

**UNIVERSIDAD AUTONOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA**



TESIS

**USE OF ARTIFICIAL INTELLIGENCE TOOLS FOR
COMPUTATIONAL-AIDED DIAGNOSTICS**

POR

MAYRA CRISTINA BERRONES REYES

**COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
DOCTORADO EN INGENIERÍA DE SISTEMAS**

2024

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO



USE OF ARTIFICIAL INTELLIGENCE TOOLS FOR
COMPUTATIONAL-AIDED DIAGNOSTICS

POR

MAYRA CRISTINA BERRONES REYES

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
DOCTORADO EN INGENIERÍA DE SISTEMAS

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
Facultad de Ingeniería Mecánica y Eléctrica
Posgrado

Los miembros del Comité de Evaluación de Tesis recomendamos que la Tesis "Use of artificial intelligence tools for computational-aided diagnostics", realizada por la estudiante Mayra Cristina Berrones Reyes, con número de matrícula 1646291, sea aceptada para su defensa como requisito parcial para obtener el grado de Doctorado en Ingeniería de Sistemas.

El Comité de Evaluación de Tesis

Dra. María Angélica Salazar Aguilar
Director

Dra. Cristian Castillo Olea
Co-director

Dr. Vincent André Lionel Boyer
Revisor

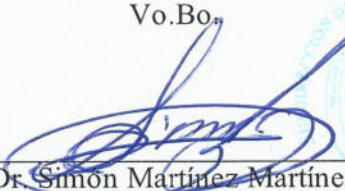
Dr. Romeo Sánchez Nigenda
Revisor

Dra. Sara Elena Garza Villarreal
Revisor

Dra. Satu Elisa Schaeffer
Revisor

Dr. José Arturo Berrones Santos
Revisor

Vo.Bo.



Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

Institución 190001

Programa 557620

Acta Núm. 342

Ciudad Universitaria, a 7 de mayo de 2024.

In loving memory of my grandfather, Ángel Berrones.

ACKNOWLEDGMENTS

I want to express my heartfelt gratitude to the Autonomous University of Nuevo León (UANL) for their continuous support throughout my academic journey, spanning from my bachelor's to my doctorate. A special appreciation goes to the Tigres triathlon representative team, whose financial assistance and encouragement mirrored the discipline and excellence ingrained in our athletic pursuits.

I extend my thanks to the Faculty of Mechanical and Electrical Engineering (FIME) for providing numerous growth opportunities and ongoing support in enhancing our facilities for the betterment of postgraduate programs. A sincere thank you is due to the Mexican National Council of Science and Technology (CONACYT) for their crucial financial support that enabled the completion of this project.

I would also like to thank the FUCAM radiology team for generously sharing vital data and lending their expertise in reviewing the validation and testing of our experiments. Special recognition goes to Ph.D. Cristian Olea and M.D. Erick, our primary collaborators, for their steadfast support.

I express my gratitude to my reviewers, Angélica Salazar Aguilar Ph.D., Satu Elisa Schaeffer D.Sc., Sara Elena Garza Villarreal Ph.D., Romeo Sanchez Nigenda Ph.D., Arturo Berrones Santos Ph.D., and Vincent Boyer Ph.D.. Their patience, dedication, and constructive feedback have been invaluable throughout the experimentation and implementation of our final work, shaping many of my current professional practices.

Thank you so much to my friends and colleagues from my master's generation and my current fellow doctorate classmates for their help and assistance throughout my career and for all the wonderful and enjoyable memories we made.

Last but not least, I am profoundly grateful to my family, the cornerstone of my success. My parents, my greatest teachers, my sister, a constant companion in our academic journey we shared since we were little, and my brother, whose tenacity continues to inspire me.

ABSTRACT

Mayra Cristina Berrones Reyes.

Candidate for obtaining the degree of Doctorado en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Title of the study: USE OF ARTIFICIAL INTELLIGENCE TOOLS FOR COMPUTATIONAL-AIDED DIAGNOSTICS.

Number of pages: 142.

OBJECTIVES AND HYPOTHESIS:

HYPOTHESIS: Combining deep learning with traditional machine learning in an ensemble approach will enhance breast cancer detection accuracy from previously designed deep learning models in mammogram images, maintaining manageable computational demands. This hybrid model is expected to offer a scalable and computational-friendly solution for medical imaging.

For this work, our primary and secondary objectives align with the overarching goal of employing a wide range of artificial intelligence tools to improve existing computer-aided diagnostics methods.

MAIN OBJECTIVES:

- **Develop an Advanced Computer-Aided Diagnostic Tool:** To develop a robust and effective tool for assisting medical experts in diagnosing anomalies in medical images, leveraging a range of artificial intelligence methodologies.
- **Classify Medical Images:** To utilize various artificial intelligence algorithms and ensemble methods to classify medical images into “with anomalies” and “without anomalies” categories with high accuracy.
- **Image Segmentation for Anomaly Detection:** After classification, combine traditional artificial intelligence methods to perform image segmentation and precisely locate the anomaly within images classified as containing an anomaly.

SECONDARY OBJECTIVES:

- **Utilize Ensemble Learning:** To integrate ensemble learning techniques, like bagging, boosting, and stacking, with deep learning to reduce the training complexity and enhance performance.
- **Experiment with Transfer Learning:** To leverage pre-trained weights from popular architectures (VGG-16, Inception, etc.) for feature extraction and examine how this affects the overall diagnostic process.
- **Ensure Data Integrity and Model Validation:** To use well-balanced and externally validated data sets to train and test the models, ensuring their real-world applicability.
- **Performance Comparison:** Highlight the differences in effectiveness between simple and deep neural network architectures and traditional machine learning algorithms in achieving high classification accuracy.

- **Optimize for Computational Efficiency:** To consider the computational time and resources, particularly when feature extraction and ensemble methods are used.
- **Benchmarking and Evaluation:** To compare the developed tool against existing methods using established datasets, focusing on popular metrics like accuracy and F1 score.
- **Understand the Impact of Data Imbalance:** To explore how an imbalance in data can introduce bias and affect the overall performance and how to mitigate this properly.

The primary and secondary objectives consider previous difficulties when modelling and testing real clinical mammography images from a Mexican hospital specialized in cancer research. The main issue was the discrepancy between the accuracy displayed in benchmark data sets and the data set provided by this hospital (YERAL).

EXPERIMENTAL FINDINGS:

In our quest to enhance breast cancer detection through computational means, we navigated a spectrum of machine learning and deep learning methodologies, highlighting each approach's merits and challenges. Strikingly, combining deep learning with machine and statistical learning methodologies resulted in a well-structured strategy, enhancing our model's predictive accuracy, notably on our target data set, YERAL.

Specifically, leveraging the feature extraction capabilities of a pre-trained VGG-16 model and integrating it with ensemble learning strategies—bagging, boosting, and stacking—brought forth good model performance. The bagging model, notably, showed a robust 91% accuracy on the development set and held 82% accuracy and F1 score on the target dataset. This strategy amplified our model's performance and achieved this without overloading the computational burden, marking a significant

stride in our exploration and application of AI in breast cancer detection.

CONTRIBUTIONS:

This research comprehensively explores the evolving landscape of Computer-Aided Diagnostics (CAD), offering critical insights across multiple dimensions. With a detailed review of the historical reliance on rule-based algorithms, this work highlights how the field has undergone significant transformations, primarily driven by machine learning and statistical learning advances.

One essential contribution is the critical evaluation of rule-based algorithms in CAD. The research highlights the limitations of these algorithms, showing that while they served as stepping stones, they had inherent limitations that necessitated the adoption of more advanced techniques. Building on this, the work delves into the role of statistical learning in CAD, offering an in-depth analysis that outlines its strengths, weaknesses, and potential avenues for future research.

Another significant contribution is the extensive overview of CAD machine learning applications, which includes exploring supervised, unsupervised, and semi-supervised learning methods. This section clarifies their respective performance metrics and suitability for various diagnostic problems, setting the stage for more targeted and practical implementations in the future.

Deep learning receives particular attention as it represents the cutting edge of CAD technologies. The research examines the types of diagnostic problems best suited for deep learning techniques, providing a roadmap for future research, including challenges that still need to be overcome.

Further, the research is open to the less technical aspects that affect CAD adoption. It investigates the resistance CAD systems have encountered within the medical community, shedding light on factors such as trust, technical limitations, and the stigma surrounding human-versus-computer errors in medical diagnoses. This work also delves into the often-overlooked aspects of technical limitations and interoperability issues. It uniquely highlights the challenges peculiar to developing countries and different from those faced by first-world countries.

Finally, the ethical and legal dimensions are examined, particularly concerning the lack of centralized data for medical imaging. This aspect contributes to understanding the ethical implications that CAD systems introduce into healthcare practices.

Based on this work, we have several practical results, where we explore the nature of different statistical and machine learning processes and methods and combine them to suit the complex task of medical imaging. These experiments have the advantage of shedding light on a process often characterized as a black box and also have the intended purpose of demystifying the inner workings of AI-based algorithms. As a result of these experiments, we developed a GitHub page with a structured explanation of different statistical learning tools, and it can be consulted by the reader in Section A.

Another notorious output of this investigation is our scientific paper describing ensemble learning to combine known convolutional neural networks and traditional machine learning algorithms. Both works share similar goals, the main difference being that in this work, we get into more detail on the previous steps that lead us to the final results highlighted in the article (Berrones-Reyes et al. [2023]).

This work's final and helpful output is the interface generated to help the medical team use the model output without needing previous coding knowledge. It is a very adaptable tool, where all the instructions on how are detailed in a technical report. This interface and the technical report can be requested from the author from the information in Section A.

In conclusion, this research is a multifaceted exploration of CAD, from its technical underpinnings to its socio-ethical implications. It aims to provide a robust foundation for academic researchers and industry professionals, aiding in the more effective design and deployment of future CAD systems.

RESEARCH OUTPUT**PUBLICATIONS**

- BerronesReyes, M. C., SalazarAguilar, M. A., CastilloOlea, C. (2023). Use of Ensemble Learning to Improve Performance of Known Convolutional Neural Networks for Mammography Classification. *Applied Sciences*, 13(17), 9639.

INVITED TALKS AND SEMINARS

- Conversatorio de mujeres en la ciencia. Universidad Santiago de Cali (March 2023)
- Department of Physical-Mathematical Sciences. Seminar of computational science: Data Science. (March 2023)
- Department of mechanical and electrical engineering. Seminar: Challenges and areas of opportunity for deep learning applied to the medical area. (November 2022)
- CONACYT and Women in Data. Seminar: Opportunities and challenges of artificial intelligence in solving tasks for social betterment. (September 2021)
- Vision to the future of systems engineering. Round table with teachers and students to enhance the graduate program of System Engineering. (December 2021)

CONTENTS

Acknowledgments	v
Abstract	vii
1 Introduction	1
1.1 Introduction	1
1.2 Problem Statement	6
2 Background	10
2.1 Machine Learning	10
2.1.1 Logistic Regression	11
2.1.2 Support Vector Machines (SVM)	13
2.1.3 Decision Trees	15
2.1.4 Neural Networks	17
2.1.5 k -nearest Neighbor	19
2.1.6 Random Forest	20
2.2 Statistical Learning	22

2.2.1	Data Pre-processing	23
2.3	Convolutional Neural Networks	24
2.3.1	LeNet-5	31
2.3.2	AlexNet	32
2.3.3	VGG-16	34
2.3.4	ResNet-50	36
2.3.5	Transfer Learning	38
2.3.6	Functions for Deep Learning	41
2.4	U-Net	46
2.5	Real-life Applications	51
2.5.1	Computer-aided Diagnostics	51
3	Literature Review	53
3.1	Evolution of Computer-aided Diagnostics	53
3.1.1	Statistical Learning in CAD	56
3.1.2	Machine Learning in CAD	57
3.1.3	Pushback and Challenges for Medical CAD Systems	59
3.1.4	Data Sharing and Privacy	61
3.1.5	Challenges Faced by Developing Countries	61
3.1.6	Professional Resistance	62

3.1.7	Current State and Future Directions: The Role of Deep Learning in CAD	63
4	Methodology	66
4.1	Materials	66
4.1.1	Data Sets	67
4.1.2	Hardware	70
4.1.3	Software	71
4.2	Classification Process	73
4.2.1	Pre-processing	74
4.2.2	Binary Classification Process	81
4.3	Segmentation process	92
4.3.1	Pre-processing	95
4.3.2	Convolutional Neural Networks	96
5	Results	98
5.1	Classification	98
5.2	Segmentation	111
6	Conclusions	117
6.1	Contributions	118
6.2	Future Work	120
6.3	Research output	122

6.3.1	Publications	122
6.3.2	Invited talks and seminars	122
A	Datasets and codes	137
A.1	Experimental results	140

LIST OF FIGURES

1.1	The evolutionary spectrum of Artificial Intelligence, Machine Learning, and Deep Learning. This illustration shows the hierarchical relationship between these three fields.	3
2.1	Behaviour of the logistic function. It outputs a number between 0 and 1.	12
2.2	Use of kernels in support vector machine to maximize the distance between classes. On the left, we see the representation of a 2D kernel, and on the right, a 3D kernel. The complexity of the SVM goes up exponentially with each kernel and becomes difficult to illustrate graphically.	14
2.3	Schematic diagram of the different terminology for decision tree. Illustration inspired by EdX.org [2023].	16
2.4	Schematic diagram of the structure of a common neural network. As it stands, we have input, hidden, and output layers. The connections between these layers have different hyperparameters, such as activation functions and weights.	18
2.5	Example of how k -NN classifies a new data point based on the different k chosen.	19

2.6	Schematic diagram of a random forest model.	21
2.7	Simplified representation of the process and parameters inside a convolutional layer.	25
2.8	Representation of the max pooling and average pooling layer process.	26
2.9	Representation of the process of the dense layer, which includes the pooling layer, fully connected layer, flattening, and output layer. . .	27
2.10	Representation of the most common activation functions for neural networks.	28
2.11	How to choose the activation function for the hidden layer based on the network type and for the output layer based on problem type. .	29
2.12	Compact visualization of the LeNet-5 architecture (from Karim [2022]) that follows the original architecture by (LeCun et al. [1998]).	32
2.13	Error rate from the IMAGENET challenge, from 2010 (ILSVRC 10) to 2015 (ILSVRC 15). The blue bars represent the error rate, and the orange line represents the complexity incline from shallow nets to 152 layers.	33
2.14	Compact visualization of the Alexnet architecture following the original architecture by Krizhevsky et al. [2017].	33
2.15	Compact visualization of the VGG-16 architecture following the original architecture by Simonyan and Zisserman [2014].	35
2.16	ResNet50 architecture, using a compact visualization by Karim [2022] following the original architecture by He et al. [2016].	37
2.17	Comparative performance of different deep learning optimization algorithms using the library Numpy. Image from Thavanani [2020]. . .	44

2.18	Confusion matrix for binary classification.	45
2.19	U-net architecture as represented by Ronneberger et al. [2015]. The arrows indicate the flow of data and the color of the type of operation performed at that level.	47
3.1	Common flowchart of a generic scheme of CADs used for lesion detection in medical images by Suzuki [2012].	54
3.2	Statistics about an American survey made in 2023 about the attitude different generations have towards AI being used in the medical field (Schmidt [2023]).	64
3.3	Graphical representation of the main concerns of the general population of an American survey regarding the use of AI in medicine (Schmidt [2023]).	65
4.1	Steps involved in the classification stage of the methodology.	74
4.2	Removal of patient information and unnecessary labels from the YERAL data set. The patient information is blurred for privacy concerns. . .	75
4.3	Pre-processing for the CBIS DDSM data set contains the DDSM, Mini Mias, and IRMA project images.	77
4.4	Morphological transformation operations from the OpenCV library. Opening is the function of erosion followed by dilation, and Closing is dilation followed by erosion. Both of them are used to remove noise from binary images.	79
4.5	Process of padding and resizing the images to 224×224 pixels. . . .	80

4.6	Example of the bagging algorithm. This example shows the CNN structures of Alexnet, then VGG-16, and lastly with Inception. Following the function of an ensemble, the final result is finally passed on in a final voting.	88
4.7	example of the stacking ensemble, where each model represents the weak learner used for intermediate predictions. In the end, consider logistic regression as a final estimator.	90
4.8	example of the boosting algorithm, where a decision tree classifier represents the weak learners. This process is set to be iterated until the complete training data fits without error. The other stop function is a specified number of estimators. In this case is set to 200.	92
4.9	Example of the difference between image classification and different segmentation types.	93
4.10	Example with bounding boxes of the intersection over unions (IoU).	95
4.11	Example with bounding boxes of the CBIS-DDSM data set.	97
4.12	example of the final summed mask for the segmentation process.	97
5.1	Training loss and accuracy for YERAL images.	103
5.2	Training loss and accuracy for YERAL images with the Alexnet architecture.	104
5.3	Training loss and accuracy for YERAL images after being subjected to a wavelet processing.	105

5.4	Graphs of the performance of the popular CNN architectures tested for this work. On each image we have the accuracy and loss for the set epochs, showing the validation and training outcome. Subfigures (a), (b), and (c) are related to the classical architectures, while subfigures (c) and (e) correspond to the ones with transfer learning.	108
5.5	Dilation and Erosion function from the OpenCV library.	112
5.6	Manual image transformations using a function of clustering that helps us identify the major pixel clusters and eliminate what we consider noise, such as the mass of the breast.	114
5.7	Manual image transformations using a function of clustering that helps us identify the major pixel clusters and eliminate what we consider noise, such as the mass of the breast.	115
5.8	Different sizes of k for the k -Nearest Neighbor experiment.	116

LIST OF TABLES

2.1	Comparison of the parameters for each architecture, LeNet-5, AlexNet, and VGG-16 (M. Swapna and Prasad [2020]).	35
4.1	Using the most popular performance metrics for AI tools, we compare the validation data set when we have an unbalanced and a balanced data. For this example, we only explore the use of a simple Alexnet architecture described by Krizhevsky et al. [2012].	70
5.1	Results of the best models for binary mammography classification extracted from the complete table by Berrones-Reyes [2019]. The number represented in the column model refers to the parameters of each model.	100
5.2	Factors of Lenet5 models.	101
5.3	Factors of Alexnet models.	102
5.4	Factors of VGG-16 models.	102
5.5	Using the most popular metrics for validation data performance, we compare the use of an unbalanced with a balanced data set using the Alexnet architecture with the same weights described by Krizhevsky et al. [2012].	105

5.6	Performance results on the more popular architectures for image classification. Rows 1, 2, and 4 are the results introduced in the Literature Review Section. Rows 3 and 5 are the architectures that combine the use of transfer learning, and uses the unfreezing method for the last pooling layers.	107
5.7	Final result of the expected performance using the ensemble algorithms compared with the predictions for the target dataset, YERAL	110

CHAPTER 1

INTRODUCTION

In this chapter, we discuss a general view of the environment for the project's development, touching on subjects such as Artificial Intelligence (AI) and its many applications in various fields, emphasizing computer vision. Then, in the problem statement, we deepen the investigation of AI applications in medicine. Lastly, we comment on a summary of the chapters ahead.

1.1 INTRODUCTION

Artificial intelligence (AI) is a rapidly advancing field within Computer Science that focuses on developing computational models capable of performing tasks like humans (Joiner [2018]). This progress is evident in the widespread use of AI-powered applications such as Siri and Cortana (Chander et al. [2022]) and the emergence of self-driving cars (Nadikattu [2016]).

In today's world, enormous volumes of data are generated from various sources, including our social media interactions, bank statements, and online purchase history (Yigitcanlar et al. [2020]). Recognizing the significance of processing and analyzing this data, companies have realized the value of leveraging customer interactions to enhance their products, retain existing customers, and attract new ones (Russell

et al. [2015]).

Consequently, the field of data analysis has evolved substantially, with computer-aided programs becoming pivotal to the success or failure of businesses (Russell et al. [2015]). To address the challenges associated with the complexity of the data and the need for real-time processing, specialized branches of AI have emerged (Vinuesa et al. [2020]).

One such branch is deep learning, which has become a critical sub-field of machine learning. Deep learning utilizes neural network algorithms as its foundation, enabling the successful resolution of highly complex problems (LeCun et al. [2015]). However, it is crucial to understand the distinctions between machine learning, artificial intelligence, and deep learning algorithms in problem-solving (Goodfellow et al. [2016]).

While these terms are often used interchangeably, they have some differences. Machine learning refers to algorithms that allow computers to learn and improve from data without explicit programming (Shinde and Shah [2018]). Artificial intelligence encompasses various techniques and methods to create intelligent systems that emulate human intelligence. On the other hand, deep learning can be interpreted as a “specialized” machine learning that uses neural networks with numerous layers to process complex data and extract meaningful patterns (Abiodun et al. [2019]).

Although these concepts share a common objective of using advanced computing capabilities to solve real-life problems more efficiently, their specific approaches and techniques differ. By understanding these distinctions, we can better appreciate the diverse landscape of AI and its various applications (Sarker [2021]).

Overall, the rapid growth of AI, along with its sub-fields such as deep learning, has revolutionized industries and opened up new possibilities for problem-solving. As technology advances, it is crucial to stay informed about these developments and their potential impact on our lives (Liu et al. [2021]).

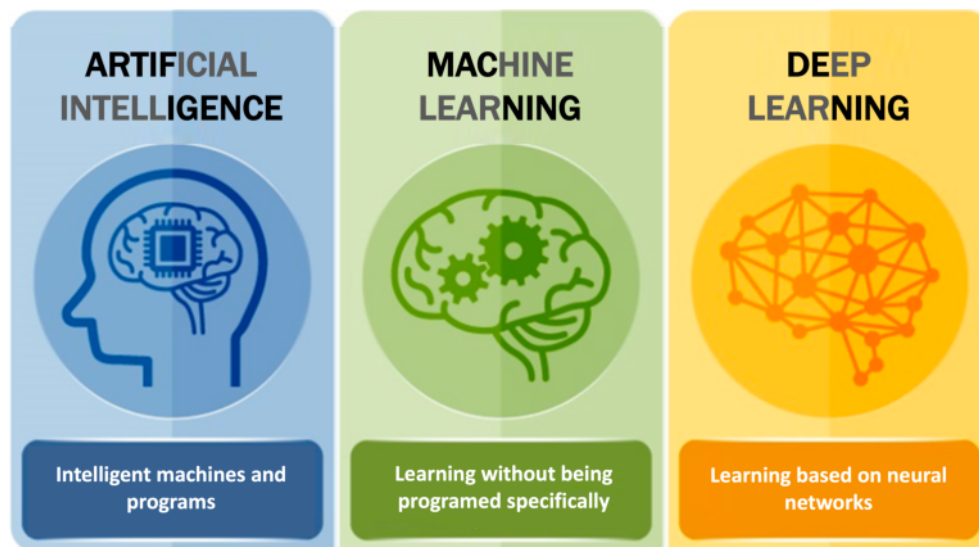


Figure 1.1: The evolutionary spectrum of Artificial Intelligence, Machine Learning, and Deep Learning. This illustration shows the hierarchical relationship between these three fields.

There is a common misconception surrounding artificial intelligence (AI) that portrays it as a recently developed field, representing a new approach to problem-solving using computers. However, this belief is only partially accurate, as AI has existed since the early days of computing when computers were employed to solve repetitive mathematical problems. Initially, AI was used to streamline processes characterized by repetitive tasks, where the precise steps, materials, and procedures were well-defined (Copeland [2022]). These processes could be optimized by leveraging AI, as computers could compute results far more quickly than humans. Beyond its practicality, AI also relieved humans of tedious tasks, enhancing workflow efficiency across numerous processes.

Since AI proved to be a significant improvement, it is natural that it followed a change to adapt to new and more challenging tasks. The complexity of the data and jobs required to be completed by the computer became more demanding and challenging (Haug and Drazen [2023]). That is when the use for more flexible algorithms came to be, and we have machine learning.

As mentioned in Figure 1.1, at the base, we have Artificial Intelligence, which focuses on emulating human-like decision-making and problem-solving. Nested inside the world of AI is Machine Learning (ML), a specialized branch that enables computers to learn from data, solving more complex problems without explicit programming. Further specializing ML, we have Deep Learning (DL), a subset that leverages neural networks for even more complex problem-solving tasks. Despite their increasing complexity and specialization, each successive section retains key attributes of its parent category, illustrating the evolution and continuity of intelligent computation.

In the previous iteration of AI, programs needed strict steps and paths since the computer would not perform anything outside of the parameters assigned by the program's instructions. If anything unexpected crossed its way, the program would crash and stop working correctly (Copeland [2022]).

Constructing a procedure that strict may have been feasible with the type of data and challenges of minor problems, but for more complex instructions, programming every single path could be time-consuming and ineffective, especially since we are trying to optimize the time to have good results. Machine learning enabled the ability of the computer to solve problems with little to no human input and make decisions based on previous actions and observations (Lazzaro [2021]).

Over time, ML algorithms could detect patterns in the data and learn from the outputs to make their predictions. The main difference between the older AI programs and this ML approach is that we do not need to program specific actions and conditions. Still, we could feed the algorithms examples of labelled data to help them make calculations, learn, and identify patterns automatically (Chollet [2018]).

ML then proved to be a very efficient tool. As long as they were provided with an accurate training example, these algorithms became efficient in labelling similar images without human assistance. This advancement helped save time and money on tasks such as data analysis.

The type of learning we described is supervised learning, where the algorithms need labelled data, which needs to be approved by humans. Since these programs will analyze the data and make a prediction based on those labels, if the program has dubious labels, the predictions for unseen data will be wrong (Chollet [2018]).

The advantage of this supervised learning is that we can measure the success or accuracy more efficiently since we can determine if the algorithm's output is good or bad depending on how many of the predictions got right or wrong. The disadvantage of these models is that their supervised part could be highly time-consuming since it needs to be fed manually and accurately tagged data to learn patterns.

There are some problems we see nowadays that would be impossible to manually label all data since it could be created at a tremendous rate, or the labelling needs a level of expertise that only a few experts could do, which then creates a bottleneck that could affect the performance of the whole process (Lazzaro [2021]).

For these types of problems, ML algorithms evolved into unsupervised or semi-supervised learning, which uses the disadvantage of having a massive amount of unlabeled data in its favour by using it to uncover patterns and relationships outside what a label could provide. These models are not trained to find the correct answer or prediction, so they are based on circumstantial evidence to find patterns independently (Géron [2020]).

Many machine learning algorithms are powerful tools for analyzing and digesting complex data. However, like every other problem we try to solve that involves data, the way these algorithms are different in their performance is how each algorithm learns, what type of data is needed, and how much of this data the algorithm will need to perform well.

Despite the many advantages of machine learning algorithms, their performance is based on factors such as the learning approach, the type of data required, and the amount needed for optimal performance. Even the unsupervised kinds depend on human intervention to perform correctly since we determine the labels we

need to predict or the hierarchy of the most important features the algorithms need to understand (e.g., determining labels or establishing the hierarchy of essential features); they require more structured data. Deep learning comes into play to tackle problems that involve unstructured data and require robust solutions.

Deep learning can be interpreted as a subset of machine learning since it is differentiated from classical or non-deep machine learning and deep machine learning. In this case, deep learning can but does not necessarily need labelled data to be trained. Going back to what we mentioned earlier, the algorithms we use to analyze data are dependent on the type, amount, and complexity of the data we have, and deep learning is primarily used for the most complex cases, such as problems that require answers in real-time, such as fraud detection or virtual assistants, to the most challenging issues we have nowadays like computer vision for self-driving cars, or tools of pattern and image recognition for medical diagnosis (Chollet [2018]).

1.2 PROBLEM STATEMENT

The advancement of artificial intelligence algorithms, particularly in deep learning, has significantly impacted fields such as finance, healthcare, and government by enhancing performance and automating labour-intensive tasks. Although many sectors strive to emulate the remarkable successes achieved through these computational tools, the applicability of algorithms substantially depends on the specificity of the data and issues at hand. While trending in numerous applications, deep learning models are only sometimes applicable due to their data and feature specificity. Specifically, image recognition in computer vision has notably benefited from deep learning, achieving unparalleled flexibility and performance through classic and deep learning methods, with neural networks emerging as a popular model.

One field that was attracted to the success of using neural networks for image recognition is the medical diagnosis field. Their main problem is the massive amount

of data they must process daily. If we compare the number of experts available to use and analyze all of the data, we come up with a bottleneck that affects the care and performance of the diagnosis of patients. This time loss becomes particularly burdensome in the medical field of radiology since cancer diagnosis relies on early detection to give the patient better treatment.

Given the necessity of applying the best tools to aid in the already existing computational tools that the experts use, there have been many attempts to implement these kinds of image recognition deep learning algorithms to improve their performance. However, in this urgency, many of the features required for a reliable and accurate model have been glossed over, resulting in dubious or non-replicable results.

This is not to say that these studies have no merit since they are all based on current methods for deep learning in the computer vision field. In the area of medical diagnosis, there needs to be a different standard in how we conduct our experimentations so as not to create false expectations on the result of our models and give them the best accuracy we can get.

In a more focused approach, we studied and compared the results of using the state-of-the-art tools for computer vision used in computational-aided diagnostics with benchmark data sets and real clinical data sets from a Mexican hospital. One of the main concerns we found in our literature review is that aspects of computer vision need to translate better to specific medical imaging problems, such as annotated images or data set diversification, especially when working with limited resources, which often happens in developing countries.

To overcome that, we explore the capabilities of artificial intelligence tools and not only trap ourselves inside the box of deep learning algorithms. In this work, we highlight the symbiosis that exists across all areas of artificial intelligence, which includes machine learning, deep learning, and statistical analysis, to solve such a complex problem as medical image analysis with limited resources. In this

instance, we focus on using mammography as the data set. (See Appendix A for further information on the data set)

Our hypothesis is that we are going to be able to find a balance between the use of deep learning mixed with traditional machine learning, and the current available resources, such as computational capabilities, and sufficient annotated data to design a model that can improve previous classification and segmentation experimentations.

Looking to identify such a balance, we also aim to do a comprehensive comparison between what are the main differences, advantages and disadvantages of what the state-of-the art instructs, and how they differ from the findings we made while experimenting with different methodologies. Another goal is to add to the balance what are the different paths we can consider in the realm of Artificial Intelligence when we encounter constraints that are not detailed on other works when computational time and resources are not a issue.

STRUCTURE OF THE THESIS

In the Introduction, we mentioned the progression and importance of all the fields in artificial intelligence. In Background (Section 2), we discuss the differences and requirements for each algorithm mentioned in the Introduction. We see the benefits and precautions we need to take when using each one and how we can determine the best fit for our data.

In Literature Review (Section 3), we explore the use of these algorithms in computer vision and medical diagnosis; we compare their benefits and outline the differences we need to consider when solving problems in the medical field. In Methodology (Section 4), we outline the steps and methods we deem appropriate for the situation we are trying to solve: to improve the classification and diagnosis of mammography by applying a combination of machine learning and deep learning algorithms. Results (Section 5) show the output of the stages mentioned in the methodology section. Since our goal is for these tools to be used in a real-life medical setting, we divide the methodology and results into three stages. They can be

separated so the experts can use our tools in the order they need.

In Conclusions (Section 6), we highlight the advantages and disadvantages of using the different methodologies we mention in our work, emphasizing the type of data we use. Finally, in Future Work (Section 6.2), we outline the proposed steps to follow our methodology once the data and resources become available.

CHAPTER 2

BACKGROUND

In this chapter, we review in depth the different artificial intelligence tools that we use for our methodology, explaining their most essential features, how to evaluate their performance, and the pros and cons of using them with different types of data, making emphasis on their use on computer vision and medical imaging fields.

2.1 MACHINE LEARNING

While deep learning, especially with CNNs, has demonstrated exceptional performance in computer vision tasks, particularly with large-scale data sets, traditional machine learning algorithms can still be valuable alternatives in specific scenarios, depending on the available data, interpretability requirements, resource constraints, and feature engineering considerations.

Deep learning models often require large amounts of labeled data to train effectively. Traditional machine learning algorithms can be a practical choice if the dataset is small or limited. Techniques such as Support Vector Machines (SVM), Random Forests, or k -nearest neighbors (k -NN) can still yield good results with smaller data sets. Some machine learning algorithms can provide more interpretable models than deep learning, so they are sometimes preferred in certain fields, such as

medical diagnosis (Chugh et al. [2021]).

Deep learning algorithms are known for their high computational requirements. Certain machine learning algorithms are more practical and efficient when faced with limited computational resources. Traditional machine learning approaches often involve feature engineering, which means extracting relevant features from raw data using domain knowledge. The presence of specific domain knowledge in a problem can significantly influence the model's performance.

This first section lists some of the more common traditional machine-learning algorithms for classification problems. In the methodology section, they are often used with their standard parameters, so if more detail is needed, there are books for traditional ML methodologies, such as works by Geetha and Sendhilkumar [2023a], Chopra and Khurana [2023], Ghosh and Math [2023] which go in-depth on these subjects.

2.1.1 LOGISTIC REGRESSION

Logistic Regression (Menard [2010]) is a popular algorithm for binary classification. It models the relationship between the data features and the binary target variable using the logistic function, which maps the input to a probability. It is known for its simplicity, interpretability, and efficiency.

The logistic function is determined by Equation 2.1 and looks like Figure 2.1. This equation forces the output to assume a value between 0 and 1. In the final result, a threshold of 0.5 is common for each category. When we have a binary categorical feature, it is easy to interpret the result as one end being closer to 0 and the other being closer to 1. When it comes to problems of more than two categories, the problem uses one hot encoding, meaning each column or feature gets its category:

$$\text{logistic}(n) = \frac{1}{1 + \exp(-n)}. \quad (2.1)$$

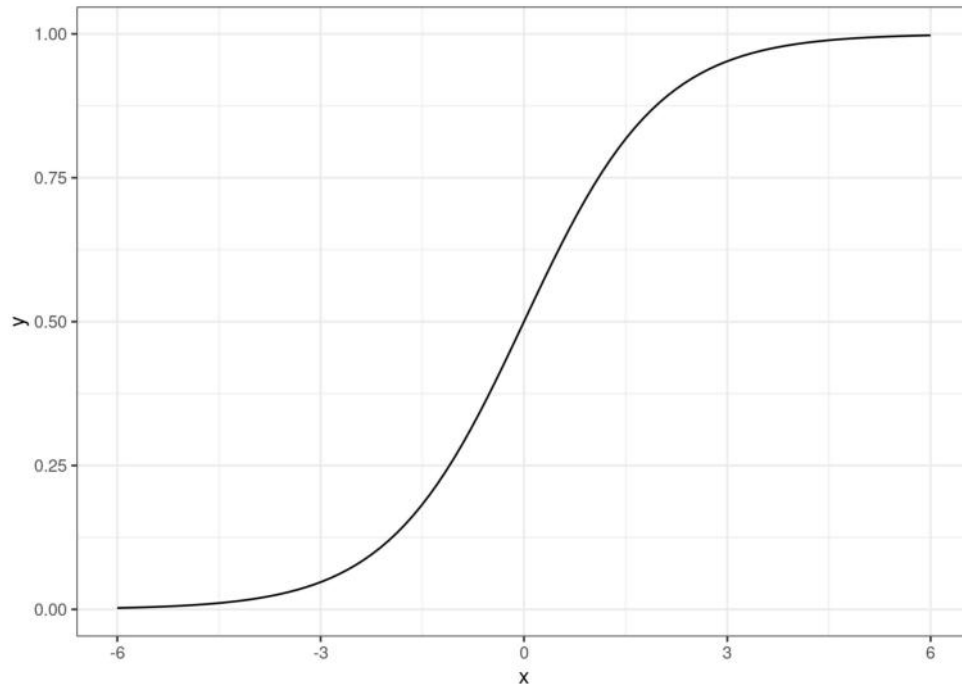


Figure 2.1: Behaviour of the logistic function. It outputs a number between 0 and 1.

The logistic regression model has the main advantage of its simplicity. It does not require extensive computational resources to be trained quickly. The model also provides interpretable results, as the coefficients associated with each feature can be interpreted as the impact of that feature on the probability of the target class. If we have a class with 80% impact, it would mean it has more significance than the probability of a class of 50%.

Finally, logistic regression tends to be less susceptible to overfitting when working with limited data, making it an excellent choice when we have small to moderate size data sets.

One disadvantage of this model is that it assumes a linear relationship between the independent variables and the log odds of the target class. It may not perform

well if the true relationship is non-linear. It also assumes that the features are not dependent on one another. If there are strong correlations or interactions among the features, it may lead to biased coefficient estimates (Singh [2020]).

Logistic regression is also designed to solve binary classification problems. Although it can be extended to handle multi-class problems (e.g., through one-vs-rest or multi-nomial logistic regression), it may not be as straightforward or effective as other models specifically designed for multi-class classification (James et al. [2013]).

2.1.2 SUPPORT VECTOR MACHINES (SVM)

SVM is a very powerful algorithm that searches for an optimal hyperplane to separate data points of different classes in a high-dimensional feature space. It aims to maximize the margin between the classes while minimizing classification errors. SVMs can handle linear as well as nonlinear classification tasks using kernel functions. These kernel functions are a strategy for dealing with scenarios where the separation of groups is non-linear and involves expanding the dimensions of the original space. An example is illustrated in Figure 2.2, where two groups are initially not separable by a linear boundary. However, introducing an additional dimension makes their separability apparent (Chollet [2018]).

This algorithm can be used for several tasks and is regarded for its high performance, efficiency, and flexibility. In some cases, using SVM has simplified the application of other machine learning algorithms, for example, combined with CNN, where the network is used to extract features and the SVM acts as the binary classifier (Ahlawat and Choudhary [2020]).

An interesting feature that we explore with this model and hyperparameters of neural networks is the regularization or penalizing feature of the concept of “maximum margin classifier” that uses a similar loss function component known as hinge loss, which can also be applied to loss-functions like cross-entropy in binary classi-

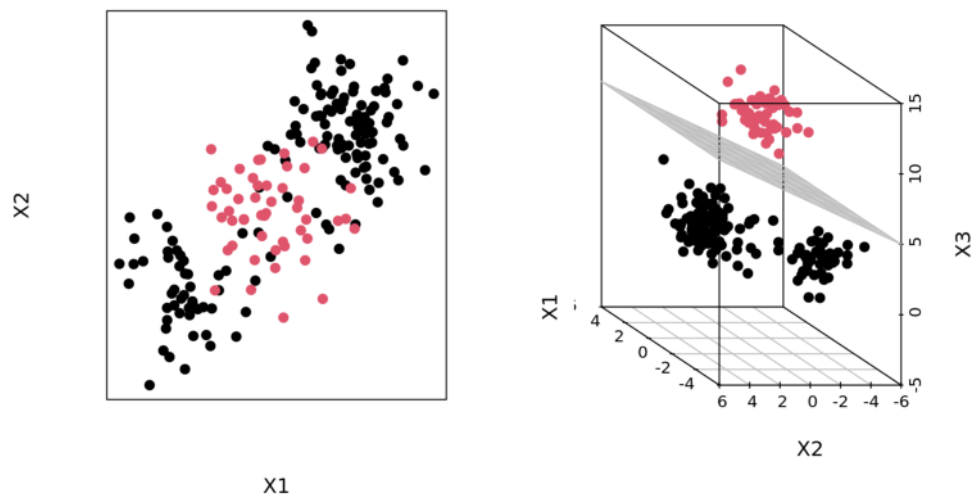


Figure 2.2: Use of kernels in support vector machine to maximize the distance between classes. On the left, we see the representation of a 2D kernel, and on the right, a 3D kernel. The complexity of the SVM goes up exponentially with each kernel and becomes difficult to illustrate graphically.

fication problems within neural network training. Sometimes, hinge loss could be beneficial when dealing with imbalanced data, as it focuses on the hardest-to-classify examples (those near the decision boundary) rather than all data points. In some cases, combining hinge loss with other regularization methods could potentially stabilize and improve training in neural networks (Hastie et al. [2009]).

One of the key advantages of SVM is its ability to utilize diverse kernel functions, including linear, polynomial, radial basis function (RBF), and sigmoid. This flexibility enables the capture of various relationships between features and the target variable. Additionally, SVM excels in handling non-linear decision boundaries by effectively utilizing kernel functions. By mapping the original features to a higher-dimensional space, SVM can effectively separate classes that are not linearly separable in the original feature space (Logunova [2022]).

Another advantage of SVM is its ability to perform well without overfitting, as

it can generalize effectively even when the amount of data is small. It is important to note that the suitability of SVM depends on the specific problem and data set since this algorithm can require significant computational resources and time when dealing with complex or large data sets. It also requires more manual tuning for optimal performance since several parameters greatly affect the outcome, such as kernel, kernel parameters, and the regularization parameter. Improper parameter selection can lead to sub-optimal results (Mammone et al. [2009]).

2.1.3 DECISION TREES

Decision Trees are versatile and intuitive algorithms that partition the feature space based on a series of decision rules. They create a tree-like structure where each of the internal nodes represents a decision based on one of the features, and each of the leaf nodes represents a class label. Decision Trees can handle both categorical and numerical features and are easily interpretable. This algorithm makes predictions based on how the previous “questions” were answered that contain the desired categorization of the data, which makes it a form of supervised learning (IBM).

The decision-making structure of the decision tree algorithm reflects its name, resembling the shape and behavior of a tree, which is why the terms used to explain them include relevant concepts. Figure 2.3 represents the schematic diagram of the different terminology for this model, where the root node represents the starting point where the decision tree begins to divide the data, while the leaf nodes signify the outcomes with no further separation. Parent and child nodes refer to the relationship between nodes, where the initial node is the parent, and subsequent nodes are its children. Splitting, sub-tree construction, and pruning are techniques employed to enhance the tree’s performance and, in some cases, reduce its size without compromising accuracy (Bansal et al. [2022]).

An important attribute of this model is the Gini index, which measures the

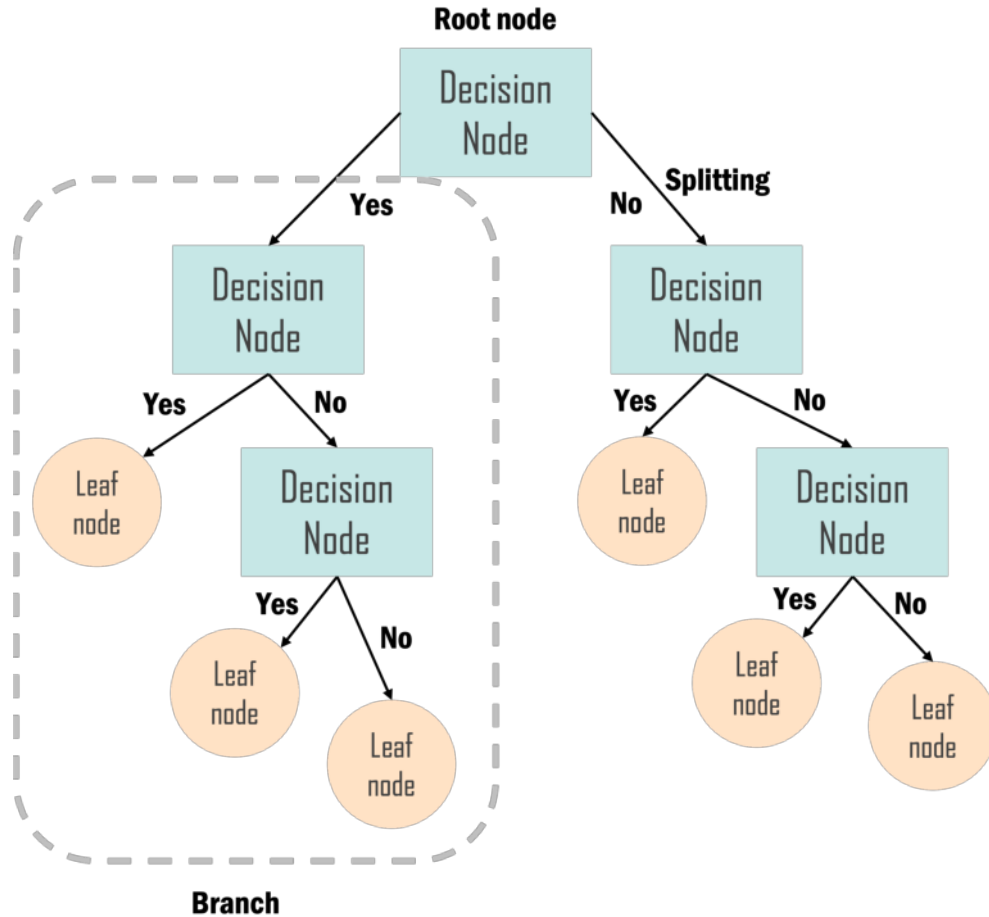


Figure 2.3: Schematic diagram of the different terminology for decision tree. Illustration inspired by EdX.org [2023].

impurity used during the creation of the tree algorithm. It helps determine the quality of a split when constructing the decision tree.

The Gini index calculates the probability of misclassifying a randomly chosen element in a node if it were randomly labeled according to the distribution of the classes in that node. It ranges from 0 to 1, where a Gini index of 0 indicates that the node is pure (all elements belong to the same class). A Gini index of 1 indicates maximum impurity. A lower Gini index indicates a more homogeneous distribution of classes after the split, which is the desirable outcome (Singh [2023]).

Decision trees provide intuitive and interpretable models for measuring the

importance of features in the classification process. By examining the tree's structure and the feature's placement, you can assess the importance of different features in decision-making. They can capture complex decision boundaries without the need for complex transformations or feature engineering and are far less sensitive to outliers than other models for linear regression (Bansal et al. [2022]).

The main disadvantage of this algorithm is that they can learn intricate patterns in the training data that may not generalize well to unseen data, leading to poor performance on test data and overfitting. They are also prone to bias towards features with many categories, as they can create more specific rules. It is important, then, when using this algorithm, to consider the various regularization techniques and ensemble methods such as random forests, gradient boosting, and pruning. (Castillo [2021]).

2.1.4 NEURAL NETWORKS

Deep learning models, particularly Artificial Neural Networks (ANNs), are increasingly popular for binary classification. ANNs consist of multiple layers of interconnected nodes (neurons) that learn complex representations of the input data. With the advancement of deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), they have shown remarkable performance on various classification tasks (Chollet [2021]).

Neural networks have gained significant popularity in machine learning applications based on their ability to model complex relationships in the data. They can adjust their internal representation based on the data they are trained with to improve their performance over time. They are a great tool when handling large-scale problems and big-data applications since they generally achieve better performance when they have more training data to use. Their success garnered many topics of investigation, leading up to deep learning and more complex architectures (Aggarwal

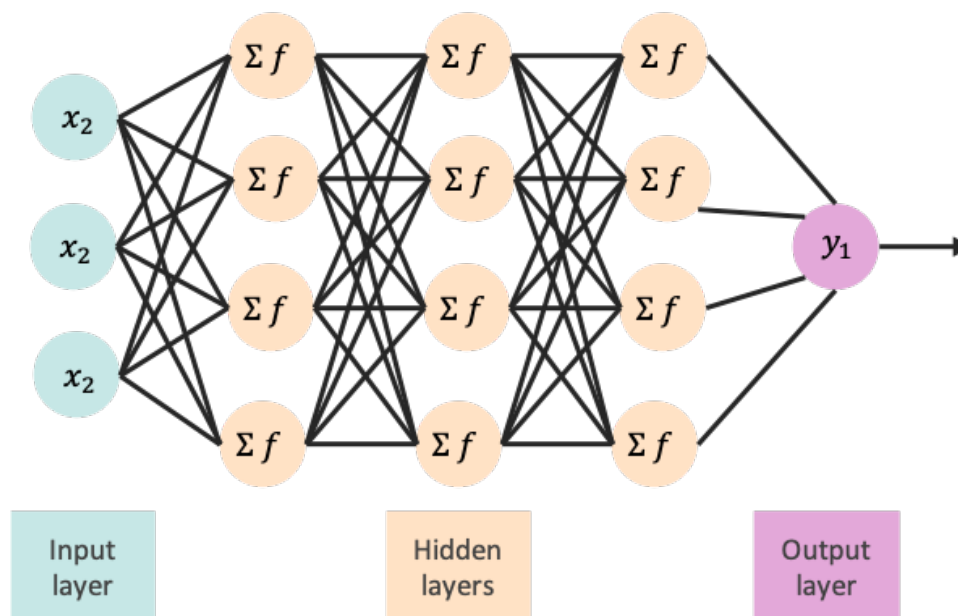


Figure 2.4: Schematic diagram of the structure of a common neural network. As it stands, we have input, hidden, and output layers. The connections between these layers have different hyperparameters, such as activation functions and weights.

[2019]).

However, one of the main drawbacks of neural networks is that they can be computationally expensive since they require substantial computational resources to compile a problem and need a large amount of training data to generalize well. If they do not have access to a big data set, they can be prone to over-fitting, where they become overly specialized in the training data and perform poorly on unseen examples. Regularization and careful hyperparameter tuning are necessary to mitigate this effect (Ng [2019]).

Furthermore, neural networks have a black-box nature, so it is difficult to interpret, understand, and explain their internal workings. This poses challenges when insights into the decision-making process are required. Considering these advantages and disadvantages in the context of specific problems, data sets, and other factors is important. While neural networks are powerful, they may not always be the best choice for every machine learning task (Géron [2020]).

2.1.5 k -NEAREST NEIGHBOR

It is a machine learning algorithm commonly used for classification tasks, including binary classification. The basic principle of k -NN is to classify a new data point by comparing the class labels of its nearest neighbors in the training data set. However, it is a non-parametric algorithm that does not make explicit assumptions about the underlying data distribution. It is a relatively simple and intuitive algorithm but can suffer from performance issues with large data sets, as the computation of distances can be computationally expensive. Additionally, selecting an appropriate value for k and handling imbalanced data sets are considerations to keep in mind when using k -NN for binary classification (Harrington [2012]).

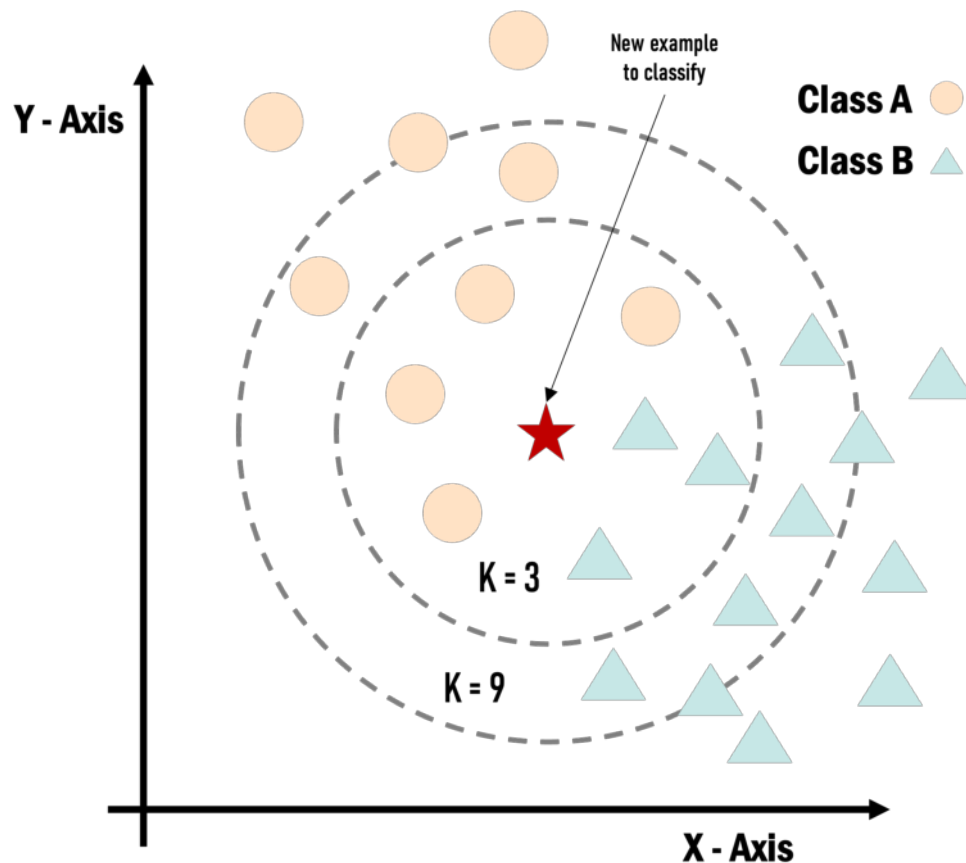


Figure 2.5: Example of how k -NN classifies a new data point based on the different k chosen.

This offers many advantages in the field of machine learning. Since it is a non-parametric algorithm, it does not assume any specific data distribution, which allows it to handle various data types and adapt well to different problems.

It also has the ability to capture local patterns in the data. While it is primarily used for classification and regression tasks, it can also be used in image segmentation to some extent. However, it is not a widely used algorithm for this task since image segmentation often requires capturing complex relationships and boundaries between regions of interest, which can not be effectively captured by a k -NN algorithm alone (Harini and Chandrasekar [2012]). So, while it can be applied to image segmentation, it may not provide the most optimal results compared to more advanced techniques tailored for this task, such as convolutional neural networks or graph-based methods.

The curse of dimensionality is also a known challenge with k -NN; as the number of dimensions increases, the distance between instances becomes less informative, and the algorithm may struggle to find meaningful neighbors. This can degrade its performance in high-dimensional feature spaces.

Lastly, k -NN is often considered a black box algorithm since it does not explicitly explain its predictions. Understanding and interpreting the decision-making process can be challenging, especially when many neighbors are used (Hu and Sejdinovic [2021]).

2.1.6 RANDOM FOREST

Random Forest is one of the most commonly known ensemble learning method. This algorithm combines multiple decision trees to improve their predictions. It is formed from a set of decision trees on different random subsets of the training data and aggregates their predictions. Random Forests offer improved generalization, robustness against overfitting, and the ability to capture complex relationships in the data (Shaik and Srinivasan [2018]). Like Figure 2.3, Random forest follows the

same terminology as shown in Figure 2.6. The main difference here is the ensemble behavior represented as multiple decision forests.

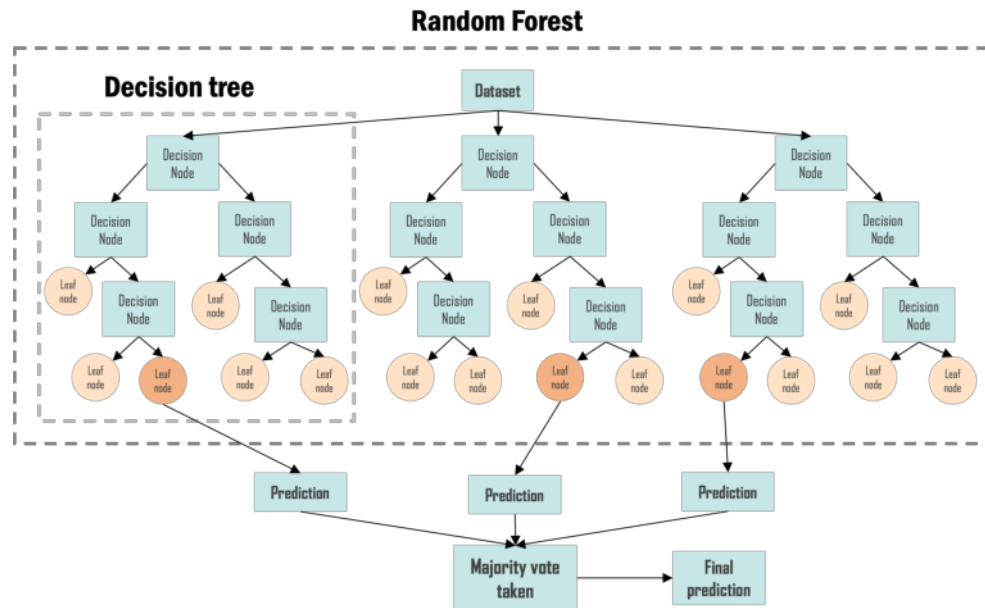


Figure 2.6: Schematic diagram of a random forest model.

Random forest algorithm is considered an ensemble method combining multiple decision trees, averaging its decisions, and reducing the impact of individual tree errors. As such, it follows the same performance metric, the Gini. This often allows the algorithm to garner a higher accuracy than a single decision tree. Combining predictions from multiple trees can capture a broader range of patterns and reduce overfitting (Slimani et al. [2023]).

Random Forest offers improved accuracy, robustness, and feature importance analysis compared to a single decision tree. However, it sacrifices some interpretability and computational complexity due to the ensemble approach. Random Forest is more suitable when higher accuracy and generalization are required, while decision trees can provide simpler, interpretable models.

Regarding computer complexity, training this model requires storing multiple decision trees in memory (often RAM), which can exhaust our resources if the problem is complex. Training and prediction times can be longer, although parallel

processing can handle this if used correctly (Cloud [2023]).

2.2 STATISTICAL LEARNING

Finally, let us explore the profound influence of statistical learning across the various domains of Artificial Intelligence discussed earlier. Statistical learning serves as a unifying factor, playing a critical role within deep learning, a subfield of machine learning that focuses on training ANN with multiple layers. While deep learning models excel at learning intricate patterns and representations from vast volumes of data, enabling great advancements in image recognition, natural language processing, and speech recognition, comprehending the specific aspects in which statistical learning contributes becomes essential for a comprehensive grasp of the underlying theoretical foundation.

By delving into the involvement of statistical learning in deep learning, we can gain valuable insights into the fundamental principles that govern the tuning of these models to achieve enhanced performance without compromising the accuracy of the outcomes. Through statistical learning, we are equipped with many techniques and concepts that aid in data preprocessing, selecting appropriate loss functions, optimizing model parameters, utilizing regularization techniques, evaluating model performance, and even hypothesis testing and model comparison. These statistical learning components seamlessly integrate with the complex machinery of deep learning, enabling us to extract meaningful insights, make accurate predictions, and unlock the full potential of artificial neural networks.

In essence, statistical learning acts as a powerful ally within deep learning, empowering us to navigate the intricacies of training deep neural networks and empowering them to unravel complex patterns hidden within vast datasets. By harnessing the synergy between statistical learning and deep learning, we pave the way for groundbreaking advancements and foster innovation across various domains,

propelling Artificial Intelligence to new frontiers of understanding and capability.

2.2.1 DATA PRE-PROCESSING

In the realm of deep learning, statistical learning techniques are commonly employed for data preprocessing purposes. This thesis focuses on applying statistical learning in the context of medical imaging. It is important to note that certain computational tools cannot be directly applied to medical images in the same way they are used with typical data, as medical images require a higher level of specificity and detail. The same holds true for statistical learning methods.

When considering a medical image dataset, statistical learning techniques are still involved in the preprocessing stage of deep learning, but there are some specific considerations to address the unique characteristics of medical images:

- **Data Normalization:** Statistical learning techniques are used to normalize medical image data. However, normalization may involve more specialized techniques tailored to the imaging modality in the case of medical images. For instance, techniques such as intensity rescaling or histogram equalization are commonly employed to ensure consistent intensity ranges across images obtained from different scanners or protocols.
- **Handling Missing Data:** Missing data in medical images can occur due to various factors, such as imaging artifacts or incomplete scans. Statistical techniques, specifically developed for medical imaging, can be used to handle missing data. These techniques may involve sophisticated interpolation methods or advanced image reconstruction algorithms to fill in the missing information.
- **Preprocessing Techniques for Specific Modalities:** Different medical imaging modalities (e.g., MRI, CT, ultrasound) have their unique character-

istics and preprocessing requirements. Statistical techniques are utilized to address these specific needs. For instance, spatial registration techniques may be applied to align images from different time points or modalities. Noise reduction algorithms may also be employed to enhance image quality while preserving important features.

- **Data Augmentation and Variability:** Data augmentation techniques specific to medical images are used to increase the size and variability of the training dataset. These techniques can include deformations, rotations, or intensity transformations that are relevant to medical imaging. It is important to apply these augmentations while preserving anatomical accuracy and clinical relevance.
- **Statistical Analysis and Visualization:** Statistical techniques are crucial in analyzing and visualizing medical image datasets. Descriptive statistics, such as mean, variance, or texture features, can provide insights into the data characteristics and guide preprocessing decisions. Statistical visualization methods, such as heat maps or scatter plots, can help identify regions of interest or anomalies inside the images.

2.3 CONVOLUTIONAL NEURAL NETWORKS

Now that we have described the different methods of artificial intelligence that range from statistical learning to machine learning, we discuss the most popular one for image classification and segmentation—Convolutional Neural Networks.

When describing the advancements of convolutional neural networks (CNN), we follow similar examples of how we described the development of artificial intelligence and computer-aided diagnostics. In the case of CNN, their early architectures were linear and simple, following a straightforward structure of layers. So to ex-

plain the functioning of a CNN, we start with the example of the first popular CNN architecture, the LeNet-5. But first, we explain the essential parts of a CNN.

- Convolutional layer:** This layer applies a convolution filter or kernel to the image to detect its features. Mathematically, this kernel is represented by a matrix of weights, as shown in Figure 2.7. We begin with the original or input image; the user determines the size of the kernel, but the numbers inside are usually random at the beginning of the process. This kernel will “slide over” the image, where at each position, we multiply the kernel by the element of the pixels it covers and sum the results. The output will be a slightly smaller image with simple features. The numbers in the kernels are not usually programmed by hand but by random allocation and change in the course of the training process thanks to the optimization function; they serve as the first stage to build on early detected features to identify more complex shapes as the layers progress. The activation functions determine the adjustment of these kernels.

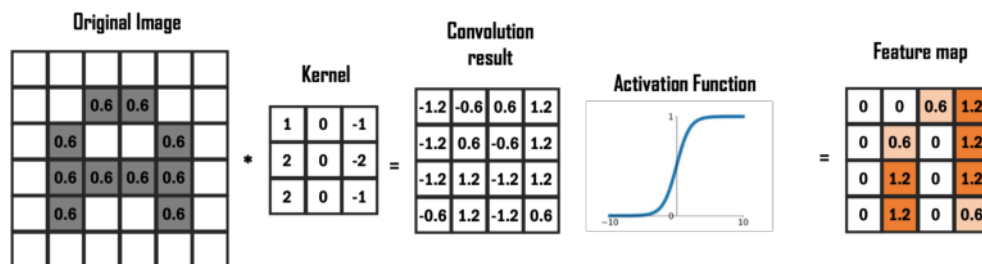


Figure 2.7: Simplified representation of the process and parameters inside a convolutional layer.

- Pooling layer:** In a structure similar to the convolutional layer, the pooling layer is responsible for reducing the features of the image, decreasing the computational power required to process the images, and extracting the dominant features. It is a common practice to always put a pooling layer right after one or two convolutional layers, though it is not mandatory for a good performance. It all depends on the complexity of the images. There are two types of

pooling: max-pooling and average-pooling, with max-pooling being the most commonly used. Figure 2.8 shows an example of the differences between each type of pooling. In this case, max pooling also has the advantage of acting as a noise suppressant, as it discards unimportant activations with dimensionality reduction. Although simple, this process reduces the network's complexity and helps improve its efficiency. The only drawback of this reduction is that it also results in information loss.

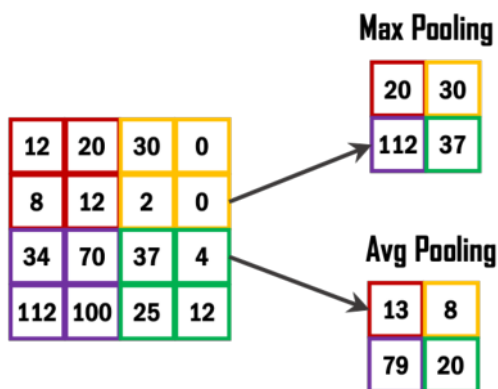


Figure 2.8: Representation of the max pooling and average pooling layer process.

- Dense layer:** We call the dense layer the last connection of the network since it holds the fully connected layer, the flattening, and the output layers. In Figure 2.9, we can see how the process starts with the receiving input of the feature extracted map, created by the convolution and pooling layers. This fully connected layer is considered a very easy and inexpensive way to learn non-linear combinations of high-level features. After going through this layer, we have the flattening layer, where the image is arranged into a column vector to feed it forward to the output nodes by a loss function. Still, this function is applied to every iteration of training and helps correct the model's performance, depending on the classification task.
- Activation functions:** An activation function decides how the weighted sum of the inputs will be transformed from one layer to the other. This parameter

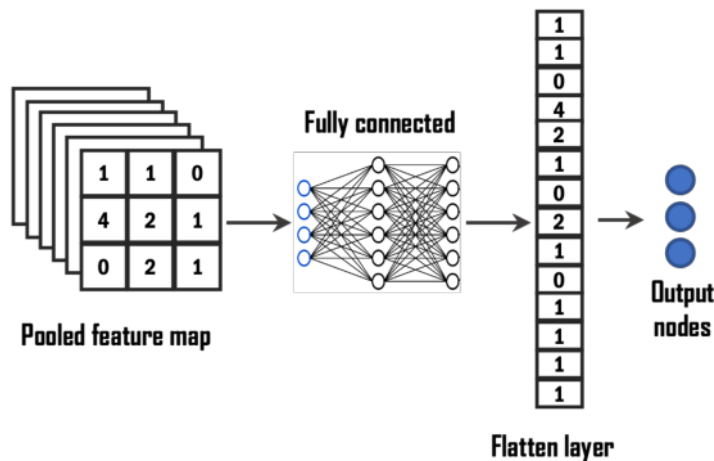
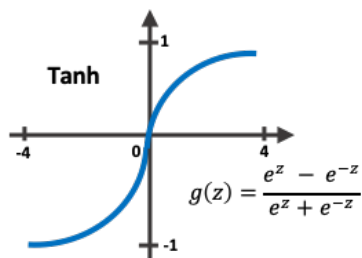
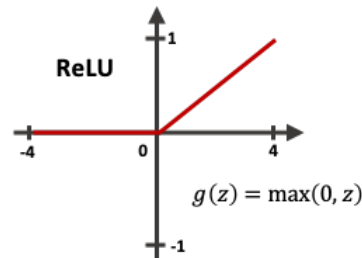


Figure 2.9: Representation of the process of the dense layer, which includes the pooling layer, fully connected layer, flattening, and output layer.

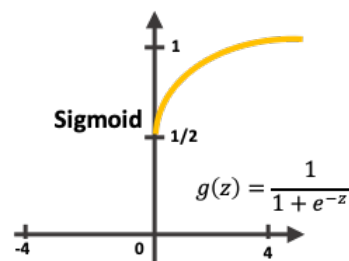
is very important to decide since it greatly impacts the neural network's performance. Different activation functions may be applied to different parts of the network. Typically, all hidden layers in the network will use the same activation function, and the output layer will have a different one corresponding to the classification process we need. Figure 2.10, illustrates the most common ones. We have in Figure 2.10b the rectified linear activation and its variant, leaky ReLU. Both are very popular for hidden layers since they are less susceptible to vanishing gradients. A very simple way of seeing this function is that when the input value is negative or 0, the value 0 is returned. Any other positive value is returned as is. Another function is the Sigmoid or logistic function. This one takes the input and output values from 0 to 1. This function is common for the dense layers since the output helps determine which class the results are more aligned with. The more positive the response, the closer it is to 1; the more negative, the closer it is to 0. Finally, the hyperbolic tangent activation function, or simply Tanh. This is very similar to the sigmoid function, only in this one, the input and output values range from -1 to 1. For the recommendations on which activation to choose in the hidden and output layers, we have a representation in Figure 2.11 taken from a web page



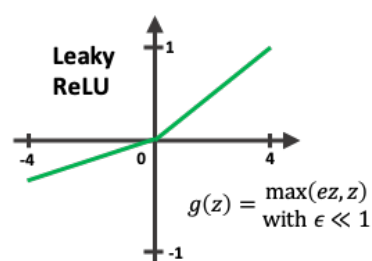
(a) Tahn activation function.



(b) ReLu activation function.



(c) Sigmoid activation function.



(d) Leaky ReLu activation function.

Figure 2.10: Representation of the most common activation functions for neural networks.

where we can learn more about the depth of each function (Brownlee [2021]).

There are other important hyperparameters that we need to mention. For neural networks, especially convolutional networks, tuning hyper-parameters is called searching for the optimum values for the model architecture. This performance enhancement varies from problem to problem, and there is no drawn-out path we can choose for these parameters, so they are often picked out with experience and trial and error.

- **Dropout:** One of the main problems to avoid when working with convolutional neural networks is the overfitting of the model. This happens when the model we use is too complex or the data set does not have enough training examples. For this, we use regularization methods, such as dropout. This parameter follows the behavior of the neural connections in the brain of a human. The more a connection between two neurons activates, the more robust it becomes.

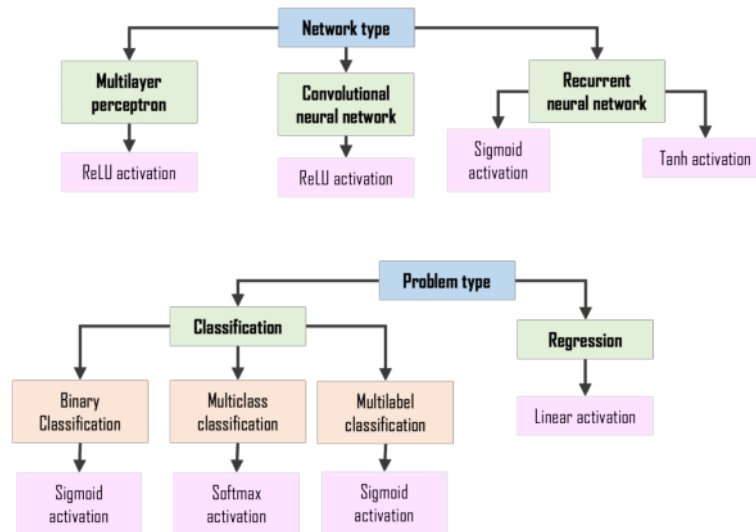


Figure 2.11: How to choose the activation function for the hidden layer based on the network type and for the output layer based on problem type.

Dropout can be used as a threshold, where set, can ignore or “drop out” some layer outputs, depending on the robustness of the connection between layers. This helps the network to learn a sparse representation as a consequence. Like pooling and other normalization strategies, dropout needs to be used carefully since one of the side effects is how it reduces the network’s capacity.

- Optimization function:** The optimization function helps the network improve the accuracy of their performance and training speed. When explaining the convolution and pooling process, we mentioned that the kernels used in those tasks start at random, and depending on the optimization function we select, it modifies the weights and aims to minimize the loss function for the dense layer. We make reference here to our master’s thesis, where we explain in detail all the different optimization algorithms available for deep learning, their advantages and disadvantages, and performance for our desired data set (Berrones-Reyes [2019]). In this work, we employ the Adam optimization algorithm. It is a recommended default optimization algorithm for many benchmark deep learning papers. It has low memory requirements, needs less tuning than any other optimizer, and has a faster running time in training. We refer

to the original work of Kingma and Ba [2014] for a more in-depth description of its properties and application.

- **Batch size:** It is a number that needs to be selected at the beginning of training since it is the number of samples the epoch will take to update the model parameters. Same as all other hyperparameters, the batch size can not be a fixed number for all problems, but it is suggested to consider the hardware capabilities for the experimentation. The bigger the batch size, the more computationally expensive it is.
- **Epochs and iterations:** The iterations is the number of batches the algorithm will execute, while the epochs are the number of times a learning algorithm sees the complete data set. The iterations are linked with the batch size since it can run out of data if we allocate a small batch number and a big iteration number. Epochs are usually the way we can see improvement in our performance. Too few epochs will result in underfitting, while too many will result in overfitting. To avoid this, there are “callbacks” that help reduce dynamically the number of iterations in each epoch.
- **Loss function:** In a convolutional neural network (CNN), the loss function is a crucial component used to measure the discrepancy between the predicted output of the network and the true or expected output. The goal of the loss function is to quantify the model’s performance by assigning a numerical value that indicates how well or poorly the network is performing on a given task. The choice of an appropriate loss function depends on the nature of the problem being solved. Specific loss functions are commonly used for tasks such as image classification, object detection, or semantic segmentation. Some examples are categorical cross-entropy, binary cross-entropy, mean squared error, etc.
- **Modules:** When working with CNN architectures, a problem arises when the network can no longer propagate useful information from the output layer to the layers from the input. This will mean that the weights can not be up-

dated, so the network will not learn, and the performance will decrease. Some more complex networks tackle this problem with modules, like the ResNet50. ResNet-50 can create a very deep network architecture by stacking multiple residual blocks together. The network is typically divided into stages, each containing a different number of residual blocks. The number of blocks per stage depends on the complexity of the problem being solved. The skip connections in ResNet-50 allow the network to effectively propagate gradients through the layers, enabling the training of very deep networks. This, in turn, helps capture more complex and abstract features from the input data, leading to improved performance in tasks such as image classification or object detection.

Having discussed the most common terms in convolutional networks, we describe the most common architectures known in the literature.

2.3.1 LENET-5

The LeNet-5 architecture of the original article by LeCun et al. [1998] is shown in Figure 2.12. In this case, and all the architectures we will discuss, we represent the convolutional layer as the red box, the gray as the pooling operation, the blue box as the dense layers, and the information inside is the actual operation of strides they use. Additionally, the yellow circles represent the different activation functions, and the white circles the other normalization functions available. We can tune many other hyper-parameters to help the network's performance, but these representations will help us to understand their complexity better.

As mentioned, the LeNet-5 architecture is the simplest one, with the standard or traditional architecture we can expect from CNN-convolutions with activation functions, pooling layers, and fully connected layers. The authors first proposed this now-familiar architecture to solve the problem of recognizing handwritten and machine-printed characters. Its simplicity allowed the hardware of the time to

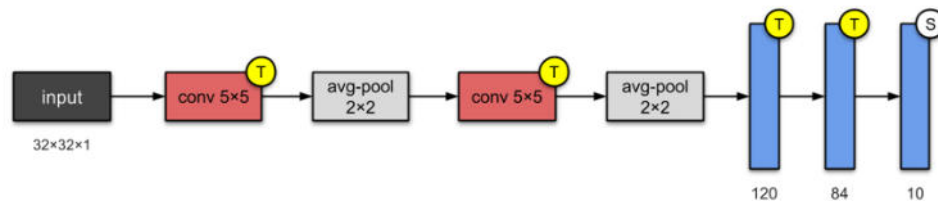


Figure 2.12: Compact visualization of the LeNet-5 architecture (from Karim [2022]) that follows the original architecture by (LeCun et al. [1998]).

achieve great results. LeCun et al. [1998] introduced the concept of automatic pattern recognition, removing the need to choose hand-crafted feature extractors. The comparison was between this newly crafted architecture and traditional machine learning and statistical learning algorithms, such as k nearest neighbor, principal component analysis, support vector machines, etc.

Even though the performance from LeNet-5 was very impressive, the comparison between the other machine learning algorithms was not out of the ordinary, when in some cases, algorithms like support vector machine could surpass it. At the time, the resources needed to train and model a CNN were still highly costly, making neural networks' popularity fade by the end of the 90s.

2.3.2 ALEXNET

When the ImageNet competition arrived, numerous researchers had already explored alternative machine learning algorithms. However, it was in 2012 that another groundbreaking moment for deep learning occurred when AlexNet's architecture was introduced by Krizhevsky et al. [2012], leading to a decisive victory in the ILSVRC competition, as depicted in Figure 2.13. Comparing the architectural differences between LeNet-5 (Figure 2.12) and AlexNet (Figure 2.14), we observe that the linear structure and certain parameters remain consistent.

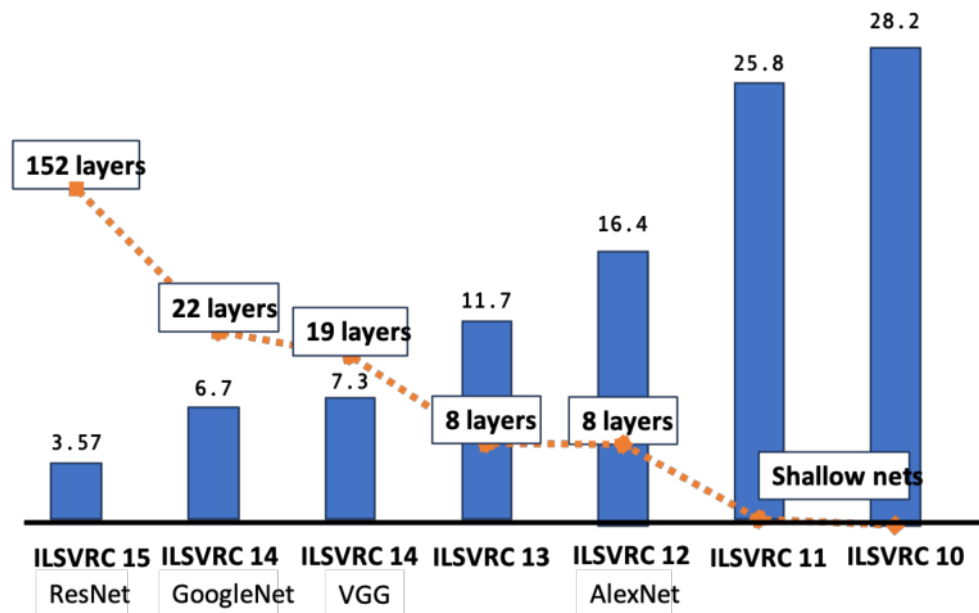


Figure 2.13: Error rate from the IMAGENET challenge, from 2010 (ILSVRC 10) to 2015 (ILSVRC 15). The blue bars represent the error rate, and the orange line represents the complexity incline from shallow nets to 152 layers.

Both models employ linear training, convolutional layers, pooling layers, and activation functions in a similar manner. Considering these similarities, it raises the question: Why did it take so long to adopt a superior architecture? The simplest answer lies in the availability of better resources.

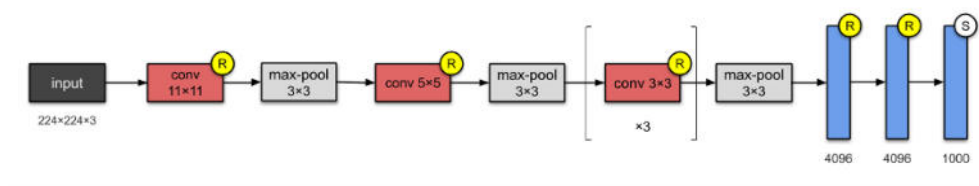


Figure 2.14: Compact visualization of the Alexnet architecture following the original architecture by Krizhevsky et al. [2017].

The key distinction between these two architectures lies in the increased parameter volume within the AlexNet architecture. This expansion of parameters became feasible due to the emergence of enhanced computational power, specifically GPUs, and the greater accessibility to a wealth of new data facilitated by the inter-

net. These advancements provided the necessary resources to scale up the parameter size, enabling AlexNet to achieve improved performance and leverage the growing availability of computational resources and vast datasets.

The concept of utilizing GPUs was not a novel idea, as parallelization had been introduced much earlier in 1958 by Cocke and Slotnick. They pioneered parallelism in numerical calculations, recognizing its potential and laying the foundation for future advancements (Sperling [2017]).

Utilizing modern GPUs for deep learning is a fascinating and intricate topic. In our methodology section, we provide a detailed explanation of how we harness the power of GPUs and the appropriate software to enhance our approach. For a more comprehensive understanding of the diverse capabilities of GPUs, we recommend exploring, for example, resources such as Dettmers [2023], Weka [2021], and Nvidia [2023].

Several additional innovations distinguished AlexNet from LeNet-5. AlexNet introduced the Rectified Linear Unit (ReLU) activation function, a departure from the previous reliance on hyperbolic tangent or sigmoid units. Furthermore, it incorporated normalization layers, which mitigated the risk of overfitting by accounting for the significantly increased number of parameters compared to LeNet-5. However, Dropout was the parameter that truly revolutionized the architectural framework, as it effectively prevents overfitting when used appropriately.

2.3.3 VGG-16

Following the linear category of CNN architectures, we have the VGG-16 model, more commonly known as VGG-16, which emerged after the AlexNet architecture. Introduced in the 2014 ILSVRC competition by Simonyan and Zisserman from the University of Oxford, VGG-16 features a configuration that includes 16 convolutional layers while still adhering to the convolutional and pooling layer pattern established

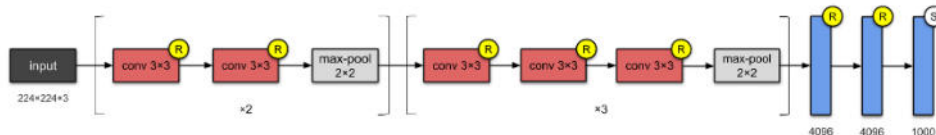


Figure 2.15: Compact visualization of the VGG-16 architecture following the original architecture by Simonyan and Zisserman [2014].

by AlexNet. At first glance, the increase from eight to 16 convolutional layers may not seem significant. However, the number of layers alone does not accurately represent the computational load imposed on the system. Table 2.1 illustrates the impact of doubling the number of layers on the parameter count, highlighting its substantial difference.

Table 2.1: Comparison of the parameters for each architecture, LeNet-5, AlexNet, and VGG-16 (M. Swapna and Prasad [2020]).

Year	CNN	Developed by	No. of parameters
1998	LeNet-5	LeCun et al. [2015]	60K
2012	AlexNet	Krizhevsky et al. [2012]	15M
2014	VGG-16	Simonyan and Zisserman [2014]	138M

VGG networks (16 and 19) introduced the concept of increasing the number of layers to enhance performance. Still, it is important to note that there are limitations to how many layers can be added. According to the study presented in its article, exceeding 20 layers hindered model convergence and introduced the issue of vanishing gradients. Among the various iterations of VGG nets, VGG-16 demonstrated superior performance, surpassing the achievements of the previous AlexNet model.

One significant drawback of the VGGNet architecture was its resource-intensive nature. Training the model required a considerable amount of time, typically lasting two to three weeks. The process relied on four Nvidia Titan GPUs (Potent but

expensive hardware), a dataset comprising 1.3 million images, and a substantial model size. This resource demand posed a notable challenge regarding computational requirements and training efficiency.

2.3.4 RESNET-50

Modern architectures now rely on skip connections or modules to reduce the number of parameters, effectively utilize computational resources, minimize training time, and enhance overall performance.

Architectures can establish direct connections between different layers by incorporating skip connections, also known as residual connections. This allows for bypassing certain layers, enabling the network to retain and propagate useful information more effectively. As a result, skip connections alleviate the burden of excessive parameters and computational demands, improving efficiency during training and inference (Adaloglou [2020]).

These skip connections offer several benefits. They enable the network to simultaneously learn low-level and high-level features, fostering better gradient flow and alleviating the vanishing gradient problem. Additionally, skip connections provide shortcuts for gradient propagation, aiding in training deeper architectures. Ultimately, these architectural enhancements contribute to more efficient and powerful models.

Understanding that adding more layers indiscriminately was not the answer to improving performance the next generation of convolutional networks focused on residual blocks and modules inside the architecture. So we have ResNet because the name comes from its nature of using Residual networks. We can observe in Figure 2.16 that the behavior of the architecture flows similarly to the ones mentioned before, but in this case, we have blocks that work differently than just adding more layers. The key component here is the skip connections (He et al. [2016]).

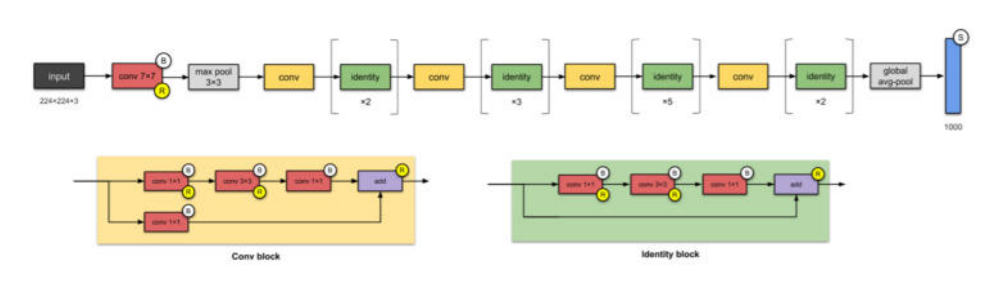


Figure 2.16: ResNet50 architecture, using a compact visualization by Karim [2022] following the original architecture by He et al. [2016].

They are designed to facilitate the training of deep neural networks by addressing the vanishing gradient problem and enabling the flow of gradients more effectively.

In a ResNet, the skip connections create shortcuts that allow information to bypass certain layers within the network. Instead of strictly following a sequential flow from one layer to the next, these connections enable the direct propagation of information from one layer to a subsequent layer, typically several layers ahead.

The core idea behind skip connections is to introduce residual mappings. Each skip connection aims to learn a residual function, which captures the difference between the desired output of a layer and the actual output. By using these residuals, the network can learn to refine the layer’s output rather than attempting to learn the entire transformation directly (Neurohive [2019]).

Mathematically, it denotes the input to a layer as x . The layer’s output can be represented as $H(x)$, where H represents the transformation applied by the layer. With a skip connection, the layer’s output becomes $F(x) = H(x) + x$, where F represents the residual mapping. The original input x is added to the transformed output $H(x)$ to create the new output $F(x)$ (Zhang et al. [2021]).

Introducing the skip connection enables the network to learn the residuals directly, making it easier for the model to optimize and train deeper architectures. These connections ensure the gradient can flow back through the network more

efficiently during backpropagation, addressing the vanishing gradient problem associated with deep networks. Using skip connections, ResNet architectures can effectively train networks with dozens or even hundreds of layers, improving performance and accuracy on challenging tasks such as object detection, image classification, and segmentation (Boesch [2023]).

2.3.5 TRANSFER LEARNING

Having explored the common architectures used in deep learning for image recognition, let us introduce another valuable tool for processing complex data: transfer learning.

Like how we perceive neural networks as mimicking the learning process in the human brain, transfer learning operates on a similar principle. To illustrate this, consider an athlete transitioning from track to field events. Their well-developed muscles from track training would give them a considerable advantage compared to someone just starting to exercise.

Likewise, in deep learning, transfer learning involves leveraging knowledge gained from one model to improve the accuracy and performance of another. A simple classifier can benefit from the learned representations and adapt them to a specific task using a pre-trained model as a starting point. This approach proves especially useful in computer vision applications, where large amounts of data and high computational power are typically required. Transfer learning provides a viable alternative when these resources are not readily available.

Transfer learning finds applications across various fields. For instance, autonomous driving can be employed to detect road signs, vehicles, and roads themselves. In the industrial sector, transfer learning aids in sentiment analysis when a company lacks extensive monitoring of customer interactions. Additionally, in the healthcare sector, where labeled data sets and expert image labeling resources are

limited, transfer learning becomes a valuable tool for image recognition tasks.

The versatility of transfer learning makes it a powerful technique to enhance model performance, accelerate training, and tackle data limitations across diverse domains. From the architectures we mentioned before, the most common to use for transfer learning are VGG-16, Inception, and ResNet50. While transfer learning can be applied with various architectures, the popularity of using transfer learning with AlexNet may have diminished compared to other deeper architectures for a few reasons:

1. **Model Capacity:** AlexNet was a groundbreaking model when it was introduced, but compared to more recent architectures, it has a smaller capacity in terms of the number of layers and parameters. Deeper architectures, such as VGG and ResNet, and more recent ones, like Inception and EfficientNet, have demonstrated superior performance and representation capabilities due to their increased depth and complexity.
2. **Feature Extraction:** Transfer learning often involves utilizing pre-trained models to extract useful features from the data before training a new classifier on top of these features. Deeper architectures generally capture more abstract and higher-level features, making them better suited for feature extraction tasks. This flexibility and expressiveness of deeper architectures make them more popular for transfer learning scenarios.
3. **Availability of Pre-trained Models:** The popularity of transfer learning with specific architectures also depends on the availability and accessibility of pre-trained models. While pre-trained AlexNet models are still available, newer and deeper architectures often have more readily available pre-trained models due to their wider usage and ongoing research advancements.
4. **State-of-the-Art Performance:** As deep learning progresses, new architectures continually push the boundaries of performance on various tasks such

as image recognition, object detection, and natural language processing. Researchers and practitioners tend to gravitate towards architectures that offer state-of-the-art results and achieve the highest accuracy on benchmark data sets.

However, it is worth noting that transfer learning with AlexNet can still be effective and useful in specific scenarios, especially when dealing with limited computational resources or smaller datasets. Ultimately, the choice of architecture for transfer learning depends on the specific task, available resources, and the state-of-the-art models and research in the field at the time of implementation.

2.3.5.1 APPROACHES FOR TRANSFER LEARNING

Transfer learning provides a valuable solution in scenarios where we encounter a task that requires deep learning but lacks sufficient data for effective training. One approach is to train a model on a dataset that addresses a similar or related problem. By doing so, we can utilize the resulting model as a starting point for our target task. However, the decision regarding whether to use all the weights of this pre-trained model or only the last layers depends on the nature of the problem at hand.

When deciding on the appropriate transfer learning strategy, we consider the level of similarity between the pre-trained model's task and our target task. If the tasks are closely related and share similar high-level features, it is beneficial to leverage the entire pre-trained model, including its weights, as a starting point. This approach allows the model to inherit the learned representations and adapt them to the specific nuances of our task.

However, in cases where the pre-trained model's task differs significantly from our target task, it may be more effective to utilize only the last layers of the model. The earlier layers of a deep neural network typically learn low-level and generic features, while the later layers capture more task-specific and higher-level represen-

tations. By freezing the weights of the earlier layers and training only the last layers on our dataset, we can fine-tune the model to specialize in our specific task while benefiting from the pre-trained model's initial feature extraction capabilities.

In summary, the decision to use all the weights of a pre-trained model or only the last layers depends on the similarity between the pre-trained task and our target task. This consideration ensures we balance leveraging existing knowledge and adapting the model to our specific problem domain.

Using pre-trained model: Pre-trained models have typically been trained on large and diverse datasets, enabling them to capture general visual features that apply to various image recognition tasks. By utilizing a pre-trained model, you can take advantage of the model's ability to generalize well to unseen data. This is especially valuable when you have limited training samples or need to classify images from different categories than the original training dataset.

Feature extraction CNNs are highly effective at learning hierarchical representations of visual features from large datasets. Pre-trained models trained on massive image datasets like ImageNet, have already learned these features. By leveraging a pre-trained model, you can benefit from its ability to extract meaningful and high-level features from images, even if you have a limited dataset. This is especially useful in scenarios where training a CNN from scratch may not be feasible due to data scarcity or computational constraints.

2.3.6 FUNCTIONS FOR DEEP LEARNING

As mentioned before, statistical learning plays a crucial role in many of the parameters we use to train a deep learning model or a machine learning algorithm. For example:

Loss functions: As mentioned before, choosing the appropriate loss functions is crucial in a CNN. Statistical learning provides various loss functions, such as mean squared error (MSE), binary cross-entropy, or categorical cross-entropy, depending on the nature of the problem. In the methodology section, we fully explain the type of loss function needed for the experimentation.

Regularization techniques: Overfitting is a common challenge in deep learning, where the model has a good performance on training data but fails to unseen data. Statistical learning techniques like L1 and L2 regularization (also known as weight decay) are employed in deep learning models to mitigate overfitting. These techniques introduce a penalty term in the loss function to discourage overly complex models, promoting generalization. In other cases, these regularization techniques can be used to ensure that the number of epochs and iterations do not compromise the learning rate of the model.

2.3.6.1 OPTIMIZATION ALGORITHMS

Deep learning models are typically trained using optimization algorithms, such as stochastic gradient descent (SGD) and its variants. These algorithms aim to minimize the loss function by iteratively updating the model's parameters based on the gradients of the loss function concerning those parameters. Statistical learning provides optimization techniques and convergence guarantees that help develop efficient optimization algorithms for deep learning.

Here we have a general description of the most commonly used optimization algorithms in CNN. For a more detailed and comprehensive explanation of each of the algorithms, we refer to a previous work by Berrones-Reyes [2019].

1. **Stochastic Gradient Descent (SGD):** SGD is the foundation of many optimization algorithms in deep learning. It updates the network parameters based

on the gradients of the loss function computed on mini-batches of training data. SGD updates the parameters by taking small steps toward the steepest descent, gradually minimizing the loss. While basic SGD is simple and easy to implement, it can sometimes be slow to converge and prone to getting stuck in local minima.

2. **Adaptive Moment Estimation (Adam):** Adam is a popular optimization algorithm combining Momentum and RMSprop ideas. It adapts the learning rate for each parameter by calculating individual adaptive learning rates based on past and squared gradients. Adam is known for its effectiveness in training deep networks, providing fast convergence and robustness to different types of networks and architectures.
3. **Adadelta:** Adadelta is an extension of Adagrad that seeks to alleviate the problem of continually decreasing learning rate during training. It addresses this using a decaying average of past squared gradients instead of the sum of squared gradients. Adadelta has no learning rate hyperparameter and can converge faster than Adagrad in certain scenarios.

These optimization algorithms and their variations and extensions are widely employed in training CNNs. The choice of the optimization algorithm is very dependent on factors such as the specific task, network architecture, and data set characteristics and empirical observations. Experimentation and fine-tuning are often necessary to find the most suitable optimization algorithm for a given CNN problem. Figure 2.17 shows the behavior we mention here for each of the algorithms from a study by Thavanani [2020].

2.3.6.2 EVALUATION METRICS

Statistical learning provides various evaluation metrics to assess the performance of deep learning models. For example, accuracy, precision, recall, and F1-score have

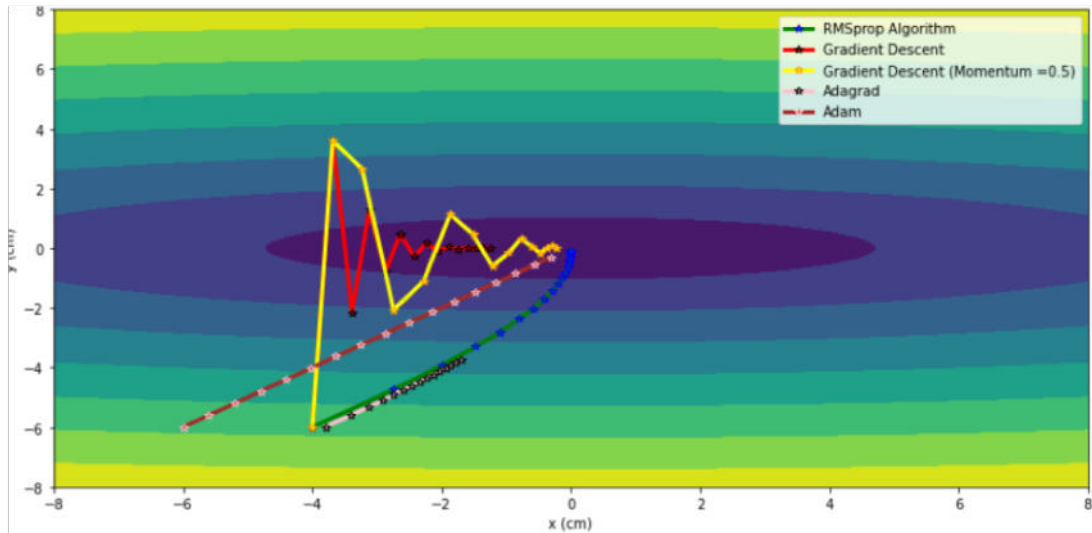


Figure 2.17: Comparative performance of different deep learning optimization algorithms using the library Numpy. Image from Thavanani [2020].

commonly used metrics for classification tasks, while mean absolute error (MAE) and mean squared error (MSE) are used for regression tasks (Chollet [2018]). These metrics help measure the effectiveness of a model and guide its improvement.

Here we explain the other three metrics commonly used for machine learning classifiers because they also provide a balanced view of the classifier performance and the data we use.

To understand the performance of the metrics, we need to discuss the confusion matrix. This takes the classification results and groups them into four categories (Geetha and Sendhilkumar [2023b]):

True positive (TP) is when the actual and predicted values are 1.

True Negative (TN) is when both the actual and predicted values are 0.

False positive (FP) is when the actual value is 0, but the predicted value is 1.

False negative (FN) is when the actual value is 1, but the predicted value is 0.

In Figure 2.18, we can see the representation of the confusion matrix for binary classification.

		Prediction	
		0	1
Actual	0	True Negative	False Positive
	1	False Negative	True Positive

Figure 2.18: Confusion matrix for binary classification.

1. **Accuracy:** The most common way to measure a classifier's performance is using the accuracy metric, as seen in the literature review. This is because we compare each input data's actual and predicted class. This metric, however, can sometimes hide the signs of an imbalanced data set, so even if the classifier were to perform poorly in the other three metrics, the accuracy would still perform well, masking the deficiency:

$$\frac{TN + TP}{TN + TP + FN + FP}. \quad (2.2)$$

2. **Precision:** Precision is a metric that focuses on the True and False positives, it can represent a view missed by the accuracy:

$$\frac{TP}{TP + FP}. \quad (2.3)$$

3. **Recall:** This metric operates on a principle similar to the precision metric but focuses on false negatives rather than false positives. Precision and recall are trade-off metrics, meaning that optimizing one often comes at the expense of the other. While precision emphasizes reducing false positives, which improves its precision, it may increase false negatives, leading to a decrease in recall.

Depending on the task's nature, we must prioritize which aspect is more critical to highlight in the problem we are attempting to solve:

$$\frac{TP}{TP + FN}. \quad (2.4)$$

4. **F1:** What if both the precision and the recall metrics are important? The F1 metric considers both precision and recall and gives us a good balance between the two:

$$\frac{2 * (\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}}. \quad (2.5)$$

2.4 U-NET

We have covered a range of algorithms for image recognition, including both classical methods and newer approaches. Although these algorithms were initially developed for general computer vision tasks, researchers have also explored their effectiveness in specialized domains.

One notable architecture is the U-Net, which has gained prominence for its application in biomedical image segmentation. By utilizing convolutional neural networks (CNNs), the U-Net architecture is specifically tailored to leverage the strengths of CNNs in this particular field. It offers a focused and effective solution for segmenting biomedical images.

The U-Net approach has demonstrated outstanding performance compared to many competing algorithms. One notable advantage is its ability to achieve accurate results even with a smaller training dataset. Additionally, it incorporates various techniques to enhance the learning rate of its architecture, further improving its performance.

Despite its specialization in biomedical image segmentation, the U-Net architecture shares several fundamental concepts with traditional CNNs. These include

cations where precise identification and localization of structures or anomalies are crucial. With its contracting and expansive paths, U-Net's architecture enables the extraction of intricate features at different scales, aiding in accurate segmentation and analysis.

Limited data availability: Biomedical data sets are often limited in size for various reasons, including privacy concerns and the difficulty of acquiring annotated medical images. U-Net has shown the ability to perform well even with a smaller training data set. This characteristic is advantageous in biomedicine, where obtaining large-scale annotated data sets can be challenging.

Handling class imbalance: In many biomedical tasks, class imbalance is prevalent, meaning certain classes or anomalies may be significantly rarer than others. U-Net addresses this issue by incorporating data augmentation techniques and using specialized loss functions such as dice loss or focal loss, which help mitigate the impact of class imbalance. This makes U-Net well-suited for biomedical applications where specific abnormalities may be infrequent.

Adaptability to various imaging modalities: Biomedical imaging encompasses diverse modalities such as magnetic resonance imaging (MRI), computed tomography (CT), and ultrasound. U-Net has demonstrated its effectiveness across different imaging modalities, showcasing its versatility and applicability in various biomedical domains.

State-of-the-art performance: U-Net has consistently achieved state-of-the-art results in numerous biomedical image segmentation challenges, competitions, and research studies. Its remarkable performance has garnered widespread recognition, making it a preferred choice among researchers and practitioners.

The U-Net architecture excels in several key areas, including effective feature

extraction, handling limited data, addressing class imbalance, adaptability to various imaging modalities, and impressive performance. These capabilities have contributed to its widespread adoption in diverse biomedical applications.

However, with U-Net, we deviate from conventional image classification techniques commonly used in deep learning. Instead, the focus shifts to semantic segmentation, a task that involves pixel classification. Each pixel in the image is assigned to a specific class. We obtain two results at the model's output: the segmented image, divided into distinct regions, and the corresponding label that assigns a class to each pixel.

So we see, the main difference between simple image classification tasks and semantic segmentation is that the latter requires not only a deep learning architecture for the classification of pixels but also a way to discriminate features that learn on the different stages of the encoder section of its architecture.

The reason why U-Net is so famous for biomedical use, and in our case, what we have a particular interest in this architecture, is because of its capabilities to use the encoder and decoder to extract and localize features in images, which in our case will be breast anomalies, and helps us solve the last two sections of our methodology.

Now we introduce two new concepts for this network: the encoder and decoder. For the encoder part, we are going to use familiar terms. This is the section where the object detection or image classification task is made. Here, we reference a classical convolutional neural network representation. We begin with an input layer, followed by the convolutional and pooling layers. The architecture has the corresponding layers contracting and passing on the important features.

We now have the intermediate or the bottom layer, where we transition from the encoder to the decoder part of our U-net shape. Here, we encounter new concepts we did not use for the classic CNN. This middle part involves the concept of **deconvolution** and **layer concatenation**.

A deconvolution involves the exact opposite behavior of the convolution. We know that convolutional layers reduce the information for the next layer of the architecture, so naturally, a deconvolution will output a bigger dimension to the next layer. Then we have the concept of pooling layers for convolution. The opposite option for deconvolution is the upsampling function.

Now that we understand the contracting and expansive path of the net (encoder and decoder), we see the skip connections. In Figure 2.19, these connections are represented by grey arrows. They play a crucial role in the flow of information and feature fusion. The skip connections provide a shortcut for gradients and help the decoder utilize both local and global context information, contributing to accurate and detailed segmentation. U-Net effectively merges multi-scale features through skip connections, leveraging high-level semantic information while preserving fine spatial details. This combination aids in producing accurate and detailed segmentations, making U-Net well-suited for various medical image segmentation tasks.

Then, in the same fashion as a CNN, we have a feature map. Throughout the network, the feature map resolution gradually decreases in the contracting path and then increases in the expanding path. This feature map resolution change allows the network to capture both high-level context and fine-grained details simultaneously.

By combining the encoder-decoder structure, skip connections and appropriate loss functions, the U-Net network is capable of capturing detailed features, preserving spatial information, and producing accurate segmentation maps for medical images. Its architecture has proven effective in various medical imaging applications, such as organ segmentation, tumor detection, and lesion delineation.

2.5 REAL-LIFE APPLICATIONS

In the same way, we discussed the topic of artificial intelligence on how it may seem like a recent field, while it has been developed and improved for many years. Computer-aided diagnostics (CAD) have grown in medical imaging since the '90s. Since then, many studies have tried to test and prove if these technologies help radiologists perform medical diagnosis. In this section, we discuss the evolution of CADs, and their use in the medical field.

2.5.1 COMPUTER-AIDED DIAGNOSTICS

In the early iterations of the use of CADs, it consisted of a checklist that the computer reviewed from the images that expert radiologists extracted from mammograms, and based on the calculations from that checklist, it made a prediction on the likelihood of malignancy for the mammogram. This was a very rigid process that did not require the computer to make the more complex learning algorithms we use nowadays.

In the case of this first CADs, the main goal was to standardize the diagnosis of different experts since the main issue then was the inconsistency in which the radiologist was able to recognize the features that lead to malignancy, which led to unnecessary biopsies, being only the 15% to 30% of cases where the biopsy resulted in the discovery of malignancy. This meant that most biopsies were performed on benign lesions (Jiang et al. [1999]).

This estimation of malignancy via automated computer feature extraction was a significant success, improving the accuracy of radiologists by up to 14%. This led to various studies that looked to improve the accuracy and efficiency of medical diagnosis.

Fast forward a couple of years, parallel to the advancement of these CAD tech-

nologies, computer vision tools were also making significant improvements, thanks to the development of deep learning algorithms.

In April 2010, the ImageNet challenge changed how researchers explored using artificial intelligence in computer vision. It dared to focus not only on the algorithms and programs but also on looking at the data to help redefine how we build models. This data set had accumulated more than 11 million labeled images by the end of 2010. It gave way to the first ever ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where researchers from all over the world competed with their programs and algorithms that allow them to classify and detect objects on these images.

Since its conception, this challenge and data set have allowed researchers to use this as a benchmark to test the newest machine learning and deep learning models, where the accuracy has reached levels greater than 95%.

Thanks to these fantastic collective contributions of artificial intelligence algorithms from different researchers, a branch of deep learning became the favorite to help solve computer vision issues and complex problems from other fields, such as big data. This field is known as neural networks.

Moving forward, we discuss some works specific to the medical use of artificial intelligence tools that, while still inside the sphere of CAD tools, have now evolved into a field better known as radiomics (Lambin et al. [2017]), which focuses more on medical imaging.

CHAPTER 3

LITERATURE REVIEW

As we mentioned, medical imaging is one of the real-life applications of deep learning in computational vision. Computer-aided diagnostics (CADs) has tried to evolve alongside the breakthroughs of computer vision. Still, as we learn from different literature reviews, we see that medical imaging has a different set of rules and limitations that we do not find in traditional computer vision problems. In this section, we discuss some of the advances that have been tested on medical images from a computer vision perspective and contrast them with the perspective of real-life clinical problems.

3.1 EVOLUTION OF COMPUTER-AIDED DIAGNOSTICS

Computer-Aided diagnostics have a history that can date back a few decades. The first iterations were notable advances in their field and helped greatly improve the accuracy of many specialists in medicine. Some examples of the early developments for CAD systems in radiology date from 1970 to 1980, when researchers explored the use of computer algorithms to interpret medical images of CT scans. In this stage, these methods aimed to consider the output of these algorithms as “second opinions” to improve the diagnostic performance of radiologists/physicians Suzuki [2012].

The first processes of a CAD scheme followed similar steps. In Figure 3.1, we can see the generic flow chart for detecting lesions in medical images. In this case, the boxes with blue color represent the major steps in the process. The boxes with orange color represent the optional steps.

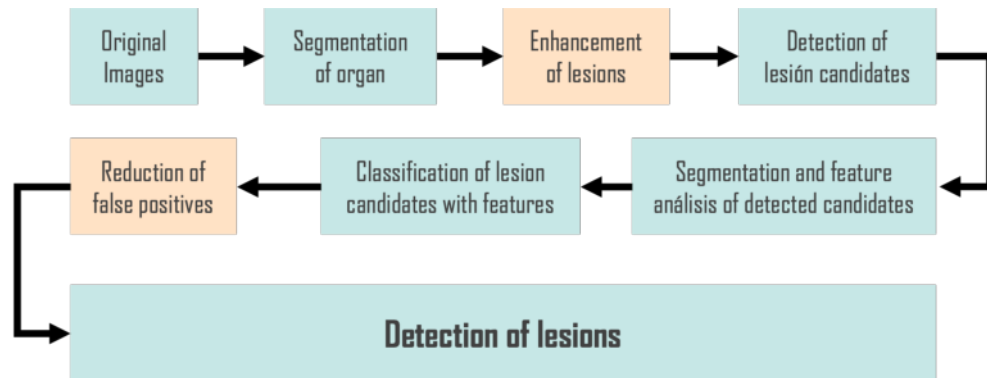


Figure 3.1: Common flowchart of a generic scheme of CADs used for lesion detection in medical images by Suzuki [2012].

In this scheme, many major steps were not optimized, as machine learning was generally not applied until lesion candidates were detected. Some of the known feature-based machine learning (or classifiers) included in the classification of the lesion candidate were LDA, QDA, ANN, a Bayesian ANN, and an SVM.

As the field advanced, Computer-Aided Diagnosis (CAD) systems began to incorporate more sophisticated pattern recognition and feature extraction techniques. These specialized methods tailored to distinct types of medical images and specific diseases required more comprehensive systems. This led to the emergence of expert systems, which integrated the knowledge of medical experts and utilized rule-based algorithms to analyze medical images. Such expert systems facilitated a more nuanced and accurate disease detection (Ribeiro et al. [2008], Matesin et al. [2001], Barrera et al. [2010]).

In the early 2000s, research efforts concentrated on the statistical aspects of rule-based and expert systems. For instance, the work by Ribeiro et al. [2008] employed an association rule-based method to analyze mammograms. This approach

aimed to expedite the diagnostic process led by specialists by automatically generating preliminary diagnostic suggestions through mining association rules. The method was tested on a real-world dataset and achieved an accuracy rate exceeding 90%.

Other notable works, such as those by Matesin et al. [2001] and Barrera et al. [2010], focused on feature extraction in medical images using expert systems. In Matesin et al. [2001], a rule-based algorithm was developed for segmenting CT brain scans into different regions, such as the background, skull, and gray matter. Meanwhile, Barrera et al. [2010] used a rule-based approach as an auxiliary tool for diagnosing cervical lesions in colonoscopic images.

These rule-based implementations provided valuable support to medical experts, significantly enhancing the accuracy and efficiency of medical diagnoses. This triggered a wave of research, surveys, and performance analyses on case-based medical data, later termed as “knowledge discovery in medicine.” Pioneering surveys, like the one by Kahn [1991], delved into the validation and evaluation of various expert systems used in medical imaging. These studies measured the potential impact and utility of such systems for future applications.

Despite the progress made with rule-based algorithms, not all such methods withstood rigorous testing and evaluation, underscoring the ongoing quest for reliable computer-aided diagnostic tools. This quest has now extended into artificial intelligence with the advent of statistical learning and machine learning. These newer, more powerful methods have significantly transformed the CAD landscape, enabling systems to analyze large and complex medical data sets better, thereby leading to more accurate and personalized diagnostics (Foster [2023]).

3.1.1 STATISTICAL LEARNING IN CAD

Statistical learning techniques, such as logistic regression and support vector machines (SVM), paved the way for more complex models in the medical diagnostic field. These algorithms, which primarily focus on finding statistically significant relationships in the data, offered improved prediction accuracy over traditional methods. Hastie et al. [2009] and Erus et al. [2014] demonstrate the application of statistical learning techniques in identifying critical features from medical images for various diagnostic purposes.

Before the rise of machine learning and deep learning, statistical learning was a cornerstone in developing early CAD systems. Works like Sande et al. [2021] show how statistical learning techniques often focus on finding a mathematical model that describes the underlying structure of the data. These models are grounded in statistical theory and offer interpretability and robustness, essential traits for medical diagnosis.

One of the seminal contributions of statistical learning to CAD was in the area of feature selection. Techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) were frequently employed to reduce the dimensionality of medical image data while preserving the most relevant information for diagnosis. Regression models like logistic regression were another staple in early CAD systems for task prediction of disease outcomes based on various features (Hastie et al. [2009]).

Statistical hypothesis testing also found its place in CAD systems, often used for validating the efficacy of diagnostic features and comparing different diagnostic models. Techniques such as the Chi-square test, t-tests, and ANOVA were instrumental in verifying whether the features used in CAD systems were statistically significant (Wang and Zheng [2013]).

Despite the significant contributions, the manual feature extraction process was

one limitation of early statistical learning methods. This was a time-consuming and often domain-specific task that required extensive medical expertise. The advent of machine learning and deep learning automated much of this feature extraction, allowing for more rapid development and deployment of CAD systems.

Nonetheless, statistical learning methods are far from obsolete. They often serve as a benchmark for newer techniques and are frequently combined with machine learning models to enhance reliability and interpretability Hastie et al. [2009]. As a result, statistical learning continues to hold a significant place in the continually evolving landscape of CAD systems.

3.1.2 MACHINE LEARNING IN CAD

The limitations of statistical learning, particularly in handling high-dimensional data and capturing complex relationships, led to adopting machine learning algorithms. Methods like decision trees, random forests, and neural networks have proven highly effective in automating the diagnosis process, as they can learn intricate patterns from data without explicit programming (Géron [2020]). For instance, Litjens et al. [2017] applied deep learning methods to achieve remarkable accuracy in diagnosing specific diseases from medical images.

The shift from statistical learning to machine learning in CAD systems represents a paradigm shift. While statistical learning methods are still been used in evaluation metrics, machine learning techniques, especially deep learning, are increasingly favored for their ability to scale with data and learn from raw features without much manual intervention. As computational power increased and large datasets became more accessible, machine learning (ML) played a critical role in advancing CAD systems. Unlike statistical learning methods, which often require manual feature extraction and selection, many machine learning algorithms can automatically identify relevant features, making them highly effective for complex tasks

like medical imaging. In this area, many medical surveys discuss the potential of these types of tools, such as works by Bakator and Radosav [2018], and Litjens et al. [2017].

Among the first machine learning techniques adopted for CAD were Support Vector Machines (SVMs). These provided a powerful mechanism for classifying images into different categories based on their features. After this, Random Forests and Gradient Boosting algorithms were employed to improve the system's performance further.

The rise of Neural Networks marked a significant shift. Initially, simpler architectures like Multi-Layer Perceptrons (MLP) were employed. These could capture more complex relationships in the data but were soon overshadowed by more specialized neural networks optimized for image data, such as Convolutional Neural Networks (CNNs) (Dong et al. [2021]).

One of the most impactful developments was the introduction of Transfer Learning. This approach allowed pre-trained models on large generic datasets to be fine-tuned for specific medical tasks, dramatically reducing the need for large annotated medical datasets and accelerating the development cycle (Esteva et al. [2021]). Natural Language Processing (NLP) techniques, another subset of machine learning, have also been applied to CAD systems, particularly in extracting useful information from clinical text data to aid in image diagnosis (Houssein et al. [2021]).

Moreover, machine learning methods have been integral in developing real-time diagnostic tools, predictive modeling for patient outcomes, and even automating the very labor-intensive data annotation process, which is crucial for training more robust models. The evolution of machine learning in CAD systems has not been without challenges. The black-box nature of many ML algorithms can be a hurdle in clinical settings where interpretability and understanding the decision-making process are critical. Yet, advancements in explainable AI are starting to bridge this gap (Dong et al. [2021]).

As computational power increases and algorithms evolve, machine learning is poised to become even more central in the development of future CAD systems, offering the promise of more accurate, faster, and more interpretable tools for medical diagnosis.

3.1.3 PUSHBACK AND CHALLENGES FOR MEDICAL CAD SYSTEMS

While Computer-Aided Diagnostics (CAD) systems have made significant advancements, they have not been without controversy or pushback within the medical community. Several factors have contributed to this resistance, ranging from technical limitations to ethical concerns. Works such as Esteva et al. [2021], Ahishakiye et al. [2021], Litjens et al. [2017], Panayides et al. [2020], Castiglioni et al. [2021], Chugh et al. [2021], show a very interesting perspective outside of the field of computer vision or artificial intelligence and more on the side of the concerns and needs of the medical community. Here, we compile some of the most prominent ones that helped us change our view of CAD systems in general.

3.1.3.1 TECHNICAL LIMITATIONS

Early versions of Computer-Aided Diagnostic (CAD) systems often faced criticism due to their high rates of false positives and negatives, resulting in clinicians' distrust. These inaccuracies could lead to incorrect treatment plans and worsen patient outcomes (Sollini et al. [2023]).

Addressing these inaccuracies, the effectiveness of a CAD system was found to be heavily dependent on the quality and quantity of the data it is trained on. Inconsistent or poor-quality data could propagate unreliable results, creating a challenge particularly when many medical datasets presented an imbalance; they contained

far more examples of common conditions than rare ones, leading to the development of biased models (Park et al. [2023], Lin et al. [2023]).

This data quality and quantity follows into the problem of overfitting, where a model, although performing exceptionally well on the training data, generalizes poorly to new, unseen data. This aspect is a critical concern when a model is trained with a benchmark data set, which is a common practice for regular computer vision problems and something that does not translate well in medical settings, emphasizing the necessity for models that can generalize effectively and reliably to real-life clinical data sets (Berrones-Reyes et al. [2023]).

Accommodating the need for more generalized models, CAD systems have evolved to incorporate more complex algorithms, including deep learning models. This evolution, while promising, has significantly increased the demand for computational resources, which holds implications for the often necessary real-time processing and analysis in medical settings (Labrada and Barkana [2023]).

This introduction to deep learning models often carries the “black-box” nature inherent to some machine learning and deep learning algorithms. This opacity in operation makes it difficult for medical professionals trained to make decisions based on understanding pathology and physiology to interpret the results, thus forming a substantial barrier to adoption and, in some cases, even encouraging other medical fields not to trust machine learning unless it goes with comprehensive explanations. (Hernandez et al. [2023]).

Lastly, a persistent obstacle facing many CAD systems is the lack of external validation. The frequent practice of training, tuning, and testing models on the same datasets has led to overly optimistic performance estimates, which often falter in real-world clinical settings. Despite these limitations and challenges, concerted efforts are underway to improve the reliability and user-friendliness of CAD systems. The focus is on mitigating false positives and negatives, elevating model interpretability, and enforcing robust validation practices to ensure the systems’ clinical efficacy

(Melarkode et al. [2023]).

3.1.4 DATA SHARING AND PRIVACY

In addressing the challenges within CAD systems, another issue emerges: standardization, which is crucial for making these systems broadly applicable. The healthcare landscape is diverse across countries, with various providers utilizing Electronic Health Records (EHR) systems, imaging modalities, and data storage formats (Wang et al. [2022]). This diversity poses a significant challenge, as it needs a level of compatibility within a CAD system to seamlessly integrate with all existing infrastructure, ensuring its widespread applicability.

Another concern is data privacy and sharing in light of the quest for standardization and integration. The need to share patient data across different platforms for enhanced diagnosis is frequently slowed down by strict privacy laws and regulations. These regulations are not uniform but vary by jurisdiction, further complicating the sharing process and adding another layer of complexity to deploying effective and universally applicable CAD systems (Yu and Shi [2023]).

3.1.5 CHALLENGES FACED BY DEVELOPING COUNTRIES

Developing countries are often faced with challenges that are less prevalent in their developed counterparts. These constraints often refer to limited financial resources and a shortage of trained medical professionals and technical experts essential for the operation and maintenance of CAD systems (Leming et al. [2023], Lv et al. [2023]).

In addition to the human resource constraints, technological limitations in these countries, particularly limited internet connectivity in rural or remote areas, can pose significant obstacles. Implementing cloud-based CAD solutions becomes

particularly challenging in these regions, unlike in developed countries, where a more robust internet infrastructure is typically in place.

Moreover, the majority of CAD systems are initially developed within the context of Western medicine. This can translate into models unsuitable for addressing local diseases or conditions prevalent in developing countries. Further complicating the landscape is the absence of centralized data for medical imaging in many developing regions. Without a centralized database, ensuring consistent, high-quality care becomes a formidable challenge, and navigating the intricate legal obligations around data storage and access becomes even more complex (Yu and Shi [2023]).

3.1.6 PROFESSIONAL RESISTANCE

Despite the promising capabilities of CAD systems, many healthcare professionals remain skeptical. For many doctors, outsourcing critical tasks to a machine can evoke feelings of losing control. Trusting a computer system with life-and-death decisions is difficult, especially when the algorithms behind these systems are often considered “black boxes” (Hernandez et al. [2023]).

Interestingly, a stigma is attached to errors made by computers instead of humans. While doctors are generally forgiving of human error as an inevitable aspect of the medical profession, mistakes made by computers are less readily accepted. This double standard can make gaining professional credibility for CAD technologies difficult (Cho [2021]). Another form of resistance comes from professionals who perceive CAD systems as threatening their role and expertise. The potential for automation to replace some aspects of human labor can create anxiety and resistance among healthcare providers.

Beyond individual healthcare providers, systemic resistance also exists. Medical facilities are often slow to adapt to new technologies due to budget constraints, a lack of understanding, or bureaucracy. Clinicians are often concerned about the

safety and security of patient data, especially in systems that require cloud-based processing. These concerns add another layer of resistance to adopting new technologies.

Overcoming this professional resistance will require concerted efforts across educational, organizational, and policy levels to ensure that CAD systems can reach their full potential in aiding medical diagnoses. With advancements in interpretable machine learning and more robust validation studies, CAD systems continue to make inroads into mainstream medical practice (Norori et al. [2021]).

3.1.7 CURRENT STATE AND FUTURE DIRECTIONS: THE ROLE OF DEEP LEARNING IN CAD

Tackling challenges in developing countries paves the way for the next step in CAD systems through machine learning. In particular, deep learning methods are ushering in an era of more precise and efficient diagnostics. Research is also focusing on integrating explainable AI to make these systems more user-friendly and gain the trust of healthcare providers.

Deep learning, a key subset of machine learning, is becoming a game-changer for CAD systems. It uses deep neural networks to learn high-level features from data, setting it apart from traditional techniques. Convolutional Neural Networks (CNNs) are gaining popularity for image-based diagnostics. They have shown great accuracy in identifying anomalies in various medical images like X-rays and MRI scans. Similarly, Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs) are useful for sequence-based tasks such as ECG interpretation. The use of transfer learning methods is also on the rise. This approach could reduce the need for large annotated medical datasets, addressing privacy concerns.

A survey conducted in the United States by Schmidt [2023] gathered insights

from 1,027 participants spanning various age groups, shedding light on their perspectives regarding integrating artificial intelligence (AI) technologies in the medical domain. Figure 3.2 illustrates a notable contrast between the attitudes of younger and older generations, underscoring the divisive nature of employing AI in medical procedures.

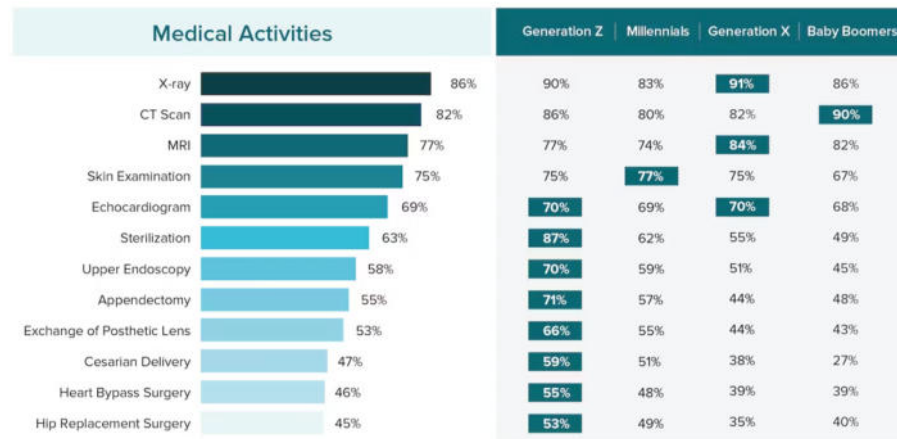


Figure 3.2: Statistics about an American survey made in 2023 about the attitude different generations have towards AI being used in the medical field (Schmidt [2023]).

The survey yielded a significant finding, indicating that 64% of respondents place more trust in a diagnosis made by AI than one provided by human doctors. Despite the evident advancements in public understanding of the scope and scientific principles underpinning artificial intelligence in recent years, there persists a degree of apprehension. The primary apprehension revolves around concerns related to the utilization and safeguarding of patient data, as highlighted in Figure 3.3. The top concerns include the accuracy of diagnoses and technical limitations, topics which we delve into extensively in this study.

Looking ahead, there is growing interest in making deep learning more interpretable. As medical professionals need clear explanations and diagnoses, future CAD systems will likely be more transparent while maintaining accuracy. Integrating deep learning with other AI forms could also lead to significant advances in personalized medicine.

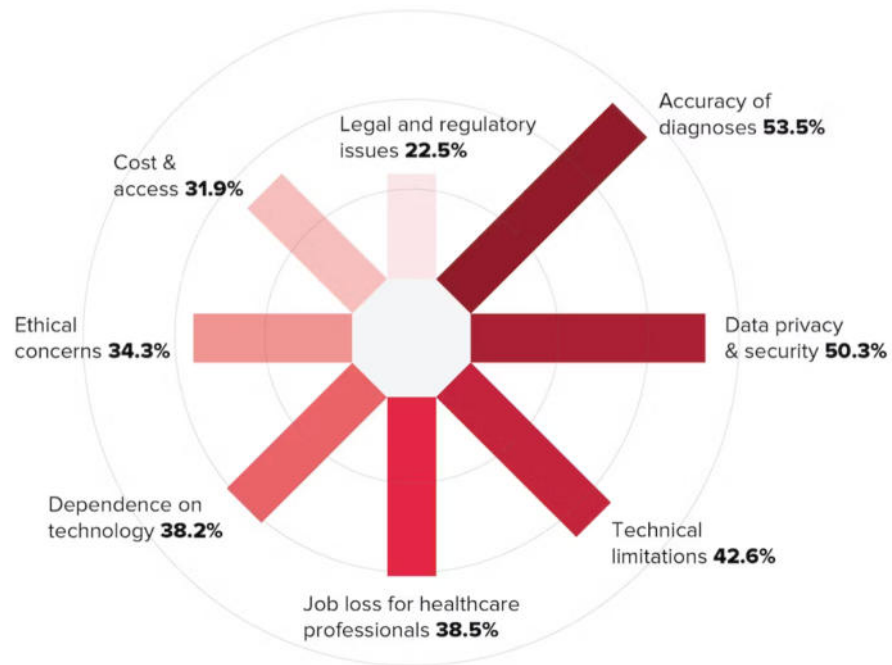


Figure 3.3: Graphical representation of the main concerns of the general population of an American survey regarding the use of AI in medicine (Schmidt [2023]).

In conclusion, combining deep learning with CAD will bring a new era in medical diagnostics, aiming for accuracy, efficiency, and personalization.

CHAPTER 4

METHODOLOGY

In this section, we describe the materials used for all of our experiments and the tools used to achieve our proposed methodology. This includes our data sets, their description, and their characteristics. We then describe the software and the crucial libraries to handle the images and construct our models. Finally, a detailed description of our hardware specifications for reproducibility.

We then move on to explain the three stages of our methodology. These are presented in this way because the result of the first is the input for the second stage, and the result of this second stage is the input for the third and final stages. Each of these steps can be done separately at the need of the medical specialist, so each one has a pre-processing stage at the beginning of the experimentation.

4.1 MATERIALS

In this Section we mention the data sets used to train, and test our tools. We present hardware specifications for our experiments' reproducibility and describe the essential libraries and software needed to complete the models.

4.1.1 DATA SETS

In the literature review, we mentioned some of the more popular benchmark data sets used to build classification and diagnostic models for mammography. There is the Mini-MIAS data set and the DDSM set.

In the case of the first two data sets presented, the Mini-MIAS and the DDSM are both public data sets that are free and available. For the case of both these public data sets, the format and way of acquiring them were old and antiquated, so to use these popular datasets as benchmarks for new algorithms, the Cancer Imaging Archive updated and standardized the version of DDSM and Minimias into the Curated Breast Imaging Subset of Digital Database for Screening Mammography (CBIS-DDSM). (See Appendix A for more information about this joint data set). Lastly, we have a third data set, YERAL, from a private Mexican hospital. This first section of the methodology will give more thorough information about each data set.

4.1.1.1 MINI-MIAS

The Mini-MIAS (Mammographic Image Analysis Society) is an organization of research groups in the United Kingdom whose work has been generated as a digital mammography data set available online since 1994. It contains 322 images, of which 121 present an anomaly, and the rest are described as normal (without anomalies). Alongside the images is a description that specifies the type of anomaly and its location. For this reason, these images have been used as a benchmark for several articles on segmentation and classification.

This data set comes with a .csv file that mentions the type of anomaly the images present, where the anomaly is located, and a summary of the type of character background tissue divided into fatty, fatty glandular, and dense glandular.

4.1.1.2 DDSM

DDSM stands for Digital Database for Screening Mammography. This database was reviewed and annotated by radiology experts from the University of South Florida and the General Hospital of Massachusetts from the Department of Engineering and Computational Science. This data set has around 2,500 case studies, each of which has two images of each breast, including relevant information about the patient (age, the density of the tissue, location of the anomaly, etc.) without having any personal feature.

In the case of this data set, a version is available that contains all the collection of scanned film mammography in a DICOM format, including the complete mammogram, the cropped anomaly, the ROI segmentation and bounding boxes, and the segmented masks. The DICOM file does not need an extra file with the description of the images since the title has all the required information.

This data set and the Mini MIAS data set that we previously described are the most often used mammography benchmark for the literature review. For this, the CBIS-DDSM data set was created. (curated breast imaging subset of DDSM). These include the DDSM, the Mammographic Imaging Analysis Society (MIAS) database, and the Image Retrieval in Medical Applications (IRMA) project.

4.1.1.3 YERAL

This data set was initially reviewed in 2012 in search only for anomalies. In 2018, it was reviewed again for feature extraction. The Instituto Nacional de Cancerología in Mexico City provided it. It was then revised by the FUCAM (Fundacion de Cancer de Mama), a private non-profit institution in Mexico and Latin America. FUCAM offers specialized breast cancer treatment through its highly specialized unit in Mexico City. This data set has 200 mammography images with anomalies and 120

without them. These images have 1024×1024 pixels, but the quality is far superior to the MiniMias data set since we already experimented with convolutions and filters. This data set could have more alterations than the Mini MIAS without being too distorted. It also has an RGB colour channel, which gives a better resolution.

One crucial remark in the surveys of deep learning applied to medical imaging referenced in the literature review section is that only some studies present externally validated results (actual clinical data). The resulting issue with this practice is that the results on the benchmark data sets give an expectation for these computational tools that are not often achievable with external data. Alternatively, in the cases where the authors used a validation set, they compared the model's performance using the same sample. Therefore, we focused on properly using data sets for training, develop, and test evaluations to avoid that weakness.

Our last validation set is referred as the test set, as it contains only images from the target data set YERAL. This work focuses primarily on giving experts a good computer-aided diagnosis tool. We use the standard practice of distributing the training and developing sets. Many ML books argue that these two data sets (training and developing) must contain at least some of the images that the model will classify (Géron [2020], Chollet [2021], Ng [2019], Harrington [2012]). Since there is already a small amount of data, the test set contains 50 images with anomalies and 50 without, only from the YERAL data set.

An important discovery in our methodology was the big impact that data imbalance had on our results, and for that, we combined the updated DDSM dataset with some of the images from the YERAL dataset. Having them all together we gathered 2,594 images without anomalies and 8,401 with anomalies. This newfound imbalance helped uncover a very big bias in the resulting performance of the two classes.

To prove this bias results, we used the most used performance metrics for

Table 4.1: Using the most popular performance metrics for AI tools, we compare the validation data set when we have an unbalanced and a balanced data. For this example, we only explore the use of a simple Alexnet architecture described by Krizhevsky et al. [2012].

	Unbalanced data set		Balanced data set	
	Normal	Anomaly	Normal	Anomaly
Precision	64%	90%	82%	86%
Recall	82%	78%	83%	85%
F1 score	76%	73%	86%	82%
Accuracy	79%		84%	

image recognition, which are F1 score and accuracy. The goal was to find a balance to improve the overall accuracy and avoid biases. At random, we removed some images only from the DDSM dataset. In the end, the balanced data set consisted of 2,594 normal images and 2,900 images with anomalies. Table 5.5 shows the impact of a balanced data set in popular metrics.

The final balanced dataset is going to be referred as DDSM_YERAL, and the additional validation test set was YERAL.

4.1.2 HARDWARE

For this section, we specify two different setups since we started with one computer at the beginning of the experimentation. We had to upgrade to another one to be able to finish our experimentations:

For the first setup, we have:

- iMac 2020
- CPU: 3.8 GHz Intel Core i7, 8 cores

- GPU: AMD Radeon Pro 5500 XT 8GB
- RAM: 16 GB 2667 MHz DDR4
- OS: Ventura 13.5.1

For the second setup, we upgraded to:

- Gigabyte AMD Ryzen 3000
- CPU: AMD RYZEN 7 5800X, 3.8GHz, 8 Cores
- GPU: ASRock AMD Radeon RX6900XT GDDR6 16GB
- RAM: DDR4 32GB 3200 MHz.
- OS: Windows 11

Since the specifications of both of the computers we used for our experimentation can be considered high specs (in particular, the second setup, which was used to sustain the bulk of the computationally heavier experimentation), in the experimental part, we will discuss the best parameters to move around in case the computer used to replicate the experiment has less computational power. That being said, it is important to note that basic computers (like Chrome books or older computers from 2012 or older) are not equipped to handle the basic requirements for machine learning algorithms. We recommend looking into cloud computing to compensate for the lack of adequate hardware for those options.

4.1.3 SOFTWARE

We do not separate the tools into the categories of segmentation, feature extraction, classification, etc., because most tools were used to complete different parts of all three stages. For all the software tools listed below, we are considering the

programming language of Python (Van Rossum and Drake Jr [1995]), version (3.8). This language is often used as a tool for data science. It is developed under an OSI-approved open-source license and hosts thousands of standard and community-contributed third-party modules.

- **Keras:** Keras (Chollet et al. [2015]) is a high-level networks API that runs with the library of TensorFlow or Theano. They allow the use of both CPU and GPU for the processing of data, which is a desirable capability when using complex information like images. This library has built-in capabilities for designing the blocks for neural networks, including convolutional and recurrent networks, that allow, alongside the more traditional blocks, to train deep learning models.
- **Tensorflow:** Tensorflow (Abadi et al. [2015]) is an open-source library for numerical computation that helps to develop machine learning and deep learning models faster. It uses programming languages like Python or JavaScript as a front-end API to build applications that help train deep learning models. This library can run on various tools, such as a cluster in the cloud, local machines, CPUs, or GPUS. Tensorflow is supported only on Python from version 3.7 through 3.10. While it may work on earlier versions, not all modules are guaranteed to do so.
- **Pillow:** Pillow (and its predecessor PIL) (Clark [2015]) is a Python library that helps to manipulate and process images. Many of the modules described in this library helped with the pre-processing part of this methodology.
- **Scikit-learn:** This is a Python library (Pedregosa et al. [2011b]) that has access to built-in modules with classification, regression, clustering, and more machine-learning algorithms. It also comes with various options for different data structures, and it has some externally built applications specific to certain problems, such as PyBrain or TorchIO.

- **Scipy:** This library (Virtanen et al. [2020]) is an essential part of the statistic portion of our methodology. It provides algorithms for interpolation, optimization, and algebraic equations. This last one is of particular importance when working with image transformations.
- **Opencv:** This library (Bradski and Kaehler [2008]) was created as the Open computer vision library in 2000 by the Intel Corporation. Today, the OpenCV library is a pillar of real-time computer vision problems. It has many areas of applications, and the main ones we are going to be using are segmentation and object recognition, as well as the human-computer interactive part of our problem.

4.2 CLASSIFICATION PROCESS

As mentioned before, the stages of this methodology are set like a process but could be used individually, depending on the stage the medical expert is interested in using. This first part considers the main bottleneck represented and discussed in the background section.

Recent efforts for the conscientization of breast cancer have generated a significant amount of information, and the number of medical experts that can use this information is dwarfed in comparison. For this, the first stage of our methodology tries to unload the information the medical personnel must go through to get to a diagnosis. With this, we aim to optimize the process by giving a “first glance” at the images and dividing between the images with and without an anomaly. For this, we use what we describe later as a binary classification.

Following our primary objective for this methodology, we want to use the methods proposed here as separate stages, so in each case, we start by pre-processing the images. If the process is used as a whole, we advise that the pre-processing stage be done just at the beginning.

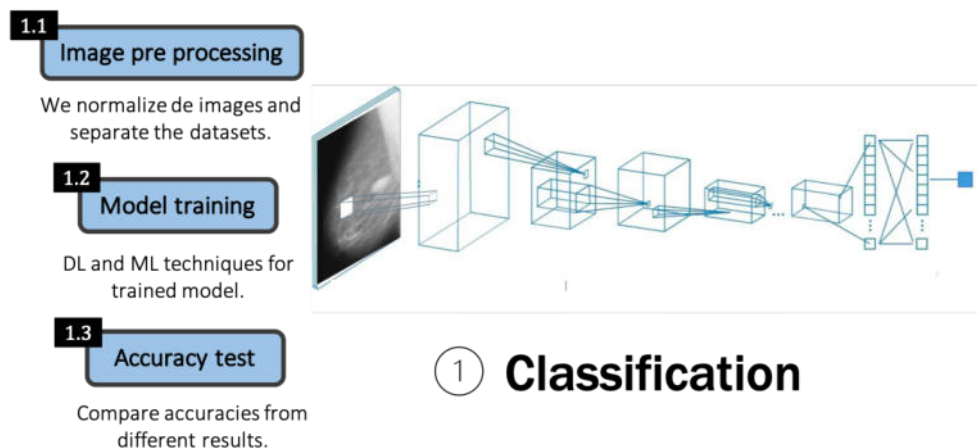


Figure 4.1: Steps involved in the classification stage of the methodology.

In Figure 4.1, we see the steps for this classification stage. After the pre-processing part, we use deep learning and machine learning models to achieve the best possible accuracy in our data set. Lastly, we must ensure our models' accuracy matches the accuracy obtained in the target data set since that is one of the main criticisms reported on medical surveys about artificial intelligence tools used for medical imaging.

4.2.1 PRE-PROCESSING

The first step in all data analysis exploration is to pre-process the information. We can explore data normalization and standardization in computer vision in many ways, but this process is much more selective for medical images. We use the different data sets explained previously for the experimentation. Normalizing the images is essential to using data sets with other distributions. The first step in both cases is to remove any unnecessary data that can identify any information about the patients.

There are labels on top of every image for the YERAL data set, which we remove by using bounding boxes in the Python code using the Pillow library. It is important to remark that this bounding box was only used for the YERAL data

set since all the other data sets from this experiment did not have the information labels on the image, so we used a binarization option that allowed us to remove the tag of the type of image. Since the text is on top of the image for the YERAL set, we needed to remove the whole label.

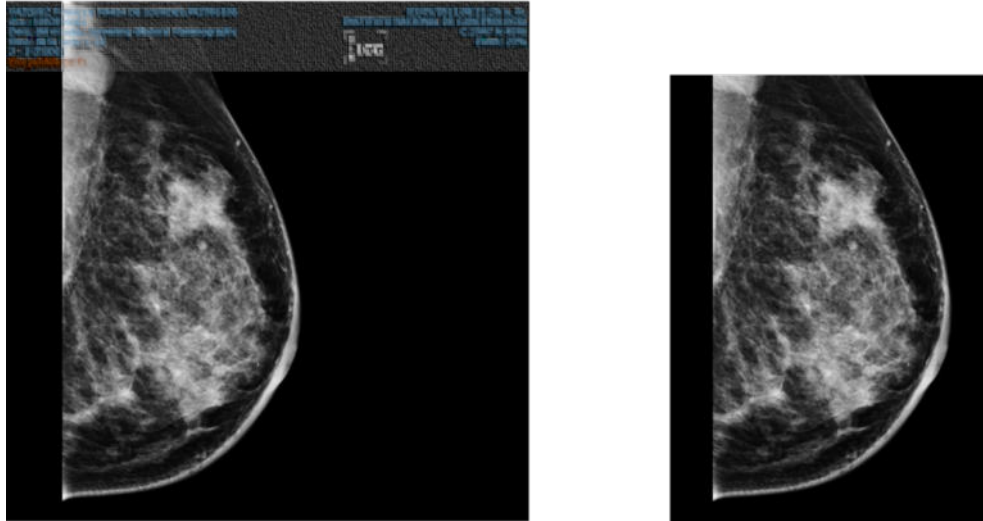


Figure 4.2: Removal of patient information and unnecessary labels from the YERAL data set. The patient information is blurred for privacy concerns.

Figure 4.2 shows the label removal before and after. For the DDSM data set, this information has already been removed. The only thing that can cause noise on the images is the labels that indicate which type of image it is (RMLO, LMO, RL, etc.). For this, we use a different algorithm first to binarize the image and identify the biggest “blob” inside. This will allow us to identify the breast section as the most prominent white part and the little label as the smaller one. We then cut out this small portion and return the image to normal. In the end, we successfully removed noise from those images. Figure 4.3 represents the process of binarization of the image, removal of unnecessary tags, and the final processed image.

```
1 import matplotlib.pyplot as plt
2 import matplotlib.patches as patches
3 from PIL import Image, ImageDraw, ImageChops
4 import numpy as np
```

```
5 import cv2
6 import os
7
8 if __name__ == "__main__":
9     for f in os.listdir('.'):
10        if f.endswith('.png'):
11            im = Image.open(f)
12            fn, fext = os.path.splitext(f)
13            img_draw = ImageDraw.Draw(im)
14            img_draw.rectangle((0, 0, 350, 95), fill='black')
15            img_draw.rectangle((1024, 0, 550, 95), fill='black')
16            try:
17                im.save('{} .png'.format(fn))
18            except AttributeError:
19                print("Not found {}".format(img))
```

Listing 4.1: Python code to crop the confidential information of the patients mammogram

In 4.1, lines 14 and 15 need adjustments depending on the images used. In our case, it is specific to the size and location of the labels. All of them had the exact coordinates. In line 6, we use the *os* library, and the way we call it in line 9 means that this code needs to be inside the folder of the images it will modify. Line 10 ensures that it only considers images with the termination of *.png*, so if there is any other format of image, it needs to be changed beforehand. It also means that it will not send an error if more code files are inside the folder.

As for the computational features that need to be considered, this format changes the original image and updates it to the censored one, so if we want to maintain the original data set without changes, line 17 can be changed to save it into another folder. If this is the case, ensure at least double the memory storage space of the original data set.

For the DDSM and some of the YERAL images that still held the indicative RMLO, LMO, and RL for the angle of the image, we used the code referenced in

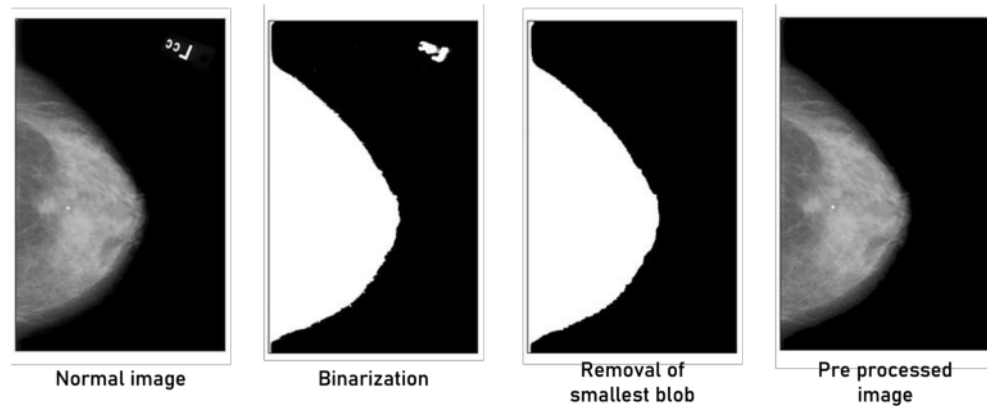


Figure 4.3: Pre-processing for the CBIS DDSM data set contains the DDSM, Mini Mias, and IRMA project images.

As `Imageprocessingforpng.py`. This code has several functions needed to clean the images. We explain each function and the order that needs to be done for each type of image.

1. `get_filepaths(directory)`: This function takes the root directory of the image folder as input. It is important to note that this function expects a folder, not a file.
2. `CropBorders(img)`: This function takes as input a single image and outputs the same images but is now borderless.
3. `Binarisation(img, maxval)`: This function takes a single image as a parameter and the maximum value of the threshold for the binarization process. The recommended value for this is 1.0. This function's output is the four image thresholds (up, down, right and left). These thresholds are used later on.
4. `OwnGlobalBinarise(img, thresh, maxval)`: This function takes an image as input and returns a binarised image's mask. Here, the `thresh` value and `maxval` serve to assign the global binarization. It needs to be inside that interval; otherwise, it is assigned a 0 pixel or black. `Thresh` is recommended to be 0.1.

5. `OpenMask(mask, ksize=(23, 23), operation = 'open')`: This function only expects as input the mask to edit (the output of the `OwnGlobalBinarise` function) to perform opening and closing functions from the `OpenCV` library. Figure 4.4 shows the use of both functions and how they affect the binarized image.
6. `SortContoursByArea(contours, reverse = True)`: This function takes the clean binarized image and forms a bounding box around it, which will output the mask as close to the border as possible.
7. `XLargestBlobs(mask, topX = None)`: This function takes the output of the `SortContoursByArea` function and finds the largest contour in the image. This, in turn, outputs only the bigger mask. This means that any label (which represents a smaller contour) will disappear, and the bigger mask (the mammogram) will remain.
8. `InPaint(img, mask, flags = "telea", inpaintRadius = 30)`: This function takes as input the original image and the generated mask from the largest blobs function. The output restores the original image to the areas the mask image indicates.
9. `HorizontalFlip(mask)`: This function inputs the restored image and figures if it needs to be flipped. The correct orientation is the breast being on the left (i.e. facing right) and it being the right side up. i.e. When the mammogram is oriented correctly, the breast is expected to be found in the bottom left quadrant of the frame.
10. `clahe(img)`: This function applies the Contrast-Limited Adaptive Histogram Equalisation filter to a given image.
11. `Pad(img)`: This function pads a given image with black pixels along its shorter side into a square and returns the square image. This normalizes all images to be the same shape, and when they are resized, they do not lose their original shape. Figure 4.5 shows the result of the padding and later resizing process for the mamograms.

At the end of all these functions, a self-called function allows all the steps to be followed. In the same way as the cropping program, there are two ways of storing the images: one changes the original images and the other stores them in a different folder. The same recommendation of allocating enough space for the second option applies here.



(a) Opening.

(b) Closing.

Figure 4.4: Morphological transformation operations from the OpenCV library. Opening is the function of erosion followed by dilation, and Closing is dilation followed by erosion. Both of them are used to remove noise from binary images.

Next comes the size of the images. Some deep learning methodologies, especially CNN architectures, expect a normalized size for the input images since the first layer is set to receive only one information size, in this case, the exact size of pixels and the same channel of the colour. For the classification process, we left the images of the format RGB, and we established the desired size as 224×224 pixels since we explored the use of transfer learning. For this type of algorithm, this size is mandatory.

Since we wanted to distort the images as little as possible, we needed to use some padding not to change the final image's shape. The area of interest is not a square shape. After removing the labels and noise, the size is a square. So, the final step for the pre-processing for the classification stage is to pad the images so they are all the same size without unnaturally stretching the image. In Figure 4.5, we see the process of padding and resizing the image. In this case, the padding will not cause too much trouble in training the models since most of them use the dropout

parameter to help, which we will explain later.

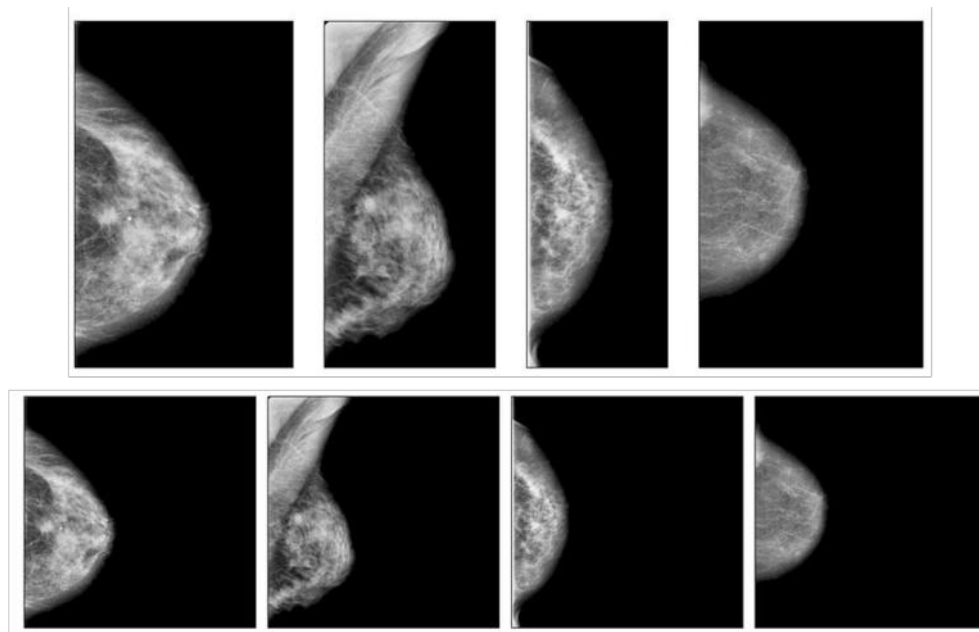


Figure 4.5: Process of padding and resizing the images to 224×224 pixels.

4.2.1.1 WAVELETS

We identified potential noise manipulation in our data set during our experimentation, further detailed in the Results (Section 5). Thus, part of our pre-processing aimed at noise removal from images using wavelet transforms, known for effectively localizing information in both spatial and frequency domains. The wavelet transform decomposed the image into varied scale components. A thresholding step minimized noise, preserving essential details, followed by an inverse transform to yield a denoised image, accentuating critical areas for medical examination.

It is vital to recognize that wavelet-based denoising, while powerful, requires precise tuning and validation for particular applications.

This code can also be found in Section A under the name of `wavelet.py`. In the same way that the binarization process is a big code, this is also way too extensive to explain line by line, so we are going to mention the three important

functions you need to consider for the wavelets to work and what aspects can be changed depending on the specific hardware features.

1. `WaveletTransformAxisY(batch_img)`: This function, as well as the transform for axis X, uses the library of Keras and the function of tensor slices and permute dimensions to apply the use of Texture classification using wavelet CNN.
2. `Wavelet()`: This function uses both transforms for axis x and y to form two decomposition levels. By experiment, we found that two decomposition levels were enough for our images since the example we found had four decomposition levels (each level adds more computational complexity).
3. `getwaveletcnnmodel()`: This whole function details the layers for the wavelet for two decompositions. This can be extended if more decompositions are needed.

4.2.2 BINARY CLASSIFICATION PROCESS

In machine learning algorithms, classification predicts a class or label for the input data. Depending on the class label we are working with is the classification algorithm we can use. That is not to say there is a road map to knowing what algorithm is indicated for specific problems. It is usually recommended that experiments take place to discover which algorithm and configuration work best for a given classification task.

The most common types of classification tasks we can encounter are:

- **Binary classification:** Typically, this type of classification involves a normal class, and another is the abnormal class. They are often assigned the value of 0 to the normal and 1 to the abnormal class labels. In the case of this

experimentation, the 0 will represent the normal images or images without anomalies, and the 1 represents images with an anomaly.

- **Multi-class classification:** This classification refers to tasks with more than two class labels. This classification is unnecessary since there can not be an image with and without anomaly simultaneously.
- **Multi-label classification:** The multi-label classification refers to the classification tasks with two or more labels. In this case, the difference between class and label is that with the labels, we can identify the different types of anomalies that we found in the classification portion of our methodology so that multi-label classification will be referenced again in the feature extraction portion of our method.

As mentioned before, we use binary classification for our method's classification stage. An output of 1 will refer to images with anomalies and 0 to normal images without any anomalies. Since this is a computer-aided diagnostic, the result of our model classifier will not be zero or one. Instead, we are using probabilities. A binary classification will typically return a probability between 0% and 100%. Since we need to go from the likelihood of the image having an anomaly to the actual decision about which class the image belongs to, we establish a **threshold**. The standard threshold for binary problems is right down in the middle. Any prediction below 50% will be interpreted as a normal image, while above that percentage will be considered in the anomaly class.

Now that the decision for the threshold has been made, how will we measure the performance of our classification model? When building a predictive classification system, we need metrics that reflect the classifier's performance. In the literature review, the most used metric is accuracy. Still, as we explain in the background section, other metrics, such as precision, recall, and F1, are also important to note since they can give us more information about the balance of our data set.

We have established now that the classification task is binary and the metrics we need to measure the performance. Now, we discuss the algorithms used to build our classification models.

4.2.2.1 CONVOLUTIONAL NEURAL NETWORKS

The literature review (Section 3) has a portion on deep learning methods for medical imaging. Popular tools used for breast cancer were CNN architectures, such as VGG-16, Alexnet, and Inception V3. The VGG-16 and Inception architectures are two of the most often used for transfer learning. The experimentation showed promising results that went from 80% accuracy to 95%. CNN architectures are divided into shallow and linear model types, such as Alexnet and VGG-16, and deep model types, like Inception and all the other architectures, that use built-in modules, which makes them more complex. Table 5.6 shows that transfer learning from VGG-16 is the most effective in keeping a slight difference between the developing and test sets' accuracy. Inception, however, only shows promising results in the training and developing sets but not in the test set. This behaviour could mean that there are better options than a deeper network, but it does not take ML algorithms off the table.

In our methodology (Section 4), we try the architectures mentioned in the literature reviews the same way they are used: first, by using the architectures and training the models from scratch with the images we have, and then by transfer learning. With transfer learning, the difference is that the weights of the architecture are already pre-trained with the data set of IMAGENET, which we mentioned in the background section.

The architectures were used with the same features described in their papers; however, they do not provide the code. We took the features from different pages and adapted the related features into Python code using the Keras library. All authors and their pages are listed in Section A. Here, we explain some features that can be

changed to adapt to specific hardware requirements.

- **Alexnet:** We need to change certain parameters for this example code. If we go to line 25, the image shape can change to smaller in case the memory load is too heavy. The number of classes also needs to be specific to the number of classes we are trying to classify. In our case, it changes to two. In the last layer, the number of neurons in the dense and convolutional layers can also be downsized if hardware constraints are important. We cannot recommend a specific number for these features, so it is more of a trial and error.
- **Vgg:** All of the changeable features from the Alexnet architecture also apply to Vgg (number of neurons in layers, size of the image, and in the case of the Vgg, we can also choose between the 16 and 19 architecture depending on the amount of data we have available and hardware restrictions).
- **Inception and ResNet50:** We will not recommend using both these architectures if there are hardware limitations. Alexnet and Vgg have the advantage of being linear architectures, and since these two networks use modules and recursive/residual learning, it is more complex to avoid heavy computational complexity. Also, for these networks, there needs to be a substantial amount of data available for training since the architecture complexity may lead to overfitting for small datasets.

Using a pre-trained structure saves a lot of processing power and time on training, which is desirable if we train the model constantly to add more images. It is also highly regarded in the literature review because it has yielded the best results.

For the transfer learning part of our process, we explored the Vgg architecture of ResNet50 since Alexnet is not commonly used for transfer learning for its simplicity. In Section A, we have mentioned the code for transfer learning, but in Listing 4.2, we show the important code snippet that could be modified depending on the resources.

```
1 from tensorflow.keras.applications.vgg16 import VGG16
2
3 vgg_model = VGG16(weights='imagenet', include_top=False,
4     input_shape=(Lng, Hg, 3))
5
6 # Freeze four convolution blocks
7 for layer in vgg_model.layers[:15]:
8     layer.trainable = False
9
10 x = vgg_model.output
11 x = Flatten()(x) # Flatten dimensions to for use in FC layers
12 x = Dense(512, activation='relu')(x)
13 x = Dropout(0.2)(x) # Dropout layer to reduce overfitting
14 x = Dense(256, activation='relu')(x)
15 x = Dense(2, activation='sigmoid')(x) # Softmax for multiclass
16
17 transfer_model = Model(inputs=vgg_model.input, outputs=x)
18
19 transfer_model.compile(loss="binary_crossentropy", optimizer="sgd",
20     , metrics=["acc"])
21
22 H = transfer_model.fit(train_datagen.flow(trainX, trainY,
23     batch_size=BS),
24     validation_data=(testX, testY),
25     validation_steps = 1000,
26     steps_per_epoch=1000,
27     epochs=EPOCHS)
```

Listing 4.2: Python code to use transfer learning from the Keras library. This example shows the Vgg16 net.

From Listing 4.2 in line 1, we import the Vgg-16 architecture from the Keras library, but this can be changed to import several different trained architectures such as Vgg, ResNet, Inception, etc.¹

¹For more pre-trained transfer learning architectures and examples of the use of each one, please refer to the Applications module of Keras <https://keras.io/api/applications/>

We freeze four convolutional blocks for this experiment and make changes to the last five layers of the architecture. In the same way, we explain how they can be downsized, a different number of neurons, the size of dropout, and the number of iterations and epochs can be adjusted to the amount of data we have or resources available. The optimizer can also be changed depending on the task, but in the Keras documentation, they have `sgd` as the primary recommendation.

Similar features can be changed for the ResNet50 and Inception architectures.

In the results (Section 5), we discuss the pros and cons of the final metrics for this experiment. Since the final performance was not what we had hoped for, we began to see alternatives to improve our accuracy and overall performance, considering the results in other reviewed works. For this, we explored the statistical part of machine learning and developed ensemble algorithms.

4.2.2.2 ENSEMBLE LEARNING

Ensemble learning refers to the type of algorithms that combine the predictions from two or more models. For this broad description, we can say there are unlimited ways to present this algorithm. However, we explore three main classes of an ensemble, the most common and discussed in machine learning work for classification.

It is also important to note that, despite looking to use machine learning algorithms, we did not want to leave altogether the deep learning methodologies we explored in the past, so in our case, we combine the use of deep learning with this type of ensemble to help improve our performance. In the literature, the terms for machine learning and for deep learning are sometimes used interchangeably. However, as established before in this work, in many applications, deep learning is “too big a hammer” for problems in which we have fewer data or limited computational resources. Ensemble methods present a powerful alternative to building a robust model in these cases.

Ensemble learning is a general technique often used to seek better predictive performance of machine learning algorithms, combining multiple models' predictions. An example of how an ensemble learning algorithm would look from a real-life perspective can apply to how a real team of doctors handles patient data to make a diagnostic decision. At the start, there is some data from patients needing a diagnosis. This data is handed out to several experts in the area. Each of them will analyze separately and independently suggest a diagnostic. Ultimately, the last expert or team leader considers all opinions and adds his own to the final diagnosis.

Thus, an *ensemble learning algorithm* can be defined as an ML algorithm that seeks to improve the performance of a task by using multiple estimators.

However, traditional machine learning is not often used as a method for computer vision, less so for medical imaging. This is due mainly to the complexity of the data. However, ensemble learning does not constrain itself by only being able to use machine learning as its component. For this methodology, we combine some of the more computationally accessible deep learning features with traditional machine learning algorithms in an ensemble learning structure.

We explain each ensemble structure shortly, but first, we need to address how we use deep learning to deconstruct the data to train a machine learning model.

The process is called feature extraction, which uses the same concept as transfer learning. Feature extraction is a part of the dimensionality reduction process, in which an initial data set is divided and reduced to a more manageable format. This process helps to get the best feature from the data sets by selecting and combining variables into features. These features are easy to process but can still accurately describe the actual data set.

In this work for the feature extraction, we focus only on using the weights of the Vgg pre-trained structure since it was the best-performing architecture for the transfer learning experiment. This code can also be found in Section A; in this case, no changes are advised since it does not use too much computational power. The

only constraint to consider is the feature file's size. Before running this code, it is necessary to have at least 20% of the size of the original dataset.

- **Bagging or bootstrap aggregation**

Bootstrap aggregation, or bagging for short, is an ensemble learning technique that seeks diverse ensemble features by varying the training data. It is a statistical technique that averages the estimates from multiple small data samples using estimates from a complete population. It is constructed by drawing observations from an extensive data set and returning them after they have been chosen. This method is called sampling with replacement.

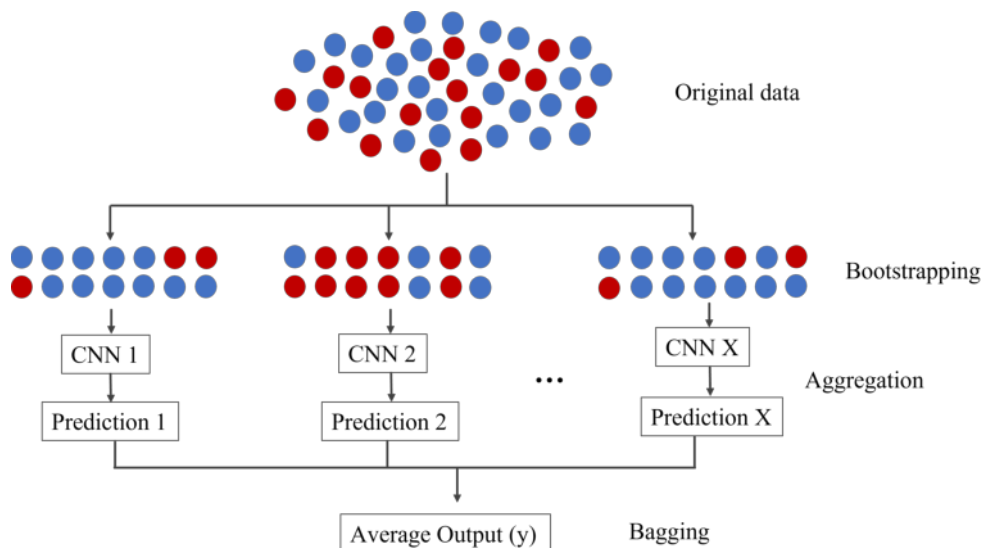


Figure 4.6: Example of the bagging algorithm. This example shows the CNN structures of Alexnet, then VGG-16, and lastly with Inception. Following the function of an ensemble, the final result is finally passed on in a final voting.

Figure 4.6 shows the Bootstrap aggregation graphically. Multiple equally sized subsets with replacements are taken from the original data, and a CNN with the same architecture runs on each one. Finally, the outputs from all the separate models are aggregated and voted into a single prediction. The experiment with Alexnet considers the same hyperparameters shown in their original publication.

For VGG-16 and Inception, we used the pre-trained weights using IMAGENET. We performed all the tests without compromising the computational complexity of other deep learning architectures by the nature of this ensemble algorithm.

Since the bootstrap method is used to estimate the quantity of the population, it takes small samples, calculates statistics, and takes the average of those statistics. Before fitting the model or tuning the hyper-parameter, data preparation must occur within the data sample's for-loop. This process is used to avoid data leakage, where the model uses knowledge of the data set to improve the model and causes overfitting.

The goal for the use of the bagging ensemble is to average the predictions across models, and it is expected that the resulting prediction will improve by having more than one model fit on the training data.

- **Stacking ensemble learning**

Stacking involves fitting different ML models or estimators. Ultimately, it does not just average the results but uses the predictions from other models to feed a new model, which takes those predictions and builds its own based on them and the model chosen for this last step. In this case, the models have their terminology where the zero-level models are the ones used in the first step of the ensemble and are known as the weak learners, and the first-level model is the one that combines the predictions. It often uses the two-level hierarchy but can also use many more layers.

In this case, transfer learning can take the stacking ensemble into the deep learning perspective. As mentioned in the Background Section, transfer learning is often used in the literature to consider pre-trained weights and architectures to lower the computational load and facilitate the use of more complex architectures.

Since these transfer learning architectures have been trained with natural images, not medical images specifically, how much they help the classification process

is often a point of discussion. In this work, the first level of the stacking process uses transfer learning with the VGG-16 architecture to apply what is known as feature extraction. It allows harnessing the power of a deep-learning algorithm and complements it with other models. Then, in place of VGG-16, the transfer learning will take place with the Inception architecture to compare the best.

It is essential to mention that, in the same case as the bagging ensemble algorithms, transfer learning was not considered for the Alexnet architecture because it is regarded as a low-complexity network that shows better results when trained from scratch.

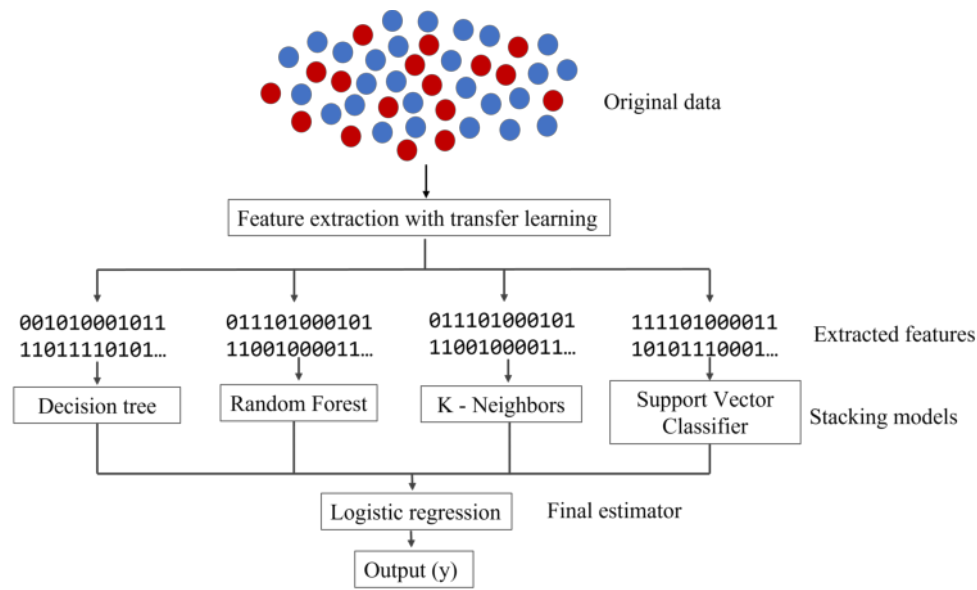


Figure 4.7: example of the stacking ensemble, where each model represents the weak learner used for intermediate predictions. In the end, consider logistic regression as a final estimator.

Using pre-trained weights to extract features from images produces a vector of numbers that can be used as an input of a new ML model since they describe the specific features of the image in a format that is more “digestible” for traditional ML algorithms. Stacking can come with more processing time, so we use feature extraction first to make the data more manageable.

As shown in Figure 4.7, we use popular ML algorithms as the receiving models of the extracted features like decision tree, random forest, k - nearest neighbors, and support vector classifier. These models can be found in the *sklearn* python library (Pedregosa et al. [2011a]).

For the stacking process, we combined the use of DL and ML algorithms by using DL only at the beginning of the process. In the end, for the process of averaging results, we use a logistic regression estimator, that functions as a final prediction. This is known as stacked generalization, and it is a method that is often used to reduce the biases of past estimators by combining them. The use of the logistic estimator at the end aligns with the decision of balancing our data set.

- **Boosting algorithms**

This type of ensemble works differently than the bagging and stacking algorithms. In Figure 4.8, the models are fitted and added to the ensemble sequentially. In the sequence, a model attempts to correct the predictions of the previous one, and so on. This ensemble aims to develop a strong learner at the end of all the iterations. The main difference between this boosting algorithm and ensembles like bagging is that, in this case, boosting takes knowledge from the previously generated classifiers and increasingly focuses on the miss-classified elements. Bagging has an individual set for each iteration, so there is no such “learning”.

In this work, we combine this ensemble with deep learning in the same way as in stacking. It means we perform feature extraction by transfer learning before we feed the models into the boost. Again, the architectures for feature extraction are VGG-16 and Inception. Although we have mentioned combining weak learners to turn them into strong learners, in this ensemble, we focus on Adaptive boosting (AdaBoost), one of the first successful boosting approaches to be implemented in classification problems.

In Figure 4.8, the weak classifier for AdaBoost is a decision tree. This algorithm

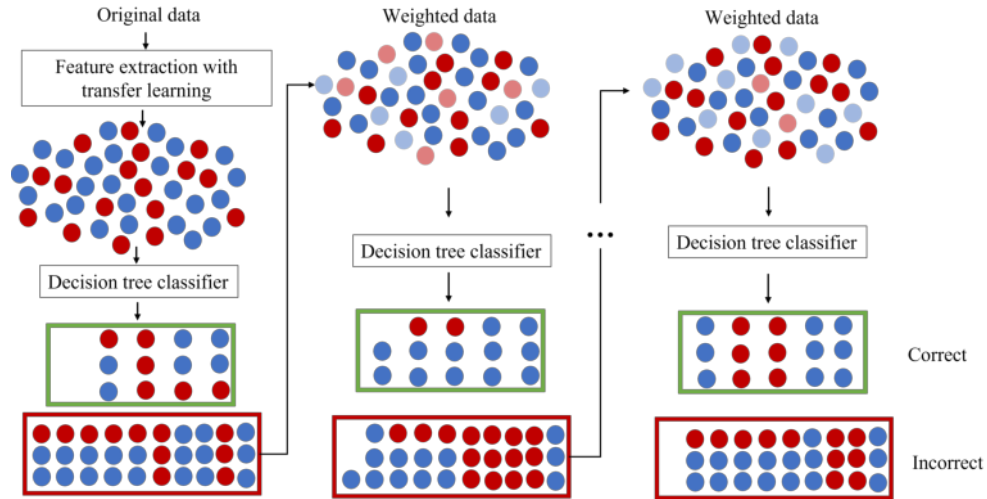


Figure 4.8: example of the boosting algorithm, where a decision tree classifier represents the weak learners. This process is set to be iterated until the complete training data fits without error. The other stop function is a specified number of estimators. In this case is set to 200.

is the most common weak classifier for AdaBoost, and in the training process, the user needs to set the number of iterations.

The desired output for this section is the division of the images into normal images and images with one or more anomalies. We present the accuracy of these experimentations in the results section. Following the methodology, we expect these results to be the input images for the next stage, the segmentation process.

The ensemble codes are referenced in Section A.

4.3 SEGMENTATION PROCESS

The goal of this next stage is to use all the images that present one or more anomalies and segment the anomalies to mask the possible position of the anomaly inside the image. For this part of the methodology, we are again using the CBIS-DDSM data set to perform some of the methods and some of the images from the YERAL data

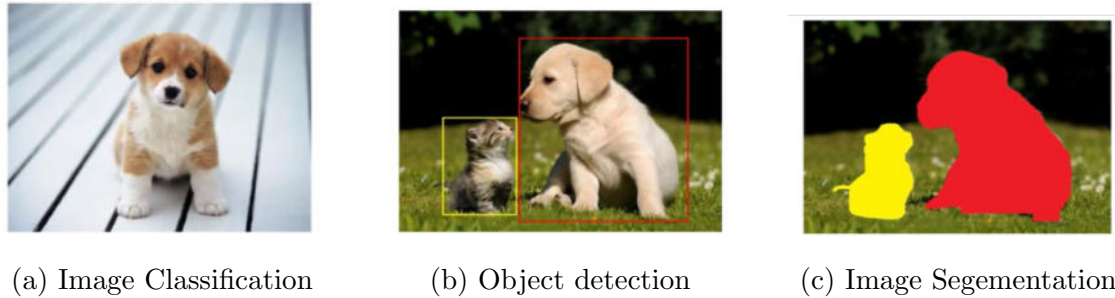


Figure 4.9: Example of the difference between image classification and different segmentation types.

set for the experimentation with the results since, at the time of the experimentation of this thesis, there is no segmentation annotation or masks made for the data set. We only have the ground truth made by other colleagues, and they are used only as guides since they are not verified.

The CBIS-DDSM data set also contains the segmented binary mask of all the anomalies found on each image. This mask is also known as the ground truth. If we use this image as an overlay, we can locate where in the image the anomaly is detected.

Same as in the case of the classification stage, the segmentation process has different types of segmentation. Since the classification process tells us what is in the image, the segmentation involves localizing, detecting, and segmenting the object of interest.

The two primary forms we have to locate the object of interest in an image are bounding boxes or masks. In Figure 4.9, we see the example of the segmentation between the bounding box and mask. We have four main techniques commonly used in computer vision problems to locate objects.

1. **Object localization:** This involves locating the class object or label from an image. This involves a cropped bounding box centred around the object. In this case, this type of segmentation usually comes with the classification label

of the object.

2. **Object detection:** This involves detecting multiple object classes in one image. Following a pattern similar to object localization, this also uses bounding boxes around every detected object in the image.
3. **Semantic segmentation:** This type of segmentation predicts the object class for each pixel in the image, which can be two or more object classes. All objects need to be labelled for this to work, and the output of this method will be the predicted mask.
4. **Instance segmentation:** This is the more elaborated version of semantic segmentation since this method can identify and differentiate two instances of the same object class. For example, if we have multiple anomalies that fall into the same type of malignancy, they will have the same label but different masks.

Unlike the classification stage, the segmentation process has another way of determining the model's performance. The most commonly used metric is the **Intersection over union (IoU)**, which measures the performance of object detection and deep learning algorithms.

The results are much the same as the accuracy for binary classification in that we have a result between 0 and 1, whereas the closer is to 1, the better the accuracy of the mask or bounding box. For the IoU, however, it is more complex. For example, in the case of segmentation, it is very unlikely that the coordinates of our predicted mask will match precisely the coordinates for the ground truth. This could be because of the feature extraction method we use or the sliding window size of the image, so instead of taking it as true or false positive/negative outcomes, we define the evaluation metric that rewards the predicted bounding boxes or masks for overlapping more heavily with the ground truth of the image.

In Figure 4.10, we have an example of how the overlapping on the ground truth

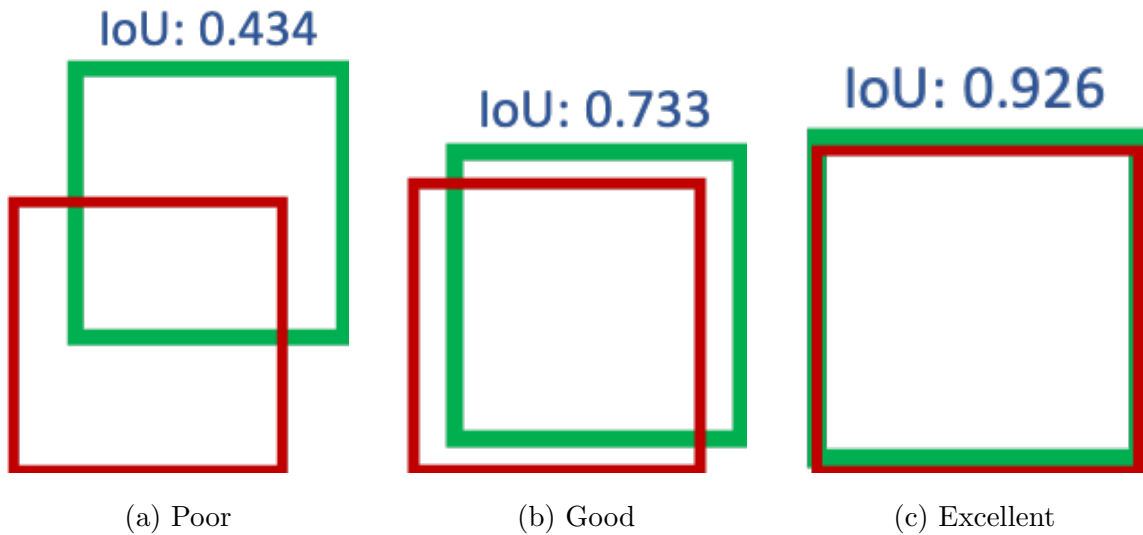


Figure 4.10: Example with bounding boxes of the intersection over unions (IoU).

computes the IoU. As we can see, a good IoU is closer to 1 than 0.

The last main difference between this stage and the classification stage is that we need other things from the data set. Besides the actual images, we need the bounding boxes or masks associated with the image. These images have the ground truth on the exact coordinates of the object in the image.

4.3.1 PRE-PROCESSING

The pre-processing for this section follows much the same process as the pre-processing from the classification stage. In this case, we will also need to apply the padding method to the images with the ground truth so they are the same size. Another thing added to the full images is the Clahe filter to improve the contrast of the images. This was not added to the classification method because previous experimentation with these enhancement tools made the performance poorer, so we discarded it.

The Clahe enhancement was introduced in this section because the majority of

the images are from the CBIS-DDSM data set, and some of those images have a poor contrast on the breast tissue, which can result in a lack of texture and meaningful difference between the mass abnormalities. With this enhancement, we leave these images as similar to our target data set as possible.

Clahe stands for Contrast Limited Adaptive Histogram Equalization and is often used in computer vision to enhance the contrast of grayscale images. Because this enhancement can also focus on the background noise of the image (although it appears black, sometimes it is not complete), we need to do this enhancement after the normalization, binarization, and removal of labels.

For the masks, there is no need for pre-processing other than the padding of the image since the images are binary, and their only purpose is to serve as labels to calculate the error during training.

In Figure 4.11, we see the example of the masks that came with the data set. Lastly, for us to use our method, it is necessary that if an image has several masks (because it has several anomalies), we must stack them together in one single summed-up mask since this stage does not require that we classify what type of malignancy, it is not necessary to differentiate them. The end goal of this stage is to create a new set of ground truths for new images and have a cropped set of the anomalies to identify the type of malignancy in the next stage. Figure 4.12 shows the example of the summed mask.

4.3.2 CONVOLUTIONAL NEURAL NETWORKS

For this next segmentation model, we consider the experimentations and results made from the binary classification portion of the methodology. In that case, the VGG-16 with transfer learning with and without the help of the ensembles showed better results than the other CNN architectures. Following that good performance, we explore using the same architecture with the U-Net architecture.

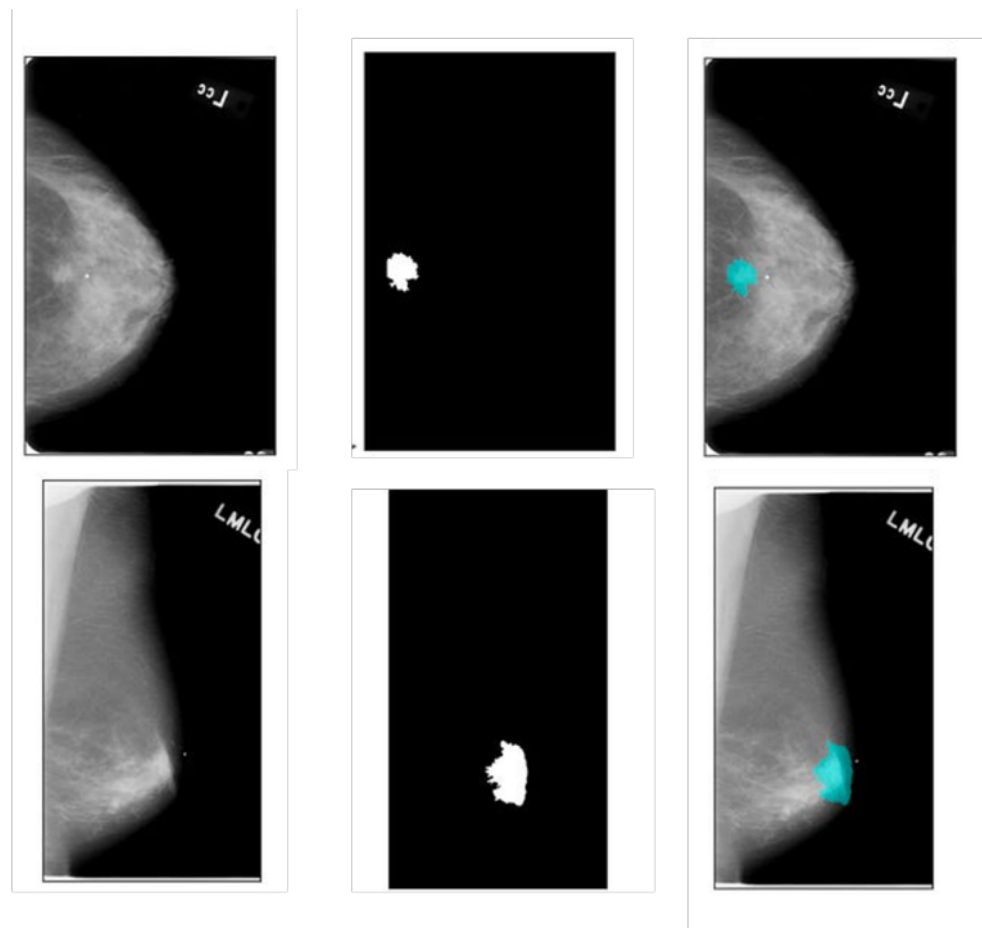


Figure 4.11: Example with bounding boxes of the CBIS-DDSM data set.

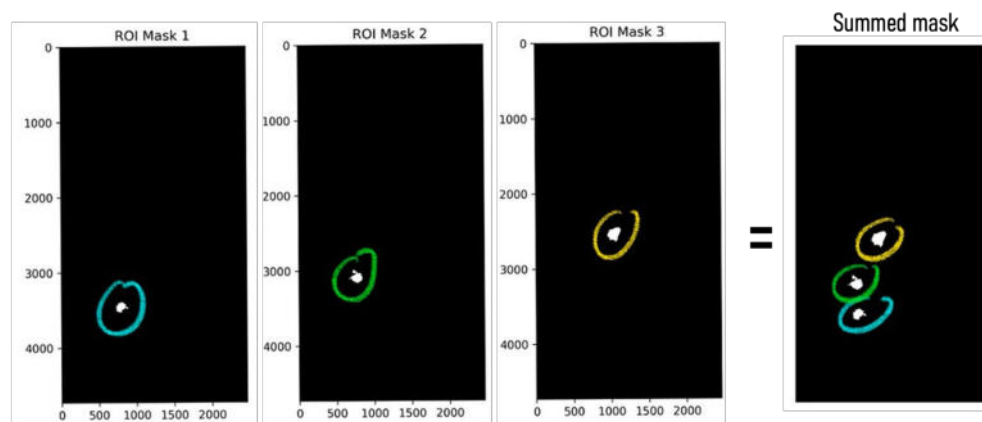


Figure 4.12: example of the final summed mask for the segmentation process.

CHAPTER 5

RESULTS

This chapter presents the diverse results of our extensive experimentation, exploring the right fit between statistical, machine, and deep learning. As we explained in the Background (Section 2) and Literature Review (Section 3), these distinct yet connected domains under the umbrella term of artificial intelligence hold the potential to improve each other, depending on the nature of the problem being addressed. Our focal point has been the enhancement of the classification model’s accuracy, a journey initiated in Berrones-Reyes [2019]. This chapter delineates the various milestones and pivotal moments we encountered during our research, beginning with stating the problem we set to address and then following with the different approaches we employed to navigate each challenge.

5.1 CLASSIFICATION

In this work, we aimed to enhance the classification model initially developed in our master’s thesis. The primary goal of the original project was to address the needs of a specific hospital that provided us with annotated images. The hospital sought a cost-effective tool to assist doctors in swiftly and accurately identifying image anomalies. This need arose due to the massive volume of annual breast cancer data. Governmental efforts to combat this disease primarily consisted of designating dates

for awareness and facilitating public check-ups.

However, the effectiveness of this system was challenged by the scarcity of specialists available to process the influx of information. Waiting periods lengthened, and measures introduced to mitigate this issue proved inadequate. As discussed in the Literature Review (Section 3), the criteria used to prioritize patients were based on global statistics and general information, leading to impractical generalizations when comparing developing and developed countries. Consequently, the incidence of breast cancer remained high.

In addressing these issues, we followed the state-of-the-art, which advocated for deep learning, to build a reliable classification model. For a detailed overview of the various iterations of our experiments, we refer to the comprehensive work by Berrones-Reyes [2019]. This document presents our new experimentation’s final outcomes and starting point. Table 5.1 displays the best models chosen for further study. In most instances, while training and validation sets achieved near-perfect accuracy, the test set experienced a notable decline in all other metrics. We recognized this as a clear sign of overfitting and aimed to correct this error.

It’s crucial to clarify that the models outlined in Table 5.1 are based on simpler architectures of convolutional networks. At that point, our focus was more on the varied hyperparameters of the layers than on the actual architecture. The parameters that we found were more influential on the best results where the optimizer, and the size of the image. While this approach enhanced our understanding of CNN architectures, it became clear that a more complex network was necessary for substantial improvement in evaluation metrics without succumbing to overfitting.

Subsequently, we explored more intricate architectures of convolutional networks. Given our ongoing computational capacity limitations, our exploration ranged from simpler to more sophisticated models. During this phase, we employed the initial hardware configuration, as described in the Methodology (Section 4). With this configuration, we were only able to achieve results up to the VGG-16 architecture.

Table 5.1: Results of the best models for binary mammography classification extracted from the complete table by Berrones-Reyes [2019]. The number represented in the column model refers to the parameters of each model.

	Accuracy			Other metrics		
Model	Training	Validation	Test	Precision	Recall	F1
1211	99%	99%	70%	80%	83%	81%
1231	99%	99%	69%	80%	81%	81%
1215	95%	97%	68%	80%	81%	80%
1315	94%	97%	68%	79%	81%	80%
1226	99%	99%	70%	79%	85%	82%
1136	99%	99%	66%	79%	80%	79%
2221	99%	100%	63%	79%	75%	77%
2131	99%	99%	72%	80%	86%	83%
2115	96%	97%	61%	78%	72%	75%
2135	96%	97%	61%	78%	72%	75%
2226	99%	99%	65%	78%	79%	78%
2236	99%	99%	64%	79%	75%	77%

Table 5.2: Factors of Lenet5 models.

Optimizer	Image size	Data set	Model	Accuracy	Time epoch	Temp (C)
Adam	80×200	Minimias	m001	0.74	43 seg	65 to 69
Adam	160×400	Minimias	m002	0.74	158 seg	65 to 69
Adam	80×200	DDSM	m003	0.87	44 seg	65 to 69
Adam	160×400	DDSM	m004	0.88	168 seg	65 to 69
Adadelta	80×200	Minimias	m005	0.75	39 seg	63 to 67
Adadelta	160×400	Minimias	m006	0.72	193 seg	63 to 67
Adadelta	80×200	DDSM	m007	0.86	38 seg	63 to 67
Adadelta	160×400	DDSM	m008	0.86	206 seg	63 to 67

We understand now that this is likely due to the linear properties of the three architectures (Lenet5, Alexnet, and VGG). Table 5.2, 5.3, and 5.4 display the name of the optimizer, the two different sizes we translated from previous experimentations, and the dataset used for each experiment in their columns.

At this juncture, we were also investigating the discrepancy in accuracy between the training and validation sets compared to the test set in our first round of experiments. A common suggestion for addressing discrepancies in computer vision problems was to separate the different datasets and assess if significant differences existed among the three datasets we mentioned in the Methodology (Section 4), Minimias, DDSM and YERAL. Accordingly, we added a column in Tables 5.2, 5.3, and 5.4 to denote the dataset we used. We focused solely on accuracy to compare the differences and considered both the time taken to train and the temperature of the hardware components. One constraint was that, since our computer was not designed to handle intensive computing tasks, the temperature was not allowed to exceed 80 degrees Celsius.

For Tables 5.2 and 5.3 we see that the difference in accuracy temperature and time favors Lenet5. While in Table 5.4 we were only able to experiment with one of the dataset (the smallest) until we reached our threshold of temperature.

As we can see in all the previous tables of results, none of them have information about the YERAL dataset, which is our target dataset that contains real clinical

Table 5.3: Factors of Alexnet models.

Optimizer	Image size	Data set	Model	Accuracy	Time per epoch	Temp (C)
Adam	80×200	Minimias	m009	0.60	222 seg	70 to 79
Adam	160×400	Minimias	m010	0.75	436 seg	70 to 79
Adam	80×200	DDSM	m011	0.76	235 seg	70 to 79
Adam	160×400	DDSM	m012	0.73	458 seg	70 to 79
Adadelta	80×200	Minimias	m013	0.70	139 seg	70 to 76
Adadelta	160×400	Minimias	m014	0.69	486 seg	70 to 76
Adadelta	80×200	DDSM	m015	0.76	144 seg	70 to 76
Adadelta	160×400	DDSM	m016	0.75	513 seg	70 to 76

Table 5.4: Factors of VGG-16 models.

Optimizer	Image size	Data set	Model	Accuracy	Time per epoch	Temp (C)
Adam	80×200	Minimias	m017	0.75	1,088 seg	80 to 90

data. DDSM and Minimias are known as benchmark datasets. In Figure 5.1, we see one example of the accuracy reached when training with Alexnet, but with all three architectures, it showed similar results. As we can see, the loss is way to high, and the accuracy never goes up the 50% mark, which is basically a random prediction.

These results marked our first checkpoint, where we had to consider alternatives to see what was happening with our images. Since our experimentations indicated that both of the benchmark datasets performed accordingly, we began to look into the pre-processing of our target dataset to see if there was something we could do to improve the accuracy.

We began to look into some topics of statistical learning and landed on the use of wavelets. This is explained in the Methodology Section 4, so here we explain how we used it and how it helped us solve this problem.

Just to set thing into perspective, we have Figure 5.2 where we see that the accuracy and loss of both datasets gets stuck in the random point, which means that it is a 50/50 chance it would classify correctly in real life. We have a score of 0.56 on accuracy and 0.72 on loss for the training set, and in the test set we have a 0.51

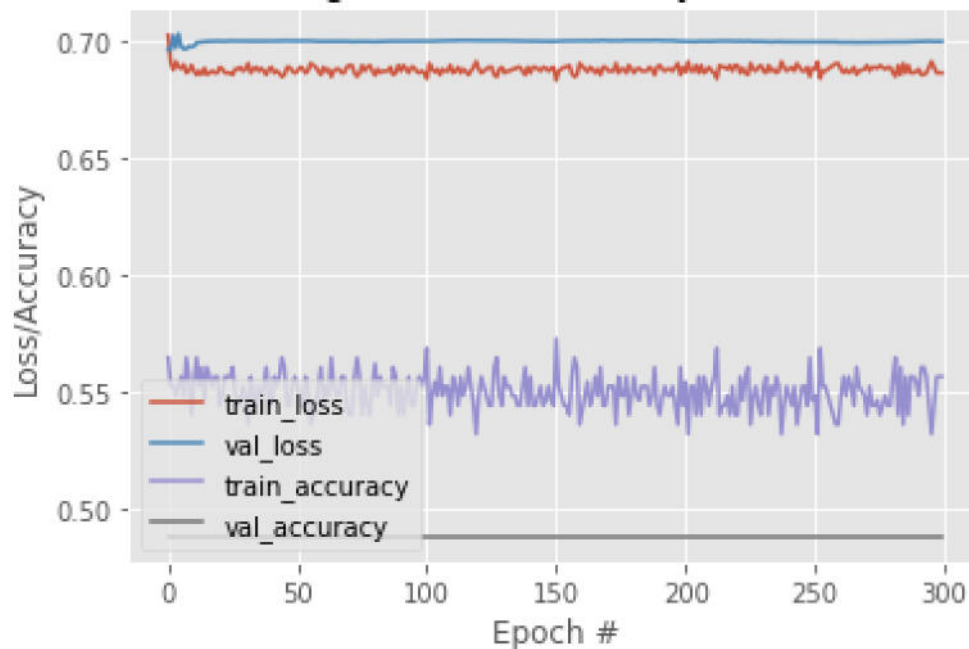


Figure 5.1: Training loss and accuracy for YERAL images.

in accuracy and 0.87 in loss. Then, we train our model with our modified Wavelet transform.

Digital image textures tell us how color or pixel intensity spreads out in space. Different patterns in this spread match up with different looks and feels of the material in the picture. Wavelet scattering moves the image through several wavelet transforms, nonlinear actions, and averaging processes (Hastie et al. [2009]).

This concept in particular, was beneficial since it gave us the idea we needed to fully understand how we could use the concept of wavelets for our classification problem.

Possible advantages of using wavelets:

- Wavelet scattering is the equivalent of a deep convolutional network. Some of the examples we investigated are very similar to the pooling method since it takes some of the features of the images to take forward.
- It has proved to yield stable representations against deformations and robust

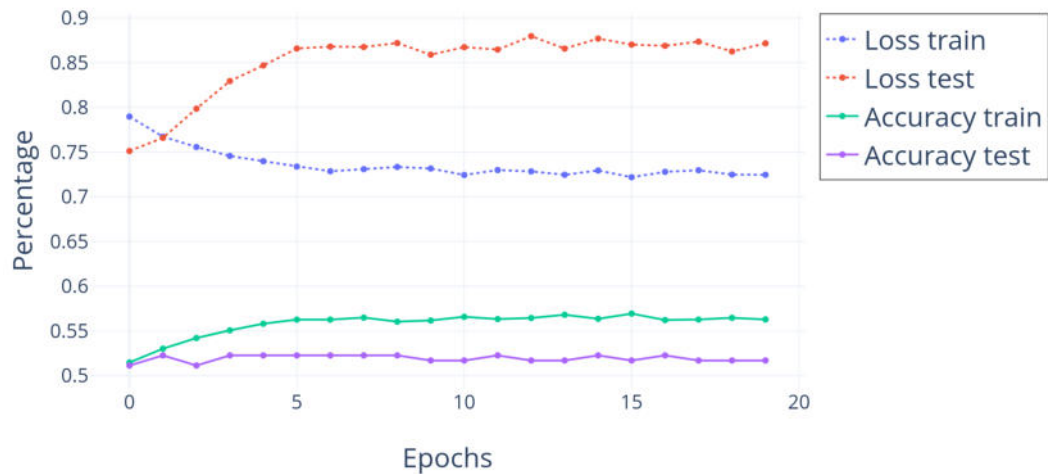


Figure 5.2: Training loss and accuracy for YERAL images with the Alexnet architecture.

to noise.

In Figure 5.3, we see how the accuracy finally escapes the random state it was stuck in, and the loss also starts to behave normally¹.

After this pre-processing, we set out to improve the primary objective again, which was classification accuracy. From this point forward, we change the hardware specification from the first to the second one described in the Methodology (Section 4).

¹The whole experimentation and moving of parameters can be seen in https://github.com/mayraberrones94/Tesis_codigos

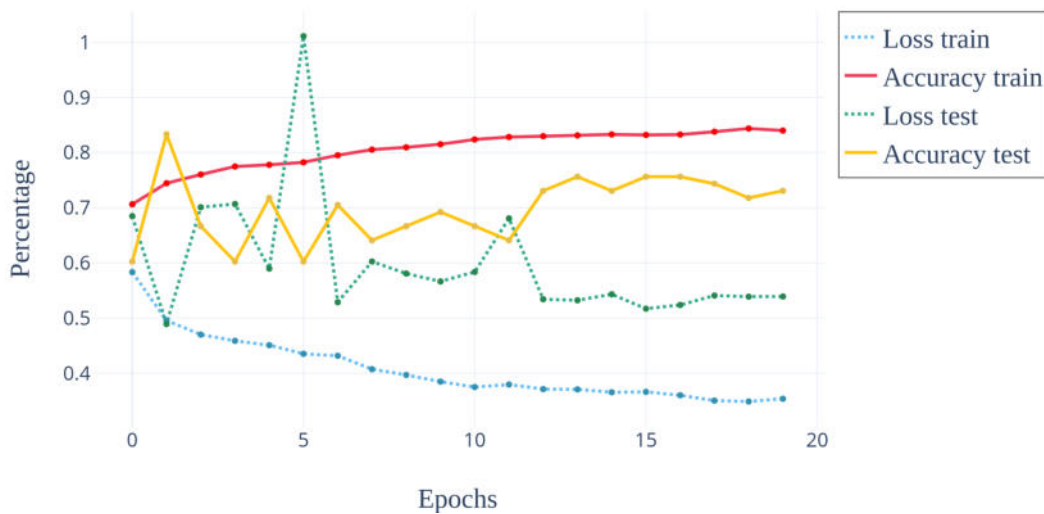


Figure 5.3: Training loss and accuracy for YERAL images after being subjected to a wavelet processing.

Table 5.5: Using the most popular metrics for validation data performance, we compare the use of an unbalanced with a balanced data set using the Alexnet architecture with the same weights described by Krizhevsky et al. [2012].

	Unbalanced Data Set		Balanced Data Set	
	Normal	Anomaly	Normal	Anomaly
Precision	64%	90%	82%	86%
Recall	82%	78%	83%	85%
F1 score	76%	73%	86%	82%
Accuracy	79%		84%	

Following common practice of computer vision problems, we need to train a model with a data set that represents well the features of a data set they may encounter in a real-life setting. In many deep-learning medical imaging studies, surveys show that most do not use genuine clinical data for external validation. In the rare cases that do, the model’s performance is often tested against the same

sample (Liu et al. [2019], Varoquaux and Cheplygina [2021]). Our study focused on correctly using data sets for training, development, and testing.

Surveys in medical imaging highlight the DDSM (Digital Database for Screening Mammography) as the primary benchmark for mammography (Liu et al. [2019], Litjens et al. [2017]). It has 2620 mammography studies of normal, benign, and malignant cases. However, the original DDSM, hosted by the University of South Florida, used an outdated lossless JPEG format (.LJPEG). This was updated in the improved DDSM, which uses the .PNG format (Lekamlage et al. [2020]).

We also utilized the private YERAL dataset from a Mexican hospital, reviewed by FUCAM (Fundación de Cancer de Mama). FUCAM is a leading nonprofit in Mexico and Latin America, specializing in breast cancer care in Mexico City. The YERAL dataset includes 641 images with confirmed anomalies and 302 without.

In this work, our “test set” only includes images from the YERAL data set, aligning with our goal to offer hospital experts a robust computer-aided diagnosis tool. Adhering to standard practices, we separated the training and development sets (Ng [2019], Chollet [2021], Brownlee [2016], Raschka and Mirjalili [2019]). Many recommend that training and development sets should encompass some images the model will later classify. Given our data constraints, the test set features 50 images with anomalies and 50 without, all from YERAL.

By merging the updated DDSM and remaining YERAL images, we had 2594 non-anomalous and 8401 anomalous images. This imbalance led to a bias, resulting in excessive false positives for non-anomalous images. As accuracy and F1 score are key metrics in medical imaging, balancing the data set was crucial to mitigate bias and enhance accuracy (Varoquaux and Cheplygina [2021], Ng [2019]). We randomly excluded some anomalous images (from DDSM only), leading to a balanced set of 2594 normal and 2900 anomalous images. Table 5.5 highlights the benefits of this balance. Henceforth, the balanced training and development sets are termed DDSM_YERAL, and the separate test set as YERAL.

Table 5.6: Performance results on the more popular architectures for image classification. Rows 1, 2, and 4 are the results introduced in the Literature Review Section. Rows 3 and 5 are the architectures that combine the use of transfer learning, and uses the unfreezing method for the last pooling layers.

	DDSM_YERAL				YERAL			
	Training Set		Developing Set		Test Set			
Model	Loss	Acc	Loss	Acc	Acc	Prec	Rec	F1
Alexnet	0.36	82%	0.31	84%	60%	57%	76%	65%
VGG-16	0.39	79%	0.35	82%	52%	51%	90%	65%
VGG-16 TF	0.25	87%	0.28	87%	79%	82%	74%	77%
Inception	0.35	81%	0.38	83%	57%	53%	93%	69%
Inception TF	0.39	80%	0.55	71%	55%	52%	90%	66%

Having defined the datasets used for training and testing, we began experimenting with prevalent CNN architectures identified in the literature review. Table 5.6 showcases the results from the training and validation sets, aligning with anticipated strong performance from these architectures. However, accuracy results shifted dramatically when tested with the target data set, YERAL.

Table 5.6 also presents the F1 score for the target data set, a metric frequently cited in the literature review due to its distinctive property of balancing error class results. As discussed in the Background Section 2, true positives, false positives, true negatives, and false negatives each uniquely influence metrics. Both accuracy and F1 scores indicate that the models are not performing optimally on the target dataset.

Figure 5.4 presents a visual depiction of each model’s behaviour, highlighting VGG-16 with transfer learning as notably stable compared to the rest.

A pivotal point in our investigation arose when our literature review and exploration of state-of-the-art image classification models indicated that these archi-

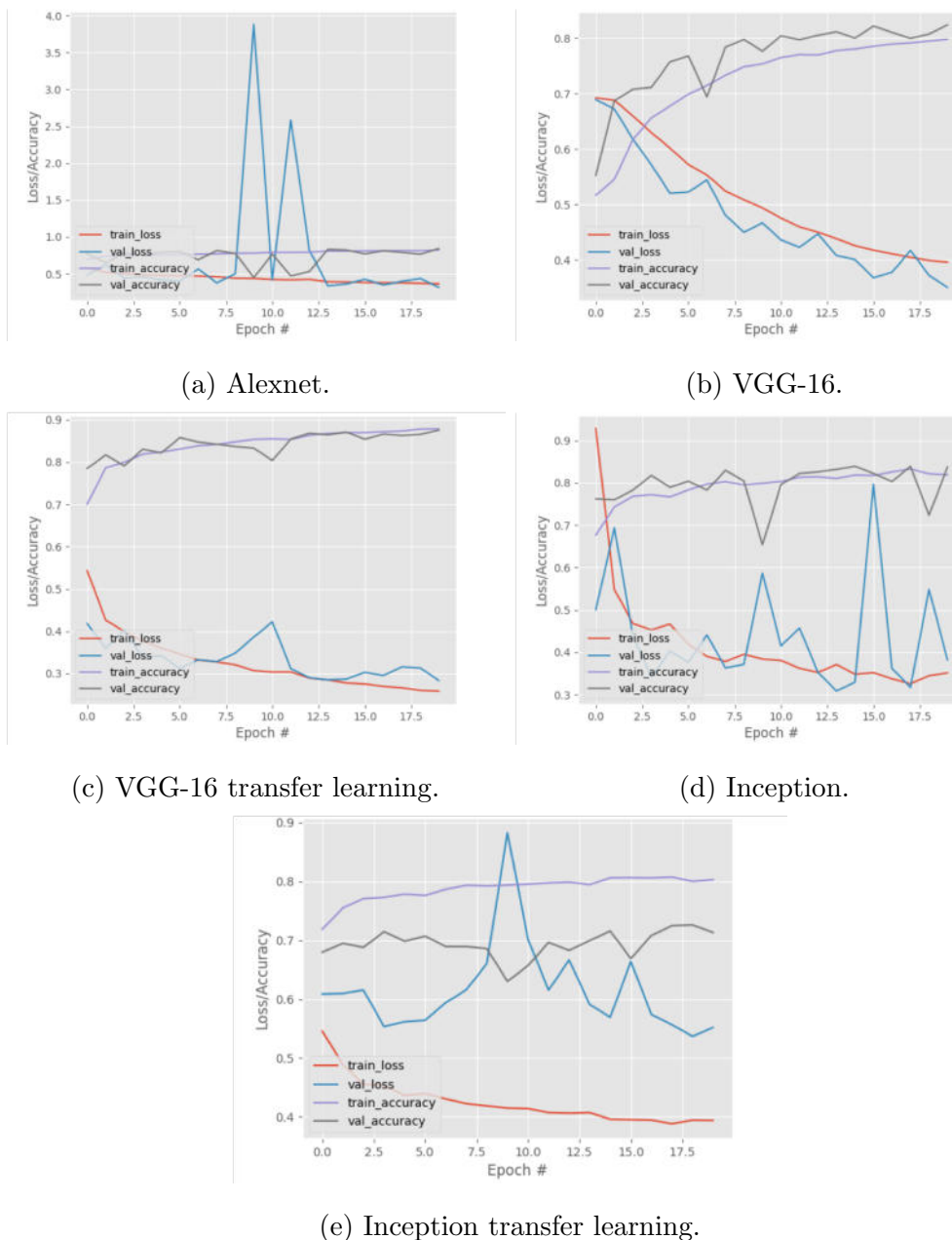


Figure 5.4: Graphs of the performance of the popular CNN architectures tested for this work. On each image we have the accuracy and loss for the set epochs, showing the validation and training outcome. Subfigures (a), (b), and (c) are related to the classical architectures, while subfigures (c) and (e) correspond to the ones with transfer learning.

tectures were optimal for adept classification within a complex dataset. Guided by the checklist necessary for training a proficient deep learning model, transfer learning emerged as a pivotal strategy, especially given our dataset—while more robust than our starting point—was significantly smaller than expansive datasets like IMAGENET.

Concern arose post-analyzing the results in Table 5.6, where the peak performance hovered just below 80%. This prompted a retrospective step back to explore alternative AI tools that could enhance our accuracy without undermining our existing advancements and insights garnered from transfer learning outcomes.

This journey brought us to a hybrid approach, intertwining statistical and deep learning methodologies. Due to our good transfer learning outcomes, our attention shifted towards feature extraction. Revisiting the concept outlined in the Background (Section 2), feature extraction in the realm of Convolutional Neural Networks (CNNs) for image processing fundamentally involves isolating critical characteristics or patterns from raw image data pertinent to specific tasks such as classification. This process translates image data into a representation that simplifies machine interpretation and analysis.

In this context, deep learning serves to “digest” intricate image features, employing feature extraction to inform other machine learning algorithms. These algorithms, not inherently designed to process image inputs, can thus leverage the distilled data effectively.

This goes back to another previously discussed concept: ensemble learning. Ensemble learning orchestrates multiple models, or “weak learners”, training them to converge and create a more robust “ensemble” model. This collective of “weak learners” amalgamates to formulate a “strong learner”. Here, predictions from various models are synthesized, with the ensemble model making a conclusive prediction, typically through a voting or averaging mechanism, generally elevating predictive performance relative to singular model applications.

Table 5.7: Final result of the expected performance using the ensemble algorithms compared with the predictions for the target dataset, YERAL

	DDSM_YERAL	YERAL			
	Developing Set	Test Set			
Model	Acc	Acc	Prec	Rec	F1
Alexnet Bagging	84%	70%	67%	76%	71%
VGG-16 TF Bagging	91%	82%	79%	86%	82%
Stacking VGG-16 TF	84%	78%	85%	78%	77%
Stacking Inception TF	82%	74%	78%	74%	73%
AdaBoost VGG-16 TF	83%	76%	82%	76%	75%
AdaBoost Inception TF	80%	71%	75%	71%	70%

We utilized bagging, boosting, and stacking as ensemble learning models, exemplified in the Methodology section 4. Table 5.7 showcases the results of each model. The feature extraction from the transfer learning model, specifically the VGG-16 architecture—since it delivered the best independent performance—was employed.

Evident from the outcomes in Table 5.7, the bagging model yields the highest overall accuracy, achieving 91% on the development set. Furthermore, the accuracy and F1 results from the same model hold strong at 82% each. These outcomes underscore how melding deep learning with machine and statistical learning enhanced the performance of our models for our targeted data set.

It is important to note that while still not achieving the promised 90 to 98 percent of accuracy shown in state of the art publications about benchmark dataset, we developed an alternative to solve the different constraints we found when trying to implement AI for our specific problem. With this work we expect to spark a conversation around the different uses of AI in medical imaging, how to work around difficulties that are not mentioned in the most highly scored algorithms, and the path moving forward with these technologies.

5.2 SEGMENTATION

For the segmentation process, we had the same perspective as the classification. In this case, at the time of this investigation, the segmentation process was halted because not all of the YERAL data set was annotated with the ground truth. In the meantime, we applied several other machine learning applications to the images to see which ones we could approach later on to compare to the use of the U-net.

5.2.0.1 KERNEL SOOTHING METHODS

Kernel smoothing methods are widely used in image segmentation to reduce noise and improve the distinction between different regions in an image (Hastie et al. [2009]). The concept involves using a kernel (a small matrix) to navigate the image and apply a mathematical operation at each pixel position. In the context of segmentation, kernel smoothing can help delineate clear boundaries between various segments by reducing the impact of noise and minor color variations.

For this, we use a code in Python that, as shown in Figure 5.5, starts with the original image, where each box represents the matrix or kernel of operations we will be using. The smoothing process aids in eliminating noise by averaging pixel values in a localized area defined by the kernel, thereby providing a smoother transition between pixel values across the image.

OpenCV has a module that allows the operations of erosion and dilation that essentially expands the white region (or foreground) in the binary image (typically used to accentuate features and join broken parts of objects) and then shrinks the white region in the binary image, typically used to eliminate noise, separate objects, and make them distinct.

These two operations, erosion and dilation, are often used in tandem for various

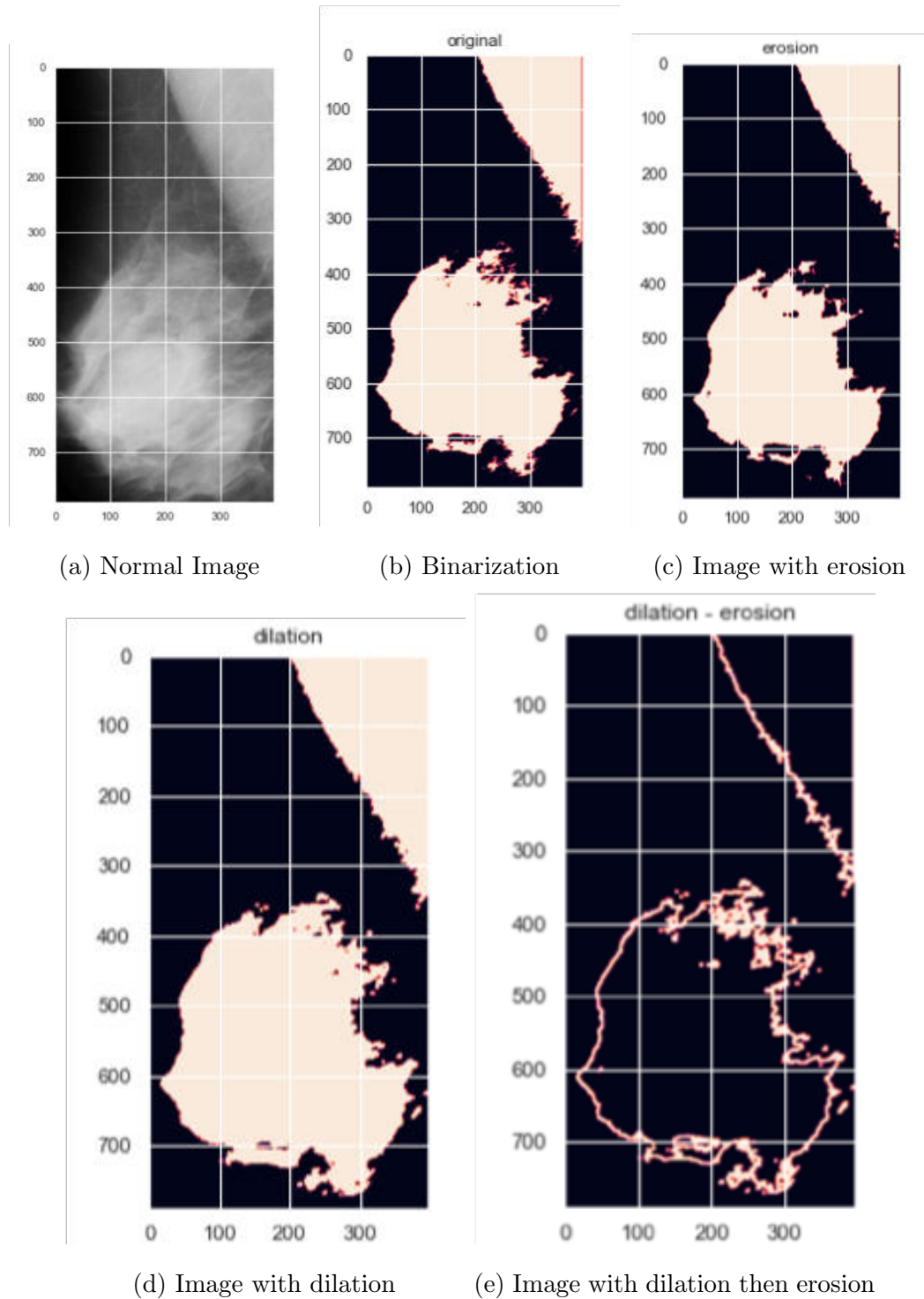


Figure 5.5: Dilation and Erosion function from the OpenCV library.

image processing tasks like getting the outline of objects, filling holes, and removing noise.

Kernel smoothing effectively minimizes localized variations and enhances global features in an image, thus improving the accuracy and robustness of subsequent segmentation.

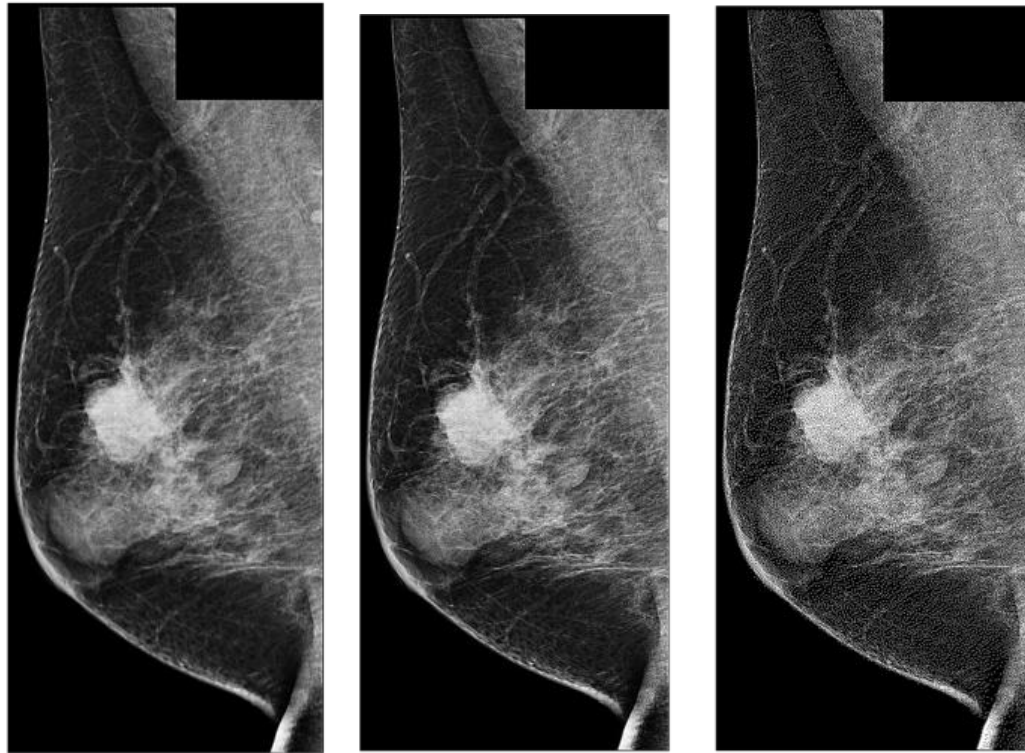
This idea was especially appealing to us since we have limited data from our target dataset. Still, after several tries, we realized that the computational load would be too much. The experiments we saw yielded very poor images as a result. We also tried to apply the best bandwidth estimator for the small image experiment we first had, but it took way too much time to compile, so we moved on to the next one.

5.2.0.2 K -NEAREST NEIGHBORS

In publications and examples, we saw when looking for information about this algorithm, we found that the best contender for image segmentation, other than simple image manipulation, is the k -Nearest Neighbors.

The k -Nearest Neighbors (k -NN) algorithm, while traditionally used for classification and regression, has a tangible potential in the domain of image segmentation as well. Image segmentation aims to partition an image into segments, where pixels in the same segment share certain visual characteristics, such as color or texture.

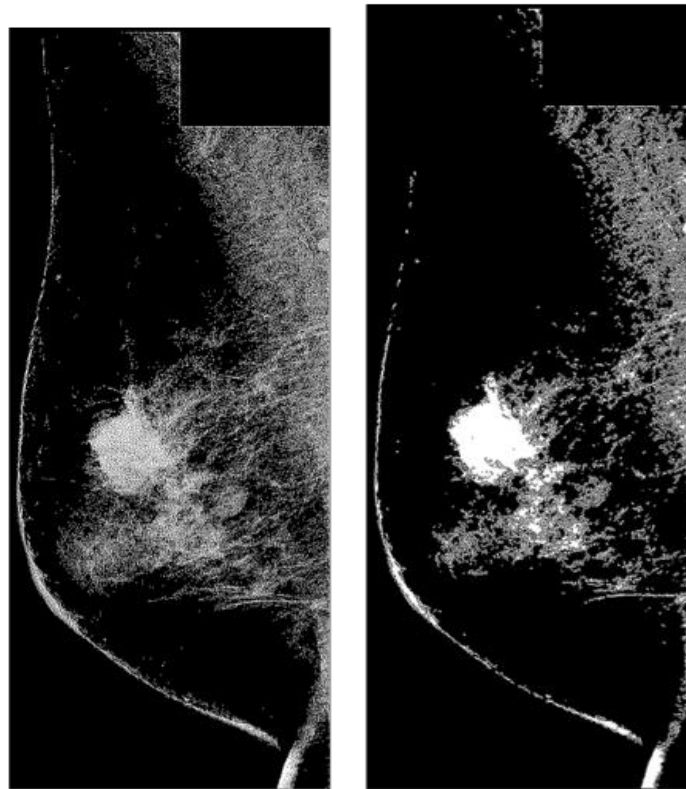
Using k -NN for image segmentation in complex datasets can introduce several challenges and may not always be the optimal choice. Images, especially high-resolution ones, inherently have high-dimensional feature spaces. The effectiveness of k -NN tends to diminish in high-dimensional spaces, so calculating distance metrics in a high-dimensional space is computationally intensive. It can be time-consuming for large or complex datasets.



(a) Detailed tool

(b) Sharpening

(c) Delete farthest pixel



(d) Binarize

(e) Join closest pixels

Figure 5.6: Manual image transformations using a function of clustering that helps us identify the major pixel clusters and eliminate what we consider noise, such as the mass of the breast.

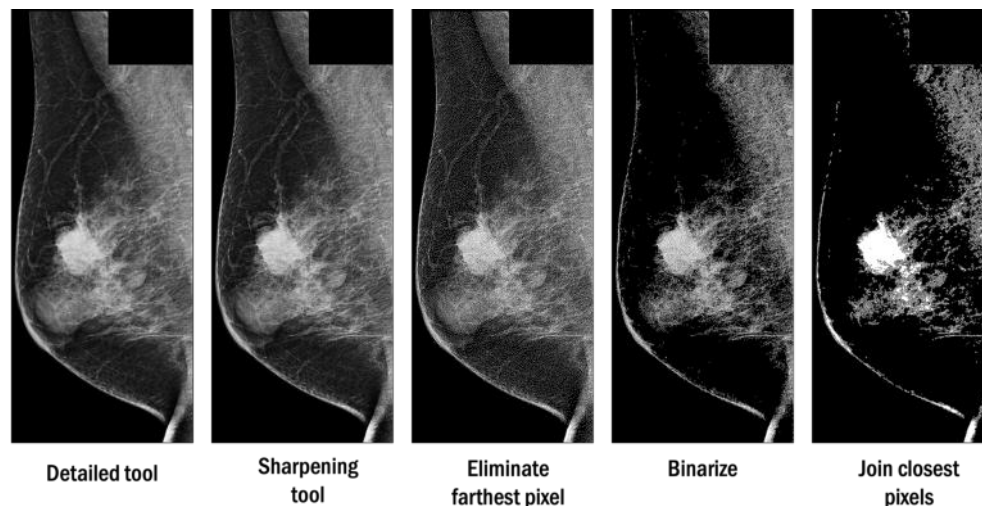


Figure 5.7: Manual image transformations using a function of clustering that helps us identify the major pixel clusters and eliminate what we consider noise, such as the mass of the breast.

Figure 5.7, we put our images through several transformations, following the behavior of clustering, where we eliminate the pixels that are farthest away from each other and cluster together the closest ones. Finally, we contour our image to find out the most prominent features to see if they match with what we know of the area of interest.

Although this process shows promise in helping clear images with too much noise due to the density of the muscle or fatty tissue of the patient, it is still a very time and resource-consuming process.

Finally, in the final iteration of k -NN algorithms, we let the threshold build until it found the suitable mask for the main anomaly on the image. The process was computationally expensive, and it took a long time to process, and in the end, we decided to stop it at 500 iterations. This yielded good results, but again, the computational load is not worth it if we had a larger dataset, since, comparing the computational load for the small amount of images we had in this experiment, the bottleneck for classification would intensify with more images.

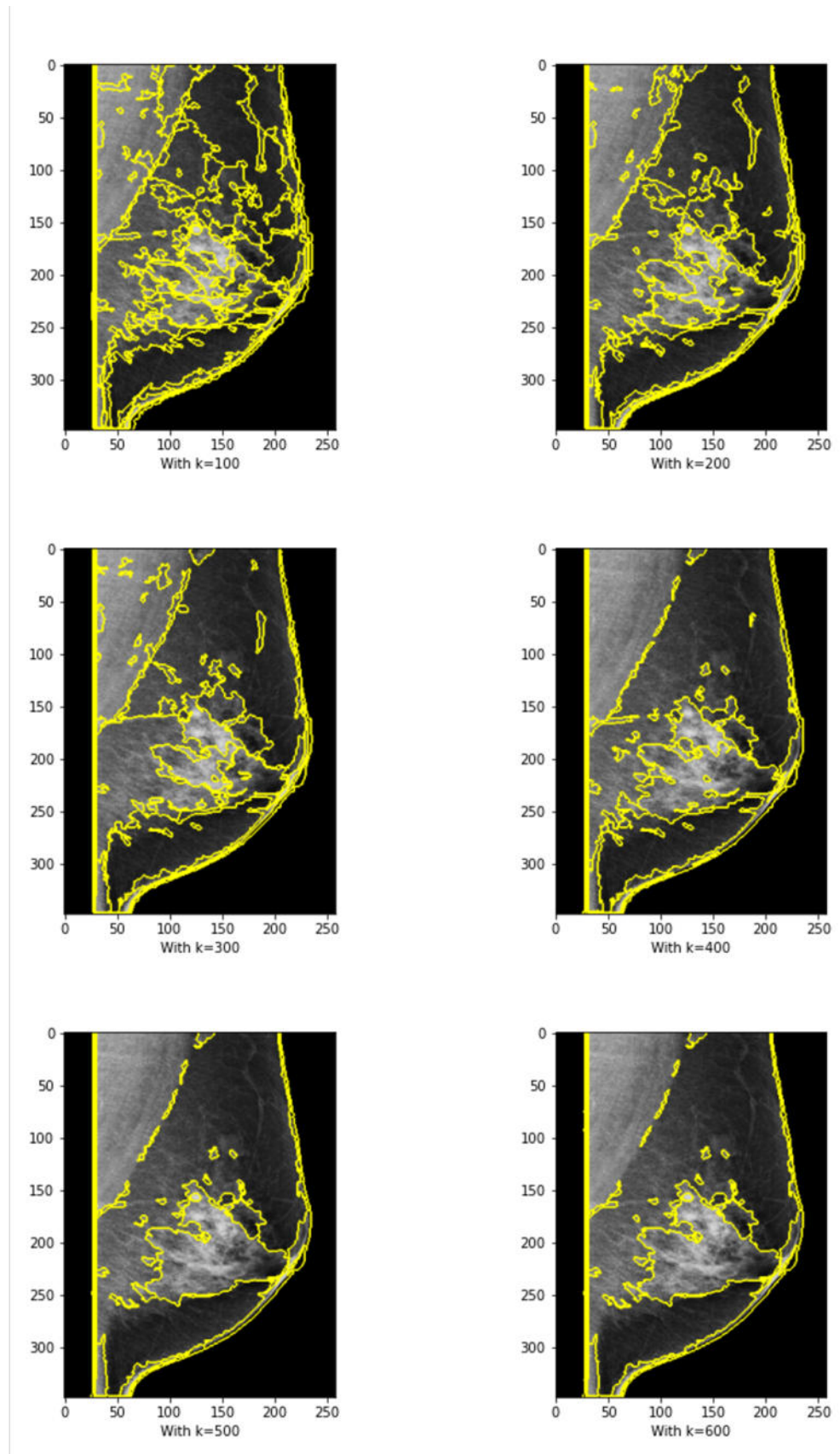


Figure 5.8: Different sizes of k for the k -Nearest Neighbor experiment.

CHAPTER 6

CONCLUSIONS

Through the intensive exploration of deep learning and its application to medical image classification, this thesis made a journey through several experimental phases to help improve methodologies suitable for breast cancer detection via mammography. With a foundational grounding in the initial model construction we developed (Berrones-Reyes [2019]), we embarked on an exploratory pathway where the relationship between statistical learning, machine learning, and deep learning was explored, illustrating the complexities and potential of artificial intelligence in enhancing each one of these fields.

The experiential knowledge gained from the initial classification model served as a base for this investigation, addressing the urgent need of a specific hospital for an economical and proficient diagnostic tool. The importance of this tool was highlighted by the conspicuous gap in specialist availability versus the number of data generated by governmental breast cancer awareness initiatives, presenting substantial waiting periods and ineffective preliminary measurements to tackle this challenge.

Our focused endeavor in refining the accuracy of the classification model was met with various challenges and learning checkpoints, particularly the realization of overfitting in our initial models and the subsequent adaptations made to mit-

igate this. Delving into more complex convolutional network architectures, our experimentation went between simplicity and complexity, constrained by computational capacity but enriched by the learning from each architectural application, from Lenet5 through to VGG-16.

In addressing the recognized limitations in external validation within medical imaging studies using deep learning, our study emphasized the importance of appropriately dividing data sets into training, development, and testing sets that were balanced correctly to avoid unnecessary bias. Notably, our use of the renowned DDSM database, despite its earlier setbacks in format, and the private YERAL data set offered a foundational base for our experimental processes.

Moreover, our experimental approach also leaned into the complexities of data balancing to mitigate bias and optimize accuracy, navigating through the intricacies of data anomalies and ensuring meticulous data validation. The subsequent creation of the DDSM YERAL and additional validation test set YERAL substantiates our commitment to ensuring a robust dataset that aligns with our research objectives.

Our work shed light on the importance of thorough pre-processing, for image denoising, and optimizing image clarity while preserving critical structural details.

The layers of challenges, adaptations, and discoveries within our experimentations highlight not only the potential within the use of artificial intelligence and all its different ramifications, but also the importance of continual exploration, adaptation, and validation in ensuring the safe and effective implementation of these technologies within medical practice.

6.1 CONTRIBUTIONS

This investigation seamlessly integrates the profundities of deep and statistical learning, establishing a robust methodology that capitalizes on the strengths of transfer

learning with the VGG-16 architecture and ensemble learning strategies within image classification contexts. The experimentation traverses various CNN architectures, offering keen insights into their dynamics and performance metrics, particularly when engaged with the challenging YERAL dataset.

We proposed in our initial hypothesis that by combining deep learning with traditional machine learning algorithm we could enhance the accuracy of previously designed classification models, keeping in mind computational constraints. In this aspect, we wanted to address the major discrepancy found in the result of using deep learning for image classification publications, and the results we where getting by using those same methods with a new data set.

Thanks to the constant feedback of our associate team on the FUCAM we where able to determine how and why these deep learning methods might not be adequate for the tasks at hand. As we have previously stated, many of the deep learning requirements and practices are not always scalable from regular computational vision problems, to medical imaging. With all of the experimentations mentioned in this work, we addressed those issues and how we went about fixing them.

In this sense, we also fulfill the main and secondary objectives we set for the completion of this work. We set out to develop a computer-aided diagnostic tool to assist medical experts on diagnosing medical images. For this, one of the outputs that we have available is the interface we developed to make the use of our trained models easy for medical experts. This also aligns to another propose we had, which was to alleviate as much of the computational load for the medical team, since we worked in our end to have good hardware to be able to experiment and choose the best artificial intelligence model for them.

With this we also made sure to ensure data integrity and optimize our combined resources for computational efficiency, which was part of some of our secondary objectives.

By using varied AI methodologies to build a medical classifier, we fulfilled

some other secondary objectives, like the integration of deep learning to reduce the complexity of our data and be able to enhance our performance by using ensemble learning techniques. Along our experimentation journey, we also made sure to follow good practices for ethical AI, such as the comparison between existing methods, exploring and mitigate the effect of data imbalance, to ensure data integrity and avoid performance bias.

Furthermore, through a detailed exploration of alternative techniques, such as k -NN for image segmentation and intricate image transformations, this work provides pivotal insights into their potential and boundaries, thereby enriching the discourse on feasible techniques in breast cancer imaging.

In the case of the segmentation process and the objectives we had planned for it, the outline of the steps was so that after the classification portion, we would perform image segmentation to locate an anomaly within the images classified as containing an anomaly.

As we faced some delays with the masking process for the target data set, the only contributions we have towards this goal were experiments based on different machine learning and statistical learning algorithms on some of the images that did have the masks. However, due to the small portion of YERAL images that had masks at the time of the experimentations, the output of this section of the methodology are solely recommended paths for the same concept we used for the binary classification; the combination between traditional machine learning and deep learning techniques to adapt to computational and data constraints.

6.2 FUTURE WORK

There are some open ended possibilities that we can continue to explore in any of the stages proposed in this work. As time passes by, many new developments have come forward to enable a more accessible use of tools for machine learning and deep

learning methods, without having the constraint of certain hardware features.

Cloud computing offers a new window of opportunity for researchers and developers of CAD systems. The only issue at the moment with this technologies remains the very restricted system still in place for the use and sharing of medical imaging among a centralized data set.

Another field that keeps improving is the research into ethical AI and the use of such tools in more critical aspects of decision making. There have been recent advances in the transparency of the use of generative AI for different fields of study to avoid data imbalances, encourages data diversity to include minorities and marginalized groups, which in turn help to avoid harmful biases.

As mentioned before, the medical field might move a little bit slower when it comes to the acceptance and use of AI tools, but as concerns about unintentional biases and misdiagnosis by computer systems are still very prevalent, it is important to continue to question and test every different scenario that can present itself so we help to avoid this barriers and encourage transparency on the use of AI tools for medical use, which in turn will avoid misconceptions.

Moving forward, the availability of labeled images for segmentation opens new avenues to merge deep and machine learning methodologies in our research, particularly leveraging the capabilities of U-Net for biomedical image segmentation. The intent is to integrate U-Net with previously utilized ensemble learning methods to refine further the segmentation and delineation of areas of interest within mammograms.

This exploration aims to enhance diagnostic capabilities by developing a hybrid model proficient in both advanced image classification and meticulous segmentation, thereby augmenting the technological toolkit in medical imaging and diagnosis.

In Section A we share the progress we made for the segmentation process with deep learning, in preparation of the availability of more revised and approved masks

for the images of our target data set. Because of time constraints, we were not able to deploy these codes, but plan to develop them in the future, and encourage readers to use them, hoping they can be helpful in any way.

6.3 RESEARCH OUTPUT

6.3.1 PUBLICATIONS

- BerronesReyes, M. C., SalazarAguilar, M. A., CastilloOlea, C. (2023). Use of Ensemble Learning to Improve Performance of Known Convolutional Neural Networks for Mammography Classification. *Applied Sciences*, 13(17), 9639.

6.3.2 INVITED TALKS AND SEMINARS

- Conversatorio de mujeres en la ciencia. Universidad Santiago de Cali (March 2023)
- Department of Physical-Mathematical Sciences. Seminar of computational science: Data Science. (March 2023)
- Department of mechanical and electrical engineering. Seminar: Challenges and areas of opportunity for deep learning applied to the medical area. (November 2022)
- CONACYT and Women in Data. Seminar: Opportunities and challenges of artificial intelligence in solving tasks for social betterment. (September 2021)
- Vision to the future of systems engineering. Round table with teachers and students to enhance the graduate program of System Engineering. (December 2021)

BIBLIOGRAPHY

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.

Oludare Isaac Abiodun, Muhammad Ubale Kiru, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Abubakar Malah Umar, Okafor Uchenwa Linus, Humaira Arshad, Abdullahi Aminu Kazaure, and Usman Gana. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846, 2019. doi:10.1109/access.2019.2945545.

Nikolas Adaloglou. Intuitive explanation of skip connections in deep learning, Mar 2020. URL <https://theaisummer.com/skip-connections/>.

Charu C. Aggarwal. *Neural networks and deep learning: A textbook. Chapter 1: Introduction*. Springer, 2019.

Emmanuel Ahishakiye, Martin Bastiaan Van Gijzen, Julius Tumwiine, Ruth Wario, and Johnes Obungoloch. A survey on deep learning in med-

- ical image reconstruction. *Intelligent Medicine*, 1(3):118–127, Sep 2021. doi:10.1016/j.imed.2021.03.003.
- Savita Ahlawat and Amit Choudhary. Hybrid CNN-SVM classifier for handwritten digit recognition. *Procedia Computer Science*, 167:2554–2560, 2020. doi:10.1016/j.procs.2020.03.309.
- Mihalj Bakator and Dragica Radosav. Deep learning and medical diagnosis: A review of literature. *Multimodal Technologies and Interaction*, 2(3):47, Aug 2018. doi:10.3390/mti2030047.
- Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. A comparative analysis of k-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning. *Decision Analytics Journal*, 3:100071, Jun 2022. doi:10.1016/j.dajour.2022.100071.
- Itzel Abundez Barrera, Eréndira Rendón Lara, Citlalih Gutiérrez Estrada, and Sergio Díaz Zagal. Diagnosis of medical images using an expert system. In *Advances in Artificial Intelligence – IBERAMIA 2010*, pages 144–152. Springer Berlin Heidelberg, 2010.
- Mayra C. Berrones-Reyes. Clasificación de mamografías mediante redes neuronales convolucionales. *Repositorio academico digital UANL*, 2019. URL <http://eprints.uanl.mx/17656/>.
- Mayra C. Berrones-Reyes, M. Angélica Salazar-Aguilar, and Cristian Castillo-Olea. Use of ensemble learning to improve performance of known convolutional neural networks for mammography classification. *Applied Sciences*, 13(17):9639, Aug 2023. ISSN 2076-3417. doi:10.3390/app13179639.
- Gaudenz Boesch. Deep residual networks (resnet, resnet50), 2023. URL <https://viso.ai/deep-learning/resnet-residual-neural-network/>.
- Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. ” O’Reilly Media, Inc.”, 2008.

- Jason Brownlee. *Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras*. Machine Learning Mastery, 2016.
- Jason Brownlee. How to choose an activation function for deep learning, Jan 2021. URL <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>.
- Isabella Castiglioni, Leonardo Rundo, Marina Codari, Giovanni Di Leo, Christian Salvatore, Matteo Interlenghi, Francesca Gallivanone, Andrea Cozzi, Natascha Claudia D'Amico, and Francesco Sardanelli. AI applications to medical images: From machine learning to deep learning. *Physica Medica*, 83:9–24, Mar 2021. doi:10.1016/j.ejmp.2021.02.006.
- Dianne Castillo. Decision trees in machine learning explained, Nov 2021. URL <https://www.seldon.io/decision-trees-in-machine-learning/>.
- Bhanu Chander, Souvik Pal, Debashis De, and Rajkumar Buyya. Artificial intelligence-based internet of things for industry 5.0. In *Internet of Things*, pages 3–45. Springer International Publishing, 2022.
- Mildred K. Cho. Rising to the challenge of bias in health care ai. *Nature Medicine*, 27(12):2079–2081, Dec 2021. ISSN 1546-170X. doi:10.1038/s41591-021-01577-2.
- François Chollet et al. Keras. <https://keras.io>, 2015.
- Francois Chollet. *Deep learning with Python*. Manning Publications Co., 2018.
- Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.
- Deepti Chopra and Roopal Khurana. *Introduction to Machine Learning with Python*. Bentham Science Publisher, Feb 2023. doi:10.2174/97898151244221230101.
- Gunjan Chugh, Shailender Kumar, and Nanhay Singh. Survey on machine learning and deep learning applications in breast cancer diagnosis. *Cognitive Computation*, 13(6):1451–1470, Jan 2021. doi:10.1007/s12559-020-09813-6.

- Alex Clark. Pillow (pil fork) documentation, 2015. URL <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>.
- Saturn Cloud. How to estimate memory usage for random forest algorithm, Sep 2023. URL <https://saturncloud.io/blog/how-to-estimate-memory-usage-for-random-forest-algorithm/>.
- B.J. Copeland. The beginning of artificial intelligence, Oct 2022. URL <https://www.britannica.com/technology/artificial-intelligence/Alan-Turing-and-the-beginning-of-AI>.
- Tim Dettmers. Which GPU for deep learning, Jan 2023. URL <https://timdettmers.com/2023/01/30/which-gpu-for-deep-learning/>.
- Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, May 2021. doi:10.1016/j.cosrev.2021.100379.
- EdX.org. What is a decision tree, 2023. URL <https://www.mastersindatascience.org/learning/machine-learning-algorithms/>.
- Guray Erus, Evangelia I. Zacharaki, and Christos Davatzikos. Individualized statistical learning from medical image databases: Application to identification of brain lesions. *Medical Image Analysis*, 18(3):542–554, Apr 2014. doi:10.1016/j.media.2014.02.003.
- Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Motaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. Deep learning-enabled medical computer vision. *npj Digital Medicine*, 4(1), Jan 2021. doi:10.1038/s41746-020-00376-2.
- Emily Foster. Choosing between a rule-based vs. machine learning system: Techtar-get, Aug 2023. URL <https://www.techtar-get.com/searchenterpriseai/feature/How-to-choose-between-a-rules-based-vs-machine-learning-system>.

- T. V. Geetha and S. Sendhilkumar. *Machine Learning*. Chapman and Hall/CRC, Mar 2023a. doi:10.1201/9781003290100.
- T. V. Geetha and S. Sendhilkumar. Performance evaluation and ensemble methods. pages 191–210. Chapman and Hall/CRC, Mar 2023b. doi:10.1201/9781003290100-8.
- Aurélien Géron. *Hands-on machine learning with scikit-learn, Keras, and tensorflow concepts, tools, and techniques to build Intelligent Systems*. OReilly, 2020.
- Tamaoghna Ghosh and Shravan Kumar Belagal Math. *Practical Mathematics for AI and Deep Learning: A Concise yet In-Depth Guide on Fundamentals of Computer Vision, NLP, Complex Deep Neural Networks and Machine Learning*. BPB publications, 2023.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- R. Harini and C. Chandrasekar. Image segmentation using nearest neighbor classifiers based on kernel formation for medical images. *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)*, 2012. doi:10.1109/icprime.2012.6208355.
- Peter Harrington. *Machine learning in action. Chapter 1: Classification*, page 18–36. Manning, 2012.
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. doi:10.1007/978-0-387-84858-7. URL <https://hastie.su.domains/Papers/ESLII.pdf>.
- Charlotte J. Haug and Jeffrey M. Drazen. Artificial intelligence and machine learning in clinical medicine, 2023. *New England Journal of Medicine*, 388(13):1201–1208, Mar 2023. doi:10.1056/nejmra2302038.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun 2016. doi:10.1109/cvpr.2016.90.
- Monica Hernandez, Ubaldo Ramon-Julvez, Elisa Vilades, Beatriz Cordon, Elvira Mayordomo, and Elena Garcia-Martin. Explainable artificial intelligence toward usable and trustworthy computer-aided diagnosis of multiple sclerosis from optical coherence tomography. *PLOS ONE*, 18(8):e0289495, Aug 2023. ISSN 1932-6203. doi:10.1371/journal.pone.0289495.
- Essam H. Houssein, Rehab E. Mohamed, and Abdelmgeid A. Ali. Machine learning techniques for biomedical natural language processing: A comprehensive review. *IEEE Access*, 9:140628–140653, 2021. doi:10.1109/access.2021.3119621.
- Robert Hu and Dino Sejdinovic. Robust deep interpretable features for binary image classification. *Proceedings of the Northern Lights Deep Learning Workshop*, 2, 2021. doi:10.7557/18.5708.
- IBM. Ibm resources - decision trees. URL <https://www.ibm.com/topics/decision-trees>.
- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer New York, 2013. doi:10.1007/978-1-4614-7138-7.
- Yulei Jiang, Robert M. Nishikawa, Robert A. Schmidt, Charles E. Metz, Maryellen L. Giger, and Kunio Doi. Improving breast cancer diagnosis with computer-aided diagnosis. *Academic Radiology*, 6(1):22–33, Jan 1999. doi:10.1016/s1076-6332(99)80058-0.
- Ida Arlene Joiner. Artificial intelligence. In *Emerging Library Technologies*, pages 1–22. Elsevier, 2018.
- C. E. Kahn. Validation, clinical trial, and evaluation of a radiology expert system.

- Methods of Information in Medicine*, 30(04):268–274, 1991. doi:10.1055/s-0038-1634850.
- Raimi Karim. Illustrated: 10 CNN architectures, Dec 2022. URL <https://towardsdatascience.com/illustrated-10-cnn-architectures>.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. doi:10.1145/3065386.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90, May 2017. doi:10.1145/3065386.
- Alberto Labrada and Buket D. Barkana. A comprehensive review of computer-aided models for breast cancer diagnosis using histopathology images. *Bioengineering*, 10(11):1289, Nov 2023. ISSN 2306-5354. doi:10.3390/bioengineering10111289.
- Philippe Lambin, Ralph T.H. Leijenaar, Timo M. Deist, Jurgen Peerlings, Evelyn E.C. de Jong, Janita van Timmeren, Sebastian Sanduleanu, Ruben T.H.M. Larue, Aniek J.G. Even, Arthur Jochems, Yvonka van Wijk, Henry Woodruff, Johan van Soest, Tim Lustberg, Erik Roelofs, Wouter van Elmpt, Andre Dekker, Felix M. Mottaghy, Joachim E. Wildberger, and Sean Walsh. Radiomics: the bridge between medical imaging and personalized medicine. *Nature Reviews Clinical Oncology*, 14(12):749–762, Oct 2017. doi:10.1038/nrclinonc.2017.141.
- Sage Lazzaro. Machine learning’s rise, applications, and challenges, Jun 2021. URL <https://venturebeat.com/ai/machine-learnings-rise-applications-and-challenges/>.
- Yann LeCun, Yoshua Bengio, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based

- learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Charitha Dissanayake Lekamlage, Fabia Afzal, Erik Westerberg, and Abbas Chaddad. Mini-DDSM: Mammography-based automatic age estimation. In *2020 3rd International Conference on Digital Medicine and Image Processing*. ACM, Nov 2020. doi:10.1145/3441369.3441370.
- Matthew J. Leming, Esther E. Bron, Rose Bruffaerts, Yangming Ou, Juan Eugenio Iglesias, Randy L. Gollub, and Hyungsoon Im. Challenges of implementing computer-aided diagnostic models for neuroimages in a clinical setting. *npj Digital Medicine*, 6(1), Jul 2023. ISSN 2398-6352. doi:10.1038/s41746-023-00868-x.
- Mingquan Lin, Tianhao Li, Yifan Yang, Gregory Holste, Ying Ding, Sarah H. Van Tassel, Kyle Kovacs, George Shih, Zhangyang Wang, Zhiyong Lu, Fei Wang, and Yifan Peng. Improving model fairness in image-based computer-aided diagnosis. *Nature Communications*, 14(1), Oct 2023. ISSN 2041-1723. doi:10.1038/s41467-023-41974-4.
- Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, Dec 2017. doi:10.1016/j.media.2017.07.005.
- Na Liu, Philip Shapira, and Xiaoxu Yue. Tracking developments in artificial intelligence research: constructing and applying a new search strategy. *Scientometrics*, 126(4):3153–3192, Feb 2021. doi:10.1007/s11192-021-03868-4.
- Xiaoxuan Liu, Livia Faes, Aditya U Kale, Siegfried K Wagner, Dun Jack Fu, Alice Bruynseels, Thushika Mahendiran, Gabriella Moraes, Mohith Shamdas, Christoph

- Kern, Joseph R Ledsam, Martin K Schmid, Konstantinos Balaskas, Eric J Topol, Lucas M Bachmann, Pearse A Keane, and Alastair K Denniston. A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. *The Lancet Digital Health*, 1(6):e271–e297, Oct 2019. doi:10.1016/s2589-7500(19)30123-2.
- Inna Logunova. Support vector machine algorithm, Oct 2022. URL <https://serokell.io/blog/support-vector-machine-algorithm>.
- Baolong Lv, Feng Liu, Yulin Li, Jianhua Nie, Fangfang Gou, and Jia Wu. Artificial intelligence-aided diagnosis solution by enhancing the edge features of medical images. *Diagnostics*, 13(6):1063, Mar 2023. ISSN 2075-4418. doi:10.3390/diagnostics13061063.
- Dr. Yogesh Kumar Sharma M. Swapna and Dr. BMG Prasad. CNN architectures: Alex net, le net, VGG, google net, res net. *International Journal of Recent Technology and Engineering (IJRTE)*, 8(6):953–959, Mar 2020. doi:10.35940/ijrte.f9532.038620.
- Alessia Mammone, Marco Turchi, and Nello Cristianini. Support vector machines. *WIREs Computational Statistics*, 1(3):283–289, Nov 2009. doi:10.1002/wics.49.
- M. Matesin, S. Loncaric, and D. Petrvacic. A rule-based approach to stroke lesion analysis from CT brain images. In *ISPA 2001. Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis. In conjunction with 23rd International Conference on Information Technology Interfaces (IEEE Cat. No.01EX480)*. Univ. Zagreb, 2001. doi:10.1109/ispa.2001.938631.
- Navneet Melarkode, Kathiravan Srinivasan, Saeed Mian Qaisar, and Pawel Plawiak. Ai-powered diagnosis of skin cancer: A contemporary review, open challenges and future research directions. *Cancers*, 15(4):1183, Feb 2023. ISSN 2072-6694. doi:10.3390/cancers15041183. URL <http://dx.doi.org/10.3390/cancers15041183>.

- Scott Menard. *Introduction: Linear Regression and Logistic Regression*. SAGE Publications, Inc., 2010. doi:10.4135/9781483348964.n1.
- Rahul Reddy Nadikattu. The emerging role of artificial intelligence in modern society. *International Journal of Creative Research Thoughts*, 2016.
- Neurohive. Resnet (34, 50, 101): Residual cnns for image classification tasks, Jan 2019. URL <https://neurohive.io/en/popular-networks/resnet/>.
- Andrew Ng. Machine learning yearning: Technical strategy for ai engineers in the era of deep learning. *Machine Learning Yearning*, 139, 2019. URL <https://info.deeplearning.ai/machine-learning-yearning-book>.
- Natalia Norori, Qiyang Hu, Florence Marcelle Aellen, Francesca Dalia Faraci, and Athina Tzovara. Addressing bias in big data and ai for health care: A call for open science. *Patterns*, 2(10):100347, Oct 2021. ISSN 2666-3899. doi:10.1016/j.patter.2021.100347. URL <http://dx.doi.org/10.1016/j.patter.2021.100347>.
- Nvidia. Deep learning, 2023. URL <https://developer.nvidia.com/deep-learning>.
- Andreas S. Panayides, Amir Amini, Nenad D. Filipovic, Ashish Sharma, Sotirios A. Tsaftaris, Alistair Young, David Foran, Nhan Do, Spyretta Golemati, Tahsin Kurc, Kun Huang, Konstantina S. Nikita, Ben P. Veasey, Michalis Zervakis, Joel H. Saltz, and Constantinos S. Pattichis. AI in medical imaging informatics: Current challenges and future directions. *IEEE Journal of Biomedical and Health Informatics*, 24(7):1837–1857, Jul 2020. doi:10.1109/jbhi.2020.2991043.
- Jae-beom Park, Han-sung Lee, and Hyun-chong Cho. Investigating effective data augmentation techniques for accurate gastric classification in the development of a deep learning-based computer-aided diagnosis system. *Applied Sciences*, 13(22):12325, Nov 2023. ISSN 2076-3417. doi:10.3390/app132212325.

- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011a.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011b.
- Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.
- Marcela X. Ribeiro, Agma J. M. Traina, Caetano Traina, and Paulo M. Azevedo-Marques. An association rule-based method to support medical image diagnosis with efficiency. 10(2):277–285, Feb 2008. doi:10.1109/tmm.2007.911837.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing, 2015.
- Stuart Russell, Daniel Dewey, and Max Tegmark. Research priorities for robust and beneficial artificial intelligence. *AI Magazine*, 36(4):105–114, Dec 2015. doi:10.1609/aimag.v36i4.2577.
- Sumaiya Z. Sande, Loraine Seng, Jialiang Li, and Ralph D’Agostino. Statistical learning in medical research with decision threshold and accuracy evaluation. *Journal of Data Science*, pages 634–657, 2021. doi:10.6339/21-jds1022.
- Iqbal H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6), Aug 2021. doi:10.1007/s42979-021-00815-1.

- Heather Schmidt. Technology and the future of healthcare, Aug 2023. URL <https://www.innerbody.com/technology-and-the-future-of-healthcare>.
- Anjaneyulu Babu Shaik and Sujatha Srinivasan. A brief survey on random forest ensembles in classification model. In *International Conference on Innovative Computing and Communications*, pages 253–260. Springer Singapore, Nov 2018.
- Pramila P. Shinde and Seema Shah. A review of machine learning and deep learning applications. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, Aug 2018. doi:10.1109/iccubea.2018.8697857.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Himanshi Singh. Gini impurity: Splitting decision trees with gini impurity, Aug 2023. URL <https://www.analyticsvidhya.com/blog/2021/03/how-to-select-best-split-in-decision-trees-gini-impurity/>.
- Naman Singh. Advantages and disadvantages of linear regression, Jul 2020. URL <https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/>.
- Camélia Slimani, Chun-Feng Wu, Stéphane Rubini, Yuan-Hao Chang, and Jalil Boukhobza. Accelerating random forest on memory-constrained devices through data storage optimization. *IEEE Transactions on Computers*, 72(6):1595–1609, Jun 2023. doi:10.1109/tc.2022.3215898.
- Martina Sollini, Margarita Kirienko, Noemi Gozzi, Alessandro Bruno, Chiara Torrisi, Luca Balzarini, Emanuele Voulaz, Marco Alloisio, and Arturo Chiti. The development of an intelligent agent to detect and non-invasively characterize lung lesions on ct scans: Ready for the “real world”? *Cancers*, 15(2):357, Jan 2023. ISSN 2072-6694. doi:10.3390/cancers15020357.

- Ed Sperling. The rise of parallelism, Apr 2017. URL <https://semiengineering.com/the-rise-of-parallelism/>.
- Kenji Suzuki. A review of computer-aided diagnosis in thoracic and colonic imaging. *Quantitative imaging in medicine and surgery*, 2(3):163, 2012. doi:10.3978/j.issn.2223-4292.2012.09.02.
- Saket Thavanani. Comparative performance of deep learning optimization algorithms using numpy, May 2020. URL <https://towardsdatascience.com/comparative-performance-of-deep-learning-optimization-algorithms-using-numpy-24ce25c2f5e2>.
- Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- Gaël Varoquaux and Veronika Cheplygina. How i failed machine learning in medical imaging—shortcomings and recommendations. *arXiv preprint arXiv:2103.10292*, 2021.
- Ricardo Vinuesa, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, and Francesco Fuso Nerini. The role of artificial intelligence in achieving the sustainable development goals. *Nature Communications*, 11(1), Jan 2020. doi:10.1038/s41467-019-14108-y.
- Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algo-

- rithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.
- Haiying Wang and Huiru Zheng. Model validation, machine learning. In *Encyclopedia of Systems Biology*, pages 1406–1407. Springer New York, 2013. doi:10.1007/978-1-4419-9863-7233.
- Liwei Wang, Sunyang Fu, Andrew Wen, Xiaoyang Ruan, Huan He, Sijia Liu, Sungrim Moon, Michelle Mai, Irbaz B. Riaz, Nan Wang, Ping Yang, Hua Xu, Jeremy L. Warner, and Hongfang Liu. Assessment of electronic health record for cancer research and patient care through a scoping review of cancer natural language processing. *JCO Clinical Cancer Informatics*, (6), Jul 2022. ISSN 2473-4276. doi:10.1200/cci.22.00006.
- Weka. Why GPUs for machine learning? a complete explanation, Sep 2021. URL <https://www.weka.io/learn/ai-ml/gpus-for-machine-learning/>.
- Tan Yigitcanlar, Kevin Desouza, Luke Butler, and Farnoosh Roozkhosh. Contributions and risks of artificial intelligence (AI) in building smarter cities: Insights from a systematic review of the literature. *Energies*, 13(6):1473, Mar 2020. doi:10.3390/en13061473.
- Zhongzhi Yu and Yemin Shi. Centralized space learning for open-set computer-aided diagnosis. *Scientific Reports*, 13(1), Jan 2023. ISSN 2045-2322. doi:10.1038/s41598-023-28589-x.
- Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

APPENDIX A

DATASETS AND CODES

As mentioned in the materials sections, there are several datasets available to perform mammography classification and segmentation, such as DDSM and Minimias, which can be found in the following repositories:

- **Mini MIAS dataset** <http://peipa.essex.ac.uk/info/mias.html>
- **CBIS DDSM dataset** <https://wiki.cancerimagingarchive.net/pages/viewpage.action?pageId=22516629>

For the Mini MIAS dataset, the download format of the data is PGM. There are ways to use the images in this format, however, all images have to be normalized in order for them to be used on a single CNN. The Pillow library in Python can help in this conversion, however, a clean version on png can be available by contacting the author of this work.

The YERAL dataset is a private dataset provided by the FUCAM Institute. All the images have been expertly annotated and labeled as normal or with anomalies. The process of segmenting the images and annotate them by hand by experts it is still an ongoing effort. The availability of the images for further experimentation can be discussed with the author.

All of the codes available below can be used and modified freely. If more

information is needed from any of the codes, please contact the author.

Code Information:

For the first part of our experimentation, we normalized our images. For this, we used the codes in the Tf_FE folder at https://github.com/mayraberrones94/Tesis_codigos/tree/main/Tf_FE:

- `black_box.py` This code draws a small black box on top of the data of the patients for the YERAL data set.
- `crop2.py` Is the code to crop the black mask of the mammography. This was only used for the YERAL data set, and it was later replaced with the binarization process for the segmentation stage.

For the experimentations with the Alexnet architecture, they can be found at: https://github.com/mayraberrones94/Tesis_codigos/tree/main/Code_Alexnet. This folder has all the iterations of the different methodologies we tried for our process. The file used for the final results is `redv4_alex.py`.

- `red_Alexnet_01.py` This code has the same features of the Alexnet architecture directed on their original paper. We only changed the flattening layer with the activation function. Different from the article, we only have two classes to classify, so we changed softmax for sigmoid in the activation function, and the last flattening layer, the number of neurons is 2. This is the same in all of the other iterations of Alexnet architecture.
- `red_Alexnet_02.py` In this code, we changed the loss function to the hinge loss function. This function is explained in more detail in the experimental repository mentioned below.
- `red_Alexnet_03.py` This is the code where we attempted our first ensemble algorithm of bootstrapping with Alexnet and the original features described

in code `red_Alexnet_01.py`. As mentioned before, Alexnet is a simple enough architecture that feature extraction was not needed. However, since bootstrapping is basically training models several times, it should consider the RAM capabilities of the computer when choosing the number of batches. In this code we have five.

- `red_Alexnet_04.py` This code is the same bootstrap experiment from code `red_Alexnet_03.py`, but this time with the features we changed for code `red_Alexnet_02.py`.
- `red_Alexnet_05.py` For this experimentation, we used the `AdaBoostClassifier` from the Keras library as a last artificial neural network estimator. The features for this architecture are the same as code `red_Alexnet_01.py`.

For the experimentation of the Vgg architecture, we have similar experiments with the Alexnet. The codes can be found at: https://github.com/mayraberrones94/Tesis_codigos/tree/main/Code_Vgg

- `red_vgg_01.py`: In this case, we had to downsize the number of neurons for the last layer, as well as make the changes to the flattening layer and last activation function.
- `red_vgg_02.py`: Same features as the last version, and this time we changed the loss function from binary to hinge.
- `red_vgg_03.py`: We added the bootstrap method with the same features as the first experiment.
- `redv4_vgg.py`: This is the version that we added for the final results, where we keep the features as they are, and use the adam optimizer at the end. We change it to binary crossentropy, but leave the softmax activation function at the end of the architecture.

- `redvgg_tf_bootstrap.py`: In this code, instead of using bootstrap with the features of the second version, we use transfer learning for each batch, which reduced significantly the training time.

For the Inception architecture, there were only two experimentations. The `red_inception_v1.py` has the same modules as the original architecture, but in all of the cases the number of neurons for each layer were downsized. For the `red_inception_tf.py` we used the library of keras with the same features described. Both these codes can be found in the `Code_V3` folder.

For the feature extraction, we experimented with both Vgg and Inception, but Inception did not have great results, so we only continued with the Vgg. The code `extract_features_02.py` is for the Vgg architecture and the `extract_features.py` is for the Inception architecture.

For the ensembles of stacking and boosting, we have the code of `red_stacked1.py` and `adaboost_class.py` respectively. Both of these codes do not take as input the images, but the features resulting from the feature extraction of the Vgg model. These codes can be found in the `Tf_FE` folder.

Lastly, we added a folder into our GitHub with the interface to load the trained models and make predictions of new images. It can be found in the `CLASIFICADOR` folder.

A.1 EXPERIMENTAL RESULTS

For the experimental results that we mention in the Results (Section 5) we have a whole description inside the GitHub page, and it can be found at: <https://github.com/mayraberrones94/Aprendizaje>

Finally, we added the several experimental code we tried for the benchmark

datasets for the segmentation process, in hope that can be useful later on. There are at: https://github.com/mayraberrones94/Tesis_codigos/tree/main/Segmentation

- `Image_enhance.py`: This code has the binarization and normalization process described in the Methodology Section 4. This code needs to be used on the mammograms only, not the masks.
- `Image_masks.py`: This code, similar to the last one is to normalize images, but in this case is only the masks. Since the mask do not need to remove noise, they just need to be the same size (and form) of the other images.
- `Segment_VggUnet.py`: This code is only to organize the data in a way that the Unet can read it. It organizes and names the folders. Please be mindful that this alters the original data set. If you want it instead to only copy and paste images in another folder, you need to change the functions to reflect that.
- `Unet_v1.py`: This, as well as versions 2 and 3 are the several changes we came up with for the Unet architecture to make it more manageable. All of them use the Vgg architecture to use the pre trained weights.
- `merging_masks.py`: This code was made to merge different masks of the same image into a single mask. This was because some of the images from the data set separated them, and the original code did not accept an image with more than one separate mask.

Contact information:

- **Author:** Mayra Cristina Berrones Reyes
- **Email:** mayra.berronesrys@uanl.edu.mx

RESUMEN AUTOBIOGRÁFICO

M.Sc. Mayra Cristina Berrones Reyes

Candidato para obtener el grado de
Doctor en Ingeniería en Sistemas
Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

USE OF ARTIFICIAL INTELLIGENCE TOOLS FOR
COMPUTATIONAL-AIDED DIAGNOSTICS

Nombre: Mayra Cristina Berrones Reyes.

Fecha de Nacimiento: 21 de Octubre de 1994.

Lugar de nacimiento: Cd. Victoria, Tamaulipas, México.

Padres: Miguel Angel Berrones Jasso y Martha Cecilia Reyes Ornelas.

Educación superior:

- Universidad Autónoma de Nuevo León. Facultad de Ingeniería Mecánica y Eléctrica. Ingeniero en Tecnología de Software (2012 a 2017)
- Universidad Autónoma de Nuevo León. Facultad de Ingeniería Mecánica y Eléctrica. Master en Ciencias en Ingeniería de Sistemas. (2017 a 2019)