Impementation of mathematical morphology algorithms on FPGA : Signal processing applications.

Mohamed Yacine Bentegri
Mahfoud Toubal Seghir

A thesis presented for the degree of
Master of Telecommunication Engineering



Department of Science and Technology
The University of Dr. Yahia Fares
Algeria

Supervisor: Prof. Rebai Mohamed

Co-Supervisor: Berkani Mohamed Rafik Aymene

Dedication

I would be pleased to dedicate this little work to my beloved family, friends and my supervisor and co-supervisor who had the great share of helping me and guiding me through until the end.

Dedication

I dedicate this thesis...

To my loving mother, you have been the shining star in my life, my guide and an inexhaustible source of love and support. Thank you for each moment shared, your encouraging words and your irreplaceable role.

To My dear father, my rock and my model, I am grateful for your unconditional support and benevolence. You are an example of perseverance and dedication. You have been an exceptional father.

To my dear brother Kheiredinne, you are the pillars of my life and my constant source of happiness. Your unconditional support, love and presence were blessings that I cherish every day.

To my dear friends Zakaria Hantabli, Abdallah Babaali, Sidaali Ayache this memory is dedicated to you as a testament to our unwavering friendship. You were there every step of the way, supporting and inspiring me.

To my thesis colleague, Mohamed Yacine Bentegri, as a testament to our partnership.

This work is especially dedicated to Mr.Berkani Mohamed Rafik Aymane, whose active collaboration, brilliant ideas and perseverance have been of crucial importance for the realization of this project.

It is with deep gratitude to all those who were present at every stage of this project that I present this brief. Their support and collaboration were essential to its achievement.

Toubal Seghir Mahfoud

Acknowledgements

It is a pleasure for us to thank our Supervisor **Mr. RRBIAI Mohamed** and Co-Supervisor **Mr. BERKANI Mohamed Rafik Aymene** who, despite his heavy tasks, was willing to lead and correct

We were particularly sensitive to its constant presence throughout the year. He deserves our gratitude for the great patience he has shown, for the advice, technical and material help and encouragement he has given us.

Our heartfelt thanks to: **Mr. BENGHERBIA Billel** for his lessons which we have taken advantage of throughout the specialty curriculum and for having kindly agreed to honour the jury with his presence and to assume the chairmanship **Mr. KHEDDAR Hamza** for having kindly examined this work and agreed to participate in the jury.

We cannot forget to extend our sincere thanks and gratitude to all those who have contributed directly or indirectly to this work. Our thanks go, of course, to all the members of our two families in especially our dear parents, our brothers and sisters and our friends(es) for their moral support and for the pleasant atmosphere they have created around us.

Abstract

This master's thesis presents the implementation of mathematical morphology algorithms on Field-Programmable Gate Arrays (FPGAs) for signal processing applications. Mathematical morphology plays a crucial role in various image and signal processing tasks. By leveraging the parallelism, reconfigurability, and low power consumption capabilities of FPGAs, efficient hardware-based implementations of these algorithms are achieved, enabling real-time processing.

The thesis explores different fields and applications that demonstrate the capabilities and flexibility of the implemented techniques and algorithms in handling both images and signals. It addresses challenges associated with adapting mathematical morphology algorithms to FPGA architectures, considering trade-offs between accuracy, performance, and resource utilization. The research findings contribute to the development of optimized and hardware-accelerated solutions for signal processing applications that rely on mathematical morphology.

This work aims to provide insights into the fundamental concepts of mathematical morphology, pushing the boundaries of its application in conjunction with AI models and FPGA implementations. It explores the intersection between mathematical morphology and other emerging technologies.

Keywords: Mathematical Morphology, FPGA, Signal Processing, Image Processing, Deep Learning, Artificial Intelligence, CNN, Prediction, PCG, Accuracy, Loss.

Résumé

Ce mémoire de master présente la mise en œuvre d'algorithmes de morphologie mathématique sur des matrices de portes programmables sur terrain (FPGA) pour des applications de traitement du signal. La morphologie mathématique joue un rôle crucial dans diverses tâches de traitement d'images et de signaux. En exploitant le parallélisme, la reconfigurabilité et la faible consommation d'énergie des FPGA, des mises en œuvre matérielles efficaces de ces algorithmes sont réalisées, permettant un traitement en temps réel.

Le mémoire explore différents domaines et applications qui mettent en évidence les capacités et la flexibilité des techniques et algorithmes implémentés dans le traitement des images et des signaux. Il aborde les défis liés à l'adaptation des algorithmes de morphologie mathématique aux architectures FPGA, en tenant compte des compromis entre précision, performances et utilisation des ressources. Les résultats de la recherche contribuent au développement de solutions optimisées et accélérées par matériel pour les applications de traitement du signal qui reposent sur la morphologie mathématique.

Ce travail vise à fournir des aperçus des concepts fondamentaux de la morphologie mathématique, repoussant les limites de son application en combinaison avec des modèles d'intelligence artificielle et des implémentations sur FPGA. Il explore l'intersection entre la morphologie mathématique et d'autres technologies émergentes.

Mots-clés : Morphologie Mathématique, FPGA, Traitement du Signal, Traitement d'Images, Apprentissage Profnd, Intelligence Artificielle, CNN, Prédiction, PCG, Précision, Pertes.

Contents

1 Generalities **4**
- 1.1 Introduction . . . 5
- 1.2 FPGA Introduction and summary . . . 6
  - 1.2.1 Why FPGA . . . 6
- 1.3 Morphological view on traditional systems . . . 7
- 1.4 Mathematical aspects of the traditional image and signal processing systems 8
- 1.5 Basic Techniques of Mathematical Morphology . . . 8
  - 1.5.1 Structuring Element . . . 8
  - 1.5.2 Binary Dilation and Erosion . . . 10
  - 1.5.3 Closing and Opening . . . 11
- 1.6 Conclusion . . . 11

2 Artificial Intelligence **11**
- 2.1 Introduction . . . 12
- 2.2 Artificial Intelligence . . . 12
- 2.3 Machine learning . . . 12
  - 2.3.1 Supervised Learning . . . 13
  - 2.3.2 Unsupervised Learning . . . 13
  - 2.3.3 Reinforcement Learning . . . 14
- 2.4 Deep learning . . . 14
  - 2.4.1 Natural language processing (NLP) . . . 15
  - 2.4.2 Computer vision . . . 16
  - 2.4.3 Robotics . . . 17
  - 2.4.4 Neural networks . . . 17
  - 2.4.5 Biological Neuron Models . . . 17
  - 2.4.6 Artificial Neural Networks (ANNs) . . . 17
- 2.5 Architecture of Artificial Neural . . . 18
  - 2.5.1 Multilayer Perceptron Architecture . . . 18
  - 2.5.2 Activation Functions . . . 19
- 2.6 Types of Neural Networks . . . 21
  - 2.6.1 Artificial Neural Networks (ANNs) . . . 21
  - 2.6.2 Recurrent Neural Network (RNN) . . . 22
  - 2.6.3 Long Short-Term Memory (LSTM) . . . 22
  - 2.6.4 Convolutional Neural Networks (CNNs) . . . 22
  - 2.6.5 Architecture of CNN . . . 23
- 2.7 Conclusion . . . 25

3 Application of Mathematical morphology **25**
3.1 Introduction . . . 26
3.2 Xilinx System Generator . . . 26
3.3 Proposed design for image processing . . . 27
3.3.1 Image pre-processing unit . . . 27
3.3.2 Image Post-processing unit . . . 28
3.3.3 Models for image processing algorithms . . . 28
3.3.4 Mathematical Morphology on Xilinx system generator . . . 30
3.3.5 Grayscale Application . . . 31
3.3.6 Conclusion . . . 33

4 Testing and implementation **34**
4.1 Introduction . . . 35
4.2 Database . . . 35
4.2.1 Database Description . . . 35
4.2.2 Database Distribution . . . 35
4.3 Material . . . 36
4.3.1 Software . . . 36
4.4 Libraries . . . 36
4.4.1 Tensorflow . . . 36
4.4.2 Keras . . . 36
4.4.3 Sklearn . . . 36
4.4.4 Matplotlib . . . 37
4.5 System Methodology . . . 37
4.5.1 Preprocessing . . . 37
4.5.2 Decomposition of the Data . . . 38
4.6 CNN Model . . . 39
4.6.1 Proposed Architecture . . . 39
4.6.2 Compiler . . . 41
4.6.3 Methods Used . . . 41
4.7 Result Evaluation . . . 41
4.7.1 Accuracy and Loss Graph . . . 42
4.7.2 Classification Report . . . 42
4.7.3 Confusion Matrix . . . 43
4.7.4 Best Model Result . . . 43
4.8 Final Test . . . 44
4.9 Conclusion . . . 44

List of Figures

1.1 FPGA card example . . . 6
1.2 Basic structuring elements examples . . . 9
1.3 Flat structuring element . . . 9
1.4 Non flat structuring element . . . 9
1.5 Example of binary dilation . . . 10
1.6 Example of binary erosion . . . 10
1.7 A , B , A⊕B , A⊖B . . . 10

2.1 Supervised Learning . . . 13
2.2 Unsupervised Learning . . . 14
2.3 Reinforcement Learning . . . 14
2.4 Structure of a Neuron . . . 17
2.5 Perceptron . . . 18
2.6 Multilayer-percepton model . . . 19
2.7 ReLU Graph . . . 20
2.8 Sigmoid Graph . . . 20
2.9 softmax Graph . . . 20
2.10 Tanh Graph . . . 21
2.11 ELU Graph . . . 21
2.12 Convolution layer . . . 23
2.13 a neural network before and after dropout . . . 24
2.14 max pooling . . . 24
2.15 average pooling . . . 25

3.1 Xilinx System Generator basic Model . . . 27
3.2 Proposed design for the hardware implementation . . . 27
3.3 Pre-processing unit in simulink . . . 28
3.4 Post-processing unit in simulink . . . 28
3.5 GreyScale image Inverter . . . 29
3.6 Input image . . . 29
3.7 Output Image . . . 29
3.8 Original input image . . . 30
3.9 Structuring element . . . 30
3.10 Dilation on XSG . . . 30
3.11 Binary dilation output . . . 31
3.12 Binary erosion output . . . 31
3.13 Dilation input image . . . 31

3.14 Dilation output Image 31
3.15 Erosion input image 32
3.16 Erosion output Image 32
3.17 Opening output from the dilated image 32
3.18 Opening output from the eroded image 32
3.19 Closing output from the dilated image 33
3.20 Closing output from the eroded image 33

4.1 Database distribution 35
4.2 Model training diagram 37
4.3 Original Signal 37
4.4 Segment 37
4.5 Resulting enveloppe 38
4.6 First method's pre-processing results 38
4.7 Second method's pre-processing results 38
4.8 Data decomposition 39
4.9 CNN Model 39
4.10 Proposed model 40
4.11 accuracy and loss graph 42
4.12 Classification report 43
4.13 Confusion matrix 43
4.14 Confusion matrix of the final test 44

List of Tables

1.1 Comparison of FPGAs to other similar options 7

4.1 Data classes 35
4.2 Best Result 44
4.3 Test Result 44

Introduction

The applications of signal and image processing have been an important task for researchers and studies in the telecommunications field and technology in general. Many techniques and tools have been combined to optimize the results.

This master's thesis explores a new approach to using mathematical morphology in a new environment. This could be the answer to the question of what steps should be taken in the near future in this field. FPGA (Field-Programmable Gate Array) is a tool that has proven its ability to perform and play an important role in the technology field.

We developed a couple of methods to test how the preprocessing of images and signals using mathematical morphology affects our final results. This includes using them to preprocess an image through an FPGA (Field-Programmable Gate Array) card and observing how effective it is, or taking them to process the signal and prepare them for an artificial intelligence that would classify the heart diseases presented to them.

In the end, the methods proved to be quite useful and effective. The accuracy of our convolutional neural networks was never below 95%, and the simulated FPGA cards proved to be a reliable tool that could be explored further in the future using such algorithms.

Chapter 1

Generalities

1.1 Introduction

Image processing is a field that involves with the manipulation, analysis, and interpretation of digital images. It can be used in many applications like healthcare, surveillance, photography, and scientific research. The massive increase of visual data in our daily lives motivated many researchers and scientists to develop and improve many techniques to obtain as much data from digital images.

Image processing has seen a huge advancement over the years, driven by the need to interpret and manipulate visual information with precise accuracy. One of the most popular techniques that proved itself to be most useful in the field of image processing is mathematical morphology.

Mathematical morphology is a theory and set of techniques that deals with the manipulation of images and shapes, it was developed by the French mathematician Georges Matheron in the 60s and then embraced and further improved by Jean Serra. Mathematical morphology mostly deals with shapes and objects within images, its main focus is to interact with the proprieties of structures (image enhancement, object recognition, noise reduction...).

Due to the robust techniques of mathematical morphology in the field of digital images, we propose a project that adopts mathematical morphology techniques and implements them within a system.

In the first chapter, we give brief introduction about the FPGA card, and the morphological view and aspects of mathematical morphology within image processing systems. Then we go through a brief introduction of mathematical morphology, and its core concepts and techniques.

Chapter 2 summarizes the AI aspect of our projects, beginning with what is AI, the different AI techniques, machine learning and its many subsets, deep learning, activation functions, and neural networks.

The third and the last chapter are about the implementation of mathematical morphology techniques within a model and the results generated by the system.

1.2 FPGA Introduction and summary

FPGA or Field programmable gate arrays is a type of integrated circuits that can be programmed by the user after manufacturing, it consists of an array of configurable logic blocks, which perform specific functions and implement complex digital circuits, the logic blocks contain an number of basic logic gates such as AND, OR and NOT gates, as well as flip flops, and other circuit elements.

FPGA cards offer a number of advantages, such as flexibility since they can be reprogrammed as many times needed for the optimal outcome, they can also be used to implement a wide range starting from making basic circuits to process complex signal algorithms.

While FPGAs are used in various fields they are very common in telecommunications due to their accuracy, fields of application and speed, taking the latter in consideration we thought of FPGA cards as a very suitable fields of application of our existing application, and aiming for any possible optimisation or improvement



Figure 1.1: FPGA card example

1.2.1 Why FPGA

- FPGAs are reprogrammable, which means that they can be reconfigured to perform different functions. This makes them ideal for applications where the requirements change frequently, such as in the telecommunications industry.
- FPGAs are fast and efficient. They can be used to implement complex algorithms and functions with high performance.
- FPGAs are versatile. They can be used in a wide variety of applications, from simple logic gates to complex signal processing algorithms.

Table 1.1: Comparison of FPGAs to other similar options

Option	Reprogrammability	Performance	Cost
FPGA	Reconfigurable	High	Medium
ASIC	Customized	Highest	Highest
Microcontroller	Fixed	Low	Low
PLD	Semi-custom	Medium	Low

In summary, an FPGA is a type of integrated circuit that can be programmed or configured by the user after manufacturing. It consists of an array of configurable logic blocks and interconnects that can be used to implement a wide variety of digital circuits. FPGAs are flexible, fast, and widely used in a variety of applications.

1.3 Morphological view on traditional systems

The study of the traditional image and signal processing systems is mainly focused to analyze, filter, sort and even manipulate the given arrays or matrices of numerical values. Those pieces of information are the constitution of the 2D signals such as images , or even the 3D signals that may contain videos or sounds.

One of the main aspects in our study is the Morphological aspect of the analysis, Which can be described as the study of the shapes, structures that defines an object while going through the image and signal data it proves to be a critical point to be pointed out, since a large piece of information can be derived from the visual constitution of the signal itself.

The morphological study relies on two main operations: dilation and erosion whereas Dilation is the application of expanding an object using a predefined Structuring element in the current signal or image , While the erosion being the one used to shrink or to delete parts of the studied target, as simple as those two operation may seem they can be generalized and flexible to use on a scale of few pixels or even to a whole selected signal.

Furthermore, feature extraction in image and signal processing is a widely important aspect, this refers to identifying and analyzing a pre-specified patterns and structures that may hold a great deal of influence on the overall data included in our images or signals, Some features that are extracted in most cases, are edges, corners and blobs.

We can also find other sets of techniques that might be applied to traditional structures such as filtering, segmentation and compression.

They aim to remove noise, separate structures in a specified space or reduce the amount of data itself. Overall, a large sum of techniques and operations are available to increase the flexibility and the fields of the practical application of the Mathematical morphology studies, these systems could provide meaningful pieces of information and make use of the complex and noisy data which is frequently presented for studies and analysis.

1.4 Mathematical aspects of the traditional image and signal processing systems

Image and signal processing systems rely on the analysis and manipulation of digital signals (including but not limited to images or sounds). The mathematical aspect is the foundation for the developpment of powerful algorithms and techniques for this study.

The Fourier analysis is by far the most practical procedure for frequency extraction in images, it does that by dividing a signal into a component of frequencies, which helps in a wide range of implementation in image processing/filtering, audio processing, and data analysis.

Linear algebra is also a determining approach in image and signal processing as it mandantary for the evaluation of matrices and vectors, as well as image filtering, segmentation, and registration.

Probablity has its importance, as it reduces noise and other obstructions within a signal with various procedures.

digital filters alter the frequency and amplitude of a targeted signal. Convolution and Fourier transforms are the basis of signal processing, making it a must in this research.

1.5 Basic Techniques of Mathematical Morphology

The main goal of Mathematical Morphology is the analysis of shapes and structures within images. When performing basic morphology, two types of images are needed: the image that we want to apply MM techniques, referred to as ”active image”, and ”the structuring element”, also called ”kernel”.

The structuring element has a specific shape, from the point of view of the active image, the structuring element acts as a filter or probe and shapes the objects within the image in a desired way. The structuring element is the core of MM, as most of the techniques that we will mention rely on it.

1.5.1 Structuring Element

A structuring element is a small matrix or kernel used in mathematical morphology operations, which are image processing techniques that analyze and manipulate the shape and structure of objects in an image. The structuring element is typically a binary matrix with a defined shape, such as a square or circle, and a specified size. The structuring element is applied to an input image to perform operations such as erosion, dilation, opening, and closing. The structuring element defines the shape and size of the neighborhood around each pixel in the input image that is used to calculate the output of the operation. By varying the shape and size of the structuring element, different types of morphological operations can be performed to extract or enhance specific features in an image.

There are two main types of structuring elements, the first one is a **flat structuring element**, defined as a binary valued neighborhood. it can be in 2D or multidimensional. true pixels are included in the morphological computation, while ones are not. The center of the

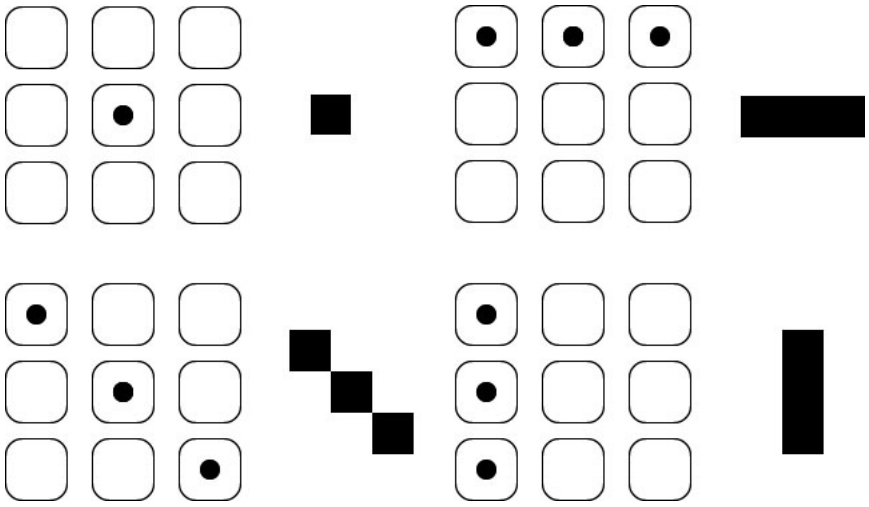


Figure 1.2: Basic structuring elements examples

structuring element is referred to as 'The Origin', it identifies the pixel in the image being processed, A flat structuring element can be used in both gray-scale and binary images.

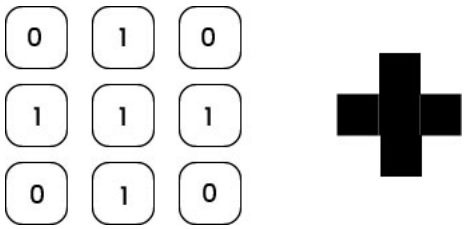


Figure 1.3: Flat structuring element

The second type of structuring element is a **non flat structuring element** , just like a flat structuring element, it a matrix, but in a double type. It identifies the pixel in the image being processed and defines the neighborhood used in the processing of that pixel. This type of structuring element contains finite values that are used in finite additive offsets in the morphological computation. Unlike flat structuring elements, a nonflat one is only used on grayscale images [1].

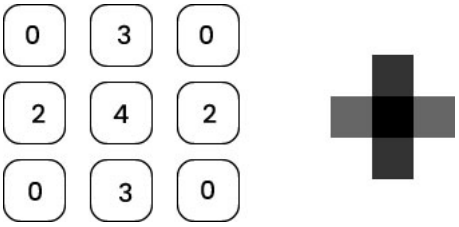


Figure 1.4: Non flat structuring element

1.5.2 Binary Dilation and Erosion

Definitions of dilation and erosion developed multiple times, We will define binary dilation as a combination of two sets using vector addition of set elements called Minkowski addition. The binary dilation of A by B, denoted A $\oplus_b$ B is defined as : convolving a binary image with a structuring element, typically a square or a circular matrix that defines the current dilation .

On top of that the structuring element is placed over each pixel in the input image, and if any part of the structuring element overlaps with a foreground pixel (a pixel with value 1), the corresponding pixel in the output image is set to 1. This operation results in the expansion of the boundaries of objects in the binary image. **Example:**
A = (0, 1), (1, 1), (2, 1), (3, 1)
B = (0, 0), (0, 2)
A $\oplus$ B = (0, 1), (1, 1), (2, 1), (3, 1), (0, 3), (1, 3), (2, 3), (3, 3).

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 1.5: Example of binary dilation

Binary erosion: It can be defined as the dual of dilation or the opposite of the erosion, as It is the subtraction of two vector set of elementsm it is denoted by A $\ominus_b$ B.

Example:
AA =(0, 1), (1, 1), (2, 1), (2, 2), (3, 0)
B = (0, 0), (0, 1)
A $\ominus_b$ B = 1(2, 1)

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1.6: Example of binary erosion

Noting that the dilation and the erosion are dual, that doesn't mean they cancel each other, we can have another form of the examples we delivered above:

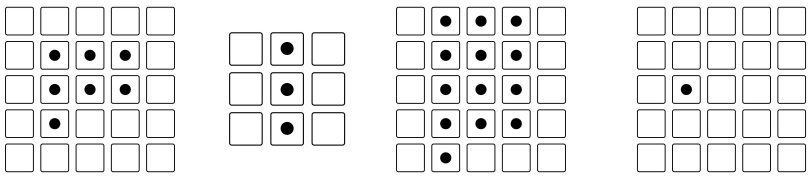


Figure 1.7: A , B , A$\oplus$B , A$\ominus$B

1.5.3 Closing and Opening

Opening and closing operations are the combination of dilation and erosion techniques in different order.

The opening operation of A by B is obtained by the erosion of A by B, followed by a dilation of A by B:

$$A \circ B = (A \ominus B) \oplus B$$

The closing operation of A by B is obtained by the dilation of A by B, followed by an erosion of A by B:

$$A \bullet B = (A \oplus B) \ominus B$$

In the opening operation, the erosion step reduces the size of large objects while the dilation restores the shape and size of the remaining objects.

The closing operation is the opposite of dilation, the dilation step precedes the erosion. The dilation expands the boundaries of the objects and fills the empty gaps within while the erosion restores the original size of the objects.

both erosion and dilation techniques are used in noise removal, segmentation, feature extraction, and shape analysis. Opening is used for to remove small objects, meanwhile closing fill in gaps and connect nearby objects[2].

1.6 Conclusion

Mathematical morphology has been proven as a robust framework for image processing, and is a popular solution for a variety of tasks like computer vision and medical imaging by providing proper techniques to manipulate and extract data from images according to their structural properties. As technology continues to evolve, mathematical morphology remains the better choice for researchers when it comes to image manipulation, and Combining it with the power of FPGA, we get a much faster and precise solution for image manipulation.

Chapter 2

Artificial Intelligence

2.1 Introduction

AI is a game changer in the technology industry, it increases efficiency and accuracy at a high-speed the human behavior cannot produce. The best part of AI is that it can be implemented in most disciplines, making life easier and saving time with little need for human interaction, it makes the monotonous tasks automatic.

Through the years, we have seen many techniques developed to make AI easier to implement and harvest its benefits such as machine learning and deep learning. These techniques add a sense of independence in AI through a self taught process by feeding it data in order for it to learn how to recognize patterns, in a way that is similar to the human brain thanks to artificial neural networks.

This chapter will go through the basic aspects of AI and its proprieties that makes it such an amazing tool to use.

2.2 Artificial Intelligence

AI, or artificial intelligence, refers to the development of computer systems or machines that can perform tasks that typically require human intelligence. The main focus of AI is is to imitate the human intelligence, allowing them to adapt their thinking accordingly in order to solve a problem. There are many distinguished AI techniques such as:

- Machine learning
- Deep learning
- Natural language processing
- Computer vision
- Robotics

These techniques' goal is to enable machines to analyze abundant amount of data, recognize patterns, and make predictions all while improving their performance over time.

AI can be implemented in all different kind of fields (healthcare, finance, networking...etc). although, with the rapid advancement of AI also raises concerns over privacy and its impact on the human decision making.

2.3 Machine learning

is a subset of artificial intelligence that focuses on the development of algorithms and models that enable computers or machines to learn and make predictions or decisions without being explicitly programmed for every specific task. in other words, the machine learns by itself without any outsiders involvement, all of this while further improving its efficiency and accuracy.

The process of machine learning can be divided into many different key elements: data: machine learning depends on a large amount of input information in order for it to learn

by itself. This data can be either structured (such as a spreadsheet) or unstructured (text, images).

- **Training**: this is where the machine starts to look for patterns and optimizes its performance from the data based on a well designed algorithm.
- **Features**: specific attributes, or features extracted from the data that are crucial for the learning process. They further help the algorithm in understanding and distinguishing the patterns in the data.
- **Algorithms**: many algorithms exist to a specific task, some are much more suited for certain problem. common known algorithms include linear regression, decision trees, support vector machines, naive bayes, and neural networks.
- **Model evaluation**: the algorithm is evaluated after the training to assess its performance, we use a separate dataset called the 'test set'. This step is essential for further optimization of the AI.
- **Predictions or decisions**: once the algorithm has been trained evaluated, the AI gains the ability to make decisions on its own and generate the desired output.

There three subset of machine learning:

- Supervised learning.
- Unsupervised learning.
- Reinforcement Learning.

2.3.1 Supervised Learning

The algorithm learns from labeled data, where both the input and the desired output are known. It learns to map the input to the output by inferring the underlying patterns. [3]

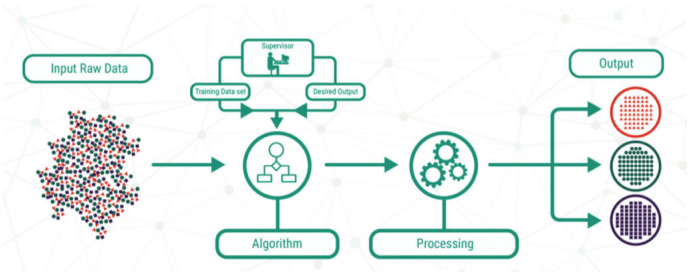


Figure 2.1: Supervised Learning

2.3.2 Unsupervised Learning

In this case, the algorithm learns from unlabeled data, where only the input data is available. It identifies patterns, clusters, or structures in the data without explicit guidance on the desired output.

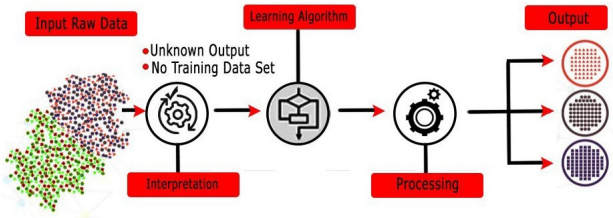


Figure 2.2: Unsupervised Learning

2.3.3 Reinforcement Learning

This approach involves an agent that learns to interact with an environment. The agent receives feedback in the form of rewards or punishments based on its actions, allowing it to learn through trial and error and optimize its decision-making.

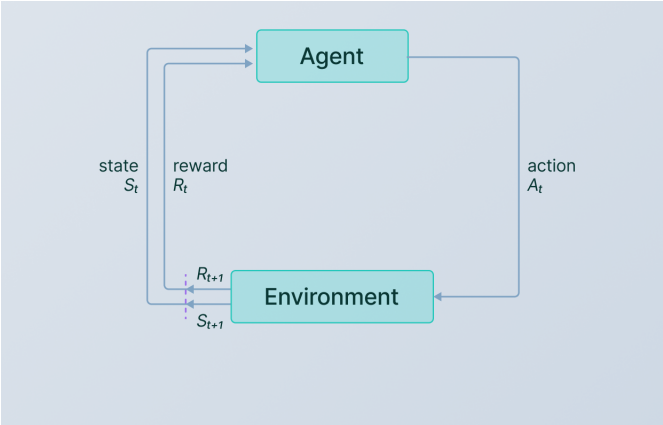


Figure 2.3: Reinforcement Learning

Machine learning is well suited for image and speech recognition, natural language processing, fraud detection, autonomous vehicles (self-driving), and more. it is really useful when there is a huge amount of data to process.

2.4 Deep learning

Deep Learning is a sub-field of machine learning that focuses on the development and application of artificial neural networks inspired by the structure and function of the human brain. The algorithms used in deep learning, also known as 'deep neural networks', can learn automatically through hierarchical representations of data past multiple layers of interconnected nodes, or artificial neurons.

tThe 'deep' part of deep learning refers to the depths of these neural networks, which can consist of multiple hidden layers. Each layer in the network input data and forwards it to the next one. each layer extracts a moderate amount of high-level features and representations.

There are many key components to deep learning:

- **Neural Networks**: The basis of deep learning, it consists of interconnected nodes (called neurons). These neurons are arranged in multiple layers, data flows through the network from the input layer to the output layer.
- **Deep Neural Network Architectures**: Feedforward neural networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversial networks (GANs) are exemples of deep neural network architectures. Each type of architecture handles different type of data and learning tasks.
- **Representation Learning**: Deep learning is quite proficient at dealing with raw data, by automatically extracting features via deep learning algorithms at distinct layers of abstractions.
- **Backpropagation**: Is an algorithm used by deep learning networks for the training process. The first step of the learning process involves feeding data toward the network while calculating the error between the predicted output and the desired output, then forwards the network in order for the network of neurons to update their behavior.
- **Big data and computing power**: Large amount of data brings are essential for training the deep learning algorithms, along with a considerable amount of computational power from GPUs (Graphics Processing Units) or a specific type of hardware to speed up the computation.

Deep learning has achieved remarkable success in various domains, including computer vision (object detection, image classification), natural language processing (machine translation, sentiment analysis, chatbots), speech recognition, recommendation systems, and autonomous driving. It has revolutionized fields like image recognition with the ability to achieve human-level or even superhuman-level performance in certain tasks.

Deep learning revolutionized various fields including computer vision (object detection, image classification), natural language processing (machine translation, sentiment analysis, chatbots), speech recognition, self-driving cars. Image recognition also gained huge improvements that often exceeds human performance.

The downsides of deep learning is that it needs a large-scale labeled data in order to maximize its performance, and the lack of interoperability in complex models. still, deep learning continues to thrive in many industries, and helps advance AI with complex problem solving.

2.4.1 Natural language processing (NLP)

is a sub-field of artificial intelligence (AI) that focuses on the interaction between computers and human language. its main focus is to design algorithms and models to understand, interpret, and generate human language in a comprehensible and valuable manner.There are many techniques involving NLP:

- **text understanding**: involves the process of understanding, translating the intent of the human language. such as text classification, sentiment analysis, named entity recognition, and information extraction.
- **Machine translation**: can be used to translate text from one language to another. it

includes inspecting a text thoroughly grasping its meaning then generating an equivalent sentence in another language.

- **Speech recognition**: turns spoken language into a text based format. This can manifest as voice assistant applications, transcription services...etc.
- **Language generation**: NLP also has the ability to generate coherent information in a text based format, including conversation and dialogues. many chatbots applications are a prime example of this technique.
- **question answering**: NLP has the ability to comprehend a question and respond with an appropriate response.
- **summarization**: goes through long documents or articles and condense a brief summary based on the original information.
- **text-to-speech (TTS)**: converts written text into spoken words. Giving off a human like speech behavior.

2.4.2 Computer vision

is a field of artificial intelligence that focuses on enabling computers to understand, interpret, and analyze visual information from digital images or videos (chatgpt). it includes algorithms and techniques that give machines the ability to recognize and extract useful information from visual data. quite helpful with object recognition, image segmentation, and motion detection.

Computer vision can be associated with many tasks similar to:

- **Image processing**: There are various computer vision techniques that deals with digital images such as filtering, image resizing, color correction, and noise reduction.
- **Feature extraction**: identifying valuable inputs from images or videos can be done with various computer vision algorithms. such information can include shapes, textures, corners, or other characteristics that help with object classification.
- **Object recognition**: being able to identify a certain type of entity within an image is a task with a significant importance, by identifying and classifying specific figures in an image. This process can be perceived in facial recognition, object tracking, and object-detection in real-time.
- **Image segmentation**: consists of dividing an image into meaningful regions based on a certain criteria. separating the background of an image for an example. especially useful when performing medical image analysis.
- **Motion analysis**: is the science of comparing sequential still images captured from photographing a body in motion in order to study the kinematics (The motions themselves) and the kinetics (the external and internal forces) involved. it helps with the development of efficient training programs for athletes, assess physical educational programs for cerebral palsy patients [4].

2.4.3 Robotics

is the art, knowledge base, and know-how of designing, applying, and using robots in human endeavors. These robotics system include robots, and other devices and systems that work in conjunction with each other. heavily used within space exploration, military purposes and more. [5]

2.4.4 Neural networks

can be described as a combination of robust techniques that excel with pattern recognition, data analysis, and control. One of the main advantages of neural networks is their ability to process and learn how to solve a problem from a set of examples in a time record [6].

2.4.5 Biological Neuron Models

Referred to as 'spiking neuron models' as well. are mathematical descriptions of the properties of certain cells in the nervous system that generate sharp electrical potentials across their cell membrane, roughly one millisecond in duration, called action potentials or spikes. Since spikes are transmitted along the axon and synapses from the sending neuron to many other neurons. every neuron is a unit, and each unit is considered as a major information processing component within the nervous system [3].

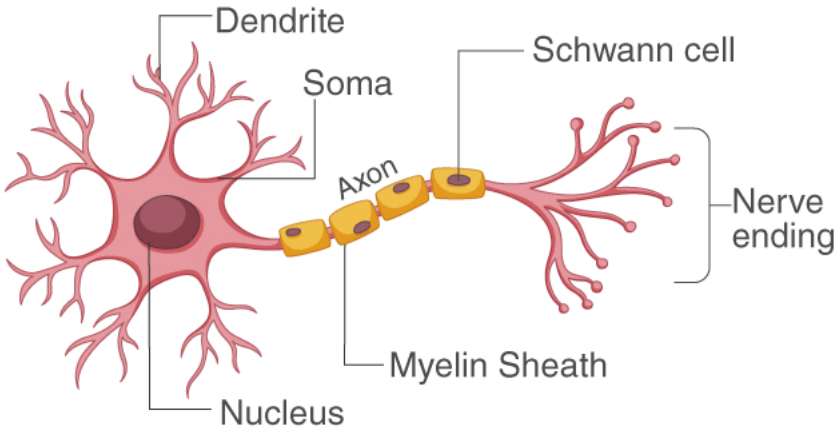


Figure 2.4: Structure of a Neuron

2.4.6 Artificial Neural Networks (ANNs)

Based on biological neural networks, ANNs are massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. Each node in the network is considered as an 'artificial neuron'. similarly to a biological neuron, it behaves in a similar manner as a biological neuron as a processing unit [7].

2.5 Architecture of Artificial Neural

There are two main categories of network

Feed-forward networks

In this case graph have no loops.

Recurrent (feedback) networks

Loops are existent because of feedback connection. produce a single result output instead of a series of values, hence why they are called 'static'.

The perceptron

introduced by the psychologist Frank Rosenblatt in 1957 as a class of artificial nerve nets, embodying aspects of the brain and receptors of biological systems. [8]

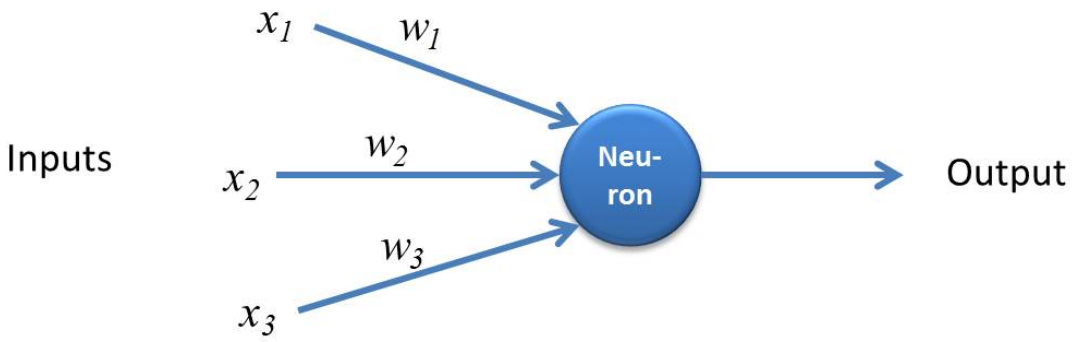


Figure 2.5: Perceptron

2.5.1 Multilayer Perceptron Architecture

Represented as an fully connected, three-layer, feedforward, perceptron neural network. The term 'Fully connected' refers to the way the output from each input and hidden neuron is propagated to all the neurons in the neighboring layer. in Each network, there's a single input and output layer, there's a variation in the number of hidden layer from network to network.

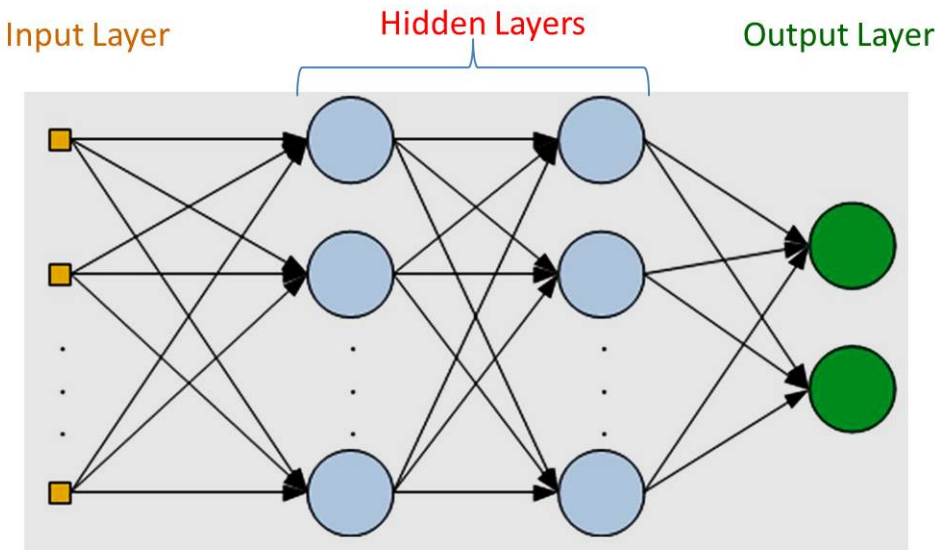


Figure 2.6: Multilayer-percepton model

Neural networks are one of the most popular Classifiers, not only for their valuable performance but for their ability to adapt their architecture to further optimize the output result by constant trial and error with activation functions, normalization schemes and other factors. Neural networks try to transform the non-linearly separable input data into more linearly separable abstract features thanks to a hierarchical structure of layers. The layers that constitute these networks are a mixture of linear and nonlinear functions, such as 'activation functions (AFs)'. Here is a list of the most common activation functions [9]:

2.5.2 Activation Functions

Activation Functions are specially used in artificial neural networks to transform an input signal into an output signal. The sum of products of inputs and their respective weights are calculated within a neural network, then we can implement a desired activation function to generate the output of that layer and forwards it as an input for the next layer [10].

The accuracy of a neural network model is proportional to how deep the network is, meaning the number of layers. Still using the right activation function for the model is as important to get concrete results. There is no perfect activation function that fits all, each one has its own strengths and weaknesses, that's way it is essential to experiment with them to get the best results.

1. **Rectified Linear Unit(ReLU)**: Rectified Linear Unit (ReLU) is a simple function which is the identity function for positive input and zero for negative input and given as:

   $$f(x) = \max(0, x)$$

   The range of ReLU is between zero and infinity. less complex from a computational point of view compared to other functions (Sigmond and Tanh). The only downside with ReLU is 'Dying ReLus', in which some neurons die out, which in their turn output 0s in training [11].

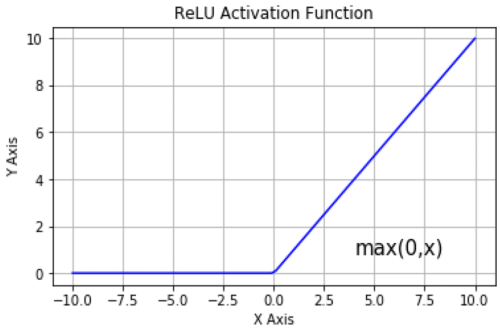


Figure 2.7: ReLU Graph

2. **Sigmoid**: without a doubt the widely used activation function due to its non-linearity nature. Sigmoid change the values in the range 0 to 1. it is identified as with the following equation:

$$f(x) = \frac{1}{\exp(-x)} \tag{2.1}$$

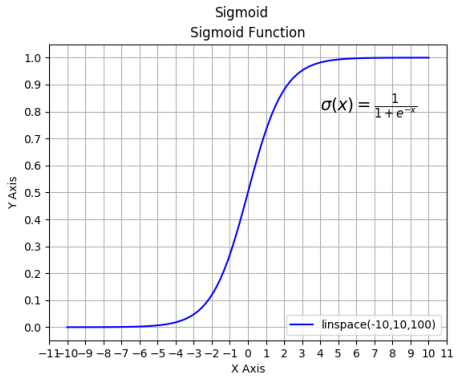


Figure 2.8: Sigmoid Graph

The sigmoid is a non-symetric about 0, meaning the signs of the outputs of the neurons will be the same, this problem can be fixed by scaling the sigmoid function.

3. **Softmax**: Is a combinaison of multiple sigmoid functions. As discussed before, Sigmoid function output values in the range 0 to 1, these values can be interpreted as probabilities of a certain class' data points.

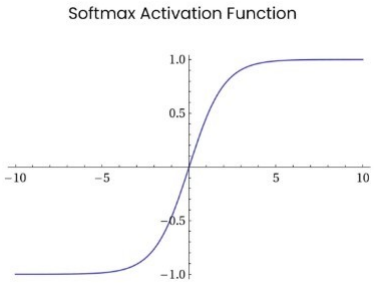


Figure 2.9: softmax Graph

sigmoid function is only used for binary classification while softmax can be used in multiclass classification. Each data point of every class returns a probability, which can be expressed as follows:

After building a network or model for multiple class classification, the output of the network will be the same number of neurons as the number of classes in the target.

4. **Tanh**: is a hyperbolic tangent function. pretty similar to sigmoid function but is symmetric to the origin. hence, it results in different signs of outputs from prior layers and will be passed toward as input to the next layer.

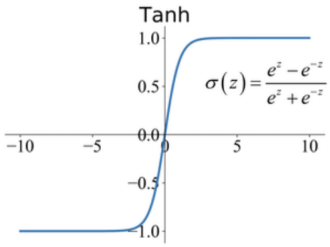


Figure 2.10: Tanh Graph

Notice that this function is continuous and differentiable, also the values range from -1 to 1. The gradient is much more steeper compared to the sigmoid function. due to its gradient being not restricted in certain direction and it is zero centered, the tanh function is much more favorable than the sigmoid function.

5. **Exponential Linear Unit (ELU)**: derived from the rectified linear Unit (ReLU) function. it uses a parameter slope for negative values and uses a log curve for defining negative values [10].

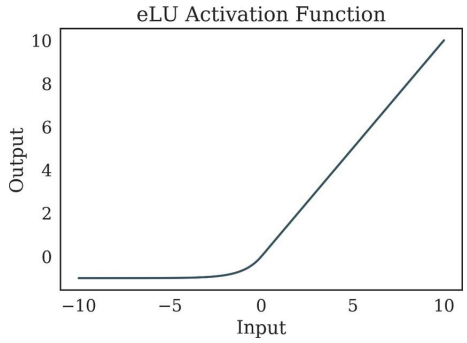


Figure 2.11: ELU Graph

2.6 Types of Neural Networks

2.6.1 Artificial Neural Networks (ANNs)

are computational processing systems which are heavily inspired biological nervous system (such as the human brain) operate. ANNs are consisted of a large number of interconnected

processing units called neurons. These neurons work in a collective manner to learn from the input data while optimizing the result output.

ANNs have a well designed structure. The input is in the form of a multidimensional vector, then this information will propagate to the hidden layers. The hidden layer then make decision based from the input layer by extracting and learning relevant features. This step is referred to as 'learning', its main goal is to optimize the final output. The way hidden layers are stacked upon each other is called 'Deep learning' [12].

2.6.2 Recurrent Neural Network (RNN)

are the neural network models which are able to take some context into account in their decision function. What puts RNNs apart from other neural networks is that there is at least one feedback loop. There are two subsets of RNNs, mainly the Elman and the Jordan RNN models. These two models both have an input layer, a hidden layer, a delay layer, and an output layer. The main difference is that the delay neurons of Elman neural network are fed from the hidden layer, while in Jordan model are fed from the output layer.

RNNs are computationally powerful and are used in many temporal processing models and applications, capable of estimating arbitrary nonlinear dynamical systems (universal approximation), particularly compact when comparing it to feedforward networks for the same precision, especially convenient in human robot interaction (HRI), and can be used as associative memories to build attractors from input-output association.

while this type of model is pretty straightforward and powerful at the same type, it is much harder to train, suffers from vanishing grading and exploding gradient problems. The network stability is hard to grasp due to the nonlinear nature of unit activation output characteristics and the weight adjustment strategies. cannot handle long sequence when using Tanh or ReLU [13].

2.6.3 Long Short-Term Memory (LSTM)

a special kind of RNN, much more suitable when it comes to learn long term dependencies. with the appropriate dataset, LSTM can outperform traditional methods, not only for short-term but long-term forecast as well. LSTM is usually involved with time-series analysis.

LSTM can store data for a long sequences. This allows them to utilize information or part of it from earlier sequences. Unlike traditional RNNs, LSTM does not suffer from vanishing or exploding gradient problem. It can be implemented with GPUs for Parallelization [14, 15].

2.6.4 Convolutional Neural Networks (CNNs)

CNN is an efficient recognition algorithm which is widely used in pattern recognition and image processing. It introduces many positive features such as less training parameters and its high adaptability, and has a simple, straightforward structure. CNNs are less complex network model and the number of weights.

The general structure of CNNs is the following: two layers, one for feature extraction . The second layer is feature map layer. CNNs are mainly used for computer vision (image classification, object detection, image segmentation, video analysis ...).

CNNs support parallelization, which allows for efficient implementation on GPUs (Graphics Processing Units), which in its turn offer faster training making CNNs suitable for real time and large scale applications. CNNs are robust to variations in input data, that makes them less sensitive to small variations and noise.

CNNs are held back by computational complexity, it needs a lot of processing power, needs a large amount of data set to reap the benefits. Also the quality of dataset is a huge factor in the training process. CNNs can also be exploited by adversaries by giving it misleading input, causing the whole network to make false predictions [16].

2.6.5 Architecture of CNN

convolution layer

a convolution layer is the most crucial building bloc or layer in convolutional neural networks (CNNS). It is specifically architectured to process grid like datasets such as images and videos. It proved itself most useful in tasks that deals in image recognition, object detection, and image segmentation.

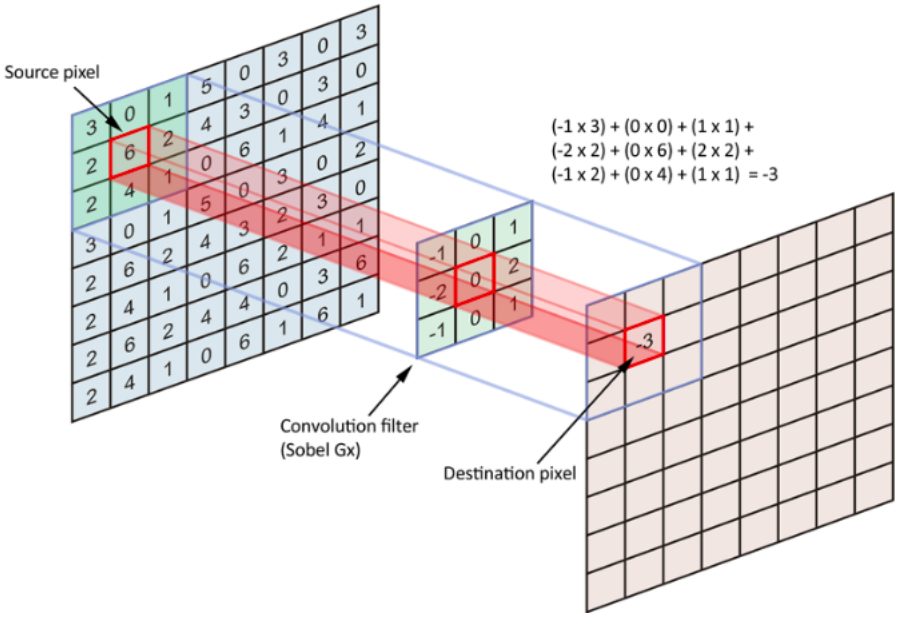


Figure 2.12: Convolution layer

a convolution layer relies on a convolution operation, which combines input data with a set of learnable filters (kernels) to generate an output, basically the filters run through input data to extract features and patterns.

dropout layer

The dropout layer is used in convolutional neural networks as a regulation technique to eliminate overfitting. Overfitting is when a model performs extremely well on the training dataset but falls short when it it comes to unseen data. The dropout layer helps in improving and regulating the performance of the model.

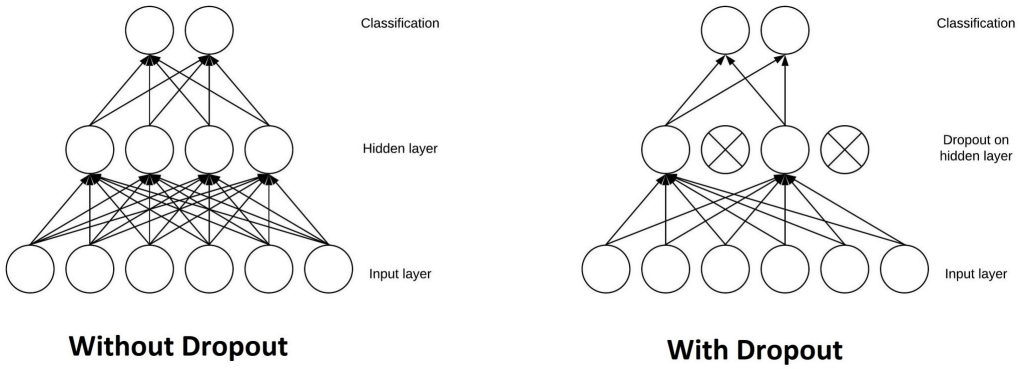


Figure 2.13: a neural network before and after dropout

max pooling

The main aim of a max pooling layer is to reduce the spacial dimensions of the input data while conserving the most important features. It is a popular choice for spacial downsampling in CNNs.

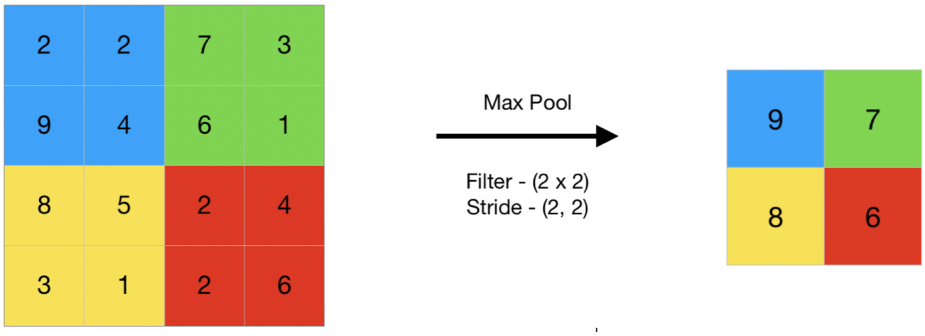


Figure 2.14: max pooling

average pooling

average pooling is another technique for spacial downsampling. It behaves in a similar way as max pooling but instead of selecting the maximum value within a pooling window, it calculates the average value.

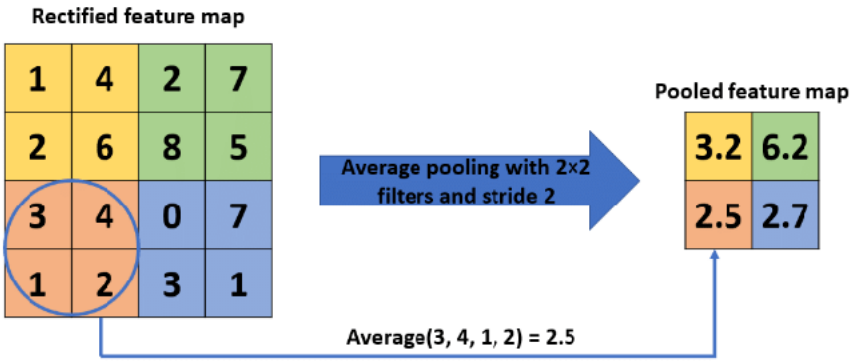


Figure 2.15: average pooling

batch normalization

Is a training process that makes the artificial neural networks faster and more stable. It can also act as a regulizer, which reduce the need for dropout.

2.7 Conclusion

Artificial intelligence (AI), machine learning (ML), deep learning (DL), and convolutional neural networks (CNNs) are all powerful tools that can be used to solve a variety of problems. AI is the broadest term, referring to the ability of machines to exhibit intelligent behavior. ML is a subset of AI that allows machines to learn without being explicitly programmed. DL is a subset of ML that uses artificial neural networks to learn from data. CNNs are a type of DL that are specifically designed for image recognition tasks.

Chapter 3

Application of Mathematical morphology

3.1 Introduction

Mathematical morphology is a powerful framework for analyzing and processing images and signals, offering valuable tools for various applications in computer vision, pattern recognition, and signal processing. Among the fundamental operations in mathematical morphology, dilation, erosion, opening, and closing play a pivotal role in shape analysis, noise reduction, and feature extraction. The integration of Xilinx System Generator and MATLAB provides a robust platform to implement and explore these operations efficiently, enabling researchers and practitioners to tackle complex image and signal processing tasks.

This chapter focuses on the basic applications of dilation, erosion, opening, and closing in mathematical morphology on images and signals, utilizing the capabilities of Xilinx System Generator and MATLAB. These operations form the building blocks of morphological transformations, allowing for the extraction of key information from digital data.

The chapter begins with an introduction to the core concepts of mathematical morphology, including structuring elements and the fundamental operations of dilation, erosion, opening, and closing. Dilation is an operation that expands the boundaries of objects in an image or signal, while erosion contracts them. Opening, which combines erosion followed by dilation, is useful for removing noise and fine details, whereas closing, which combines dilation followed by erosion, helps in filling gaps and restoring broken structures.

The integration of Xilinx System Generator with MATLAB offers significant advantages in implementing these morphological operations on FPGA platforms, enabling hardware acceleration and real-time processing capabilities. We discuss the benefits and challenges of utilizing Xilinx System Generator for efficient deployment of morphological algorithms on FPGA devices, highlighting its potential for high-performance image and signal processing applications.

In conclusion, this chapter provides a comprehensive overview of the basic applications of dilation, erosion, opening, and closing in mathematical morphology on images and signals. By utilizing Xilinx System Generator and MATLAB, researchers and practitioners can leverage the power of these operations to address a wide range of image and signal processing challenges. The insights gained from this chapter will serve as a solid foundation for further exploration and development of advanced morphological techniques, opening avenues for innovative solutions in the field of digital signal and image processing.

3.2 Xilinx System Generator

Xilinx System Generator is a high-level design tool that enables the rapid development of digital signal processing (DSP) systems using Xilinx FPGAs. It provides a graphical environment for designing and implementing complex algorithms by connecting predefined functional blocks and modules. With System Generator, designers can efficiently prototype and deploy signal processing algorithms on FPGA platforms, taking advantage of the parallel processing capabilities and hardware acceleration offered by Xilinx devices.

Figure 3.1 presents how we can model in simulink using the Xilinx system generator, as the Gateway In and Gateway Out blocks present the input and output from the FPGA, respectively. On the outside, simulink blocks are used to do the required processing needed

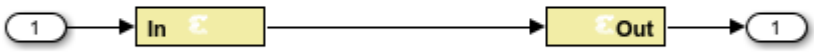


Figure 3.1: Xilinx System Generator basic Model

for the images and signals. Depending on the model, the system generator will define the type of FPGA board that will be used. With over 90 blocks available to us and the ability even to inject C, Matlab code, and other perks that the system generator presents, it sure is a great resource for modeling, visualizing, and finally implementing the project needed [17].

3.3 Proposed design for image processing

In order to fulfill the real-time application of the FPGA cards, image processing techniques can be carried out on the system generator. However, we must first go through a number of processes before supplying the input to our FPGA simulation environment. To enter and see the image needed to show off our work, we need an image input and an image viewer blocks. As he design flow of hardware implementation of image processing using XSG is given in the next figure

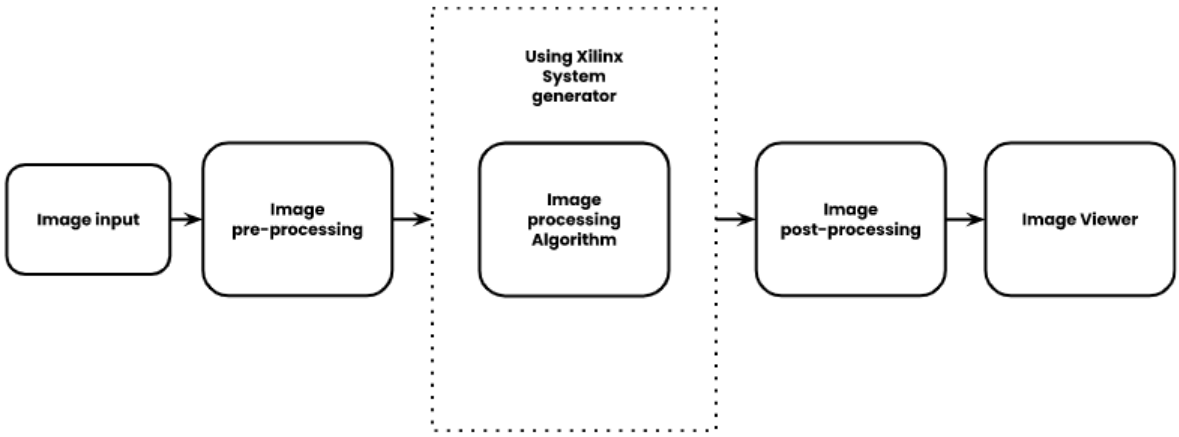


Figure 3.2: Proposed design for the hardware implementation

3.3.1 Image pre-processing unit

Since the gateway in block allows determined data types, a preprocessing step is required before proceeding and presetting the data as an input to the FPGA environment. The preprocessing will be done in several steps:

Resize:

The input image may need to be adjusted to match the FPGA design's required input size. This step assures that the image's size and resolution are correct for the next step. We will mainly be using bicubic interpolation as it helps preserves the image's details.

Data formatting

This step in used to put the data in the correct form to be submitted into the FPGA environment, We will be transforming the image from 2D into 1D, followed by frame conversion and buffer which assures the right format for the desired image.

The blocks used to create the pre-processing unit are shown in the figure below, but many additional blocks might be used, such as conversion to frames or as we will see later in the case of colored images.

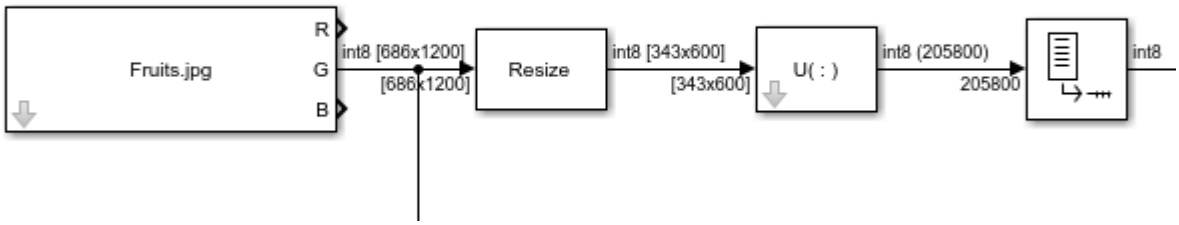


Figure 3.3: Pre-processing unit in simulink

3.3.2 Image Post-processing unit

As previously stated, the FPGA environment will not output the image in the appropriate format, thus we must perform post-processing before viewing the final image output.

This phase is essentially the inverse of how we prepared the data in the pre-processing step; in our instance, reconstructing the image from one dimension is the way to go.

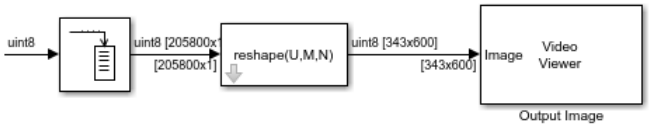


Figure 3.4: Post-processing unit in simulink

3.3.3 Models for image processing algorithms

Since we established the necessary block for the processing, we only have the part where our logical and mathematical algorithms go.

Image inverter

We'll start with one of the most basic and widely utilized processes, the color inverter. We begin by presenting the entire model utilized to carry out the activity, including the post- and pre-processing components [18].

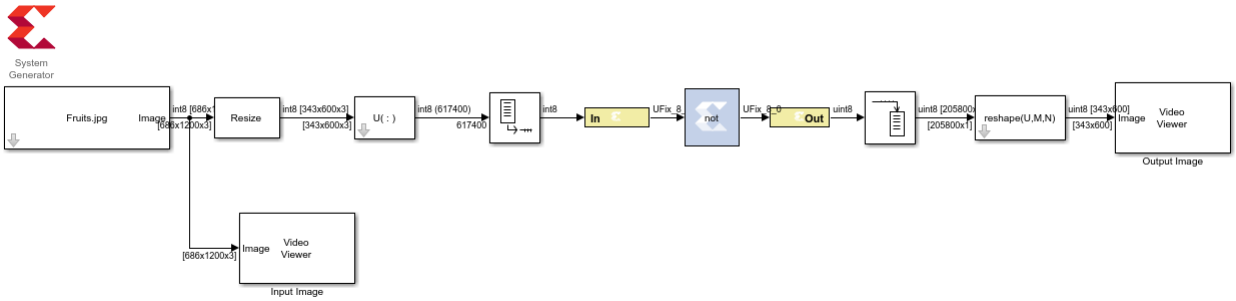


Figure 3.5: GreyScale image Inverter

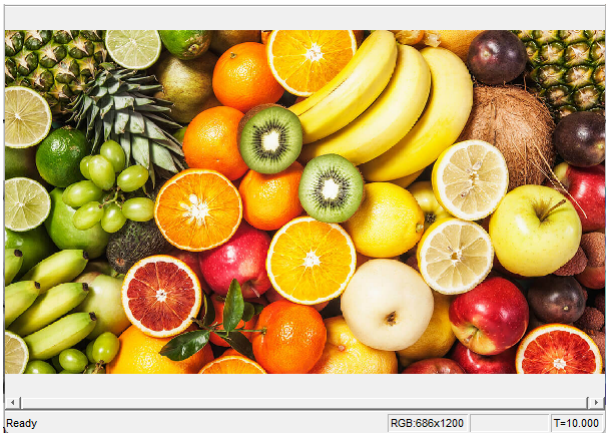


Figure 3.6: Input image

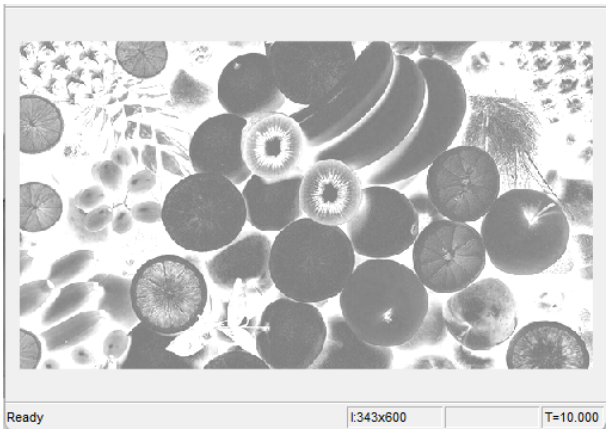


Figure 3.7: Output Image

3.3.4 Mathematical Morphology on Xilinx system generator

We would proceed to apply the fundamental algorithms for mathematical morphology after we've established the basics and basic structure of the model on the Xilinx system generator. Firstly to demonstrate how the simplest operation are performed we will be performing a binary dilation and erosion at the very basic state following with the following input image [19]:

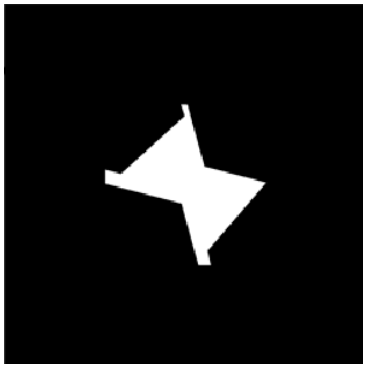


Figure 3.8: Original input image

We will be having the following structuring element on the image:

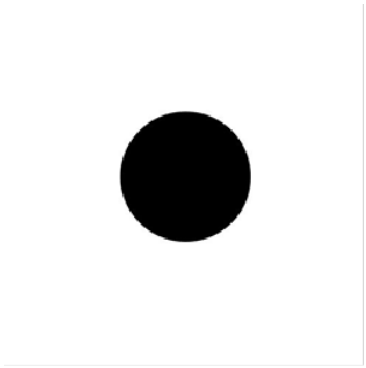


Figure 3.9: Structuring element

By simply having this simple blocks we are able to perform the dilation and the erosion of the image above : And finally we obtain the following results:

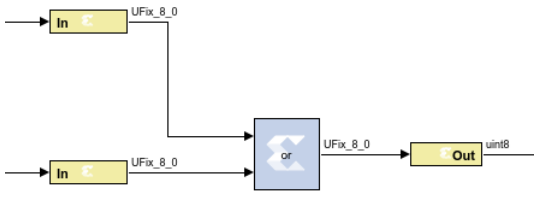


Figure 3.10: Dilation on XSG

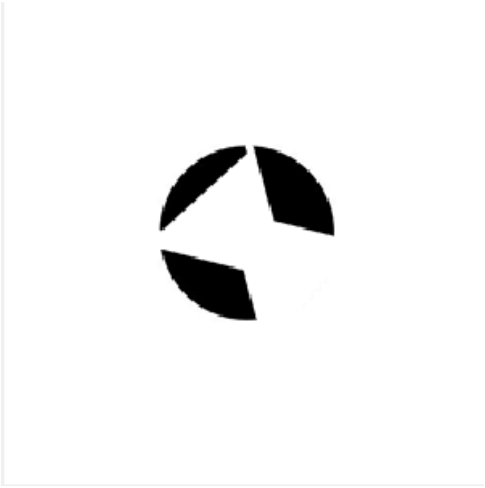


Figure 3.11: Binary dilation output



Figure 3.12: Binary erosion output

3.3.5 Grayscale Application

Dilation

The difference between the two photographs is obvious, as the dilated image has lost details while improving the exterior of the shapes:



Figure 3.13: Dilation input image



Figure 3.14: Dilation output Image

Erosion

The difference could be observed from a long distance away; the man has begun to become lost on the inside as a result of the image losing the information of the pixels on the inside.



Figure 3.15: Erosion input image



Figure 3.16: Erosion output Image

Opening

We used the dilation result in this phase, and to improve the image's information, we created an algorithm to darken the surrounding area using subtraction and addition for the dilated and eroded images, respectively, and then used Opening.



Figure 3.17: Opening output from the dilated image



Figure 3.18: Opening output from the eroded image

Closing

We follow up with same concept with closing and seeing what changes occur on our image.



Figure 3.19: Closing output from the dilated image



Figure 3.20: Closing output from the eroded image

3.3.6 Conclusion

To conclude this chapter, we have looked at the basic implementation of the binary applications of MM, Grayscale MM, using XSG and Matlab's Simulink alongside with the presentation of the effects of the algorithms on grey and binary images.

Chapter 4

Testing and implementation

4.1 Introduction

The following will be providing a real life application of Mathematical Morphology, we chose PCG signal to test the effect of the MM algorithms on, and to see how accurate the model can be using the techiques we have chosen.

4.2 Database

We chose the michigan heart sounds database, we selected a set of 6 classes along with the normal heart sound, having each class with a signal to segment to about 75 segment that would be transferred later into and image to be treated by our model.

4.2.1 Database Description

To test our approach to classifying different heart diseases, we use different heart data (PCG) that are obtained from the Library of Heart Sounds and Whispers of the University of Michigan School of Medicine in the site given by [20]. Our study is done with this database which contains 23 cardiac sound, recorded as .wav format in 16 bits and a frequency of 44.1 khz, we proceeded with only 7 classes .

Class	PCG
1	Normal S1 S2, Supine, Bell
2	Split S1, Supine, Bell
3	S4, LLD, Bell
4	Mid Sys Click, Supine, Bell
5	S3, LLD, Bell
6	Early Sys Mur, Supine, Bell
7	Mid Sys Mur, Supine, Bell

Table 4.1: Data classes

4.2.2 Database Distribution

Our classes are distributed as the following, we have chosen about 65 to 80 sample per class as per to be used in our run thourgh this process.

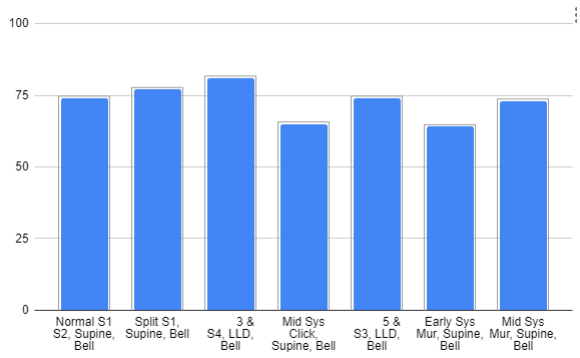


Figure 4.1: Database distribution

4.3 Material

4.3.1 Software

Matlab

Matlab is a high-level programming language and environment for numerical computation, visualization, and data analysis. It provides a wide range of tools and functions for scientific and engineering applications. Matlab allows users to create algorithms, manipulate matrices, and generate plots, making it a powerful tool for research, simulation, and development in various fields such as mathematics, physics, and engineering. In our work we used the 2017b version because it was the only version compatible win Xilinx system generator .

Python

Python is a versatile and powerful high-level programming language known for its simplicity and readability. It supports multiple programming paradigms and offers a vast array of libraries and frameworks. Python is widely used for web development, data analysis, artificial intelligence, scripting, and automation, making it a popular choice among developers [].

4.4 Libraries

4.4.1 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It enables the creation of neural networks and deep learning models, allowing users to build, train, and deploy artificial intelligence applications efficiently. TensorFlow provides a flexible and scalable platform for tasks like image recognition, natural language processing, and predictive analytics.

4.4.2 Keras

Keras is a high-level open-source neural network library written in Python. It is built on top of TensorFlow and provides a user-friendly interface for building and training deep learning models. Keras allows developers to quickly prototype and iterate on their models, with support for various neural network architectures and pre-trained models. It is widely used for tasks such as image classification, text generation, and sentiment analysis.

4.4.3 Sklearn

Sklearn, short for scikit-learn, is a popular open-source machine learning library for Python. It provides a comprehensive set of tools for data preprocessing, model selection, and evaluation. Sklearn includes various algorithms and utilities for classification, regression, clustering, and dimensionality reduction, making it a go-to library for machine learning tasks.

4.4.4 Matplotlib

Matplotlib is a Python library used for creating visualizations and plots. It provides a flexible and extensive set of tools for generating line plots, scatter plots, bar plots, histograms, and more. Matplotlib is highly customizable and widely used in data analysis, scientific research, and data visualization tasks.

4.5 System Methodology

In this step, we need to manage our data that we generated using matlab, we had a set that cosists of 508 images. we used 70% as training data, 10% for validation and 20% for the testing.

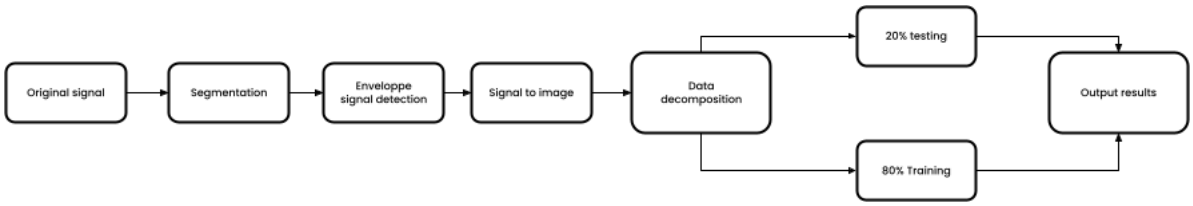


Figure 4.2: Model training diagram

4.5.1 Preprocessing

This was the preprocessing of our data before we began training the model. We started with signals that were about 3052800 points long and cut them into signals that were about 75000 points wide apiece. The envelope signal was identified using mathematical morphology, specifically dilation with a line-shaped structuring element of 100 points width. Finally, to obtain the final data, we proceeded by sketching the signals in 500x500 pixel pictures. The first approach involved drawing each point with a point on the image, while the second involved drawing a line from the 0 point, which is the image's center, into the value of the point.

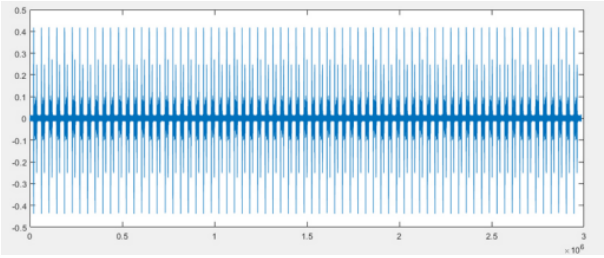


Figure 4.3: Original Signal

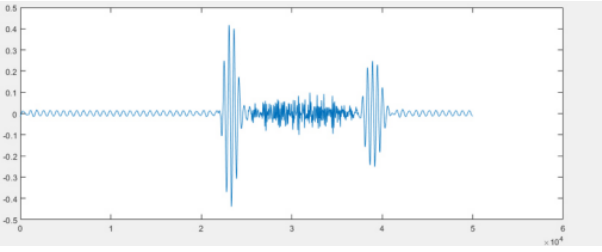


Figure 4.4: Segment

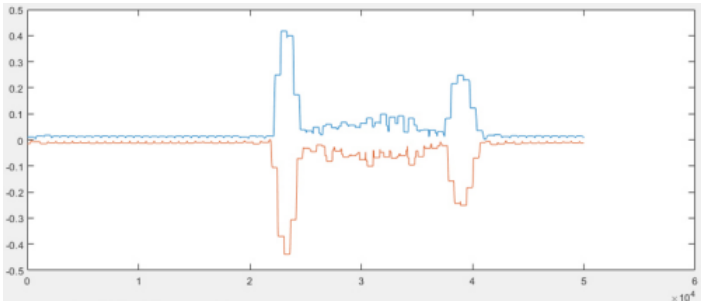


Figure 4.5: Resulting enveloppe

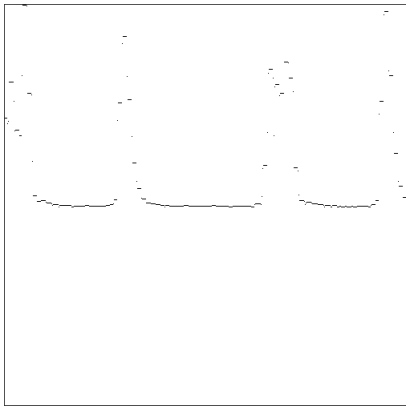


Figure 4.6: First method's pre-processing results

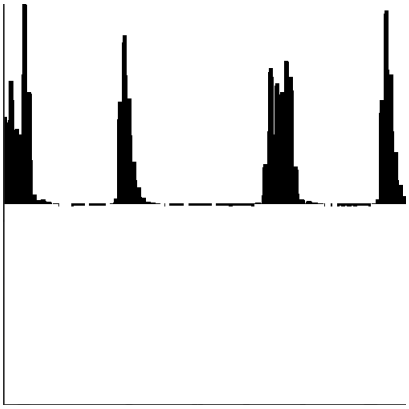


Figure 4.7: Second method's pre-processing results

4.5.2 Decomposition of the Data

We split the data as depicted in the Figure 4.8:

- Train: 70% of all data is for the training set.
- Validation: the remaining 30% it will be split it on two set 20% validation.
- Test: the remaining 10% is for the final test, this testing set that not seen by the model during the training.

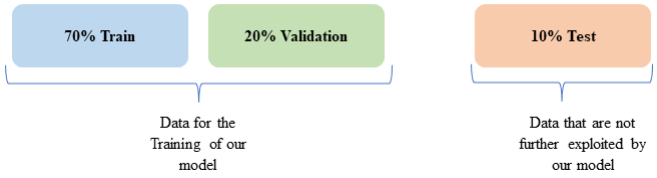


Figure 4.8: Data decomposition

4.6 CNN Model

After the last step, we ended up with images such as the one below, which are the building blocks of our database. Our CNN model is roughly presented as follows :

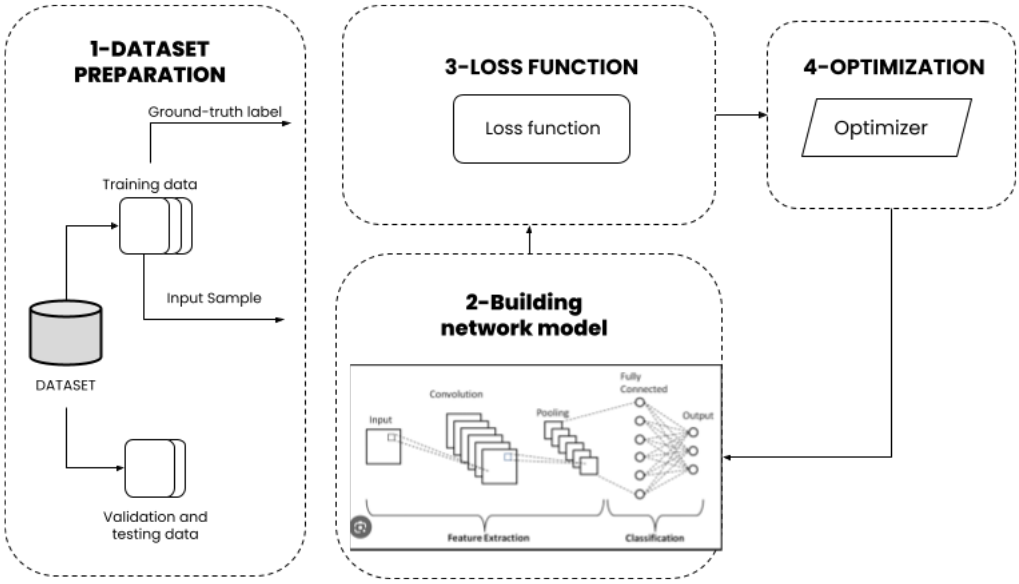


Figure 4.9: CNN Model

4.6.1 Proposed Architecture

After several tests, we were able to develop the best architecture to have good result. (Figure 4.10). To improve the efficiency of the treatment, an activation function "ReLu (Rectified Linear Unit)" is inserted between the treatment layers on the output images.

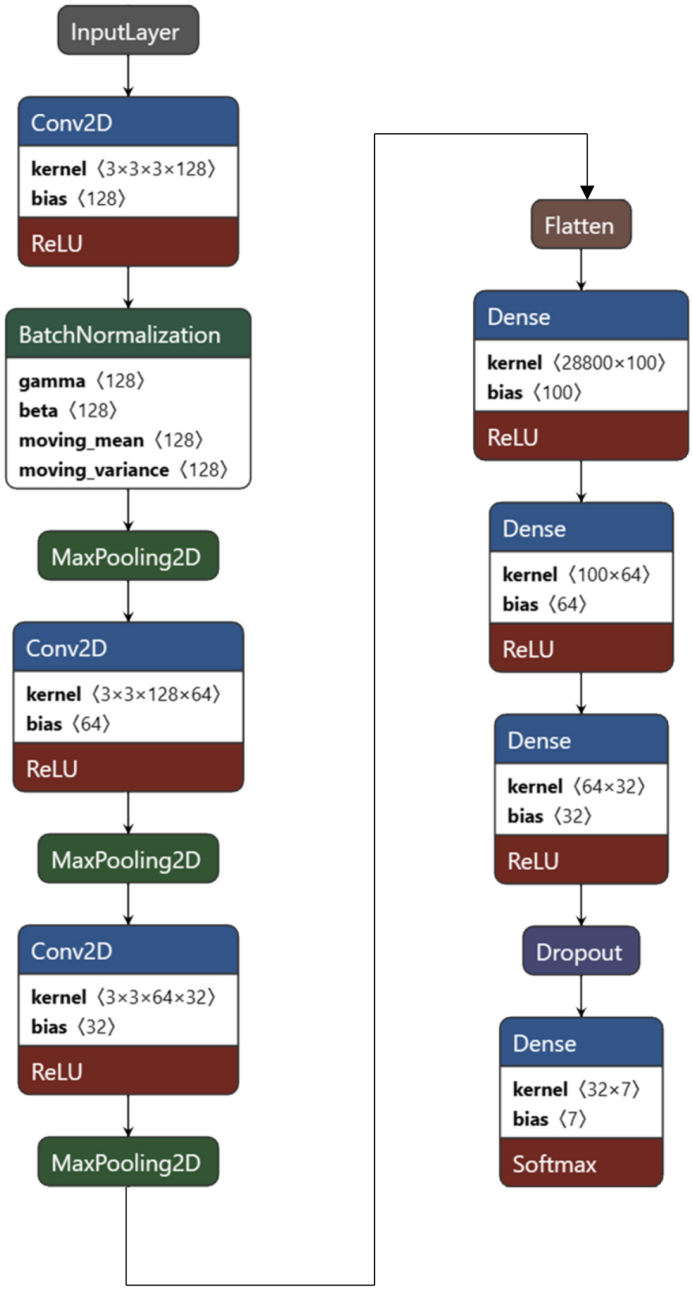


Figure 4.10: Proposed model

The architecture presented in Figure 4.10 consists of: is a 7-layer CNN with batch normalization and dropout. It takes a 28x28x3 image as input and outputs a prediction of one of 7 classes.
The first layer is an InputLayer with 28800 neurons, which is the size of the input image.
The next 4 layers are convolutional layers. Each convolutional layer has a kernel size of 3x3 and a stride of 1. The number of filters in each layer is 128, 64, 32, and 16, respectively.
The activation function used in all convolutional layers is ReLU.
After each convolutional layer, there is a BatchNormalization layer. Batch normalization helps to improve the performance of the model by normalizing the output of the convolu-

tional layer.
After the last convolutional layer, there is a MaxPooling2D layer with a pool size of 2x2. This reduces the size of the feature maps by half.
The next layer is a Dense layer with 100 neurons. The activation function used in this layer is ReLU.
The final layer is a Dense layer with 7 neurons, one for each class in the dataset. The activation function used in this layer is Softmax, which outputs a probability distribution over the 7 classes.
The dropout rate is 0.5, which means that 50% of the neurons in the Dense layer are randomly dropped out during training. This helps to prevent overfitting.

4.6.2 Compiler

Before starting the training, we need to configure the following processes:

- **Loss:** This is a function used by the model to minimize errors. We utilize 'categorical-crossentropy' as we have a multi-class classification.
- **Optimizer:** We employed the stochastic optimizer 'Adam' as it delivers excellent results.
- **Metrics:** In the case of a classification problem, we utilize metrics=accuracy.

4.6.3 Methods Used

We have put in our code a lot of methods to save our best model of the classification process, they are as follow :

Callbacks Model Check Point

We utilized this Keras Callbacks function because it can play a vital role. Training our model can take hours or even days to complete, so it is necessary to have a function to monitor our model during execution.

We employed this function to save the most accurate model and compare it with the accuracy of the next epoch. If the accuracy increases, we can take this model and overwrite the previous one to retain the best model [21].

4.7 Result Evaluation

In order to showcase and evaluate the results achieved by our 2D CNN model, we presented the different outcomes in terms of:

- Accuracy / Loss
- Classification report
- Confusion matrix

We conducted a total of 50 epochs to obtain these results.

4.7.1 Accuracy and Loss Graph

Figure 4.11 represents the graphs of: accuracy and loss of training and validation as a function of the epochs number, we note that the accuracy for train and validation increases proportionally with the number of epochs. On the other hand the loss marks a decrease inversely proportional with the number of epochs for the train and validation.

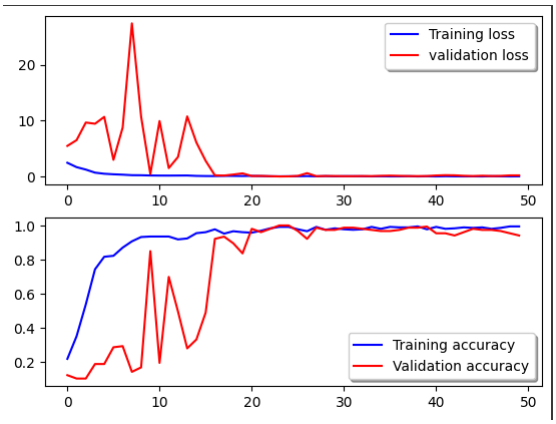


Figure 4.11: accuracy and loss graph

4.7.2 Classification Report

- **Precision**: It is the ratio of true positives to the sum of true positives and false positives. It is also the ratio of correct positive predictions to the total positive predictions.
- **Recall**: It is the ratio of true positives to the sum of true positives and false negatives. It measures the number of positive points that our model is able to recall from the data.
- **F1-Score**: It is the weighted harmonic mean of precision and recall. The closer the F1-score value is to 1.0, the better. It is considered as a harmonic average of precision and recall.
- **Support**: It refers to the number of signals for each class in the database.

Based on the classification report, we can see 91% precision in the fourth class, compared to 100% for the others, as well as 93% recall in the first class and 96% and 95% for the f1-score for the first and fourth classes, respectively."

	precision	recall	f1-score	support
0	1.00	0.93	0.96	28
1	1.00	1.00	1.00	22
2	1.00	1.00	1.00	27
3	0.91	1.00	0.95	21
4	1.00	1.00	1.00	22
5	1.00	1.00	1.00	17
6	1.00	1.00	1.00	16
accuracy			0.99	153
macro avg	0.99	0.99	0.99	153
weighted avg	0.99	0.99	0.99	153

Figure 4.12: Classification report

4.7.3 Confusion Matrix

Figure 4.14 depicts the confusion matrix, which summarizes the prediction results and allows for the evaluation of our classification model's performance. Each row corresponds to a true class, and each column corresponds to an estimated class. Since we have 7 classes, our confusion matrix is of size 7x7



Figure 4.13: Confusion matrix

4.7.4 Best Model Result

The best results of our model are listed in the table below 4.2:

Table 4.2: Best Result

	Accuracy	**Loss**
Train	100%	0.001
Validation	99%	0.01

4.8 Final Test

In this section we use the 10% (41 images) of the final test set that are not used by our CNN model. We will represent the results of the prediction are given:

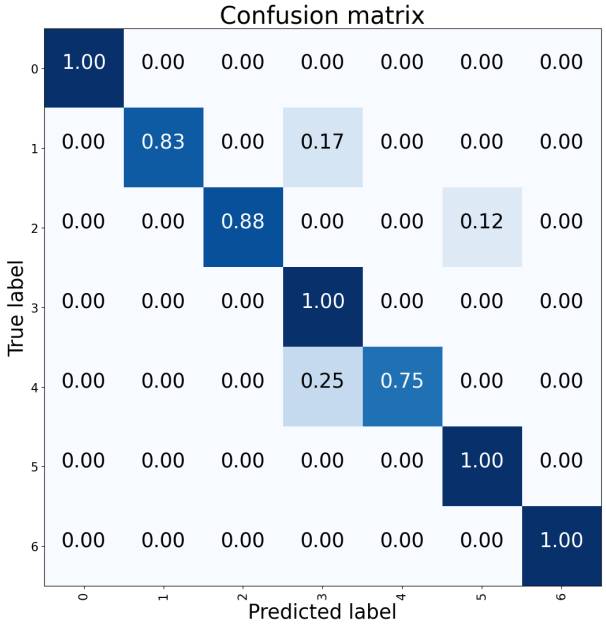


Figure 4.14: Confusion matrix of the final test

From this matrix we note that there is a problem in the classification of the images of the second, third and fifth classes which are miss classified as another classes because the number of images is few and same similarity between the classes.

Table 4.3: Test Result

	Accuracy	**Loss**
Test	95%	0.1

4.9 Conclusion

For the final chapter we chose a real life application for our proposed MM algorithms, we would preprocess our signal and present it to our model that eventually will give us a very accurate classification according to the data base that we generated.

Conclusion

objective of this master's thesis is to provide a comprehensive understanding of mathematical morphology algorithms, their applications, and implementation. Various fields and tools have been defined, including environments like XSG and Matlab, as well as libraries capable of modeling and creating artificial intelligence.

To demonstrate the practical implementation of these algorithms, we selected an image processing example involving grayscale and binary images. This allowed us to observe the algorithms' performance and behavior within the XSG environment, providing valuable insights.

Additionally, we sought to apply these tools to a real-life scenario. After careful consideration, we focused on the classification of phonocardiogram signals. By leveraging the preprocessing capabilities of the algorithms we employed, we developed a model that achieved remarkable accuracy in signal classification.

Through this research, we aim to establish a clear understanding of mathematical morphology algorithms, exploring their potential applications and evaluating their effectiveness in different domains. The combination of theoretical analysis, practical implementation, and real-world application enables us to contribute to the advancement of this field and showcase the benefits of utilizing these algorithms in various contexts.

Bibliography

[1] MathWorks, "Matlab structuring element article," https://www.mathworks.com/help/images/structuring-elements.html.

[2] F. Y. Shih, *Image processing and mathematical morphology: fundamentals and applications*. CRC press, 2009.

[3] "Biological neuron model, wikipedia,"

[4] I. W. Griffiths, "Principles of biomechanics & motion analysis," 2005.

[5] P. SAEED BENJAMIN NIKU, PHD, "Introduction to robotics: Analysis, control, applications," 2020.

[6] "Review of scientific instruments," 1994.

[7] N. Gupta, "Artificial neural network," vol. 3, 2013.

[8] L. N. Kanal, "Encyclopedia of computer science," 2003.

[9] "arxiv:1702.05659," 2017.

[10] D. o. C. S. Anidhya Athaiya Assistant Professor and J. Engineering, Global Institute of Technology, "International journal of engineering applied sciences and technology," 2020.

[11] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[12] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[13] A.-N. Sharkawy, "Journal of advances in applied computational mathematics„" 2020.

[14] S. Muzaffar and A. Afshari, "Short-term load forecasts using lstm networks," *Energy Procedia*, vol. 158, pp. 2922–2927, 2019. Innovative Solutions for Energy Transitions.

[15] X. W. P. C. Y. C. J. L. Zheng Zhao, Weihai Chen, "Lstm network: a deep learning approach for short-term traffic forecast," 2017.

[16] Y. Z. P. W. J. Z. Tianyi Liu, Shuangsang Fang, "Implementation of training convolutionalneural networks,"

[17] T. Šušteršič, V. Milovanović, V. Ranković, N. Filipović, and A. Peulić, "Medical image processing using xilinx system generator," pp. 104–116, 2020.

[18] A. Mohammed, E. Rachid, and H. Laamari, "High level fpga modeling for image processing algorithms using xilinx system generator," *International Journal of Computer Science and Telecommunications*, vol. 5, no. 6, pp. 1–8, 2014.

[19] D. P. Guragain, P. Ghimire, and K. Budhathoki, "Implementation of fpga based image processing algorithm using xilinx system generator," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 01, pp. 2395–0056, 2018.

[20] F. Richard D. Judge, M.D., "Heart sound murmur library," 2015. https://open.umich.edu/find/open-educational-resources/medical/heart-sound-murmur-library.

[21] "Keras open source library," https://keras.io/api/callbacks/model_checkpoint/.