have a random frequency offset in the vicinity of 250 Hz and are centered at 0 Hz.
- The signals also receive random phase offsets.
- Signal strength is normalized to one.
- SNR: The dataset contains a range of Signal-to-Noise Ratio (SNR) values, which correspond to various noise levels: 25, 20, 15, 10, 5, 0, -5, and -10 dB.
- Fading channel: According to CCIR 520, the fading channel conforms to the Watterson Model. This explains how the signals were impacted by ionospheric propagation.

- Transmission Modes / Modulations: The dataset contains 18 different transmission modes or modulations.

1.2 Transmission modes

The following are the 18 transmission modes available in the dataset:

1. Morse: It is a classic communication method that relies on patterns of short and long signals to transmit text messages via telegraphy.
2. PSK31: It is a popular digital mode, enables low-power and highly efficient communication by phase-shifting the carrier signal to encode data.
3. QPSK31: It is a quadrature phase-shift variant of PSK31, improving spectral efficiency and robustness in amateur radio communications.
4. RTTY: RTTY45_170, RTTY50_170, and RTTY100_850 are Radio Teletype modes with different baud rates, offering reliable text transmission over radio waves.
5. Olivia: Olivia8_250, Olivia16_500, Olivia16_1000, and Olivia32_1000 are Olivia digital modes, known for their resistance to fading and interference.
6. DominoEX11: It is a versatile digital mode for ham radio operators, offering different submodes for text and data transmission.
7. MT63_1000: It is a digital mode optimized for weak-signal communication, known for its robust error correction.
8. NAVTEX: It is a dedicated mode for maritime safety information broadcasts over short-wave radio.
9. USB and LSB: USB (Upper Sideband) and LSB (Lower Sideband) are modulation modes commonly used for single-sideband (SSB) voice communication in amateur radio.
10. AM: AM (Amplitude Modulation) is a classic modulation method used in broadcasting, transmitting both voice and music over the airwaves.
11. FAX: FAX, short for facsimile, is a mode used to transmit images and documents over radio, particularly in weather charts and document broadcasting.

Mode Name	Modulation	Baud Rate
Morse Code	OOK	variable
PSK31	PSK	31
PSK63	PSK	63
QPSK31	QPSK	31
RTTY 45/170	FSK, 170 Hz shift	45
RTTY 50/170	FSK, 170 Hz shift	50
RTTY 100/850	FSK, 850 Hz shift	850
Olivia 8/250	8-MFSK	31
Olivia 16/500	16-MFSK	31
Olivia 16/1000	16-MFSK	62
Olivia 32/1000	32-MFSK	31
DominoEx	18-MFSK	11
MT63 / 1000	multi-carrier	10
Navtex / Sitor-B	FSK, 170 Hz shift	100
Single-Sideband (upper)	USB	-
Single-Sideband (lower)	LSB	-
AM broadcast	AM	-
HF/radiofax	radiofax	-

Table I
TRANSMISSION MODES

Figure 1.1: Transmission modes

Chapter 2

SHORT-TIME FOURIER TRANSFORM AND SPECTROGRAM

2.1 Short-Time Fourier Transform (STFT)

The Short-Time Fourier Transform (STFT) is a method employed in signal processing and time-frequency analysis to examine the evolution of frequency content within a signal over time. It proves especially beneficial for analyzing non-stationary signals, which are signals whose frequency characteristics vary over time.

Overlapping Segments are:

nperseg = 2048; Window width

noverlap = 1024; Overlap between segments

$$f, t, Sxx = stft(iq_samples, fs = fs, nperseg = nperseg, noverlap = noverlap)]$$

Where,

f: Frequency bins.

t: Time segments.

Sxx: Short-time Fourier Transform.

The STFT function is implemented on IQ samples, representing a portion of the input signal. The size of the window for each segment is determined by the nperseg parameter, while the degree of overlap between these segments is governed by the noverlap parameter.[2]

2.2 Spectrogram

A spectrogram serves as a graphical depiction of the spectrum of frequencies present in a signal, showcasing how these frequencies change over time. It provides a method for examining the frequency characteristics of a signal across different time intervals. Spectrograms find extensive utility across diverse fields such as audio processing, speech analysis, radar, sonar, and other domains where comprehending the frequency distribution of a signal over time holds significance.

Within a spectrogram, time is depicted along the horizontal axis, frequency along the vertical axis, and the intensity or amplitude of each frequency component is portrayed through color or brightness variations.

A dark region within the spectrogram suggests minimal energy at that frequency during a specific time interval. Conversely, a brighter area signifies greater energy or the presence of that frequency component.

For instance, in audio processing, a spectrogram proves invaluable for visualizing the variation of energy across different frequency bands over time. It serves as a crucial tool for tasks such as speech recognition, music analysis, and the identification of particular sounds within an audio signal.

Spectrograms are produced through the use of the Short-Time Fourier Transform (STFT) on a signal. This technique involves dividing the signal into small, overlapping segments and computing the Fourier Transform for each segment to analyze its frequency content. The outcome is displayed as a two-dimensional image, with time advancing horizontally from left to right and frequency ascending vertically from bottom to top.[3]

Here Figure. 2.1 and Figure. 2.2 are some spectrogram images which are obtained by performing STFT on the dataset.

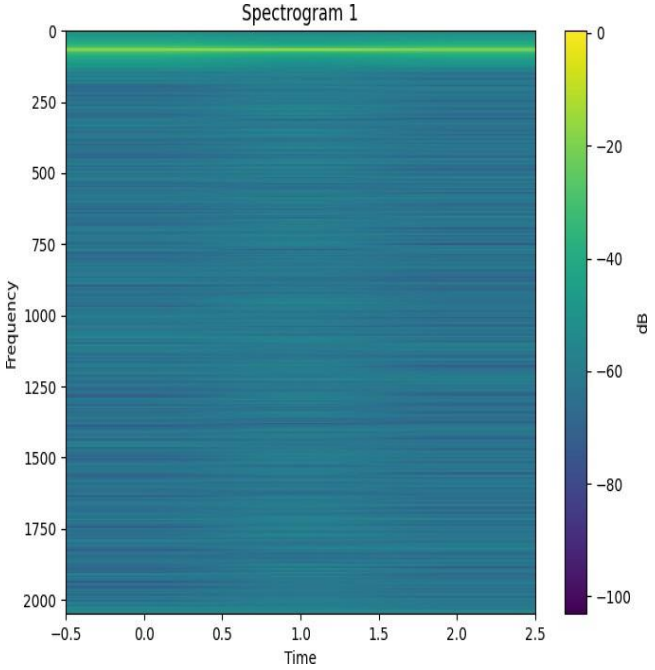

Figure 2.1: Spectrogram of a sample radio signal

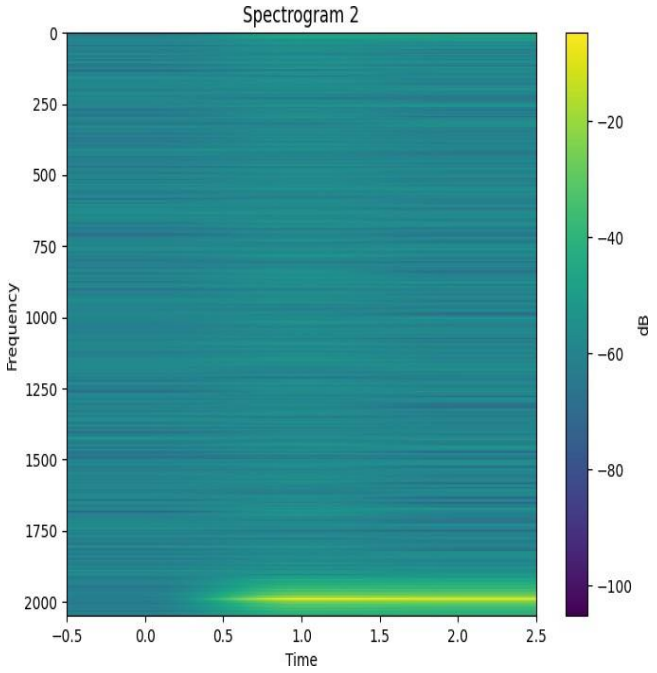

Figure 2.2: Spectrogram of another sample radio signal

2.3 Methodology

This section elaborates on the Short-Time Fourier Transform (STFT) and Spectrogram. In the original dataset, the signal values are initially in the time domain. Through the application of STFT, these values undergo conversion into the frequency domain. The Spectrogram then visually represents the frequencies present in the signal, illustrating how they change over time.

The code produces spectrograms for a dataset comprising IQ samples, with each spectrogram linked to a corresponding label. It utilizes the Short-Time Fourier Transform (STFT) to calculate spectrogram data, generating visual depictions of frequency content variations over time. These spectrograms are both displayed and saved as PNG images, with filenames incorporating row indices and labels. The script iterates through the dataset, generating spectrograms for every IQ sample-label pair. Ensure that the necessary libraries are imported and that the sequence of displaying and saving the spectrograms aligns with your requirements.

Chapter 3

HAAR WAVELET FEATURES

3.1 Feature Engineering - Haar wavelet Features

Accurate classification of high-frequency radio signals is vital for communication systems, requiring efficient methods for feature extraction to accurately differentiate modulation types. This report explores the implementation of Haar wavelet feature extraction within a project centered on high-frequency radio signal classification using Convolutional Neural Networks (CNNs).

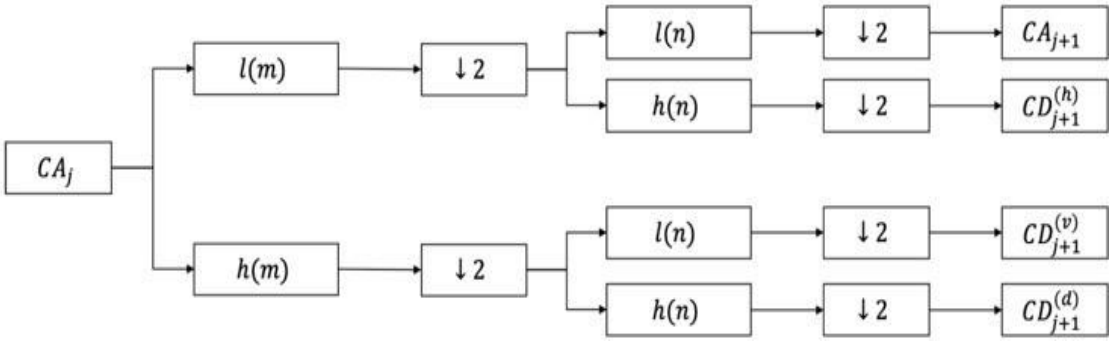

Figure 3.1: Process of HAAR features extraction

Haar-like features are the linchpin of our system, facilitating a comprehensive and efficient signal analysis. The distinctive capacity of Haar-like features to interpret intricate signal attributes ensures that our radio signal classification is not only highly accurate but also adaptable to a diverse array of scenarios. [4]

3.2 Haar Wavelet Transform

Extracting Haar wavelet features from spectrograms entails employing the Discrete Wavelet Transform (DWT) to decompose the spectrogram into approximation coefficients (cA) and detail coefficients (cD). The DWT is a mathematical method that dissects a signal or image into various frequency components, enabling the extraction of both low-frequency approximation information (cA) and high-frequency detail information (cD). With spectrograms representing

the frequency content of a signal over time, DWT proves instrumental in identifying pertinent patterns and features across both the time and frequency domains.

To extract Haar wavelet features from a spectrogram, the initial step involves applying the Discrete Wavelet Transform (DWT) to the spectrogram data. This procedure includes a sequence of filtering and down-sampling operations aimed at acquiring the approximation coefficients (cA) that portray the broad, low-frequency components and the detail coefficients (cD) that encapsulate the finer, high-frequency details. Notably, the Haar wavelet employs a straightforward, orthogonal wavelet basis, adept at efficiently representing abrupt changes or discontinuities in the signal.

After decomposing the spectrogram into its approximation (cA) and detail (cD) coefficients across multiple scales or levels, pertinent features are extracted from these coefficients. The approximation coefficients (cA) typically encapsulate the general trend or overarching structure of the spectrogram, whereas the detail coefficients (cD) accentuate specific transient or high-frequency components within the signal. Features such as energy distribution across various frequency bands, statistical properties (mean, variance, skewness, etc.), or entropy measures can be derived from these coefficients to delineate the spectrogram's distinct characteristics.

For example, analyzing the energy distribution across distinct frequency subbands within the approximation and detail coefficients can unveil crucial insights into the spectral content of the signal across different scales. Statistical metrics computed from these coefficients offer understanding into the signal's variability, texture, or anomalies across various frequency ranges over time. These extracted features, stemming from the Haar wavelet transform of the spectrogram, can subsequently serve as inputs for machine learning algorithms or pattern recognition endeavors, assisting in tasks like classification, denoising, or feature-based analysis of audio signals.[16]

3.3 Levels of HAAR Wavelet

Haar wavelet is one of the simplest wavelets used in signal processing and image compression. It's particularly popular due to its computational efficiency and simplicity. The Haar wavelet transform breaks down a signal or image into approximation and detail coefficients at various scales or levels. This process is commonly executed over five levels in the following manner:

- Level 1: At the first level, the signal or image is divided into two parts of equal length (or width, in the case of images). These parts represent the approximation (average) and detail (difference) coefficients at this level.
- Level 2: Each part obtained from the previous level is further decomposed similarly. So, each of the two parts at level 1 is divided into two subparts, resulting in four parts altogether. This level provides more detailed information about the signal or image.
- Level 3: Similar to level 2, each part from level 2 is further decomposed into two subparts, resulting in a total of eight parts at this level. This decomposition continues to provide more detailed information about the original signal or image.
- Level 4: Following the pattern, each part from level 3 is decomposed into two subparts, resulting in a total of sixteen parts at this level. The process continues, providing even more detailed information about the signal or image.
- Level 5: At this stage, each part from level 4 is decomposed into two subparts, resulting in a total of thirty-two parts. The decomposition process can be continued further if more levels of detail are desired.

Each level of decomposition provides information about the signal or image at different frequencies or resolutions. The approximation coefficients represent the low-frequency components, while the detail coefficients represent the high-frequency components. This multi-resolution analysis is useful in various applications such as compression, denoising, and feature extraction.

3.4 Methodology

This section introduces Haar features and technical aspects. Extracting Haar-like features from spectrograms involves a multi-step process. Initially, generate spectrograms from your audio data, often using methods such as Short-Time Fourier Transform (STFT) to produce a 2D matrix representing frequency content over time. Subsequently, define the Haar-like features you intend to extract, specifying their types and sizes. Common selections encompass horizontal, vertical, and diagonal features of various dimensions.

Following that, employ a sliding window technique, utilizing windows of the selected feature size to traverse the spectrogram. At each window position, compute the Haar-like feature value

by determining the difference between the sums of pixel intensities in particular regions based on the feature type. This extraction procedure is executed iteratively across the entire spectrogram. Subsequently, one might fine-tune the step size and scaling factors to regulate the extent and detail of the Haar-like feature extraction, aligning them with particular analysis objectives. Upon extraction, these features can be employed for diverse purposes, including visualization or as inputs for machine learning algorithms. Such features possess the potential to assist in tasks such as classification, object detection, or pattern recognition within your spectrogram data, contingent upon the demands of your application.

3.5 Data Flow

The image 3.2 is a block diagram of a wavelet processor. Wavelet processors are used in a variety of applications, including signal processing, image compression, and feature extraction. The diagram shows the different stages of the wavelet processing pipeline, from the input vector to the output.

- Input Vector: This is the signal that you want to process. It can be a time-series signal, an image, or any other type of data.
- PRE: This stage pre-processes the input vector. This may involve scaling the data, normalizing it, or padding it with zeros.
- FEATURE MAPS: This stage extracts features from the input vector. The features are represented as a set of wavelet coefficients.
- HAAR WAVELET: This stage uses the Haar wavelet to decompose the signal into its constituent frequencies. The Haar wavelet is a simple type of wavelet that is often used in wavelet processing because it is easy to compute.
- CONVOLUTION POOLING: These stages perform convolution and pooling operations on the wavelet coefficients. Convolution is a mathematical operation that is used to extract features from an image. Pooling is a mathematical operation that is used to reduce the dimensionality of the data.
- FULL CONNECTION: This stage is a fully connected layer that is used to make predictions based on the wavelet coefficients.

- Output: This is the final output of the wavelet processor. It can be a compressed version of the input signal, a set of features, or a prediction.

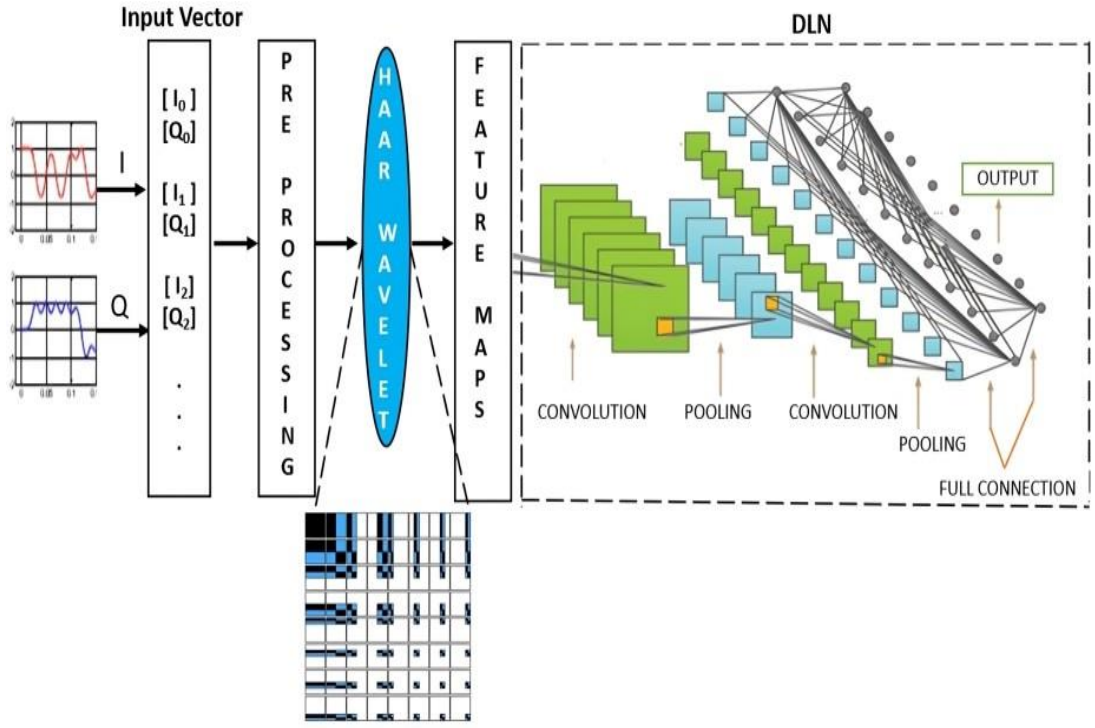

Figure 3.2: Data flow for a NAS-optimized Deep Learning Network using frequency domain signals (Haar-Wavelet transforms).

3.6 Selection of the best level

The selection of Haar wavelet level 2 as the main level for signal classification was based on thorough analysis and comparison of the deep CNN and ResNet models across various SNR levels. As we can see in Figure 3.3 and Figure 3.4, at level 2, both models consistently demonstrated superior performance compared to other levels, showcasing their ability to effectively capture and represent signal features for classification.

The deep CNN model exhibited remarkable accuracy at level 2, achieving up to 98.81% accuracy at SNR level 10. This indicates that the model was able to learn and differentiate signal patterns with high precision, making it a reliable choice for signal classification tasks. Similarly, the ResNet model also performed impressively at level 2, achieving accuracies as high as 98.53% at SNR level 25, highlighting its capability to effectively classify signals.

Comparing the overall performance of both models, the deep CNN model with level 2 Haar wavelet transformation emerged as the superior choice, achieving an overall accuracy of

93.77%. In contrast, the ResNet model achieved an overall accuracy of 84.26% at level 2. This significant difference in performance underscores the effectiveness of the deep CNN model for signal classification, particularly when combined with level 2 Haar wavelet transformation

In conclusion, the selection of level 2 for Haar wavelet transformation was a strategic choice based on the robust performance of the deep CNN model across various SNR levels. This decision ensures that the model is equipped to accurately classify signals, making it a valuable asset in signal processing applications where accuracy and reliability are paramount.

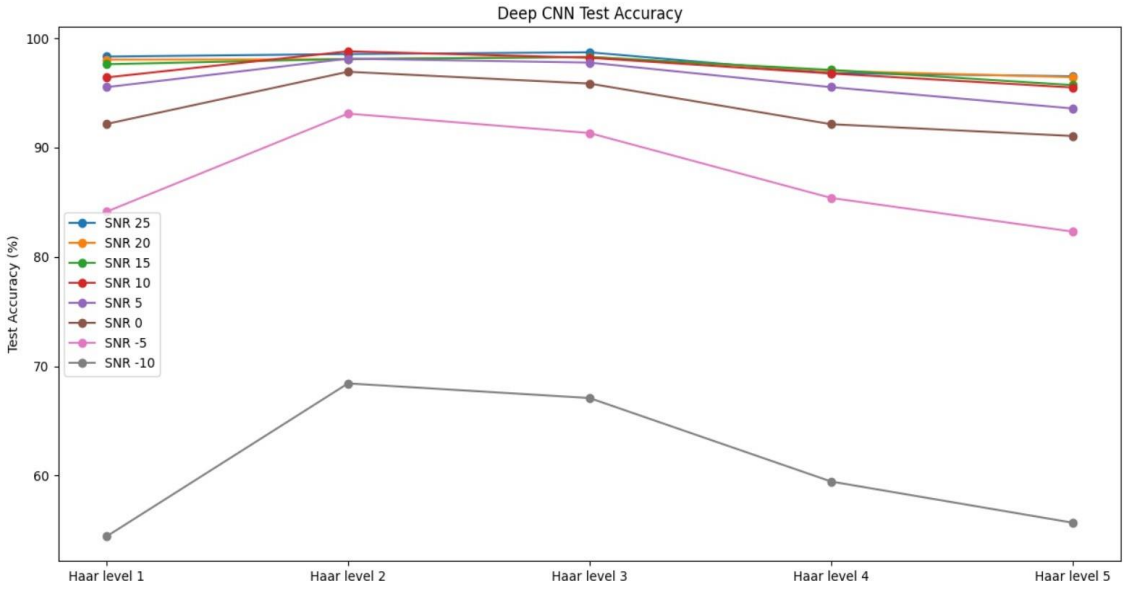

Figure 3.3: Selection of best HAAR level for CNN model

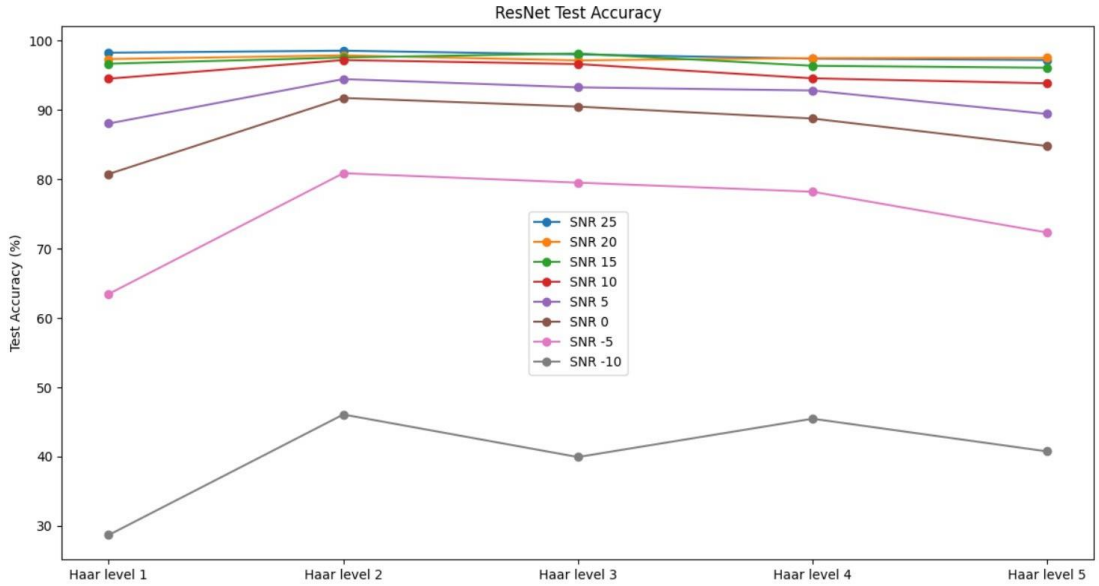

Figure 3.4: Selection of best HAAR level for Resnet model

Chapter 4

CONSTELLATION DIAGRAM

4.1 Feature Engineering - Constellation diagram

A constellation diagram visually represents a signal modulated through digital modulation techniques like phase-shift keying or quadrature amplitude modulation. This diagram visually displays the signal's characteristics during symbol sampling points on a two-dimensional scatter plot in the complex plane. Additionally, the distance of a point from the origin indicates the signal's amplitude or power. [14]

In digital communication systems, information is frequently represented by altering the amplitude, phase, or frequency of a carrier signal. Each combination of these properties denotes a distinct symbol or state. Constellation diagrams offer a method to visually represent these states, with each point on the diagram corresponding to a specific symbol or combination of signal properties.

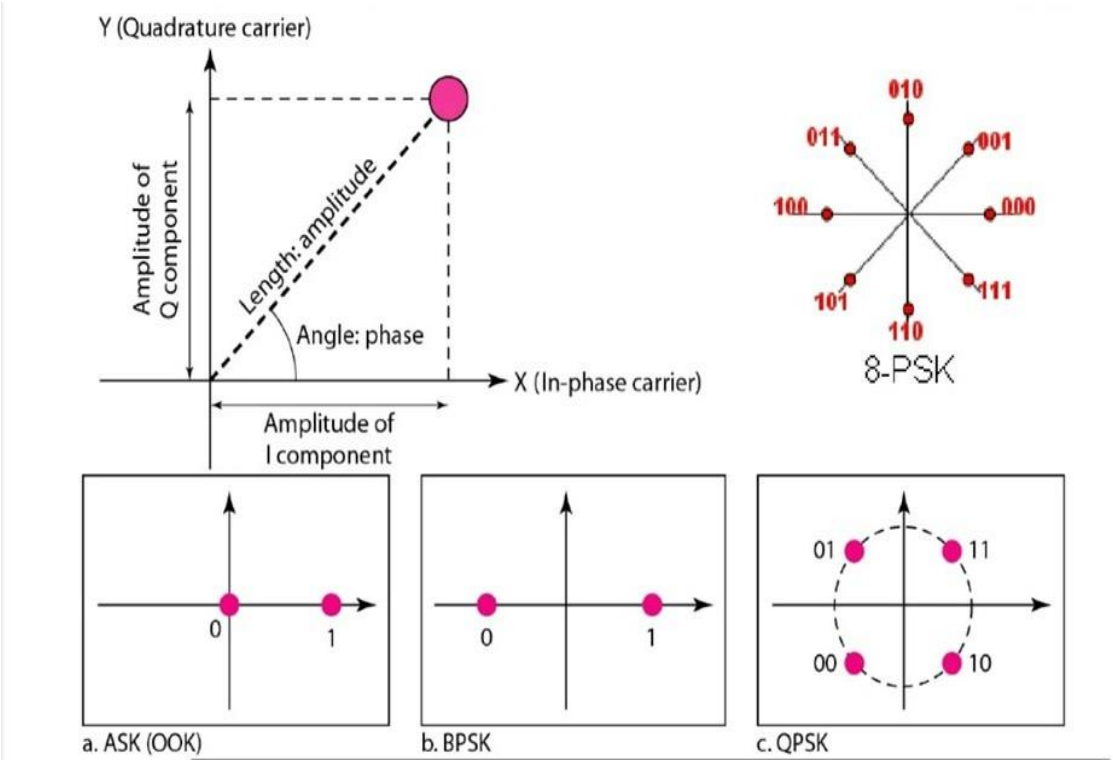

Figure 4.1: General idea of constellation diagrams.

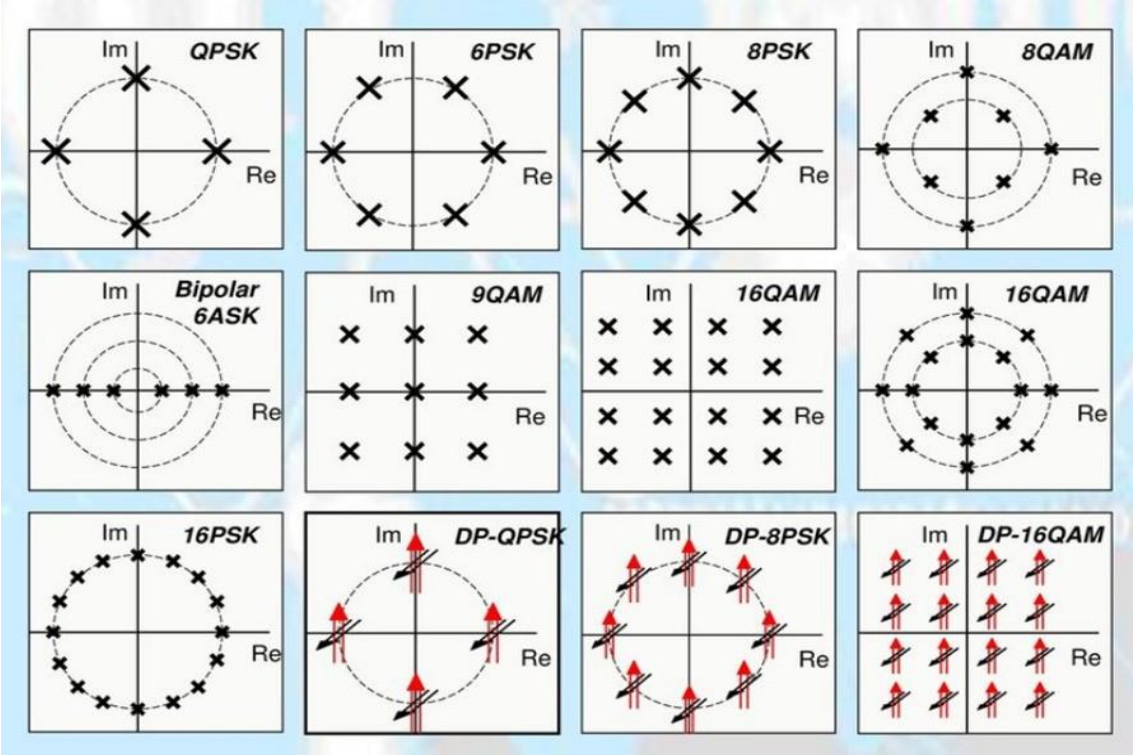

Figure 4.2: Constellation diagrams for Modulation types.

4.2 Diagram analysis

In digital modulation systems, sequential samples are taken at fixed time intervals to transmit data. The carrier wave, constrained to a finite range, maintains constant amplitude and phase in each sample. Each sample represents a finite set of symbols, each encoding one or more binary digits (bits) of data. These symbols are encoded using various combinations of carrier amplitude and phase, which are depicted as constellation points on a diagram. These points, representing all possible symbols transmitted by the system, are located on a circle centered around the origin in frequency or phase-modulated signals, where the signal's amplitude remains unchanged.

A sine wave that has been 90° shifted—known as the ”Q” (quadrature) carrier—and a cosine wave that represents the ”I” (in-phase) carrier are combined to create the carrier for each symbol. As a result, the constellation diagram becomes a complex plane since each symbol can be represented as a complex number. The Q component is represented vertically by the imaginary axis, and the I component is represented horizontally by the real axis. Quadrature modulation is based on the independent demodulation of these carriers made possible by coherent detection.

When a modulating symbol is in perfect phase with the carrier, the modulated signal is best represented. This is known as pure phase modulation.

A 'signal space diagram,' an ideal constellation diagram, accurately depicts the position of each symbol. However, as the signal passes through a communication channel, electronic noise or distortion may alter the received amplitude and phase, causing the demodulator to deviate from the correct symbol position. When plotted on a constellation diagram, the point representing the received sample is offset from the correct symbol position. A vector signal analyzer, an electronic test instrument, can display the constellation diagram of a digital signal by sampling and plotting each received symbol as a point. The outcome is a 'ball' or 'cloud' of points surrounding each symbol position, allowing the identification of interference and distortion types in the signal.

The size of the "alphabet" symbol that each sample in a diagram can transmit is determined by the number of constellation points, which also determines the number of bits transmitted per sample, which is usually a power of 2. To enable the transmission of two bits per sample, a modulation scheme that can encode all four combinations of two bits—00, 01, 10, and 11—is represented by a diagram with four points. A modulation with N constellation points typically transmits log2N bits for each sample.

Following transmission through the communication channel, the signal undergoes decoding by a demodulator. The demodulator's role is to categorize each sample as a symbol, and the set of sample values classified as a particular symbol forms a region in the plane around each constellation point. If noise causes a sample's representation point to stray into another symbol's region, the demodulator misidentifies the sample, resulting in a symbol error. Demodulators often employ maximum likelihood detection, estimating the transmitted symbol as the constellation point closest (in Euclidean distance) to the received sample. The detection regions on the constellation diagram are easily represented by dividing the plane with lines equidistant from each adjacent pair of points.

Half the distance between neighboring points corresponds to the amplitude of additive noise or distortion required to cause misidentification and symbol error. Consequently, modulation systems aim to maximize the minimum noise necessary to induce a symbol error, implying equal distances between adjacent points on the constellation diagram for optimal noise immunity.[12]

The quality of the received signal is analyzed by displaying the constellation diagram using a vector signal analyzer. Various types of distortion manifest as characteristic patterns on the

diagram:

- Gaussian noise results in samples forming a random ball around each constellation point.
- Non-coherent single frequency interference appears as samples circling each constellation point.
- Phase noise causes constellation points to spread into arcs centered on the origin.
- Amplifier compression causes corner points to move toward the center.

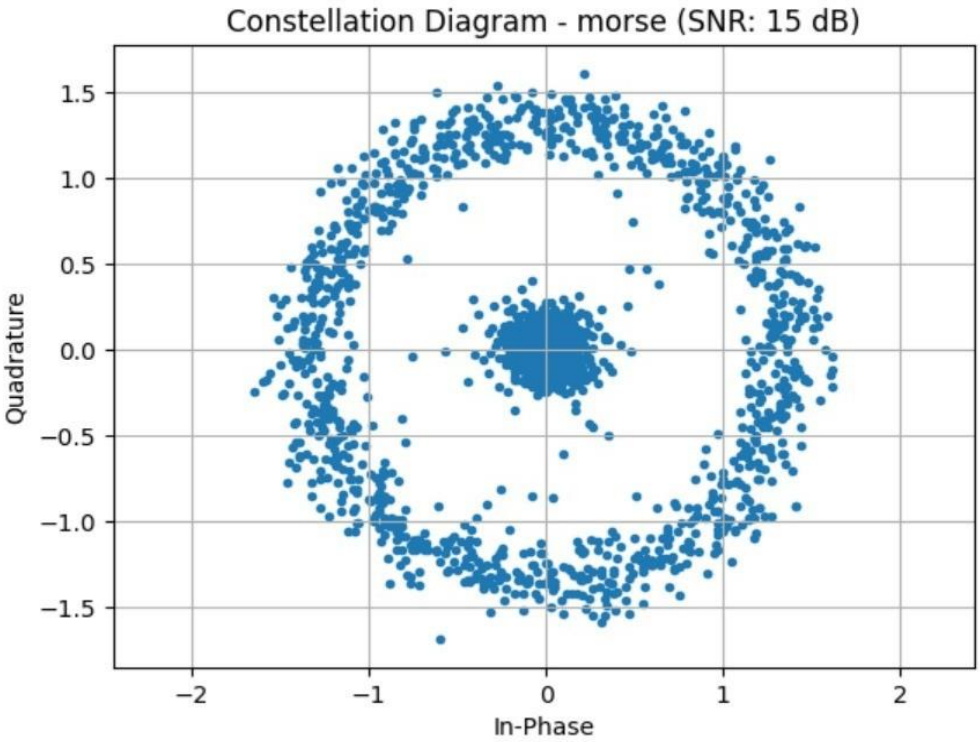

Figure 4.3: Constellation diagram generated for signal IQ samples.

4.3 Data flow

The image 4.4 is a diagram of a clustering process using a constellation diagram to generate a feature map.

- Input Vector: This represents the data points being clustered. It's denoted by a blue circle on the left side of the diagram.
- Constellation Diagram: This is a visual representation of the input data, where each data point is mapped to a star-like symbol (constellation) in a 2D space. The positions of the

stars are determined by the features of the data points. You can see this constellation diagram in the center of the image.

- DLN: This likely refers to a Deep Learning Network, which is used to process the constellation diagram and extract features from it. The specific details of the network are not shown in the diagram.
- PRE: This stage preprocesses the input data before it is fed into the DLN. The exact preprocessing steps are not shown in the diagram.
- FEATURE MAPS: This refers to the output of the DLN. It is a 2D representation of the input data, where each pixel in the map corresponds to a specific feature or cluster. You can see two feature maps on the right side of the diagram, labeled [0] and [6].
- PROCESSING: This stage likely refers to additional processing that is performed on the feature maps, such as dimensionality reduction or clustering. The specific details of the processing are not shown in the diagram.
- OUTPUT: This represents the final result of the clustering process. It could be a set of cluster labels for each data point, or a visualization of the clustered data points. The output is not shown in the diagram.

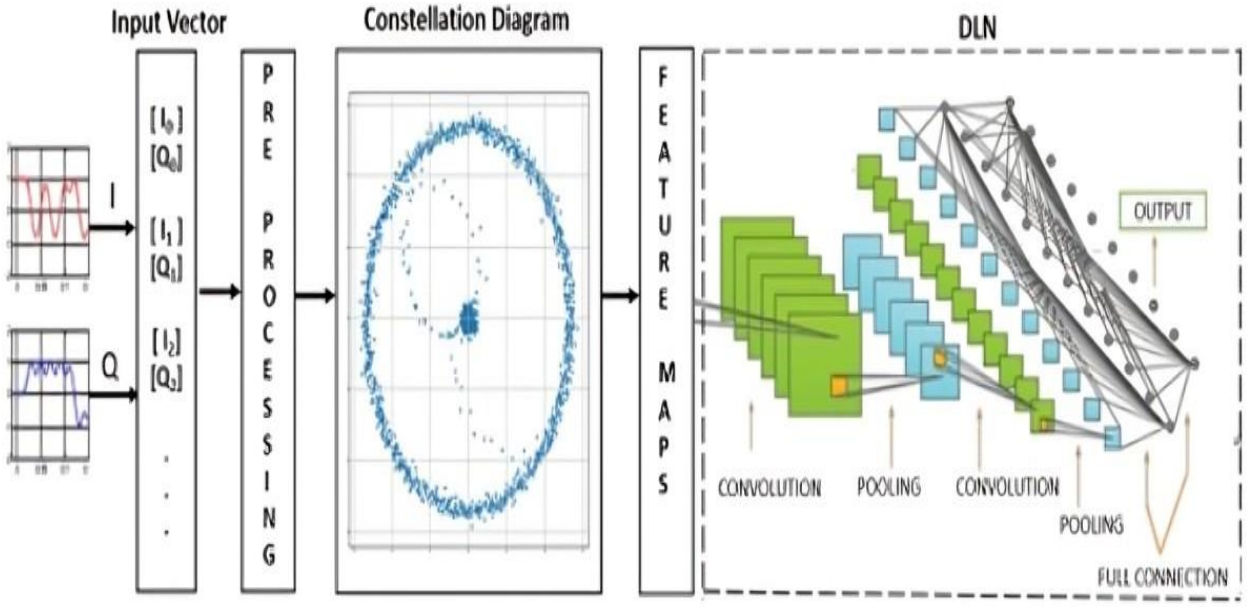

Figure 4.4: Data flow for an NAS-optimized Deep Learning Network using Constellation Diagrams (Constellation Diagrams).

Chapter 5

RESIDUAL NET AND DEEP CNN

5.1 Data Flow

The image 5.1 depicts a block diagram of a process mapping process. This process takes input data and transforms it into output data through a series of intermediate stages. Here's a breakdown of the different stages:

- Input Vector: This is the initial data that you want to process. It can be represented in various forms, such as numbers, text, or images. In the diagram, it's denoted by the blue circle on the left side.
- PRE: This stage preprocesses the input data to prepare it for further processing. This might involve tasks like scaling the data, normalizing it, or padding it with zeros.
- FEATURE MAPS: This stage extracts relevant features from the preprocessed data. These features are essentially characteristics or patterns that the process aims to identify or utilize. The extracted features are then arranged into "feature maps," which are visual representations of the data's key characteristics. In the diagram, you can see two feature maps on the right, labeled "[I0]" and "[Q0]".
- Convolution and Pooling: These are two fundamental operations used in many processing tasks, including image and signal processing. Convolution involves applying a filter to the data to extract specific features. Pooling then reduces the dimensionality of the data by summarizing or combining elements. The diagram depicts these operations happening iteratively, with two stages of "Convolution Pooling" followed by another two stages of "Convolution Pooling".
- Full Connection: This stage integrates the extracted features from the previous stages. It essentially involves connecting all the features in a comprehensive way, allowing the process to make sense of the relationships between them. In the diagram, this is represented by the "FULL CONNECTION" block.

- OUTPUT: This is the final result of the process mapping, represented by the green circle on the right side. The output data can be in various forms, depending on the specific application. It could be a compressed version of the input data, a set of extracted features, or a prediction based on the processed information.

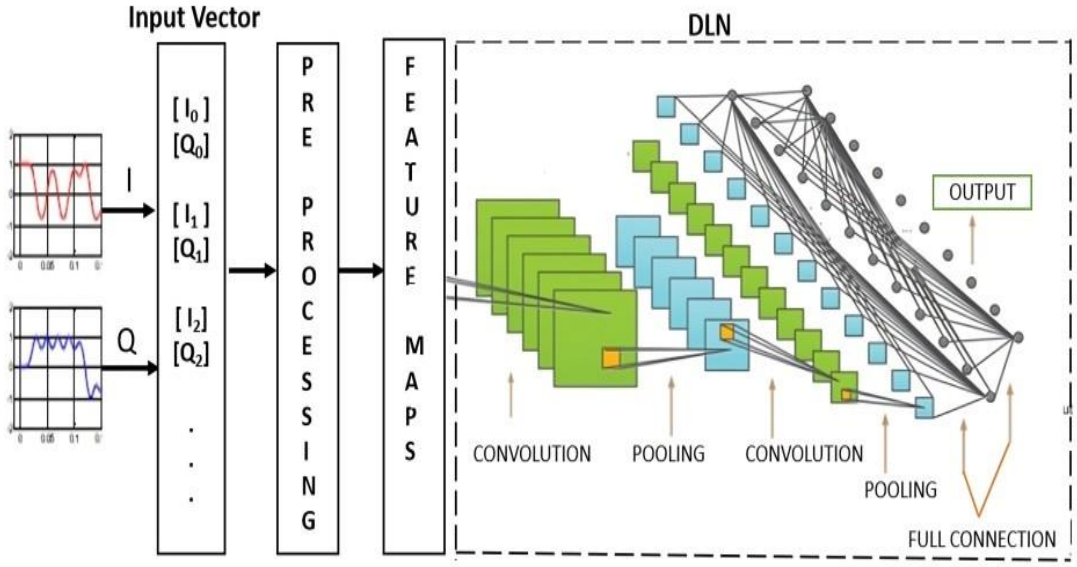

Figure 5.1: Data flow for a NAS-optimized DLN using time domain signals (I/Q samples)

5.2 Residual Net (Resnet)

ResNet, short for Residual Networks, is a deep neural network architecture introduced to address the problem of vanishing gradients in very deep convolutional neural networks (CNNs). ResNet's key innovation is the use of residual connections, which allow information to bypass certain layers during training. This mitigates the vanishing gradient problem and enables the training of exceptionally deep networks, making it easier to optimize and leading to improved accuracy in image recognition tasks.[7]

5.2.1 Residual Stack

A "Residual Stack" typically refers to a series of residual blocks, which are fundamental building blocks in Residual Neural Networks (ResNets). A residual block consists of several layers,

usually including convolutional layers, batch normalization, and ReLU activation functions. The key innovation in a residual block is the presence of skip connections, which allow the network to learn residual functions.

Each ”Residual Stack” you mentioned has a specific number of filters. A filter corresponds to the number of output channels from the convolutional layers within the block. For example, ”Residual Stack 96” has 96 filters, and ”Residual Stack 128” has 128 filters. The number of filters determines the dimensionality of the feature maps produced by these blocks.

5.2.2 Dropout

Dropout is a regularization technique commonly used in neural networks to prevent overfitting. The ”dropout 0.5” and ”dropout 0.7” layers indicate that dropout is applied with different dropout rates. The dropout rate represents the fraction of neurons (or units) that are randomly deactivated during each training iteration. A dropout rate of 0.5 means that, on average, 50% of the neurons are deactivated during training, and a rate of 0.7 means that, on average, 70% are deactivated.

5.2.3 Dense Layer

The ”dense, 18” layer indicates a fully connected (dense) layer with 18 units. This is typically used for the final classification layer in many neural network architectures. The output of this layer will be used for making predictions, and it’s common to apply a softmax activation function to obtain class probabilities.

In the architecture of Resnet, there are Multiple stacks of residual blocks with 96 filters, Two dropout layers with rates of 0.5, Multiple stacks of residual blocks with 128 filters, One dropout layer with a rate of 0.7, and A dense layer with 18 units for classification.

5.3 Deep CNN

Deep CNNs, or Deep Convolutional Neural Networks, are a class of neural network architectures designed for tasks involving visual data, such as image and video analysis. They consist of multiple layers of convolutional and pooling operations that automatically extract hierarchical features from input images. Deep CNNs have revolutionized computer vision and are essential in various applications, including image classification, object detection, and image segmentation, where they excel at learning and representing complex patterns and structures in visual data.[7]

5.3.1 Residual Stack

The model architecture consists of multiple stacks of convolutional layers with varying numbers of filters, similar to the concept of residual blocks used in Residual Neural Networks (ResNets). Each stack contains convolutional layers, batch normalization, and ReLU activation functions. These stacks form the core building blocks of the model's feature extraction process.

First Residual Stack: Multiple convolutional layers with 64 filters.

Second Residual Stack: Multiple convolutional layers with 128 filters.

Third Residual Stack: Multiple convolutional layers with 256 filters.

5.3.2 Dropout

Dropout layers are employed for regularization, mitigating overfitting during training. Two dropout layers are included in the model with different dropout rates. The dropout rate represents the proportion of neurons randomly deactivated during each training iteration.

First Dropout Layer: Dropout rate of 0.5.

Second Dropout Layer: More aggressive dropout rate of 0.7.

5.3.3 Dense Layer

Dense Layer: The architecture concludes with a fully connected (dense) layer containing 18 units. This layer is responsible for classification, converting the extracted features into class probabilities. A softmax activation function is commonly applied to obtain these probabilities.

In summary, the model architecture leverages multiple stacks of convolutional layers with varying filter counts, utilizes dropout for regularization, and culminates in a dense layer for classification into one of 18 categories.

In the architecture of Resnet, there are Multiple stacks of convolutional layers with 64 filters, Multiple stacks of convolutional layers with 128 filters, One dropout layer with a rate of 0.5, Multiple stacks of convolutional layers with 192 filters, One dropout layer with a rate of 0.7, A dense layer with 18 units for classification.

Chapter 6

GENETIC ALGORITHM NEURAL ARCHITECTURE SEARCH

6.1 Neural Architecture Search

The automatic process of architecture searching is called NAS (Neural Architecture Search). It is a technique for automating the design of artificial neural networks (ANN), a widely used model in the field of machine learning. NAS has been used to design networks that are on par with or outperform hand-designed architectures. [11]

There are three components to the NAS method:

- The collection of potential architectures that can be created and optimized is expressed by the search space.
- Search Strategy outlines the space research needed to locate quality architecture.
- A performance estimation approach that evaluates each architecture's performance after taking the precise search strategy into account.

6.2 Genetic Algorithm

GA is a search method that is based on populations. It begins with a population of people chosen at random. Every point in the research space represents a single GA individual in an optimization issue. A population's individual members are also referred to as chromosomes. Every chromosome was additionally composed of gene sets. Every gene is a representation of an individual's basic traits. It adheres to the same concepts of crossover, mutation, and selection as does natural evolution. The selection mechanism attempts to highlight the population's most suitable elements for a specific criterion. Conversely, crossover and mutation referred to as GA's genetic operators, keep the search space explored.[15]

To advance a person from one generation to the next, GA uses the following processes. First, using a selection method, individuals are chosen from the provided population. To make children, the crossover operator then chooses two chromosomes and partially swaps them. A mutation can maintain elitism in the population by arbitrarily flipping some of the genes of

freshly created progeny. Every person's fitness is evaluated by GA's fitness function. The fittest people are finally introduced for the following generation. Until the maximal generation is reached or a predetermined condition is satisfied, this GA process usually repeats itself.

The image 6.1 explains the working of Genetic Algorithm.

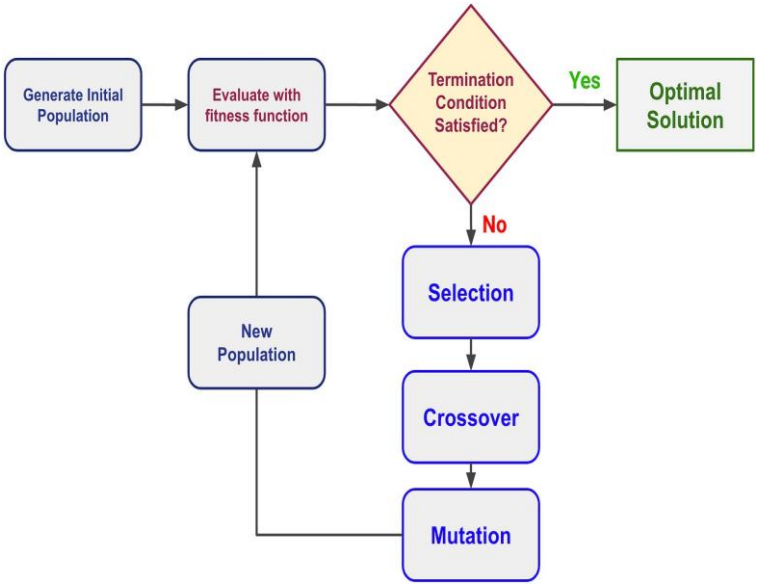

Figure 6.1: Genetic Algorithm

6.2.1 Chromosome Structure

When dealing with discrete gene values, the proposed chromosome forms a linear structure. Each chromosome encompasses multiple hyperparameter values. Within the solution set, each chromosome denotes a distinct neural network architecture. The genes within the chromosome encode various architectural aspects, including the count of hidden layers, the number of units (neurons) within each hidden layer, the chosen activation function, and the network optimizer, in respective order.

6.2.2 Fitness Evaluation

Crafting a fitness function is pivotal in shaping the efficacy of a genetic algorithm paradigm. A robust fitness function assumes the role of an objective evaluator in single-objective optimization scenarios. It translates a feasible solution into a scalar value, offering a concise assessment of its proximity to predefined objectives. In the realm of machine learning, the accuracy of the validation dataset serves as a crucial metric, indicating the performance of a given model in

classification tasks. Opting for the validation dataset's higher prediction rate, often synonymous with accuracy, becomes the model's fitness value, reflecting its effectiveness.

6.2.3 Selection

At the outset of the genetic algorithm (GA), the initial population is randomly assembled. Various selection mechanisms exist within GAs, and for this paper, a tournament selection mechanism has been chosen. In tournament selection, individuals are pitted against each other based on their fitness values. The fitness function utilized in this paper is termed noisy, meaning it produces varying output values for identical input values across successive evaluation steps. This is due to the absence of assurance that training the same neural architecture multiple times will yield identical results. Despite this noise, the fitness function yields a consolidated fitness value. Individuals exhibiting high accuracy (fitness) are then favored for inclusion in the subsequent population.

6.2.4 Crossover

By combining the genotypes of two parents, an offspring's genotype is created through a crossover operation. A pair of genotypes, parent1 and parent2, chosen through tournament selection, are subjected to crossover. Initially, a random number between 0 and 1 is selected. Parent 1 and Parent 2 undergo the crossover operation if it is less than the threshold; if not, they are merged into the next generation following a mutation step. A point has been arbitrarily selected as the crossover point from each parent. Every genome has a crossover point that divides it in half. The first offspring is created by concatenating the left portion of parent1 to the right part of parent2, and the left part of parent2 to the right part of parent1.

6.2.5 Mutation

The mutation operation makes it possible to change one gene value per person. Modify a few hyper-parameter values at random. Every individual will undergo the mutation operation, which has a random mutation probability.

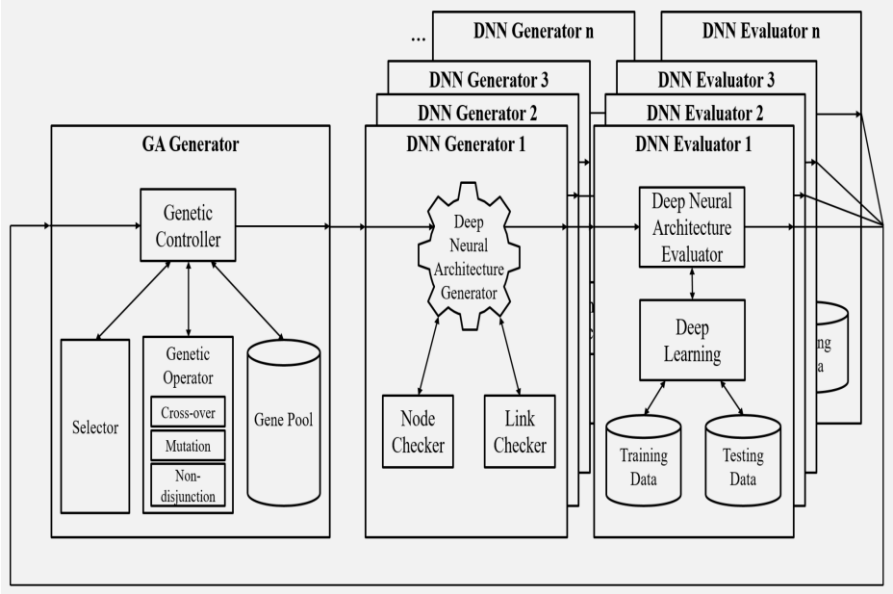

Figure 6.2: Working of GNAS

6.3 Implementation

Main aim of genetic algorithm of is to choose the best combination between the different choices given for each parameter.

Parameters of GA:

- Filters - Same as neurons in this case - total 5 filters for five layers
- Kernels - a small matrix used for the convolution operation in convolutional neural networks (CNNs) - 5 kernels for architecture(1,2,3,4,5,6,7)
- Activation function - the functions used are relu, gelu, sigmoid, tanh
- Dropout - the practice of disregarding certain nodes in a layer at random during training - 0.01, 0.02, 0.05, 0.07, 0.1, 0.2, 0.5
- Fitness Evaluator – Roulette Wheel

Chapter 7

TRAINING OUTPUTS AND RESULTS

7.1 Using CNN Model

Implementing the deep learning model, Convolutional Neural Network (CNN), for the classification of high-frequency radio signals. This is done in two different ways, which are by using IQ (In-phase and Quadrature-phase) samples and HAAR Wavelet features as inputs. The process involves pre-processing the data, defining and training the models, and evaluating their performance.

7.1.1 Training with IQ Samples

When the data was trained and tested using IQ samples as input, the Training accuracy we got was 93.22 percent and the Testing accuracy was 92.64 percent.

From Fig.7.1, we can observe that SNR 25 gave us the highest accuracy whereas SNR -10 gave us the lowest accuracy, this happens because in the SNR -10 range, the external noise for the signals is greater. From Fig.7.2 we can observe the confusion matrix for all the 18 signal types and we can conclude that the signal type ”olivia16_500” is the one that is most differing from others. Fig.7.3 and Table 7.1.1 give the complete details of the training and testing accuracy plots as well as the classification report.

7.1.2 Training with HAAR Features

The data when trained and tested using HAAR wavelet as input, the Training accuracy we got was 97.18 percent and the Testing accuracy was 93.77 percent.

From Fig.7.4, we can observe that SNR 10 and 25 gave us the highest accuracy whereas SNR -10 gave us the lowest accuracy, this happens because in the SNR -10 range, the external noise for the signals is greater. From Fig.7.5 we can observe the confusion matrix for all the 18 signal types and we can conclude that the signal types ”olivia16_500 and psk31” are the ones that most differ from others. Fig.7.6 and Table 7.1.2 give the complete details of the training and testing accuracy plots as well as the classification report.

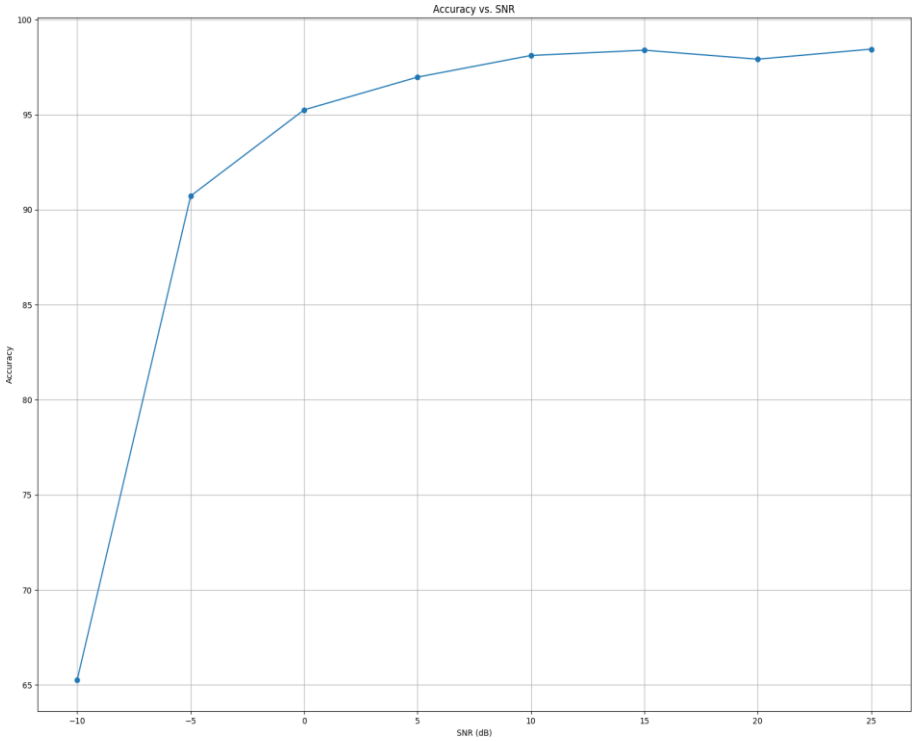

Figure 7.1: Plot between SNR values and their accuracies

Signals	Precision	Recall	f1-score	Support
am	0.81	0.93	0.86	1600
dominoex	0.88	0.93	0.91	1600
fax	0.92	0.90	0.91	1600
lsb	1.00	0.96	0.98	1600
morse	0.99	0.91	0.95	1600
mt63_1000	0.96	0.99	0.97	1600
navtex	0.98	0.95	0.96	1600
olivia16_1000	0.99	0.96	0.97	1600
olivia16_500	0.81	0.87	0.84	1600
olivia32_1000	0.94	0.88	0.90	1600
olivia8_250	0.83	0.88	0.86	1600
psk31	0.81	0.92	0.86	1600
psk63	0.97	0.91	0.94	1600
qpsk31	0.90	0.77	0.83	1600
rtty100_850	0.99	0.98	0.99	1600
rtty45_170	0.90	0.90	0.90	1600
rtty50_170	0.93	0.90	0.91	1600
usb	0.98	0.97	0.98	1600

Table 7.1: Classification table for IQ samples using CNN model

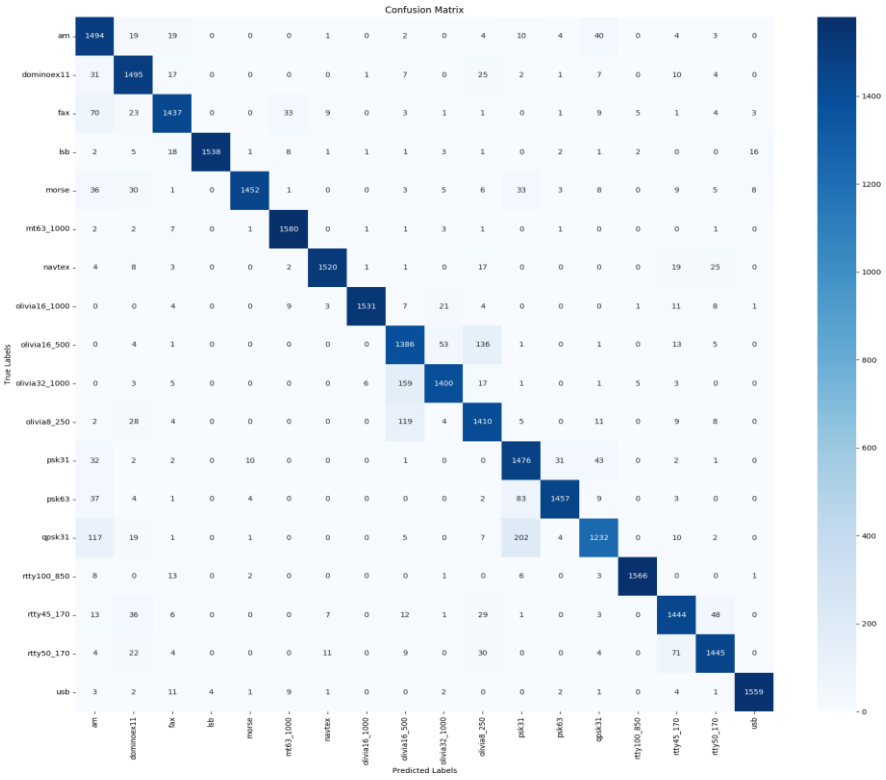

Figure 7.2: Confusin matrix for IQ samples using CNN model

Signals	Precision	Recall	f1-score	Support
am	0.87	0.92	0.89	1600
dominoex	0.95	0.95	0.95	1600
fax	0.92	0.91	0.92	1600
lsb	0.97	0.94	0.96	1600
morse	0.98	0.96	0.97	1600
mt63_1000	0.97	0.98	0.98	1600
navtex	0.99	0.99	0.99	1600
olivia16_1000	0.98	0.97	0.97	1600
olivia16_500	0.93	0.85	0.89	1600
olivia32_1000	0.96	0.92	0.94	1600
olivia8_250	0.87	0.97	0.92	1600
psk31	0.82	0.94	0.87	1600
psk63	0.93	0.92	0.93	1600
qpsk31	0.89	0.81	0.85	1600
rtty100_850	0.98	0.98	0.98	1600
rtty45_170	0.96	0.95	0.96	1600
rtty50_170	0.96	0.96	0.96	1600
usb	0.98	0.95	0.96	1600

Table 7.2: Classification table for HAAR features using CNN model

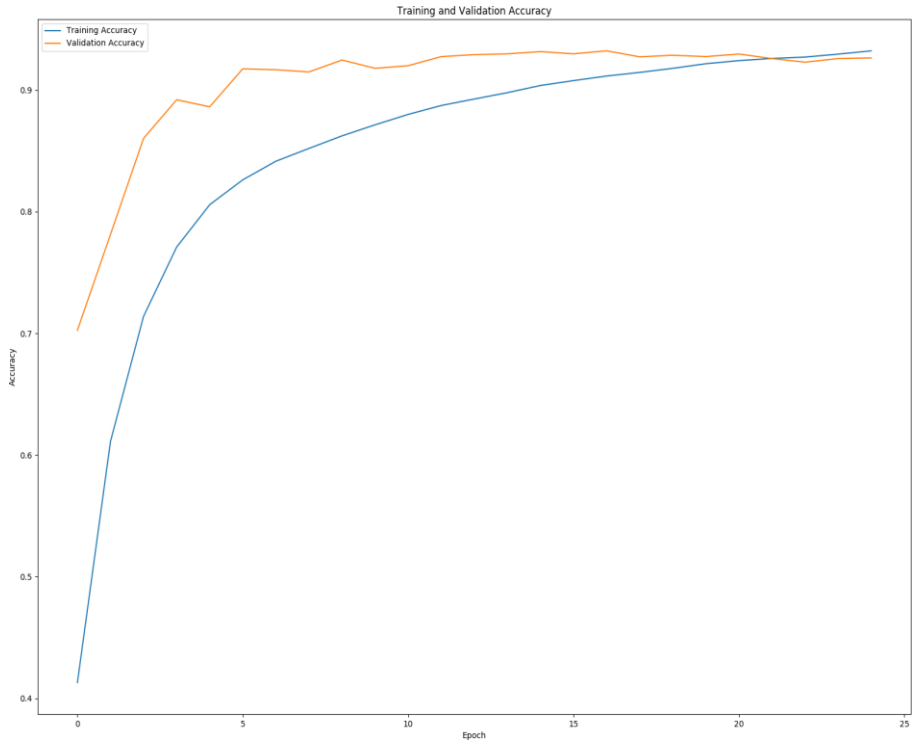

Figure 7.3: Training and Testing accuracies for for IQ samples using CNN model

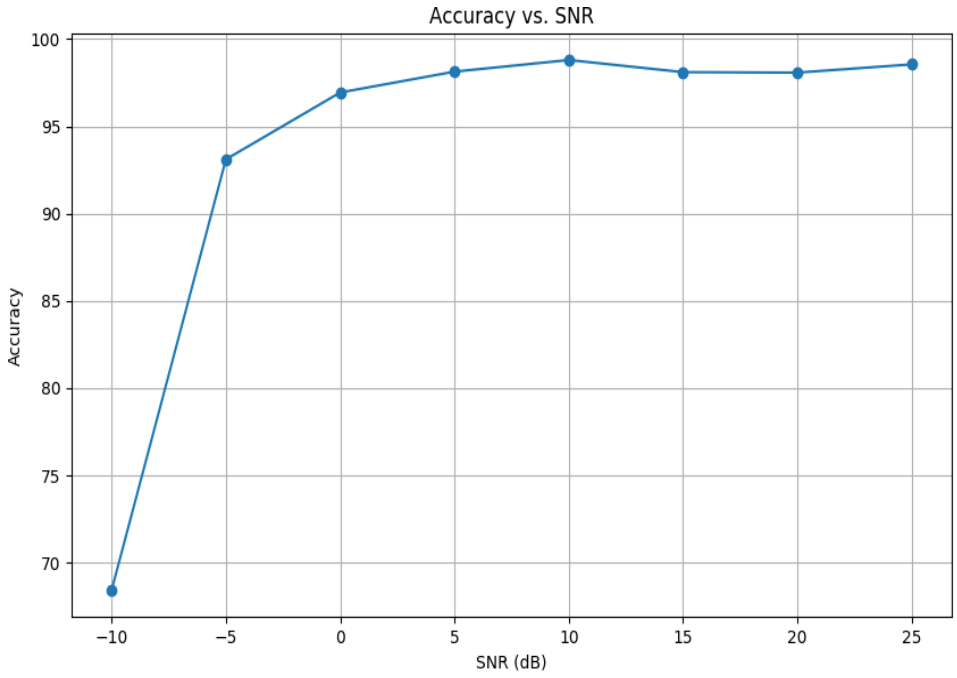

Figure 7.4: Plot between SNR values and their accuracies

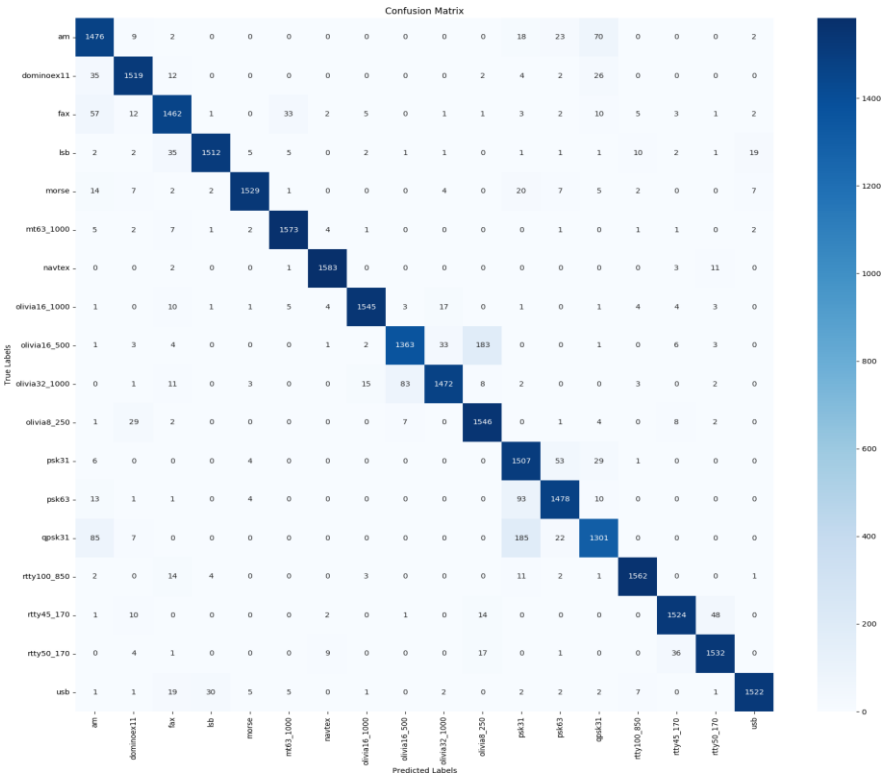

Figure 7.5: Confusin matrix for HAAR features using CNN model

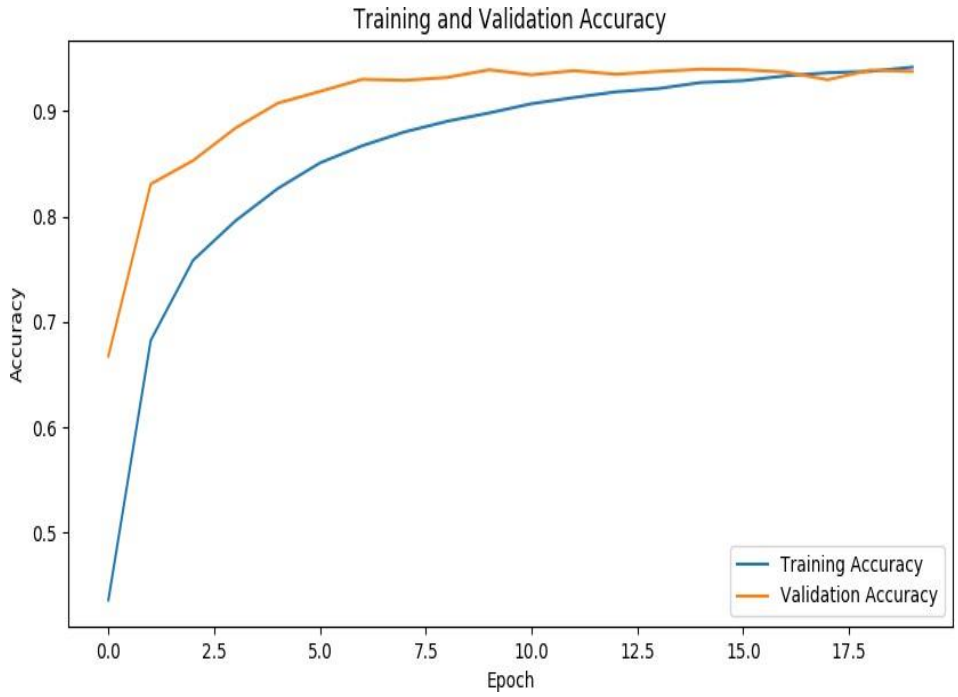

Figure 7.6: Training and Testing accuracies for HAAR features using CNN model

Signals	Precision	Recall	f1-score	Support
am	0.91	0.53	0.67	1600
dominoex	0.68	0.68	0.68	1600
fax	0.89	0.84	0.86	1600
lsb	0.99	0.97	0.98	1600
morse	0.92	0.90	0.91	1600
mt63_1000	0.91	0.98	0.94	1600
navtex	0.74	0.75	0.75	1600
olivia16_1000	0.92	0.69	0.79	1600
olivia16_500	0.58	0.76	0.66	1600
olivia32_1000	0.79	0.76	0.78	1600
olivia8_250	0.59	0.60	0.60	1600
psk31	0.74	0.73	0.74	1600
psk63	0.60	0.93	0.73	1600
qpsk31	0.85	0.52	0.64	1600
rtty100_850	0.96	0.94	0.95	1600
rtty45_170	0.45	0.21	0.28	1600
rtty50_170	0.38	0.72	0.50	1600
usb	1.00	0.96	0.98	1600

Table 7.3: Classification table for IQ samples using Resnet model

7.2 Using Resnet Model

Implementing the deep learning model, Resnet, for the classification of high-frequency radio signals. This is done in two different ways, which are by using IQ (In-phase and Quadrature-phase) samples and HAAR Wavelet features as inputs. The process involves pre-processing the data, defining and training the models, and evaluating their performance.

7.2.1 Training with IQ Samples

The data when trained and tested using IQ samples as input, the Training accuracy we got was 72.21 percent and the Testing accuracy was 74.66 percent.

From Fig.7.7, we can observe that SNR 25 gave us the highest accuracy whereas SNR -10 gave us the lowest accuracy, this happens because in the SNR -10 range, the external noise for the signals is greater. From Fig.7.8 we can observe the confusion matrix for all the 18 signal types and we can conclude that the signal types ”rtty45_170” are the ones that most differ from others. Fig.7.9 and Table 7.2.1 give the complete details of the training and testing accuracy plots as well as the classification report.

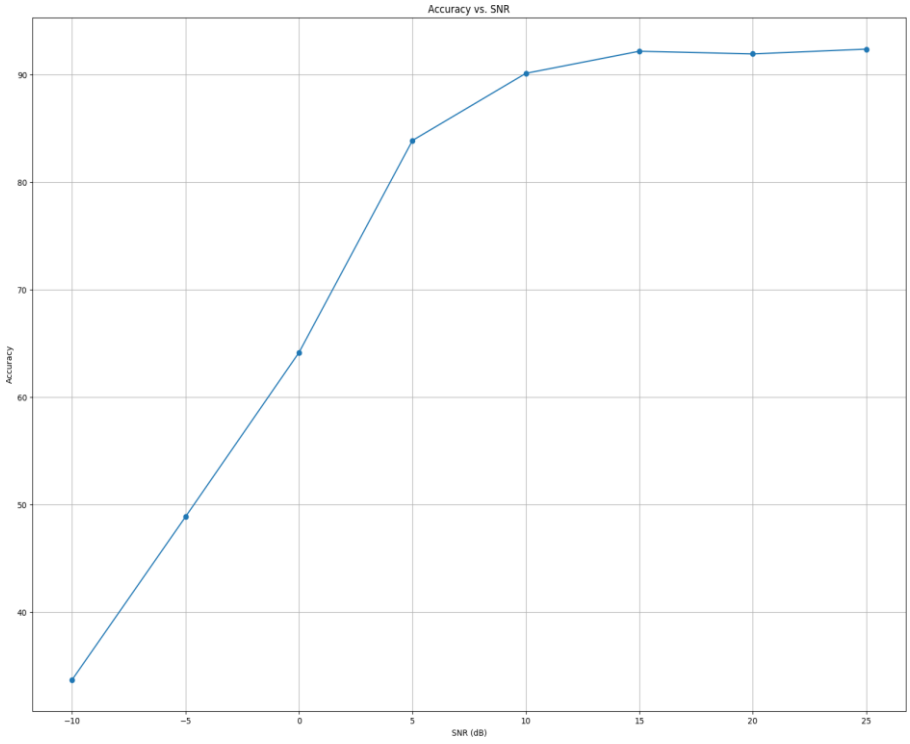

Figure 7.7: Plot between SNR values and their accuracies

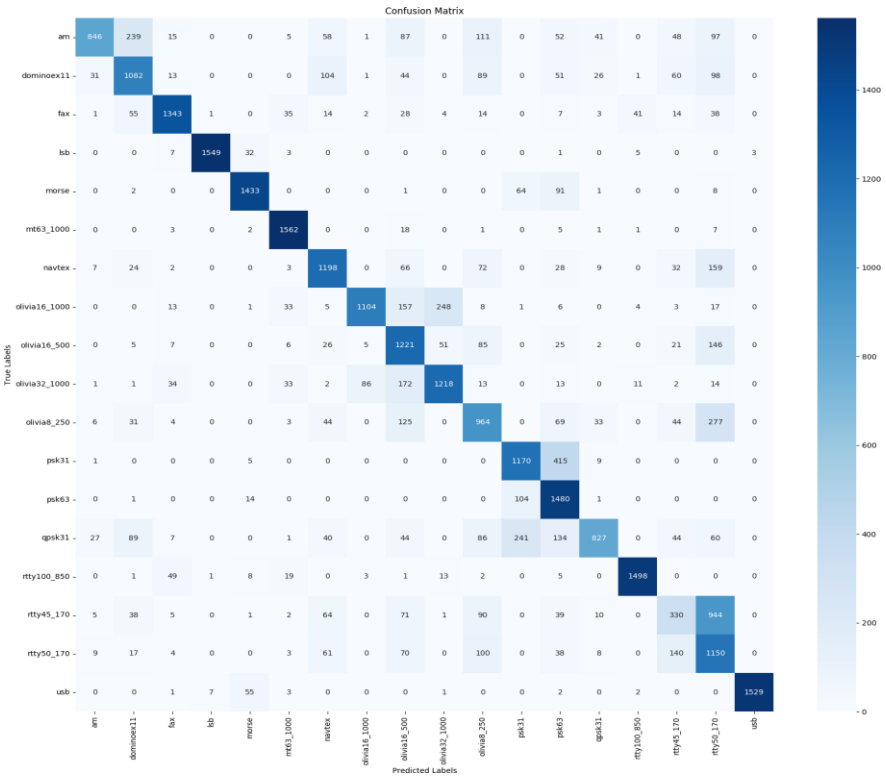

Figure 7.8: Confusin matrix for IQ samples using Resnet model

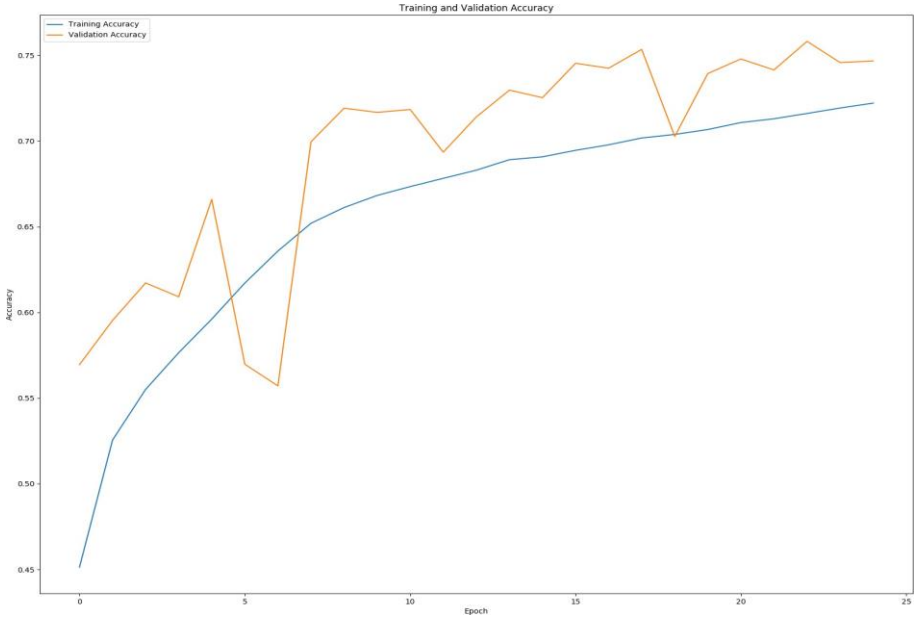

Figure 7.9: Training and Testing accuracies for IQ samples using Resnet model

7.2.2 Training with HAAR Features

The data when trained and tested using HAAR features as input, the Training accuracy we got was 81.71 percent and the Testing accuracy was 84.25 percent.

From Fig.7.10, we can observe that SNR 25 gave us the highest accuracy whereas SNR -10 gave us the lowest accuracy, this happens because in the SNR -10 range, the external noise for the signals is greater. From Fig.7.11 we can observe the confusion matrix for all the 18 signal types and we can conclude that the signal types ”rrty50_170” are the ones that most differ from others. Fig.7.12 and Table 7.2.2 give the complete details of the training and testing accuracy plots as well as the classification report.

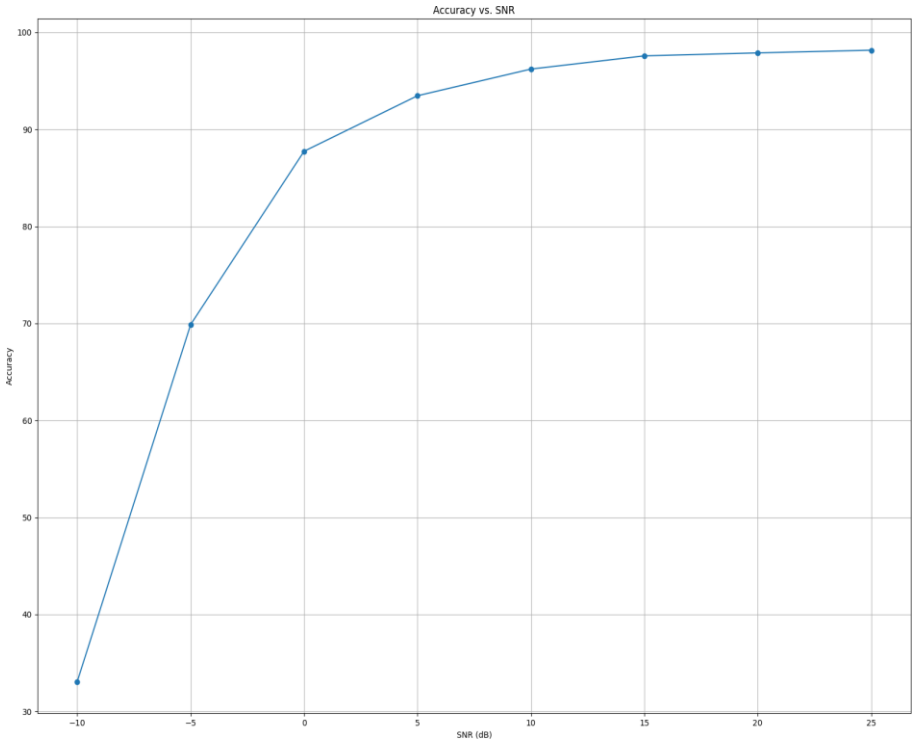

Figure 7.10: Plot between SNR values and their accuracies

7.3 Implementing GA-NAS

The main aim of implementing NAS on CNN models for IQ samples and HAAR features is to select a better architecture to increase the accuracy and reduce the model size. Here, we will

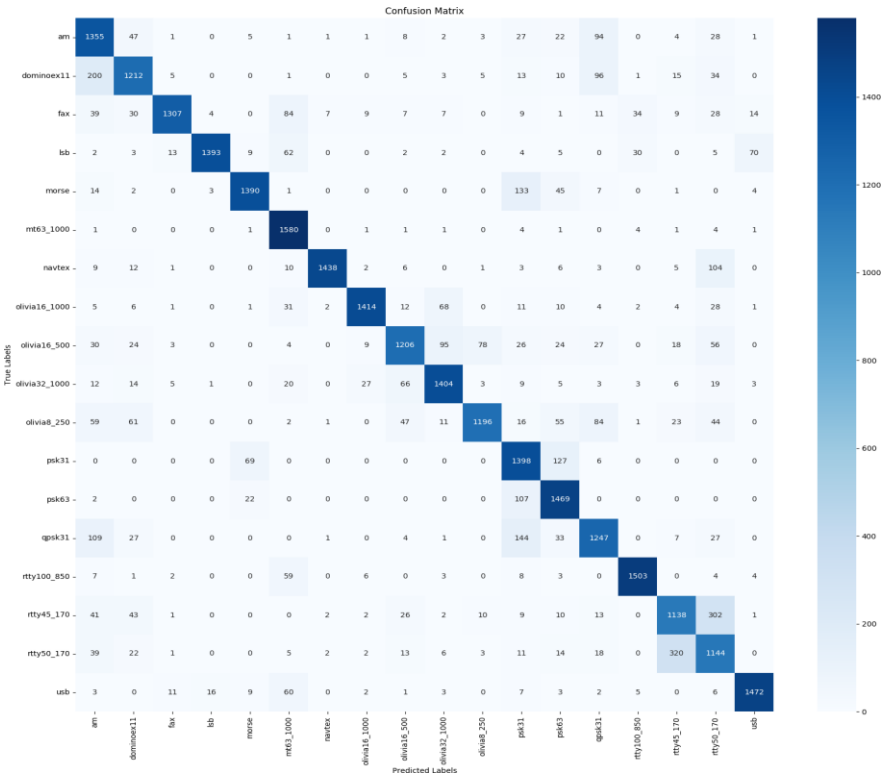

Figure 7.11: Confusin matrix for HAAR features using Resnet model

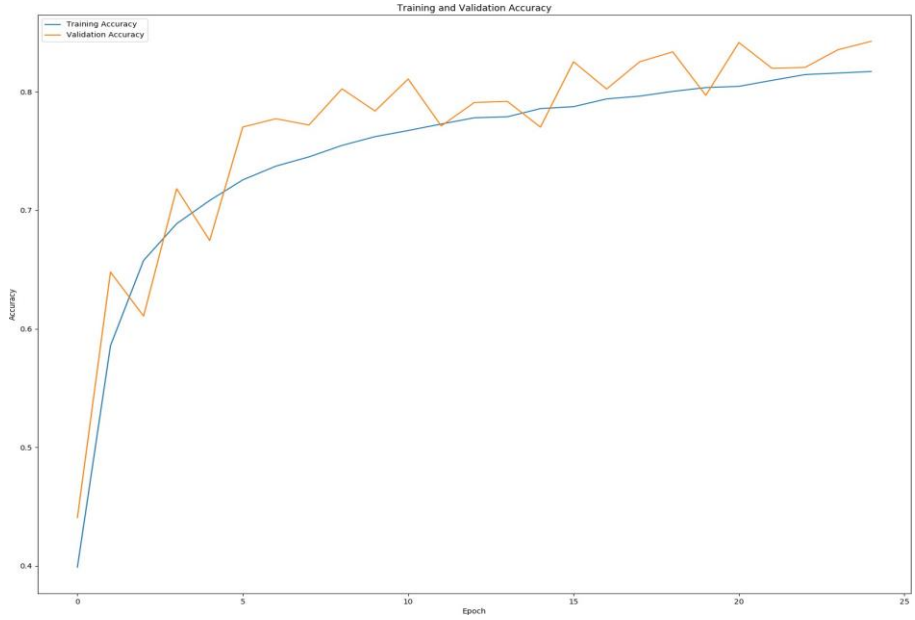

Figure 7.12: Training and Testing accuracies for HAAR features using Resnet model

Signals	Precision	Recall	f1-score	Support
am	0.70	0.85	0.77	1600
dominoex	0.81	0.76	0.78	1600
fax	0.97	0.82	0.89	1600
lsb	0.98	0.87	0.92	1600
morse	0.92	0.87	0.90	1600
mt63_1000	0.82	0.99	0.90	1600
navtex	0.99	0.90	0.94	1600
olivia16_1000	0.96	0.88	0.92	1600
olivia16_500	0.86	0.75	0.80	1600
olivia32_1000	0.87	0.88	0.88	1600
olivia8_250	0.92	0.75	0.83	1600
psk31	0.72	0.87	0.79	1600
psk63	0.80	0.92	0.85	1600
qpsk31	0.77	0.78	0.78	1600
rtty100_850	0.95	0.94	0.94	1600
rtty45_170	0.73	0.71	0.72	1600
rtty50_170	0.62	0.71	0.67	1600
usb	0.94	0.92	0.93	1600

Table 7.4: Classification table for HAAR features using Resnet model

be comparing the original architecture as well as the architecture we got by using NAS and finalizing which model gives us the best accuracy as well as the one with less size.

7.3.1 Training of IQ samples for CNN model using NAS

From 7.3.1, we can see the difference in the architecture of the original CNN and the architecture we got using NAS when using IQ samplesa as input. Even though the model's accuracy decreased, network size also drastically decreased when we used NAS architecture.

7.3.2 Training of HAAR features for CNN model using NAS

From 7.3.2, we can see the difference in the architecture of the original CNN and the architecture we got using NAS when using HAAR features as input. The model's accuracy increased and the network size also drastically decreased when we used NAS architecture.

Aspect	Original Architecture	GA-NAS architecture
Number of Layers	5	5
Neurons per Layer	32, 32, 64, 64, 128	256, 32, 64, 16, 256
Kernel Sizes	3, 3, 3, 3, 3	6, 4, 1, 5, 6
Activation Functions	ReLU, ReLU, ReLU, ReLU, ReLU	ReLU, ReLU, ReLU, Sigmoid, GELU
Dropout Rates	0.5	0.1
Overall Accuracy	92.64%	90.41%
Trainable Parameters	4,332,114	1,785,970
Network size	49.7 MB	20.6 MB

Table 7.5: Comparing Original architecture and NAS architecture of CNN model by using IQ Samples as input

Aspect	Original Architecture	GA-NAS architecture
Number of Layers	5	5
Neurons per Layer	32, 32, 64, 64, 128	128, 64, 256, 128, 64
Kernel Sizes	3, 3, 3, 3, 3	2, 6, 4, 5, 3
Activation Functions	ReLU, ReLU, ReLU, ReLU, ReLU	GELU, Tanh, ReLU, ReLU, Sigmoid
Dropout Rates	0.5	0.1
Overall Accuracy	93.77%	94.33%
Trainable Parameters	8,527,698	962,514
Network size	97.7 MB	11.1 MB

Table 7.6: Comparing Original architecture and NAS architecture of CNN model by using HAAR features as input

Chapter 8

HIGH PERFORMANCE COMPUTING (HPC)

8.1 High Performance Computing

HPC is linux based system and it refers to the use of supercomputers and parallel processing techniques to solve complex computational problems that require significant computational power and large-scale data analysis.[10] The following are some of the commands used for HPC environment:

- ls - List
- cd - change Directory
- nano - text editor
- cat - for displaying the file. (eg cat randc.py)
- qsub filename.sh - For Submitting the Job
- pbsnodes -aSj - For knowing the node status
- qstat -ans - For knowing the status of the job

8.2 Portable Batch System

PBS (Portable Batch System) scripts are used for job scheduling and management HPC clusters. These scripts allow users to submit and control batch jobs, which are typically long-running, resource-intensive computational tasks. PBS scripts often include environment setup, such as loading necessary modules or activating virtual environments.

RESULTS AND CONCLUSIONS

The deep Convolutional Neural Network (CNN) model, alongside the ResNet model, exhibits varying degrees of performance in classifying signals with different signal-to-noise ratios (SNR). The CNN model demonstrates robust performance for signals with high SNR, achieving high accuracies for SNR values of 0, 5, 10, 15, 20, and 25. However, its effectiveness diminishes notably for signals with low SNR, particularly at -10 and -5, where the classification results are poor. This disparity indicates that the model struggles when signals are heavily corrupted by noise, revealing a potential limitation in its ability to accurately classify such signals.

Similarly, the ResNet model showcases a similar trend in performance. It achieves strong accuracies for signals with high SNR but struggles with signals at low SNR levels. Despite achieving a higher testing accuracy than the CNN model for most SNR levels, the ResNet model still shows a significant drop in performance for low SNR values, indicating a similar sensitivity to noise in classification tasks.

The performance of both models is closely tied to the strength of the signal relative to the noise; as SNR increases, so does the accuracy of the models. This suggests that both models are more proficient at distinguishing signals from noise when the signal is stronger. The considerable performance gaps between high and low SNR signals suggest that both models may be sensitive to noise levels and may require enhancements to better handle noisy input data.

Applying the Genetic Neural Architecture Search (GNAS) algorithm to both the CNN and ResNet models could potentially address their sensitivity to noise and improve their performance, especially for signals with low SNR values. GNAS systematically explores different network architectures to identify those better suited for extracting features from noisy signals. By evolving and evaluating candidate architectures based on their performance, GNAS could lead to the discovery of structures that enhance the models' ability to distinguish signals from noise, potentially narrowing the performance gap between high and low SNR signals for both models.

Additionally, integrating Haar wavelet transformation into both models could further enhance their performance by extracting important features from the signals. Haar wavelet transformation is effective at capturing sudden changes or discontinuities in the signal, which are often indicative of important signal characteristics. By incorporating Haar wavelet transformation, both models can focus on the most relevant features, especially in scenarios with varying SNR levels, potentially improving their overall accuracy across a wider range of SNR values.

REFERENCES

[1] S. Scholl: Classification of Radio Signals and HF Transmission Modes with Deep Learning, 2019, https://arxiv.org/abs/1906.04459

[2] CMU ECE, ”STFT Notes ADSP,” Carnegie Mellon University, Available: https://course.ece.cmu.edu/ ece491/lectures/L25/STFT_Notes_ADSP.pdf

[3] Mathuranathan,”Spectrogram Analysis using Python”,2022.[Online].Available: https://www.gaussianwaves.com/2022/03/spectrogram-analysis-using-python/

[4] Demirkir and Sankur, ”Object Detection Using Haar Feature Selection Optimization,” 2006 IEEE 14th Signal Processing and Communications Applications, Antalya, Turkey, 2006, pp. 1-4, doi: 10.1109/SIU.2006.1659787.

[5] Xin Wang, ”Moving window-based double haar wavelet transform for image processing,” in IEEE Transactions on Image Processing, vol. 15, no. 9, pp. 2771-2779, Sept. 2006, doi: 10.1109/TIP.2006.877316.

[6] J. Zhu and Z. Chen, ”Real Time Face Detection System Using Adaboost and Haar-like Features,” 2015 2nd International Conference on Information Science and Control Engineering, Shanghai, China, 2015, pp. 404-407, doi: 10.1109/ICISCE.2015.95.

[7] Stefan Scholl. “Classification of Radio Signals and HF Transmission Modes with Deep Learning.” Jun, 2022.

[8] T. O’Shea and J. Hoydis, ”An Introduction to Deep Learning for the Physical Layer,” in IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563-575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[9] 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.

[10] X. Zhang, F. Reveriano, J. Lu, X. Fu and T. Zhang, ”The Effect of High Performance Computer on Deep Learning: A Face Expression Recognition Case,” 2019 IEEE International

Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), New York, NY, USA, 2019, pp. 40-42, doi: 10.1109/CSE/EUC.2019.00017.

[11] A. Ghosh and N. D. Jana, ”Neural Architecture Search with Improved Genetic Algorithm for Image Classification,” 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, 2020, pp. 344-349, doi: 10.1109/ComPE49325.2020.9200164.

[12] Y. He, B. Ma, Y. Shi and M. Han, ”Research on Modeling and Value Evaluation of Optical Transmission Link Based on Constellation Diagram,” 2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE), Xiamen, China, 2019, pp. 424-428, doi: 10.1109/EITCE47263.2019.9094833.

[13] L. Liu et al., ”Improving Deep Convolutional Neural Network Performance for Signal Classification under Low Signal-to-Noise Ratio Conditions,” in IEEE Transactions on Signal Processing, vol. 68, pp. 4188-4198, 2020.

[14] X. Wang, Y. Chen, and Z. Li, ”Feature Extraction and Signal Classification of High-Frequency Radio Signals Using Constellation Diagrams,” in IEEE Transactions on Vehicular Technology, vol. 65, no. 7, pp. 5432-5445, July 2016.

[15] J. Zhang et al., ”Genetic Neural Architecture Search for Noise-Robust Signal Classification,” in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, pp. 2913-2925, 2020.

[16] X. Chen et al., ”Haar Wavelet Transformation for Feature Extraction in Noisy Signal Classification,” in IEEE Transactions on Instrumentation and Measurement, vol. 69, pp. 5981-5990, 2020.