UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Biológicas e Ingeniería

Design and Construct a Robotic Arm for Sign Language Interpretation with a Neural Network

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero/a Biomédico/a.

Autores:
Jordan Herrera Dioselyn Anyeline
Ubilluz Ortega Christian Nathanaél

Tutor:
Mgs. Cruz Varela Jonathan David

Urcuquí, Noviembre 2024

Autoría

Yo, ***Christian Nathanaél Ubilluz Ortega***, con cédula de identidad ***0604701276***, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad del autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Noviembre 2024.

Firmado electrónicamente por:
CHRISTIAN
NATHANAEL
UBILLUZ ORTEGA

Christian Nathanaél Ubilluz Ortega

C.I: 0604701276

Autoría

Yo, ***Dioselyn Anyeline Jordan Herrera***, con cédula de identidad ***0959378589***, declaró que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad del autor del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Noviembre 2024.

Firmado electrónicamente por:
DIOSELYN
ANYELINE JORDAN
HERRERA

Dioselyn Anyeline Jordan Herrera

C.I: 0959378589

Autorización de publicación

Yo, ***Christian Nathanaél Ubilluz Ortega***, con cédula de identidad ***0604701276***, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Noviembre 2024

Firmado electrónicamente por:
CHRISTIAN
NATHANAEL
UBILLUZ ORTEGA

Christian Nathanaél Ubilluz Ortega

C.I: 0604701276

Autorización de publicación

Yo, ***Dioselyn Anyeline Jordan Herrera***, con cédula de identidad ***0959378589***, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Noviembre 2024

Firmado electrónicamente por:
DIOSELYN
ANYELINE JORDAN
HERRERA

Dioselyn Anyeline Jordan Herrera

C.I: 0959378589

Dedicatoria

A mis padres, Diego Fernando Chávez y María Patricia Ubilluz.

A mis abuelitos, Cristóbal Vicente Chávez, María Angelica Chávez, Martha Cecilia Chavéz
Luis Flavio Gonzalo Ubilluz y Ana María Ortega.

A mis hermanos, Diego Alejandro y Fernando Sebastián.

A mis tíos, Liliana Chávez, Edwin Chávez, Xavier Chávez, Gonzalo Ubilluz y Oscar Ubilluz.

A mis primos, Esteban Chávez, Pablo Ubilluz y Ana Ubilluz.

Christian Nathanaél Ubilluz Ortega

Dedicatoria

Dedico este trabajo a mi familia, quienes han sido mi pilar en todo momento. Su apoyo incondicional, su amor y su comprensión me han dado la motivación necesaria para continuar en los momentos difíciles. A cada uno de ustedes, que ha creído en mí y me ha acompañado en este proceso, les agradezco profundamente por ser mi fuerza y mi inspiración diaria.

Dioselyn Anyeline Jordan Herrera

Agradecimientos

Agradezco a mi familia, por ser el impulso a superarme cada día, por ser mi soporte y siempre apoyarme en mis decisiones. Gracias en especial a mis padres que me han imbuido con sus valores y me han guiado por este sendero llamado vida, enseñándome que los sueños se pueden alcanzar dedicándole el suficiente tiempo y sacrificio. A mis hermanos por ser su modelo a seguir y tratando de ser lo mejor para ellos. A mis cuatro abuelitos, por ser mis segundos padres que me han instruido en cada una de mis acciones, enseñándome que no importa cuantas veces nos caigamos siempre tendremos que levantarnos con la frente en alto. Agradezco profundamente a la vida, que me dio la oportunidad de estudiar esta carrera en esta Universidad, y a todas las personas maravillosas que formaron parte de esta gran experiencia de crecimiento personal e intelectual. A Daniel y Héctor por ser mis compañeros de camino en esta aventura que empezamos juntos llamada Universidad. Agradezco en especial a Dioselyn que ha sido mi compañera de aventuras y mi confidente a través de estos años, por creer en mí, en mis ideas locas y sobre todo por nunca perder la fé en mis capacidades. También agradezco a mi gatito Blacky, por desvelarse conmigo en esas épocas de estudio y ayudarnos en la realización de la tesis. Agradezco a mi tutor por compartir sus conocimientos y siempre impulsarme a salir adelante y plantar la semilla de nuevos proyectos. A todos mis maestros que me han inspirado en el camino de la ciencia e investigación. Y en general a todos mis amigos y compañeros de carrera por ser parte de estos 5 años de experiencias inolvidables. Agradezco a todas las personas que fueron parte de toda mi vida universitaria que compartieron conmigo mis logros. También agradezco a Dios por cada cosa buena y mala en esta etapa. Cada uno de ellos es una pieza fundamental de la persona que soy ahora, todo esto se lo debo a ustedes.

Christian Nathanaél Ubilluz Ortega

Agradecimientos

A mi familia, quiero expresar mi más profundo agradecimiento. Gracias por ser mi pilar en todo momento, por su apoyo incondicional y por brindarme el ánimo necesario en cada paso de este proceso. Su amor, comprensión y paciencia han sido mi mayor motivación, especialmente en los momentos de dificultad. A ustedes, que siempre han creído en mí y me han dado la confianza para seguir adelante, les debo mucho más de lo que puedo expresar con palabras. Este logro también es suyo, porque sin su respaldo nada de esto habría sido posible.

A mi tutor, le agradezco su paciencia y dedicación. Su guía constante y sus enseñanzas me ayudaron a enfrentar cada reto de este trabajo.

Dioselyn Anyeline Jordan Herrera

I. Resumen

La discapacidad auditiva es un problema que afecta a las personas en diferentes etapas de la vida. Solo en Ecuador, esta cifra se extiende al 14,12% de la población total siendo aproximadamente 62 mil personas que presentan problemas de audición. El uso del lenguaje de señas se presenta como una alternativa para garantizar una buena comunicación entre personas con discapacidad auditiva y personas sin esta afección. Sin embargo, pocas personas conocen sobre este lenguaje. Para ello, desarrollamos un brazo robótico con características humanas para que trabaje en conjunto con redes neuronales convolucionales para la interpretación de gestos mediante visión artificial, promoviendo el aprendizaje del lenguaje de señas de manera didáctica.

Los resultados obtenidos demuestran la capacidad de las redes neuronales para detectar patrones y distinguir los diferentes tipos de señas que posee este lenguaje. Además, el uso de diversas redes neuronales permite una comparación sobre las limitaciones que poseen los diferentes modelos. La eficiencia de los modelos, en términos de porcentaje de precisión, fue del 99,27% y 98,2% para los modelos AlexNet y GoogleNet, respectivamente. En conclusión, se demuestra la importancia de implementar técnicas de aprendizaje automático para mejorar la comunicación y generar un impacto en el entorno de personas que desconocen esta problemática. Se espera que en un futuro esta tecnología pueda ser implementada en el sistema educativo para enseñar este lenguaje a niños y jóvenes, reduciendo así la brecha de comunicación desde una etapa temprana.

Palabras clave: Brazo robótico, Visión artificial, Redes neuronales convolucionales (CNN), MATLAB, Aprendizaje Automático.

II. Abstract

Hearing impairment is a problem that affects people at different stages of life. In Ecuador, this affection extends to 14.12% of the total population being approximately 62 thousand people with hearing problems. The use of sign language is presented as an alternative to ensure good communication between people with hearing impairment and people without this condition. However, few people know about this language. For this, we developed a robotic arm with human characteristics to work together with convolutional neural networks for the interpretation of gestures through artificial vision, promoting the learning of sign language in a didactic way.

The results obtained demonstrate the ability of neural networks to detect patterns and distinguish the different types of signs in this language. In addition, the use of different neural networks allows a comparison of the limitations of the different models. The efficiency of the models, in terms of percentage accuracy, was 99.27% and 98.2% for the AlexNet and GoogleNet models, respectively. In conclusion, the importance of implementing machine learning techniques to improve communication and generate an impact on the environment of people who are unaware of this problem is demonstrated. It is hoped that in the future this technology can be implemented in the educational system to teach this language to children and young people, thus reducing the communication gap from an early stage.

Keywords: Robotic arm, Artificial vision, Convolutional Neural Network (CNN), MATLAB, Machine Learning.

Contents

Thesis Overview..1

Chapter I..3

1. Introduction..3

  1.1. Justification..4

  1.2. Problem Statement..4

  1.3. Objectives and work methodology..5

    1.3.1. General Objective..5

    1.3.2. Specific Objectives..5

Chapter II..7

2. Theoretical Framework..7

  2.1. Robotics arm..7

    2.1.1. Types..7

    2.1.2. Materials..8

  2.2. Control System..9

  2.3. Actuators..10

    2.3.1. Kinds of Actuators..10

    2.3.2. Stepper Motors..12

    2.3.3. Servomotors..13

  2.4. Hearing Impairment..15

    2.4.1. American sign language..16

  2.5. Image Processing..17

  2.6. Artificial Intelligence..18

    2.6.1. Robotics and planning..19

    2.6.2. Natural Language Processing..20

    2.6.3. Expert systems..21

    2.6.4. Speech and Vision..22

    2.6.5. Machine learning..22

    2.6.6. Google-Net..27

    2.6.7. Alex-Net..27

  2.7. Optimizers and techniques to avoid overfitting..28

    2.7.1. Types of optimizers..28

    2.7.2. Techniques to avoid overfitting..29

  2.8. Software..30

    2.8.1. Python..30

    2.8.2. Arduino IDE..32

    2.8.3. Lab View..33

  2.8.4. Matlab..34

    2.8.5. Microcontrollers..38

2.9. Related Works for “Design and construct a robotic arm for sign language interpretation with neural network” 40
2.10. Main Challenges 49
2.10.1. Hardware Limitations 49
Chapter III 50
3. Methodology 50
3.1. Methodology of work 50
3.2. Fundamental Workflows 50
3.2.1. Requirements 50
3.2.2. Analysis 51
3.2.3. Design 51
3.2.4. Testing 51
3.3. Life cycle phases 51
3.3.1. Start 51
3.3.2. Elaboration 52
3.3.3. Construction 52
3.3.4. Transition 52
3.4. Software Development 52
3.4.1. Database 52
3.4.2. Convolutional Neural Network 54
3.4.3. Arm Motion 56
3.4.4. Artificial Vision 58
3.5. Practical development – prototype 61
3.6. System Integration 66
3.7. Experiment Phase 69
3.7.1. Experiment 1: Database 69
3.7.2. Experiment 2 : Convolutional Neural Network 72
3.7.3. Experiment 3: Artificial Vision 74
3.7.4. Experiment 4: Prediction Applications 75
Chapter IV 77
4. Results and Discussion 77
4.1. Experiment 1: DataBase 77
4.1.1. Human hand 77
4.1.2. Robotic Hand 78
4.2. Experiment 2: Convolutional Neural Network 78
4.2.1. Human 78
4.2.3. Grad-CAM 90
4.3. Experiment 3: Artificial Vision 94
4.3.1. Human 94
4.3.2. Robotic Hand 95
4.4. Experiment 4: Prediction Applications 96

4.4.1. Real-Time....96
4.4.2. Single Image.... 97
4.5. Discussion....97

Chapter V....103

5. Conclusions and future work....103
5.1. Conclusions....103
5.2. Future works.... 104
5.3. Limitations.... 104
6. Reference....106
7. Attachments.... 123

List of Figures

Figure 1. Cartesian robots, articulated robot, polar or spheric robot, Scara robot and angular or anthropomorphic robots [17 - 18]....8
Figure 2. Robot arm [26]....10
Figure 3. Pneumatic actuators [28]....10
Figure 4. Hydraulic actuators [30]....11
Figure 5. Mechanical actuators [31]....11
Figure 6. Electric Actuators [33]....11
Figure 7. Hydraulic actuators [33]....12
Figure 8. Parts of servomotors [41]....14
Figure 9. Percentage of persons with hearing disabilities registered in Ecuador in 2022 [53]....16
Figure 10. ASL Alphabet [59]....17
Figure 11. Artificial Intelligence applications [66]....19
Figure 12. Machine learning [66]....22
Figure 13. DeepLearning [66]....24
Figure 14. The basic architecture of CNN based on [103]....26
Figure 15. GoogleNet scheme based on [122]....27
Figure 16. Structure of Alex Net [125]....28
Figure 17. Python ecosystem [136]....30
Figure 18. Arduino IDE software....33
Figure 19. Deep Network Designer Start Page....35
Figure 20. Deep Network Designer Interface....36
Figure 21. Image Region Analyzer interface....38
Figure 22. Mini Maestro 18 channel Pololu USB servo controller [165]....38
Figure 23. Part of Raspberry Pi [168]....39
Figure 24. Arduino Mega 2560 connection diagram [171]....39
Figure 25. Arduino Sensor Shield v5.0 diagram [173]....40
Figure 26. Main phases and workflows of the Unified process [191]....50
Figure 27. Flowchart to make our database....54
Figure 28. A flowchart to use in transfer learning on Pre Trained Neural Networks....56
Figure 29. Flowchart for arm movement....58
Figure 30. Flowchart of Artificial vision....60
Figure 31. Scheme of Artificial vision....61
Figure 32. Robotic arm exploded view using Fusion 360....64
Figure 33. Scheme of prototype creation....66
Figure 34. Global Scheme....67
Figure 35. Robotic hand prototype....68
Figure 36. Different filters to make our DataBase....70
Figure 37. ASL is made by the Human hand....77

Figure 38. ASL is made by robotic hand.. 78
Figure 39. Training progress for Alexnet with Human DataBase..79
Figure 40. Confusion Matrix for Alexnet with Human DataBase.. 80
Figure 41. Important Metrics for AlexNet with Human DataBase...82
Figure 42. Training progress for GoogleNet with Human DataBase......................................82
Figure 43. Confusion Matrix for GoogleNet with Human DataBase.................................... 83
Figure 44. Important Metrics for GoogleNet with Human DataBase.....................................84
Figure 45. Training progress for Alexnet with Robotic DataBase..85
Figure 46. Confusion Matrix for Alexnet with Robotic DataBase... 86
Figure 47. Important Metrics for AlexNet with Robotic DataBase..87
Figure 48. Training progress for GoogleNet with Robotic DataBase.....................................87
Figure 49. Confusion Matrix for GoogleNet with Robotic DataBase.....................................88
Figure 50. Important Metrics for GoogleNetNet with Robotic DataBase.............................. 89
Figure 51. Grad-CAM for AlexNet model in Human DataBase...91
Figure 52. Grad-CAM for GoogleNet model in Human DataBase...91
Figure 53. Grad-CAM for AlexNet model in Robotic Hand DataBase..................................92
Figure 54. Grad-CAM for GoogleNet model in Robotic DataBase.......................................93
Figure 55. Multiple steps to apply Artificial Vision for Background elimination.................. 94
Figure 56. Multiple steps to apply Artificial Vision for Background elimination on Robotic Hand...95
Figure 57. Comparison of prediction in Real Time between AlexNet and GoogleNet models... 96
Figure 58. Comparison of prediction on Single Image between AlexNet and GoogleNet models..97
Figure 59. 3D printing of the robotic arm thumb..124
Figure 60. 3D printing of a part of the robotic hand.. 124
Figure 61. 3D printing of a part of the robotic hand.. 125
Figure 62. 3D printing of a part of a robotic hand... 125
Figure 63. Power source 10A/5V... 136

List of Tables

Table 1. Characteristics of each class of Database.................. 8
Table 2. Kinds of Actuators.................. 10
Table 3. Types of stepper motor.................. 12
Table 4. Type of servo motors.................. 14
Table 5. Comparison between several models of actuators.................. 14
Table 6. Types of NLP.................. 20
Table 7. Types of Optimizers.................. 28
Table 8. Methods to avoid overfitting.................. 29
Table 9. MLT & DLT.................. 34
Table 10. IP & CV.................. 36
Table 11. Multiple controllers for Servo Motors.................. 38
Table 12. Main Characteristics of each author.................. 46
Table 13. Characteristics of each class of Database.................. 53
Table 14. Necessity for the robotic arm.................. 61
Table 15. Concurrent design on an as-needed basis for the robotic arm.................. 62
Table 16. Properties of material (ABS and PLA) [199-200].................. 63
Table 17. Features of arm parts.................. 64
Table 18. Total cost and unit price of material costs for the construction of the robotic arm.. 66
Table 19. Human Database characteristics.................. 70
Table 20. Robot Database characteristics.................. 72
Table 21. CNN configuration options.................. 73
Table 22. CNN configuration options.................. 74
Table 23. TP, FP, FN and TN values for the AlexNet model with Human Database.................. 81
Table 24. TP, FP, FN and TN values for the GoogleNet model with Human Database.................. 83
Table 25. TP, FP, FN and TN values for the AlexNet model with robotic Database.................. 86
Table 26. TP, FP, FN and TN values for the GoogleNet model with robotic Database.................. 88
Table 27. All important data obtained of each model.................. 89
Table 28. Comparison of other studies with our model.................. 100

Abbreviation

WHO: World Health Organization
dB: Decibels
CV: Computer Vision
AI: Artificial Intelligence
AV: Artificial Vision
CNN: Convolutional Neural Networks
RNNs: Recurrent Neural Networks
GANs: Generative Neural Networks
ASL: American Sign Language
PLA: Polylactic acid
ABS: Acrylonitrile Butadiene styrene
NLP: Natural Language Processing
IR: Information Retrieval
MLP: Multilayer Perceptron
ROS: Robot Operating System
IP: Image processing
PIL: Python Image Library
ML: Machine learning
IDE: Development Environment
DL: Deep learning
SNNs: Siamese Neural Network
UML: Unified Modeling Language
RUP: Rational Unified Process
TP: True Positive
FP: False Positive
FN: False Negative

THESIS OVERVIEW

For our Thesis Project, we have divided all our work into five chapters.

The first is a brief introduction that shows the background, justification, and objectives of this research, which involves creating/assembling a Robotic Hand and implementing it with a CNN to classify American Sign Language (ASL).

In the second chapter, we have our theoretical framework that deals with all types of Robotic Hands, every material that we can use in a 3D printer and successfully assemble for our model, and every kind of motor controller. Another important point we have is the study of hearing impairment and the implementation of sign language. In addition, we have a strong review of artificial intelligence (AI), including all types of learning and neural networks. Finally, we will describe the software and its important toolbox and applications.

The third chapter is about the methodology, mainly the creation of the prototype of our thesis project based on AlexNet and GoogleNet Neural Networks. Also, all the modifications that are required on AV and the System integration of Computer/Hand prototype with their indispensable Experiment and Optimization process.

The fourth one shows the results that we obtained by the AlexNet and GoogleNet model, their learning plots and confusion matrix of each, the Grad-Cam, the visualization mode, and their prediction with their probabilities in a bar plot. The GoogleNet model indicates fewer variable results than AlexNet, but in some letters like B or X, the AlexNet model had more accuracy of detection than GoogleNet.

The fifth one describes the main characteristics of the comparison between seven authors, each of them with different models of Sign Language and different costs of a robotic hand. The distinguishing difference in our model was the cost (App $164) of the creation of the robotic hand. Also, we used several processes to improve the detection algorithm, such as color detection and background elimination in comparison with other authors.

Finally, in the last chapter, we discuss the conclusion of our research work, our future perspectives, and our limitations. We propose using another current model with more degrees of freedom to improve our system and help sign language become more practical.

CHAPTER I

INTRODUCTION

Hearing loss affects a significant portion of the world's population, with more than 5% of individuals experiencing disabling hearing loss; this number is expected to rise to 2.5 billion people by 2050, according to the World Health Organization (WHO) [1]. For many individuals, hearing deteriorates progressively, impacting their ability to communicate effectively [2]. Communication is fundamental to social interaction, as it enables the sharing of ideas, emotions, and needs. Thus, inclusive communication tools are crucial, especially for those with hearing impairments who rely on alternatives to spoken language. While sign language serves as a powerful medium, its adoption faces challenges, including the need for specialized interpreters and training for non-signers.

Previous approaches to bridge this communication gap have included wearable technologies, such as gloves equipped with sensors and camera-based systems that recognize hand gestures to translate sign language. For example, Naranjo-Zeledón's systematic review (2019) discusses various methods, such as Visualfy (which translates sign language to visual signals) and SignLanguageGlove (designed to convert sign language to voice), while other methods use Deep Learning algorithms for gesture recognition, as seen in Google's AI tracking algorithms using a DataSet of 30000 images. Also, other projects, such as Huawei's StorySign, employ avatars to translate written stories into sign language for children, aiding in their learning process [3].

Despite these advances, current solutions still encounter limitations, such as low accuracy in gesture recognition, high processing times, and difficulties in real-time response, especially when translating complex alphabetic signs. Our project aims to overcome these difficulties by designing and developing a robotic arm that interprets the ASL alphabet, enhanced by a graphical interface using neural networks for accurate alphabet recognition using MATLAB to detect the hand by color recognition and trained using a database with images from 3 different angles and applying data augmentation to ensure its accuracy.

This approach addresses the technical obstacles and aims to improve the social inclusion of hearing-impaired people by offering a solution that is potentially more accurate, responsive, and widely applicable to different social environments. On the other hand, the use of the robotic arm will ensure the interest of youngsters in learning more about sign language and generate an inclusive approach towards deaf people at an earlier stage. This project is designed to be implemented in educational centers so that students with and without hearing disabilities can learn and practice sign language in order to foster an inclusive community without communication barriers.

1.1. JUSTIFICATION

Hearing loss is a condition that prevents people from perceiving sounds optimally, compared to hearing a person with normal hearing ability, which is considered to be around 20 decibels (dB), according to the definition provided by the WHO. In other words, hearing impairment is experienced, although it is not necessarily classified as total hearing loss. According to WHO, "loss" implies a decreased hearing ability rather than a complete elimination of hearing [4].

It is important to note that hearing loss is not only a binary condition of "hearing" or "not hearing" but encompasses a range of degrees and types. About 18% of adults between the ages of 20 and 69 experience decreased hearing in different frequency ranges [5].

This variability in hearing loss may be due to various factors, such as prolonged exposure to loud noise, natural aging of the auditory system, and other medical conditions [6]. Individuals experiencing this hearing impairment may encounter difficulties in everyday communication, highlighting the importance of hearing health awareness and appropriate prevention and care measures to preserve hearing throughout life.

Communication plays an essential role in our lives, and hearing loss can create obstacles to participation in society. People with hearing impairment often encounter difficulties when interacting with others, leading to frustrating situations and, in the most severe cases, social isolation [7]. This could hurt their self-esteem and self-confidence, as they may feel ununderstood or fail to express themselves fully.

Sign language (SL) was created to overcome these difficulties. This system represents letters and words through hand gestures and body movements [8]. SL has evolved, and its development has provided a powerful tool for people who are deaf or hard of hearing.

1.2. PROBLEM STATEMENT

Different SL systems have been developed worldwide, each with variations and dialects. These systems have been enriched by incorporating specific grammar, vocabulary, and linguistic structures. In addition, with the advancement of technology, deaf people have found new ways to disseminate and share sign language through digital media and online platforms [9].

Just as the human mind designed a method for SL communication, many technological tools exist. The most widely used are computer vision (CV) models, where you can identify patterns of certain objects in real time, allowing computers to interpret and replicate human visual perception ability using algorithms and computational techniques [10].

On the other hand, neural networks are another more simplified form of Artificial Intelligence inspired by the structure and functioning of neurons in the brain. These networks are designed to process information, learn patterns, and perform specific tasks. Their main system comprises "neurons or nodes" organized in connected layers. Several types of neural networks are designed for particular tasks [11-12].

Among these types are feed-forward neural networks, which transmit information in one direction from the input layer to the output layer. Also, we have CNN, designed to process gridded data, such as images, and are essential in CV applications due to their ability to detect visual features; recurrent neural networks (RNNs), especially useful in tasks involving data sequences and temporal dependencies; generative neural networks (GANs), which generate new realistic data based on a given data set.

1.3. OBJECTIVES AND WORK METHODOLOGY

A series of objectives, detailed below, have been taken into account to carry out this work of curricular integration.

1.3.1. General Objective

- To develop an innovative tool focused on people with hearing disabilities.

1.3.2. Specific Objectives

- Design and build a robotic arm focused on joint movement, controlled by an electronic board with a microcontroller.
- Create a database with images for ASL.
- Implement a neural network for ASL classification.
- Perform the preprocessing of the database images for ASL.

- Represent the ASL letters through the robotic arm.
- Develop an AV system for the detection of ASL letters.
- Development of a code for positioning the servomotors of the robotic arm.
- Recreate the ASL with a robotic arm and interpret the movements through a neural network.

CHAPTER II

THEORETICAL FRAMEWORK

2.1. ROBOTICS ARM

The word robot comes from the Czech word robot, a, which means "Forced labo.' r' This term was first used in a play by Karel Capek in the 1920s [13]. According to the Oxford Dictionary, a robot is a machine capable of carrying out a complex series of actions automatically [14].

2.1.1. Types

There are many types of robots, among which we have Cartesian robots, spherical robots, scary robots, articulated robots, and anthropomorphic robots, see Figure 1.

A Cartesian robot consists of 3 prismatic joints whose axes coincide with the Cartesian coordinate [15]. This type of robot stands out for its precision, speed, and constant load capacity in the x, y, and z directions [16]. Its applications are assembly operations, machine tool manipulation, spot welding, etc. [17].

Spherical robots were the first robots to appear on the market [15]. Their axes form a polar coordinate system [16] consisting of two perpendicular rotational axes and a linear one. Spherical robots are more accessible and have better load capacity than Cartesian robots [15].

Robot Scara is composed of three parallel axes, two rotational and one linear. This configuration results in very fast robots with very high precision, which are often used in assembly or packaging operations and simple parts-taking movements [15].

Anthropomorphic robots are robotic arms similar to the human hand, consisting of fingers and independent thumbs [16]. They consist of three rotating axes, the first one perpendicular to the ground and the other two parallel to each other. They are very fast robots that occupy little space and allow complex trajectories [15].

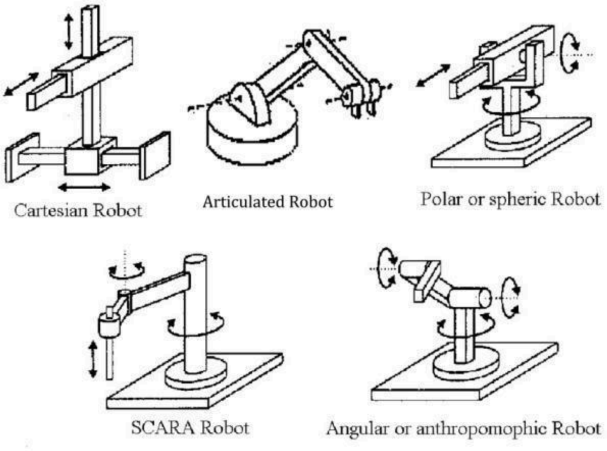


Figure 1. *Cartesian robots, articulated robot, polar or spheric robot, Scara robot and angular or anthropomorphic robots [17 - 18].*

On the other hand, degrees of freedom are the independent movements that a joint can perform [15]. Generally, the different types of robotic arms can have two or more degrees of freedom, except for the anthropomorphic robot, which contains more than 5 degrees of freedom as it simulates the human arm.

2.1.2. Materials

Robotic arms can be made by 3D printing. The most commonly used materials are PLA and ABS, but 3D printing is a much simpler and cheaper method than the materials used for industrial arms, see Table 1. In addition, due to their industrial use, robotic arms can be made with different types of material, such as metal.

Table 1. *Characteristics of each class of Database.*

Material	Image	Description
Polylactic Acid (PLA)		PLA is mainly composed of Lactic Acid [19]. This material is thermoplastic due to its biodegradable and biocompatible characteristics and is used for medical purposes [20]. Moreover, this material is used in the manufacture of robotic arms because it can be used to print 3D parts.

Acrylonitrile Butadiene styrene (ABS)		ABS was implemented in the 1950s and used to describe multiple acrylonitrile blends and copolymers [21]. It has a good impact, good resistance, and high aesthetic qualities [22].

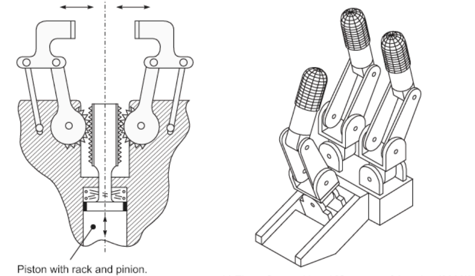
2.2. CONTROL SYSTEM

A control system is an efficient combination of the software and hardware of a machine with the aim of automating and streamlining processes as well as minimizing errors that these can commit. In 1959, we learned of the first machine controlled by a computer through a wiring system that allowed it to follow a set of basic commands. In 1968, there was an evolution in the previous systems, allowing the use of modular control and implementing a rustic system of automation in the processes of assembly lines [23].

In robotics, the implementation of control systems is essential to achieving the desired objectives. In simple terms, it is crucial to have methods to adjust a robot's movement or action as needed. For this purpose, control systems are used, which consist of sets of devices or components designed to maintain predefined safety parameters in an optimal and efficient state and to be easily programmable [24].

2.3. ACTUATORS

Actuators are an essential part of any control system. They are devices responsible for receiving an input signal (usually electricity) and transforming it into motion or force, which is essential for carrying out automation processes [25], see Figure 2.



a) Single fastener [Courtesy of Mair (1988)]. b) Three-fingered hand [Courtesy of Angeles (2003)].

Figure 2. *Robot arm [26].*

2.3.1. Kinds of Actuators

Several types of actuators, including pneumatic, hydraulic, electric, and mechanical actuators, are adapted to different application areas shown in Table 2.

Table 2. *Kinds of Actuators.*

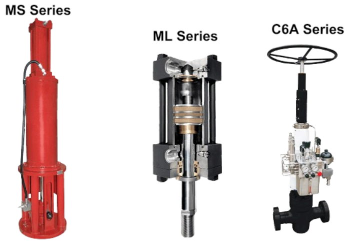
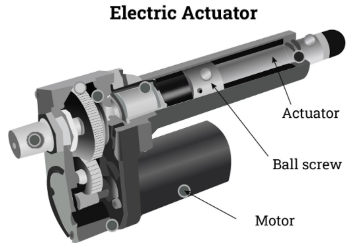
Kind of Actuators	Image	Description
Pneumatic Actuators	 **Figure 3.** *Pneumatic actuators [28].*	It is a device that transforms air pressure into mechanical movement. Its main use is operating valves and switches when efficient pressure control and quick responses are required, as shown in Figure 3 [27].
Hydraulic Actuators	 **Figure 4.** *Hydraulic actuators [30].*	It is a mechanical device that uses the energy of a fluid in confinement to generate mechanical movement, such as the movement of valves or the control of the flow of gasses or fluid, Figure 4 [29].
Mechanical Actuators		These devices implement various mechanisms to generate motion. They mainly use gears, levers, or cranks. Their use is simple, and they have applications in

	Figure 5. *Mechanical actuators [31].*	low-cost systems [25]. For example, the mechanical actuator presented in Figure 5 mainly functions as a zipper mechanism.
Electric Actuators	 **Figure 6.** *Electric Actuators [33]*	It is a device that uses electrical energy and transforms it into physical movement. These can be operated by DC or AC motors, see Figure 6. The advantage of this type of actuator is that they are highly accurate, generate little noise compared to other actuators, and require little maintenance [32].

2.3.2. Stepper Motors

A stepper motor is an electric motor that transforms electrical impulses into incremental mechanical movement. This means that it is not continuous but a movement in discrete steps. A stepper motor has several components for its operation: a Rotor and a Sator [34].

A rotor is a rotating component consisting of an electromagnet which allows it to rotate continuously around its axis, see Figure 7.

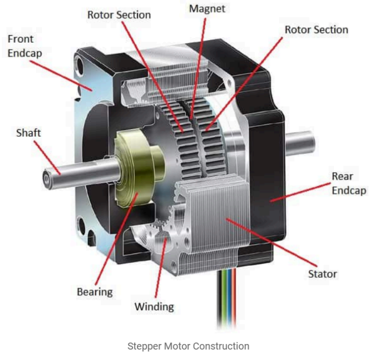


Figure 7. *Hydraulic actuators [33].*

A stator is a fixed part where the areas in which the electric current is generated through the rotor are housed [36]. There are several types of stepper motors which are shown in Table 3.

Table 3. *Types of stepper motor.*

Name	Descriptions
Unipolar Motor	They are characterized by having one coil per phase and can be used to change their direction by inverting their polarity. Due to their ease of use and low costs, they are commonly used in power circuits.
Bipolar Motors	They differ from the Unipolar by having two coils per phase connected in series or parallel. They present a configurable change of direction, which is obtained by reversing the current in the coils [37].

2.3.3. Servomotors

A servo motor is an electronic component that works by direct current and accurately controls aspects such as speed and position. It has three main pins(Power, GND, and PWM) that connect it to a microcontroller [38]. It works in most cases with a 5 V power supply and, depending on the force with which it is handled, uses between 1 mA for proper operation.

Unlike stepper motors, servo motors are used mainly because of their high precision and higher torque force application. These devices are equipped with feedback mechanisms that allow for the control of real-time information about the position and speed at which the servomotor is adjusted. This allows the continuous adjustment of the position of the servomotor by means of a built-in potentiometer, preventing its movement if it is not in the adjusted position, thus helping to perform precise movements [39]. Its application is very diverse, from its use in industrial automation components to its use in the field of robotics.

The main components of a servomotor are the electric motor, regulation system, control system, and potentiometer, shown in Figure 8.

The electric motor is in charge of moving the servo motor shaft. The regulation system is responsible for acting on the motor in order to set its speed; it is formed by gears that are adjusted to the motor [40]. The control system monitors the movement of the motor by

constantly sending electrical pulse signals. Finally, the potentiometer, a variable electrical resistance, allows us to determine in real-time the angle at which the motor is configured.
It is important to emphasize that they have different features that make them unique, from a metal chassis that allows you to move loads of more than 10 kg to 60 kg depending on the mode to an almost plastic chassis used for servo motors below 5 kg. On the other hand, these devices are highly compatible with microcontrollers such as Arduino and Raspberry electronic cards, which can be configured easily.Also, there are two types of servo motors implemented in the use of robotics, seen in Table 4.

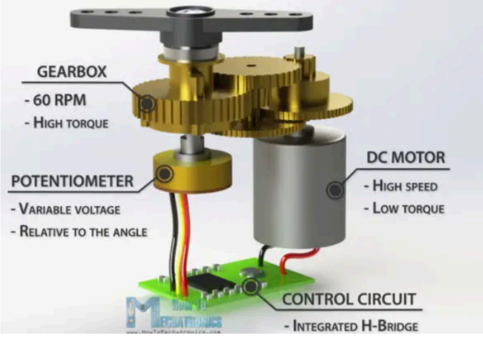


Figure 8. *Parts of servomotors [41].*

Table 4. *Type of servo motors.*

Servomotor Degrees	Characteristic
180°	The main characteristic of this servomotor is that its movement is limited to forming an Angle of 180° in its potentiometer. An electronic card or microcontroller is essential to control it. Generally, its manipulation is done by determining the angle at which you want to change its position.
360°	Unlike the 180° servo motor, this one performs an entire 360° movement. Because it does not present any limitation in its movement, it can be used in several applications, such as the use of wheels in electric cars. However, the configuration method of this servomotor does not work in degrees but at the speed that you want to operate.

The characteristics that helped us to decide the best actuator to implement in our model are detailed in Table 5. In it, we can see that controlling the position of the joints of the robotic arm is essential to make precise movements for this electric motors are a viable option; on the other hand, both models, stepper motors and servo motors, are viable options; however, by needing six motors the stepper motor option would exceed $ 164 while servomotors guarantee us the most economical option being $ 72 so this is the most economical option without losing accuracy.

Table 5. *Comparison between several models of actuators*

Actuator Type	**Model**	**Dimensions**	**Max Force/ Torque**	**Operating Temperature**	**Operating Pressure**	**Approximate Cost**	**Application**
Pneumatic	DMSP-5-50N-RM-CM [42]	50 mm x 12 mm	14 kg	N/A	0-6 bar	154,93 €	Light applications
	DMSP-40-120N-RM-CM [43]	120 mm x 80 mm	600 Kg	N/A	0-6 bar	281,64 €	Heavy applications (up to 250 kg)
	DMSP-20-100N-RM-CM [44]	100 mm x 40 mm	150 kg	N/A	0-6 bar	159,60 €	Medium loads (up to 80 kg)
Hydraulic	Bettis G-Series Hydraulic [45]	N/A	3,000,000 lbs-in (339,000 Nm)	-29°C to +93°C	Up to 345 bar	N/A	High power and durability
	Bettis GVO Linear Pneumatic [46]	N/A	160 kg	-60°C to +100°C	12 bar	$100-$500	High torque in linear operations
Mechanical	AC Input 60 mm Horizontal [47]	100-800 mm (Stroke)	10 kg	N/A	N/A	$953.00	High speed in horizontal movements
	AC Input 60 mm Vertical [48]	100-800 mm (Stroke)	5.2-9.5 kg	N/A	N/A	$953.00	High speed in vertical movements
Electric (Stepper Motor)	5.7V, 1 A NEMA 23 POLOLU [49]	56.4 mm x 41 mm	Four kg-cm	N/A	N/A	$41.99	High precision at low cost
	7.4V, 0.28 A NEMA 14 POLOLU [50]	35 mm x 26 mm	650 g-cm	N/A	N/A	$26.99	Compact and low-power

Electric (Servomotor)	TOWER PRO MG996R 180° [51]	40.6 x 19.8 x 42.9 mm	13 kg-cm	-30 to +60 °C	N/A	$11.99	Angle control with moderate torque
	TOWER PRO MG995 360° [52]	40.7 x 19.7 x 42.9 mm	11 kg-cm	0 to 55 °C	N/A	$14.99	Continuous angle control

2.4. HEARING IMPAIRMENT

Hearing impairment is defined as hearing loss greater than 35dB, which is why, within this concept, we have hyperacusis and deafness [1]. According to the WHO (World Health Organization), it is expected that by 2050, there will be 2,500 million people with some degree of hearing loss and that at least 700 million need rehabilitation [1]. Only in Ecuador (2022) does hearing disability occupy the third place among the most common disabilities in the country, with 14. 12%, approximately 66,538 have some degree of hearing disability, Figure 9 [53].

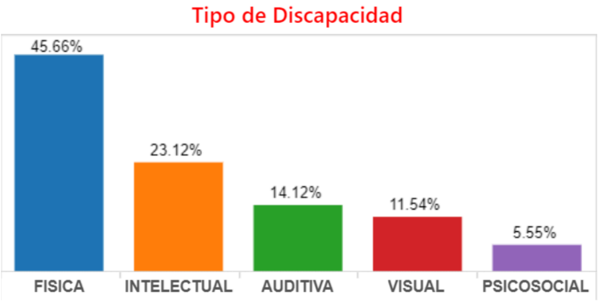


Figure 9. *Percentage of persons with hearing disabilities registered in Ecuador in 2022 [53].*

Hearing impairment can occur due to various factors, hereditary, noise exposure, meningitis [54], or age, as most older people have some degree of hearing loss [55]. A significant hearing loss can affect daily life; the main limitation that people with hearing disabilities have is at the intrapersonal level, unable to relate to other individuals in the various environments of daily life to the point of low or zero possibilities in the labor field in front of hearing people [56].

2.4.1. American sign language

In the 1970s, linguistic analysis began, and ASL included different parameters such as hand shape, movement, location, and palm orientation [57]. The American alphabet forms an important part of ASL, Figure 10; it has a set of 26 known letters, of which 19 different manual forms are used to express the various letters of the alphabet only with different orientations and are used to spell words in English. It is commonly used for proper names, technical terms that have no native ASL equivalent, very short English words, or very long English abbreviations [58].

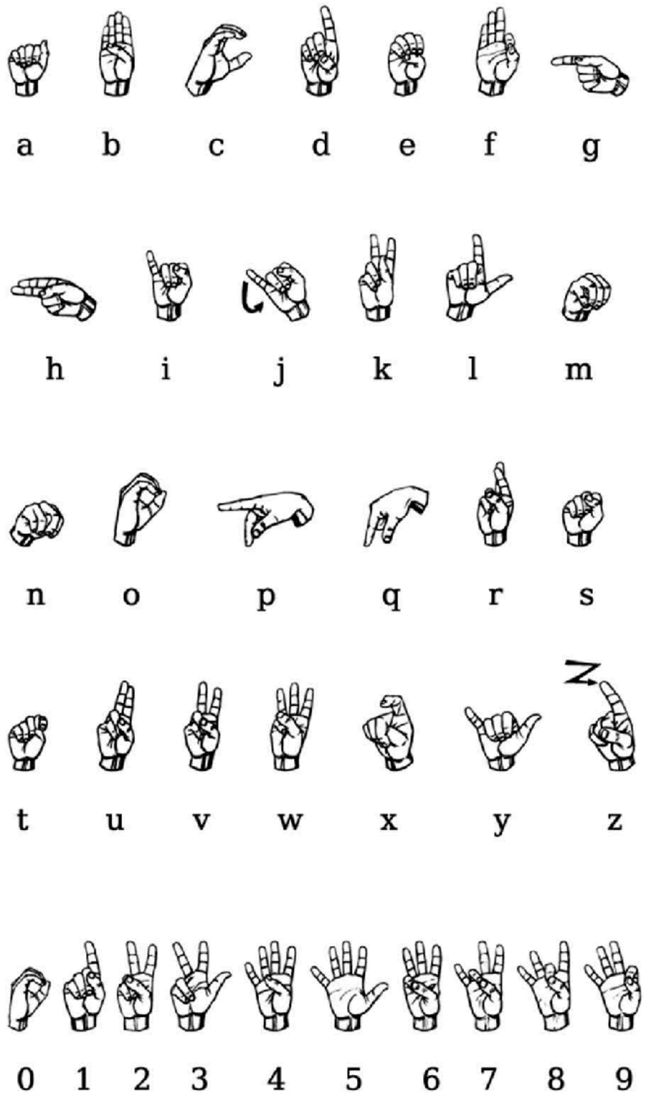


Figure 10. *ASL Alphabet [59].*

2.5. IMAGE PROCESSING

Neural networks require various input models for training. In the case of CNNs, images are necessary. Using several image samples allows us to perform convolution processes in which the most noticeable features are extracted to find patterns with respect to the desired objectives.

The quality of the sample images represents an important point in the result of a neural network. In addition, they need to know the accuracy with which they can detect patterns. However, only some images present an ideal environment for obtaining them, so they can present certain errors and impair the desired features. Therefore, certain algorithms are implemented to improve image detection through AV and feature extraction in neural networks. An image enhancement leads to a better interpretation of patterns and highlights desired features to obtain more accurate results.

According to the compilation of image processing (IP) processes [60], several types of processing can be performed. The ones used for the proposed work are median filtering or Medfilt2 [61]. This performs noise reduction on the image, replacing the value of each pixel with the average value in nearby areas, thus avoiding edge smoothing.

The median filter is a common IP technique for reducing noise in an image. It works by replacing each pixel in the image with the median value of the pixels in its vicinity, which helps to remove noise without smoothing the edges too much. Morphological operations were performed to modify the shape and size [62] of the objects in the images.

Finally, two toolboxes, the Image Processing Toolbox [63] and the Computer Vision Toolbox [64] were used to limit important characteristics in the image, such as color, eccentricity, orientation, and equivalent diameter, with the objectives of filtering and object detection.

2.6. ARTIFICIAL INTELLIGENCE

AI aims to endow machines with human intelligence, focusing on intelligent agents that perceive and act, as described [65] and Figure 11. This term is applied when machines perform human functions. In other words, AI refers to any software and hardware technique that reproduces the behavior and thinking of human beings and also includes ML, natural language processing (NLP), text and speech synthesis, CV, robotics, system planning, etc [66].

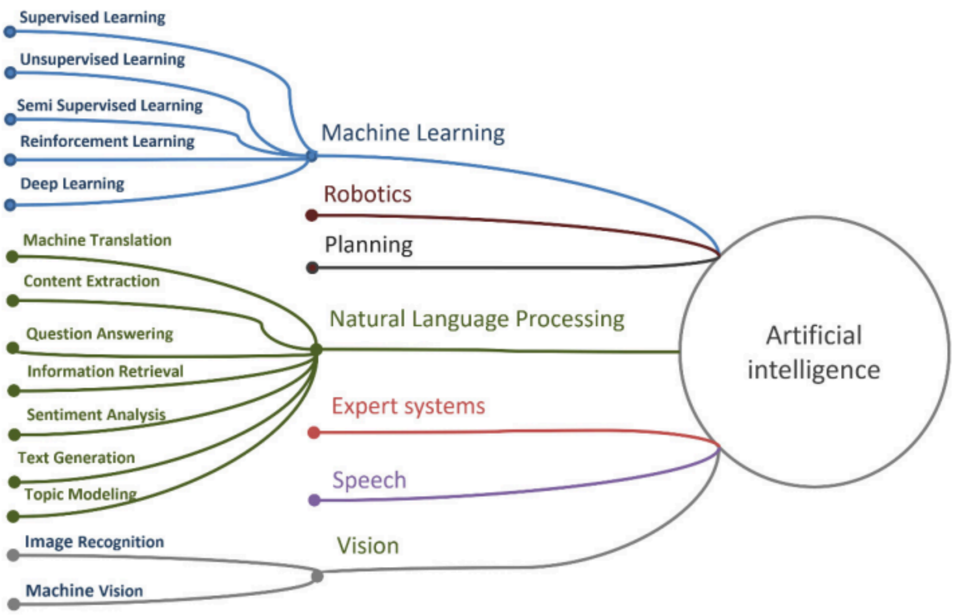


Figure 11. *Artificial Intelligence applications [66].*

2.6.1. Robotics and planning

Robotics is a discipline that focuses on designing and building machines programmed to perform specific activities automatically, in addition to being able to emulate human behavior. Also, robotics is defined as “the intelligent connection of perception to action” [67]. In simple words, it describes the movement that can be performed by a computer thanks to a moving part, either an arm or a gripper. The use of various tools allows artificially created devices to use “reasoning” in a primitive state, capable of perceiving space and distances between objects.

The rules that a robotic device must follow allow it to plan a method or a sequence of algorithms to obtain an efficient or desired result. Furthermore, planning in robotics can be defined as “all the steps and operations that transform the initial state into the ultimate goal state” [68], thus arriving at the most optimal solution.

However, the progress made in robotics has been overwhelming. Nowadays, several robots are focused on specific tasks with a minimal range of error. Industrial robots, for example, have revolutionized manufacturing by increasing precision and efficiency in mass production. Robots, such as robotic assembly arms used in automotive manufacturing, can perform welding and assembly with impressive accuracy [69].

In addition, the complement of AI represents unprecedented advances and constant evolution. In medicine, surgical robots, such as the Da Vinci [70], allow minimally invasive surgeries to be performed with a precision that surpasses human capabilities, reducing recovery time and postoperative complications.

2.6.2. Natural Language Processing

Neuro-linguistic programming has its roots in the 1950s with Alan Turing in his 'Machine and Intelligence' paper [71]. NLP is a discipline of computer science that analyzes text using specific theories and technologies, the goal of which is to achieve human-like language processing for various applications [72]. NLP is divided into Machine translation, Content extraction, Question answering, Information retrieval (IR), Sentiment analysis, Text generation, and Topic Modeling, see Table 6.

Table 6. *Types of NLP*

Kind of NLP	Description
Machine Translation	Machine translation is a subfield of NLP focused on translating text or sounds from one language to another automatically. In simple terms, it is an algorithm that is in a loop trying to obtain a promising result or with significant features higher than other results to ensure a translation closer to the original language [73]. However, this does not guarantee an efficient translation. An example of this system [74] is the software created by Google to translate various languages: " Google Translate. "
Content Extraction	Feature extraction is essential to transforming unprocessed data into useful information, and feature selection is key to processing dimensional data [75-76]. In other words, this field focuses on identifying, labeling, and extracting key information such as person names, companies, locations, or organizations [72].
Question Answering	As part of NLP, it is essential that the system is calibrated to obtain the best result. In this section, the best result must be found using the available resources by associating different interactions or processes

	[73]. On the other hand, [77] details that the language can be defined by means of its syntactic structure with which functions or transformations can be applied, mixing it with the operation of the program. Thus, it is essential to analyze these sections in order to obtain conclusions and a useful prediction.
Information Retrieval	IR systems, also called document or text retrieval systems, aim to find documents that match the user's information needs [78]. In addition, IR has used many NLP techniques such as stemming, chunking, tagging, recognition, etc [79]. The result of an IR process is usually a group of documents containing data on a particular topic [80].
Sentiment Analysis	Sentiment analysis is performed using the NLP technique. It studies the user's expressions and relates them to the emotions conveyed [81]. Sentiment expressions include adjectives, adverbs, and verbs, for example, "'good product"' where we want to convey a positive emotion towards a product [82].
Text Generation	Another section within NLP is text generation. These follow the aspects previously mentioned in the planning stage. Thus, the aim is to emulate fluent and understandable writing for users [83]. Currently, there are a multitude of structures for text generation using RNNs, dilated convolutional networks, and tokens generated previously [84].
Topic Modeling	Topic modeling is a process described as the identification of a group of words in a set of text. It is part of text mining which uses data mining algorithms, NLP, ML,etc. [85].Also, topic models are a powerful and practical tool for analyzing huge text documents in NLP [86].

2.6.3. Expert systems

In past years, the term “expert system” was attributed to software developed to achieve the performance of a human being with extensive knowledge in a specific area, obtain detailed results, and explain them efficiently [87].

Currently, an expert system is considered a subgenre of AI whose objective is to function as a decision-making system that emulates the knowledge of an expert in an area of work. These systems operate primarily through a database containing “knowledge and expertise” along with a set of rules [88].

2.6.4. Speech and Vision

Speech recognition gives computers the ability to transcribe spoken words into text for analysis. This is complemented by the NLP technique, which is also integrated into various electronic devices, providing them with intelligent and automated functions [89]. CV simulates the human ability to see by employing software and hardware to analyze and process visual information. It has applications in a wide range of fields, from medicine to autonomous driving [90].

2.6.5. Machine learning

ML classification is divided into three categories: supervised, unsupervised, and semi-supervised [91], see Figure 12.

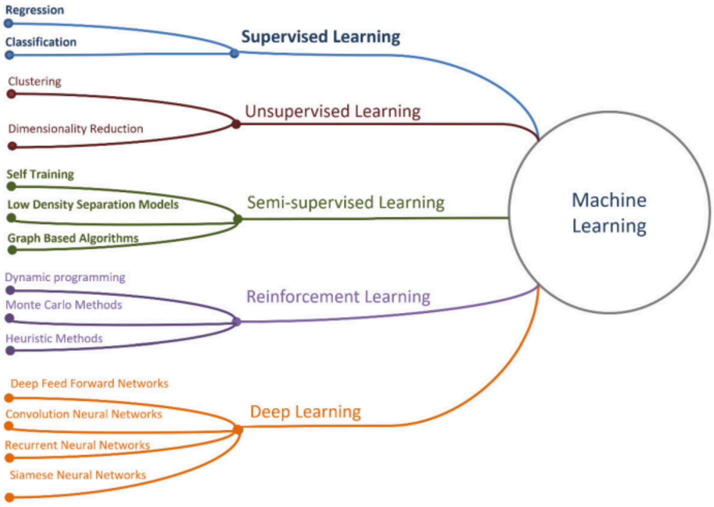


Figure 12. *Machine learning [66].*

- **Supervised machine learning**

Supervised learning is a model that performs decision-making based on the classes or labels assigned to the initial data set and configures both input and output parameters to perform prediction efficiently [92]. It adjusts internal parameters iteratively using methods like stochastic gradient descent to minimize errors. Deep learning (DL), characterized by a multilayer architecture, autonomously learns intricate features, addressing selectivity-invariance challenges without manual engineering. Shinde, P. P., & Shah, S. (2018) explain that evaluation is conducted on an independent test suite to assess generalization [65].

- **Unsupervised Learning**

In contrast to supervised learning, which makes decisions based on the configuration of specific inputs and outputs to obtain its results, unsupervised learning only uses the inputs. It seeks to discover patterns or features in the data to generate its results [93]. This approach is especially useful in situations where labeled output data is not available, allowing the algorithm to explore and analyze the data without direct human supervision.

Furthermore, unsupervised learning is a type of adaptive algorithm, as it dynamically adjusts to the input data to identify and extract relevant and meaningful features [94]. This adaptability is crucial in contexts where the data is complex or unknown, allowing the algorithm to adapt to new information and discover hidden structures without prior guidance.

Finally, unsupervised learning is ideal for feature selection and reduction tasks on very large datasets [95]. Identifying and removing irrelevant features simplifies data analysis, improves efficiency, and facilitates the interpretation of results. This is particularly beneficial in fields such as data mining, bioinformatics, and big data analysis.

- **Semi-supervised Learning**

A combination of the previous models gave rise to semi-supervised learning. This is characterized by using an unsupervised scheme (unlabeled output data) in supervised learning problems, where we are given labeled input and output data [96]. Furthermore, Hady (2013) highlights that semi-supervised learning is divided into four groups: semi-supervised classification, semi-supervised regression, semi-supervised clustering, and semi-supervised dimensionality reduction [96]. Finally, Van Engelen (2019) points out that semi-supervised learning has a weakness, which is the reduction in performance when applying unlabeled data; this is an aspect that needs to be strengthened to avoid minimal improvement compared to other methods [97].

- **Reinforcement Learning**

Reinforcement learning is a significant point in the types of ML because it responds to a situation of action and reward; in this case, it is not programmed to follow an established pattern but must find the set of actions that can generate a positive or approving response, based on an automated trial and error analysis [98]. Kaelbling (1996), in his article, points out that the action of an agent changes depending on its interaction with the environment by means of a scalar reinforcement signal, thus developing learning by reinforcement that complies with pre-established algorithms for its correct functioning [99]. Finally, current studies mention that this learning method can develop solutions that a trained person can take a good amount of time to deduce, thus developing skills and behaviors in an exponential way [100].

● Deep Learning

DL classification is divided into four categories: DeepFeed-Forward networks, Convolution Neural Networks, RNNs, Siamese Neural Networks (SNNs,) and Transformer, shown in Figure 13.

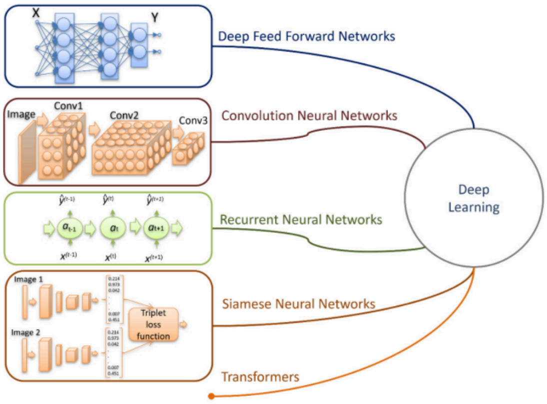


Figure 13. *DeepLearning [66].*

○ Deep Feed Forward Network

The first generation of neural networks were Feed-Forward networks. These are called one-way because the information moves in only one direction, from the input layer to the output layer. According to Ketkar (2021), it is essential to understand their architecture, which consists of multiple layers of neurons, where each neuron is connected to each neuron in the next layer [101]. In addition, these networks use transfer functions, such as the Heaviside threshold function, to introduce nonlinearities in the model [102].

According to Huang's (2020) research, it is uncommon to consider a feedforward neural network with more than 30 layers. These types of networks tend to have a higher error rate compared to their simpler counterparts [103]. This is due to problems such as gradient degradation, which makes training more difficult.

A typical example of a feedforward network is the multilayer perceptron (MLP). This nonlinear neural network receives a sum of the incoming signals, and applying a nonlinear activation function produces an output [104]. The MLP is widely used in various applications due to its ability to model complex and nonlinear relationships between input and output data.

○ Recurrent Neural Networks

The term “recurrent neural network” refers to any type of neural network that is capable of sending feedback signals, just like a biological neural network, and capable of storing large amounts of data such as patterns, scales, and colors, thus modulating learning [105]. These neural networks were created to emulate pattern learning in dynamic systems with a sequence of events. Some examples of this type of neural network are the Hopfield network, the Boltzmann machine, and recurrent backpropagation networks [106].

Unlike the previously mentioned networks, this type of neural network uses all the available input information to guarantee a good prediction instead of using a fixed number of inputs [107]. In fact, their performance is given by the configuration prior to training, such as the activation function, the number of layers, etc.

Currently, the applications of RNNs are much broader, and they are used to predict various variables more accurately. An example is the research of Nketiah (2023), who focused on predicting atmospheric temperature using meteorological parameters such as temperature, dew, and wind speed, guaranteeing an optimal result with minimal errors in the five models he performed [108].On the other hand, they can also be incorporated into CV, detecting complex patterns and activating a response system [109].

Finally, the versatility and accuracy of this type of neural network for pattern prediction make it a powerful tool in research, significantly expanding the fields of study.

○ Siamese Neural Network

SNNs, which began in 1994 by Bromley and his collaborator, are described as an algorithm for verifying a signature written on a tablet with pen input [110]. As the name implies, they are inspired by the concept of Siamese twins. They are formed by twin models entered by supervision and are notable for their outstanding approach to one-time learning in face recognition [111]. Furthermore, SNNs are symmetrically structured to process multiple inputs and determine similarity within a specific feature space [112].

The SNNs consist of two parallel CNN architectures, two segments, one convolutional layer followed by pooling, and three fully connected layers. They are designed for 128x128x3 pixel input. In addition, they consist of max-pooling layers operating in 2x2 non-overlapping regions [113].

○ Convolution Neural Network

CNNs, ConvNet, are a type of DL model for processing data as grid-like patterns, such as images that contain a series of pixels, which are values that indicate color and brightness [114]. A CNN learns according to the characteristics using convolution and clustering and is fully connected to extracting and classifying patterns from images [115-116], see Figure 14.

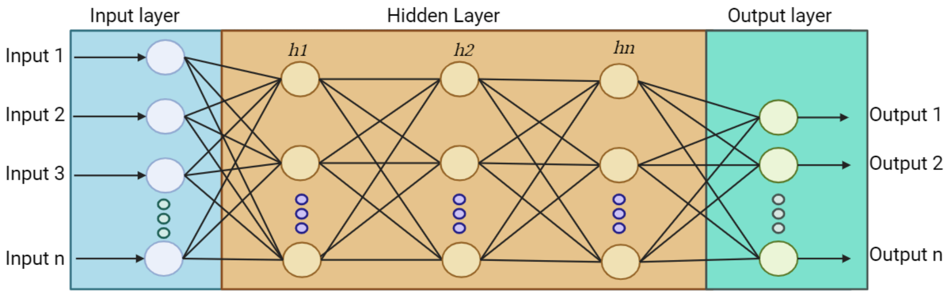


Figure 14. *The basic architecture of CNN based on [103].*

Several pre-trained CNNs are already trained to classify images, such as SqueezeNet [117] is a CNN; this network is smaller and designed to replace AlexNet as it is smaller than other models with only 18 layers, which is almost 50 times fewer parameters than AlexNet. SqueezeNet [118] features an architecture with "compression" and "expansion" layers. In the compression layer, filters of size 1 × 1 are used, fed into an expansion layer that combines 1 × 1 and 3 × 3 convolution filters, called the "trigger module".

The AlexNet model consists of 5 convolutional layers, three max-pooling layers, two normalization layers, two fully connected layers, and one softmax layer. Although the input size is usually stated to be 224x224x3, the padding makes it 227x227x3. Overall, AlexNet has a total of 60 million parameters. GoogleNet [119], the overall structure comprises 22 layers. It was conceived to focus on computational efficiency and is designed to process images of 224 x 224 dimensions with RGB color channels. DarkNet-19 [120], a 19-layer CNN, serves as the backbone of YOLO.v2. Pretrained on a subset of ImageNet with more than one million images, this network can classify images into 1000 categories, standing out for its strong recognition capability.

2.6.6. Google-Net

GoogleNet was an important breakthrough in CNNs, demonstrating efficient results with optimized architectures. According to the research of Szegedy et al. (2015), this model was characterized by using inception [121]. The module was efficient by greatly reducing the parameters needed for training the network. Its architecture comprises 9 inception modules, containing 22 layers along with 4 maxima clustering layers and one averaging clustering layer,see Figure 15. In addition to the ReLu function in all convolutional layers. Finally, a dropout of 40% was applied to the SoftMax classification function [122].

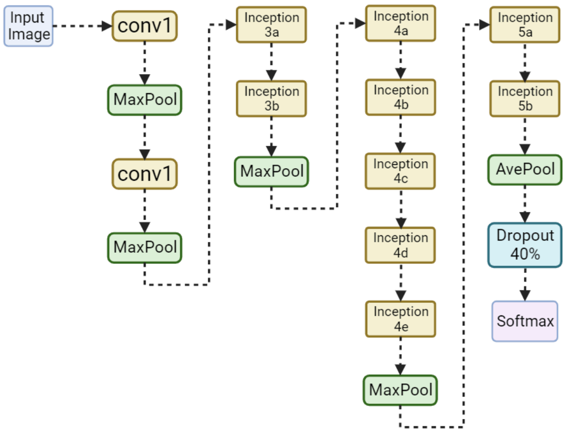


Figure 15. *GoogleNet scheme based on [122].*

2.6.7. Alex-Net

In 2012, Alex Krizhevesky and his team presented a version of a deeper and more extensive CNN model compared to its predecessor LeNet. AlexNet represented a significant breakthrough in the field of ML as it boasts visual recognition and classification accuracy compared to other pre-trained networks [123]. The model is composed of 8 learning layers, five convolutional layers, and three fully connected layers [124], see Figure 16.

The first convolutional layer filters an input image of (224x244x3) with 96 nuclei of size(11x11x13) with a phase of 4 pixels, the second convolutional layer has as input the first convolutional layer and filters it with 256 nuclei (5x5x48), the third convolutional layer has 384 cores (3x3x256) connected to the normalized outputs of the second convolutional layer, the fourth convolution layer has 384 cores (3x3x192) and finally the fifth convolution layer has 256 cores (3x3x192)(Adam, Mohd & Younis, 2021). The fully connected layers have 4096 neurons [124].

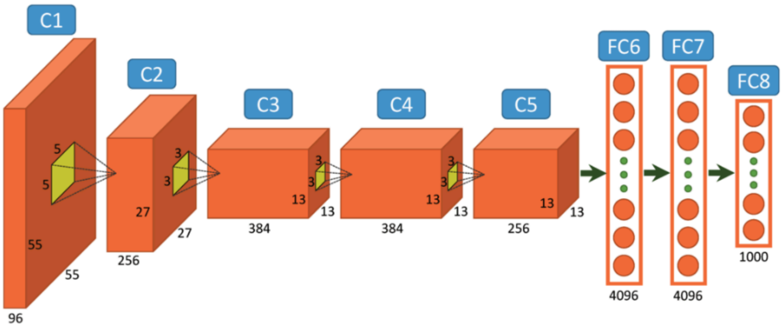


Figure 16. *Structure of Alex Net [125].*

2.7. OPTIMIZERS AND TECHNIQUES TO AVOID OVERFITTING

2.7.1. Types of optimizers

The different optimizers involve a calculation depending on their algorithm. For our project, we implemented the Stochastic Gradient Descent (SGD) as our optimizer for neural network training because it is simple and effective, shown different optimizers in Table 7.

Table 7. *Types of Optimizers*

Type of optimizers	**Optimizers**	**Description**
Adaptive learning rate methods	**Adam (Adaptive estimation of momentum) [126-127]**	Converges quickly and stores the average of previous gradients.
	Adagraf (Adaptive gradient) [126, 127]	Upgrades the walls individually as required by the importance of the upgrades.

Gradient Descent	**Batch Gradient Descent [126]**	It is used to minimize the cost function and achieve the optimal weight matrix and bias.
	Stochastic Gradient Descent [126]	It requires less calculation since it divides the data in half and updates the weights of the first half to obtain weights for the second half.
Moment-based optimizers	**Momentum [126]**	The momentum accelerates the direction of the position of the weights using velocity instead of gradient.
	Nesterov Accelerated Gradient(NAG) [126]	Slight variation from the normal gradient descent can accelerate the training process significantly.
Mini Batch Optimization [126]		It is recommended for large networks that process redundant data.

2.7.2. Techniques to avoid overfitting

Several techniques exist to avoid overfitting a neural network mode, including L1 and L2 regularization, dropout, data augmentation, cross-validation, batch normalization, and early stopping, see Table 8. For the training of our mode, we applied data augmentation, dropout, and frequency validation.

Table 8. *Methods to avoid overfitting*

Technique	**Description**
L1 Regularization [128]	Using Lasso regression, we use the cab distance, which is the sum of the absolute values of all the weights as the penalty term. Some features are removed from our model, and only the most valuable features are retained.
L2 Regularization [128]	It uses ridge regression theory, which makes networks prefer to learn features with small weights

Dropout [128]	It is an efficient learning combined with an ingenious approximate inference scheme that was remarkably accurate in the case of rectified linear networks.
Data augmentation [129]	It focuses on generating more training data from existing training samples. The goal is that the model never sees the same image twice; the image changes by rotating, flipping, or shifting it vertically or horizontally.
Cross Validation[130]	It is one of the most widely used data resampling methods for model selection and evaluation. It is used to fit hyperparameters of statistical and machine learning models.
Batch Normalization [131]	Normalizes the outputs of each layer, stabilizing and accelerating training.
Early Stopping [128]	The accuracy of the data is calculated as the network is trained, and training is stopped when the accuracy stops improving.

2.8. SOFTWARE

2.8.1. Python

Guido Van Rossum developed Python in late 1989 to be released in early 1991 [132]. Python is a versatile programming language that includes object-oriented programming, structured programming, functional programming, and aspect-oriented programming [133]. In addition, Python is distinguished by its accessibility and is extremely easy to learn and use, making it the preferred language for beginners [134].

Anaconda, the Python ecosystem, is the complete and prepackaged version of Python that includes a wide range of libraries and environments, such as Jupiterr, SymPy, pandas, SciPy, Numpy, etc. [135], shown in Figure 17.

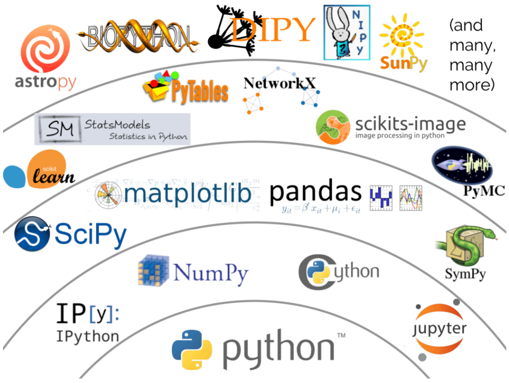


Figure 17. *Python ecosystem [136].*

Python Robot Operating System (ROS)

The Python robot operating system (ROS) was released in 2007 [137]. It is a platform intended for the development of robotic software and is notable for evaluating algorithms designed for various robotic tasks, such as robot navigation [138]. ROS supports an open-source framework for developing packages to interact with real sensors, actuators, and robots, among others [139].

Artificial Vision in python

In the field of machine vision or CV, Python is one of the most complete languages, capable of encompassing several pre-established algorithms and libraries that only need to be imported in order to use them. The main task is to give a machine the ability to detect patterns and “see.” This is achieved by combining software (libraries and algorithms) and hardware (cameras and sensors), starting with a system capable of capturing images of the environment and then performing IP to extract essential features and obtain the expected result [140].

Python is a language with specialized features for these processes, such as face detection, feature extraction, and color detection. Asaad (2023) points out several libraries capable of performing CV and IP, such as Numpy, Matplotlib, OpenCV, Scipy, PIL (Python Image Library), Simple ITK, Mahotas, Scikit-image, TensorFlow, and Keras [141]. However, TensorFlow and Keras are used more in the field of neural networks.

These libraries process images through the system of pixels contained in an image. The scale of these pixels can vary depending on the color grade, either RGB, CMYK, or B&W, and will depend on the value of the data that each pixel possesses [142]. The best-known Python library focused on IP is OpenCV, which was created in C++. Because it is open access, it can be applied in various fields, such as facial recognition, surveillance, game applications, and object counting, among others [143].

Neural networks in python

An artificial neural network consists of an input layer of neurons, one or more hidden layers of neurons, and a final layer of output neurons [144]. In Python, there are several ML and DL libraries, such as Fata, which simplifies the training of neural networks quickly and accurately; Lasagne, which is used to build and input neural networks; and Elepha, where distributed DL models can be executed [145].

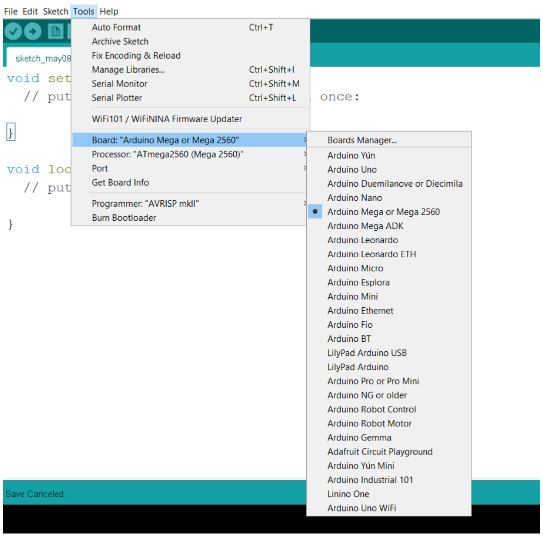
Artificial neural networks have applications in many fields, such as medicine, where they detect diseases through pattern detection and synthetic vision. They can also be used to detect and classify objects or characters. Due to their accuracy in predicting responses, artificial neural networks have recently been introduced to models of Tesla brand cars [146].

2.8.2. Arduino IDE

Arduino offers not only boards(Arduino Uno, Arduino Shield, Arduino Mega, etc.) but also software that includes a development environment (IDE) [147]. The Arduino IDE environment is shown in Figure 18.

The integrated development environment or IDE of Arduino is a multiplatform software of free license that is used to write and load programs in Arduino boards and also in those that are compatible; in addition, it allows debugging, editing, and saving programs or sketches in development boards in a fast and simple way [148].

On the other hand, Arduino IDE is an environment focused on the control and programming of boards that have robotics as an application. In addition, due to its limited environment, this software does not have applications focused on AV and artificial neural networks. To compensate for its limited environment, Arduino works with other software such as Python, which, with the TensorFlow lite library, makes it possible to install and run a neural network on the Arduino board to recognize voice commands or gesture recognition [149].



```
void set
  // put                          once:

}

void loc
  // put

}
```

Save Canceled.

Figure 18. *Arduino IDE software.*

2.8.3. Lab View

LabView, whose acronym is "virtual instrument engineering laboratory" [150], is both a programming language and a graphical environment with which applications can be developed in an agile and accessible way [151].

LabVIEW, unlike other software, is based on add-ons or Toolkits, which are essential to implement various actions in the software. However, their use is limited to the type of license with which this application works. It is mainly based on a graphical environment to configure the processes to be carried out at the industrial and research level. Its applications are varied but can also be limited to IP and ML. The main disadvantage is that the user must configure all the necessary parameters in its interface.

According to Ortiz (2020), LabVIEW relies on several main features to implement a machine vision system, such as the Input matrix, the Reordered matrix, and the distance between pixels [152]. In addition, the differential matrix is calculated using the AVA algorithm, errors are removed, and corrections are applied to the model. Another application in the field of CV is IP using external tools [153].

On the other hand, image acquisition can be performed by using the webcam and performing simple processing, such as applying noise and implementing various filters to improve clarity [154]. Finally, in the field of AI, it is essential to use the LabVIEW Analytics and ML Toolkit to perform efficient predictive analytics and optimally implement ML in order to extract and analyze patterns from a large database [155]. Such is the case of Ramirez (2007) and Dirik (n.d.), who implemented basic and complex systems, respectively, for pattern detection and data processing focused on anomaly detection [156-157].

2.8.4.MATLAB

It is programming software focused on numerical computation and data visualization in various fields of physics, mathematics, and computational engineering [158]. Unlike Python, this is paid software that limits its use. However, it allows the implementation of various applications focused on different areas, from aerospace engineering to ML processes.

Its use is very wide. Its friendly language is oriented to matrices, which makes it easy to learn to perform mathematical operations efficiently. It also has several integrated libraries that streamline and solve complex problems with exceptional efficiency. Pre-designed applications are indispensable for adjusting parameters and adding them to lines of code quickly and easily. Image Region Analyzer and Deep Network Designer are examples of applications used in this project to optimize the code.

- **Machine learning and deep learning ToolBox**

Several applications in Matlab perform ML and DL. Among them are the deep network Quantizer, experiment manager, neural net clustering, classification learner, cluster data, and reduced dimensionality [159]. Next, we will describe some of the toolboxes used in Table 9.

Table 9. *MLT & DLT.*

Kind of Machine learning Toolbox	Description
Statistics and Machine Learning Toolbox	Provides tools and applications for describing, analyzing, and modeling data, including functions for descriptive statistics, data visualization and clustering, fitting probability distributions, working with random numbers for Monte Carlo simulations, performing hypothesis testing, regression algorithms, etc. [160].

Text Analytics Toolbox	This toolbox works with text data by providing the user with algorithms and visualizations for preprocessing and analyzing text data. A common example of the use of this toolbox is in sentiment analysis, predictive maintenance, and topic modeling. These features can be combined with other sources to create ML models using textual, numerical, and different types of data [161].
Deep Network Designer	On the other hand, Matlab also offers a tool capable of creating and using pre-trained models of CNNs, thus facilitating their design, training, and testing [162]. It has a user-friendly graphical interface and an intuitive design, see Figure 19-20.

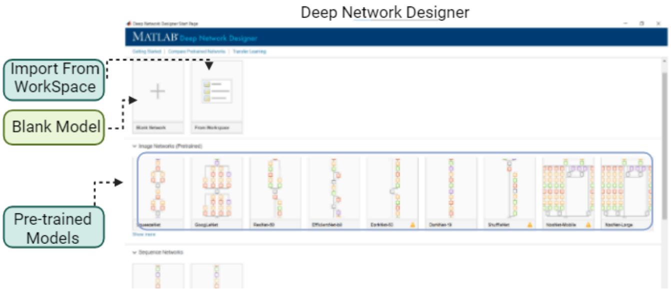


Figure 19. *Deep Network Designer Start Page.*

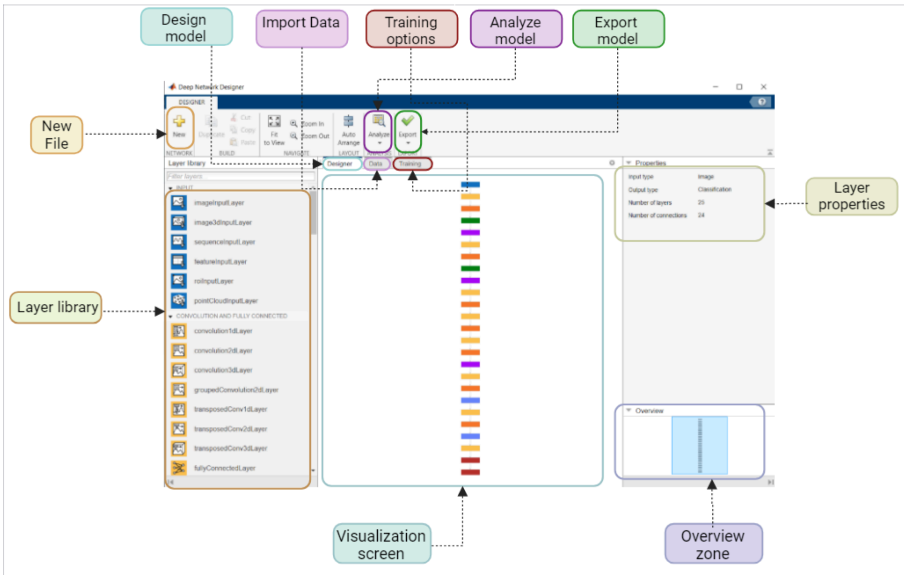


Figure 20. *Deep Network Designer Interface.*

- ## Image Processing and computer vision

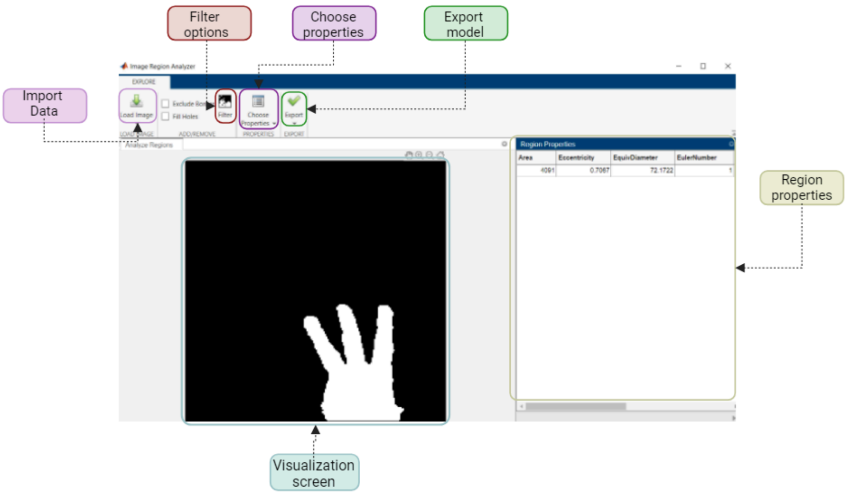
IP and CV, located in the applications section of Matlab, are tools that allow you to perform end-to-end processing, from acquisition to data preprocessing, enhancement, and analysis, along with deployment in an integrated vision system. For example, it can be used for image, video, point cloud, lidar, and hyperspectral data where we can interactively visualize, explore, label, and process data using applications, perform semantics, object detection, image-to-image classification, and translation using DL, etc. [163].

In addition, we have several apps that are included in IP and CV applications, see Table 10.

Table 10. *IP & CV.*

Image processing apps from Matlab	Description
Image Region Analyzer	It is an app that provides Matlab with the objective of characterizing binary images from the various regions of interest for the extraction of specific information in particular areas of the image [164]. It has several parameters to analyze, such as Area, Convex Area, Eccentricity, EquivDiamete,

	and Orientation, which are the parameters used in part of the project, shown in Figure 21.
DICOM Browser	DICOM Browser allows you to select and save a specific series within the working environment. It works with files in DICOM format, organizes them by study and series, and stores the data as a volume with separate variables for color mapping and spatial details.
Image Segmenter	The Image Segmenter tool facilitates the creation of segmentation masks using various automatic, semi-automatic, and manual methods. This tool uses automatic algorithms for diffusion filling and offers tools to enhance masks using interactive techniques such as active contours [165].
Volume Segmenter	Volume Segmenter is a tool for creating and refining semantic or binary segmentation masks for an RGB or 3D grayscale image to segment an object drawn in the region of interest [166-167].



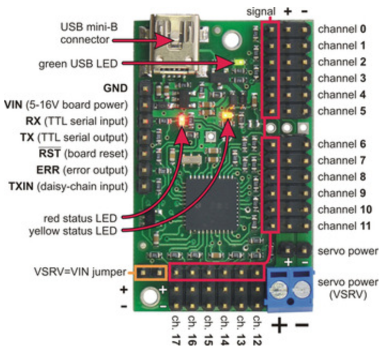
Area	Eccentricity	EquivDiameter	EulerNumber
4091	0.7067	72.1722	1

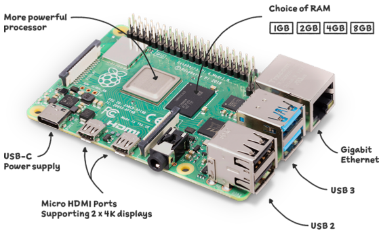
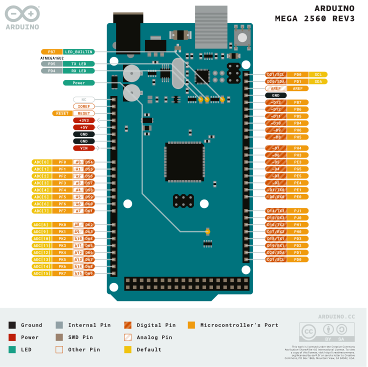
Figure 21. *Image Region Analyzer interface.*

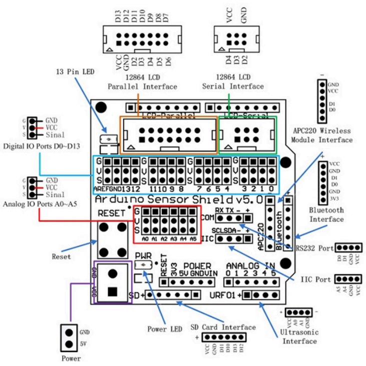
2.8.5. Microcontrollers

There are different types of servo motor controllers that help with the movement of servo motors. We have Pololu mini masters of 12 and 24 channels, Raspberry, Arduino one, Arduinoo mega 256, and Arduinoo mega shield, see Table 11.

Table 11. *Multiple controllers for Servo Motors*

Kind of Servo Motor Controllers	Image	Description
Pololu	 USB mini-B connector green USB LED GND VIN (5-16V board power) RX (TTL serial input) TX (TTL serial output) RST (board reset) ERR (error output) TXIN (daisy-chain input) red status LED yellow status LED VSRV=VIN jumper + - signal + - channel 0 channel 1 channel 2 channel 3 channel 4 channel 5 channel 6 channel 7 channel 8 channel 9 channel 10 channel 11 servo power servo power (VSRV) ch. 17 ch. 16 ch. 15 ch. 14 ch. 13 ch. 12 + - **Figure 22.** *Mini Maestro 18 channel Pololu USB servo controller [165].*	The Pololu mini master is a 12, 1, 8, and 24-channel servo controller, see Figure 22, which connects to the computer via a USB cable. The external power supply comes from 5-16 V [168].

Raspberry Pi	 More powerful processor Choice of RAM 1GB 2GB 4GB 8GB Gigabit Ethernet USB 3 USB 2 USB-C Power supply Micro HDMI Ports Supporting 2 x 4K displays **Figure 23.** *Part of Raspberry Pi [168].*	Raspberry Pi is a small open-source microcontroller. It consists of a Broadcom BCM2835 chip, including ARM1176JZF-S, GPU video core IV, etc. (Figure 23). It runs using the Raspbian operating system and can be programmed using GNU Octave version 3.6.4 and Python 2.7, which are open-source [169].
Arduino Mega 2560	 ARDUINO ARDUINO MEGA 2560 REV3 Ground Power LED Internal Pin SWD Pin Other Pin Digital Pin Analog Pin Default Microcontroller's Port ARDUINO.CC **Figure 24.** *Arduino Mega 2560 connection diagram [171].*	The most representative microcontroller, which is an improvement over the original Arduino UNO, consists of a microcontroller board that includes 54 digital pins, 16 analog inputs, 4 hardware serial ports, a 16MHz oscillator crystal, USB port, power connector, ICSP header, and reset button, see Figure 24. For its operation, it is only necessary to connect an AC to a DC adapter or even a battery [170].

Arduino Shield V5	 13 Pin LED 12864 LCD Parallel Interface 12864 LCD Serial Interface APC220 Wireless Module Interface Bluetooth Interface RS232 Port IIC Port Ultrasonic Interface SD Card Interface Power LED Power Reset Digital IO Ports D0~D13 Analog IO Ports A0~A5 GND VCC Sinal Arduino Sensor Shield v5.0 RESET PWR COM IIC SD URF01 POWER ANALOG IN APC220 Bluetooth **Figure 25.** *Arduino Sensor Shield v5.0 diagram [173].*	This expansion board includes a digital and analog interface, IIC interface, 32-road digital address controller interface, Bluetooth module communication interface, SD card communication interface, APC220 RF module communication interface, RB URF v1.1 ultrasonic transducer interface, 12864 LCD interface, and parallel interface [172], see Figure 25.

2.9. RELATED WORKS FOR "DESIGN AND CONSTRUCT A ROBOTIC ARM FOR SIGN LANGUAGE INTERPRETATION WITH NEURAL NETWORK"

It's important to understand previous models to ensure our project is successful. The research of Liam, K. (2019) used a CNN that they created to recognize hand modeling. First, it's important to select the correct Database; they use 31492 annotated hands from the dataset to train the right-hand model and left-hand model. Also, the images need to be processed; they use a simple background and apply Otzu's model to make an automatic binarization of the images, and two different databases are used to validate it CNN [174].

Another important study was the project of Nihal, R. (2021), who focused his research on the creation of a humanoid robot for the correct interpretation of sign language. The use of a flexible and lightweight material was essential for the creation of his model, and he chose thermoplastic polyurethane (TPU) as the raw material. The robotic arm operates with 15 degrees of freedom and five servomotors, which will allow for controlled movement, as well as its use with an Arduino UNO microcontroller. Nihal emphasizes that the model of his robot costs approximately $800 to prototype. For the neural network model, he used DenseNet 201, DarkNet 53, and ResNet 50, as well as a database of Bangla sign language (BdSL) and medical signs interpretation for training. Finally, the results obtained for sign language recognition were 87.5%, and for BdSL using DenseNet-201, they were 98.19% [175].

While Nihal used models such as DenseNet, DarkNet, and ResNet to realize his detector, Chavan, S. (2021) used neural network models based on LeNet-5, Vgg16, and MobileNet v2 for his study in detecting ASL. RGB to Gray conversion was applied as image processing to reduce training time. In addition, the Otsu method was applied to perform an automatic binarization. On the other hand, Canny's method was also used, which is divided into 5 stages: noise reduction, gradient calculation, non-maximum suppression, double thresholding and hysteresis thresholding, Mirroring, Cropping, Rotating, shearing, local warping, and Color shifting as data augmentation parameters were applied to train the neural network.

The database used is from Turkey Ankara Ayrancı Anadolu High School, which comprises approximately 205 images per class. The dataset consists of hand gestures for 0 to 9 digits in RGB format with a resolution of 100x100 px. Finally, Chavan's model obtained an accuracy of 91.37% and a validation of 86.30% [176].

For the study of Rastgoo, R. (2020), a multi-skeleton model of the hand was implemented for sign language recognition. Both 2D and 3D convolutional neural analytic networks were used together with an LSTM model to capture spatial and temporal features, the main one being the ResNet50 model. Hand skeleton processing is performed by projecting the 3D points onto surface images in 3 different views. For the database, a video was used as input data, RKS-PERSIANSIGN, characterized by 10,000 videos of 100 words in Persian sign language. For training, 50 frames per video and an AdaGrad optimization algorithm were used to train the neural networks. Rastgoo used 21 3D estimated hand key points from RGB inputs and scaled these inputs to the interval [0:95, 0:95, 0:95, 0:95]. Finally, a hand gesture recognition accuracy of 91.12% was obtained [177].

On the other hand, Zhi, D. (2018) implemented the multiclass-SVM classifier and N-Dimensional DTW (ND-DTW) classifier for static posture recognition and dynamic gesture recognition. The use of a robotic arm was fundamental to test the detection model, the model used was the ADA Robotic Arm. The main dataset for training consisted of ten digits based on ASL. Factors such as the position of the palm and fingers, along with their speed and direction, must be taken into account when determining the detector. For the implementation of the recognition model, the SVM library was added to provide greater accuracy to the model. For the model to be used in real-time, a linear kernel was selected to improve the training and classification speed and to reduce the resources needed for processing. Finally, the experiment showed that the proposed method achieved an accuracy of 98.25% compared to other studies with an accuracy of 95.5% [178].

Previous studies by Cao Dong, M. (2015) laid the groundwork for low-cost detection systems such as Microsoft's Kinect for detection. The aim of this research was to demonstrate that a low-cost method with good results can be used to detect ASL. He used several configurations to obtain the position segments: a dimension-adaptive mean-shift mode-seeking algorithm. In addition, the assemblies of the 13 key angles of the hand skeleton were used as the features to describe hand gestures. Random Forest was used because previous studies indicated a high probability rate for segmenting body parts using depth contrast features.

To improve this algorithm, a glove with different colors was implemented to facilitate the detection of each section of the hand. In addition, an increase in the color saturation of the images was performed in order to create an efficient dataset consisting of 3,000 images generated using the color glove, of which 2,000 images were randomly picked for training, and the rest were used for validation, with a resolution of 256x256. This method proved to be novel and efficient, reaching an accuracy of 90% compared to previous models that ranged from 59% accuracy to 87% accuracy [179].

Barbhuiya, A. (2020) conducted his research with AlexNet and VGG16 CNN models for ASL recognition applied on a robotic arm. The implementation of the SVM classifier was fundamental to improving the processing performance by means of support vectors. For his DataSet, he started from the Barczak Dataset, which included 70 images in each class and had a total of 36 classes of sign language; 26 classes were of gestures, and ten were of numbers. In addition, data augmentation (Image translation, image Shedding) was applied to provide more variety in the training. 70% of the images were taken for the training configuration and 30% for testing the models. Finally, the results show that the model (AlexNet and SVM classifier) had an accuracy of 70%, which is better but not enough than other models [180].

The use of a robotic hand to perform sign language translation activities was also the brainchild of Johnson, S. (2021) and his team, who implemented TATUM, designed to facilitate communication with deaf and deafblind people tactilely and visually. The design called for 15 degrees of actuation to ensure that the robotic hand could execute the full ASL alphabet. The hand was constructed from materials such as thermoplastic polyurethane (TPU) and polylactic acid (PLA), making it affordable and durable. The hand's movement is controlled through a cloud-based service called Interpres, which translates text into servo motor instructions, allowing the hand to emulate ASL gestures. The system was validated in a study with deaf and deafblind participants, with recognition rates of 94.7% for visual and 71.7% for tactile recognition. The results indicate that the potential of TATUM will improve communication between the deaf and deafblind communities [181].

In the research work of Mazhar, O. (2019), a robotic arm called Kuka LWR 4+ was created, where Microsoft Kinect V2 sensors were used to capture visual information. The robot consisted of a mobile base called Neobotix MP700. For the dataset, an OpenSign dataset was used, which contains 20950 images in which there are ten gestures, among which we have letters, numbers, and the gesture of none (there is no interaction). In addition, we used the pre-trained network of Inception V3 with the ImageNet dataset, which is coupled with the objectives of this research. On the other hand, real-time data augmentation was applied with Keras running transformations such as displacement, zoom, rotation, and translation. Finally, this model achieved an accuracy of 98.9%. In conclusion, this research work is working collaboratively in real-time with the human, where the user performs hand gestures, and the robot performs others [182].

On the other hand, in the work of Meghdari, A. (2018), a robot with arms, hands, and a head called RASA, which has a total of 29 degrees of freedom, was created. The robot parts were made of polyamide PA2200 and aluminum AL7075 for the movement of the joints; the Dynamixel Mx64 and Mx28 servo motor were used for the movement of the arm joints. The total cost for the development of the RASA robot was less than $10k. This robot was designed to simulate the movement of sign language, which is 70 signs of sign languages, including 60 words of the Persian language and 10 of the Baghcheban phonetic alphabet. Finally, the average recognition rate was 77% in the first test and 100% in the second test. In conclusion, the researchers developed a robot called RASA designed to teach sign language to hearing-impaired children [183].

In the research of Bulgarelli, A. (2016), improvements were made to a model of an anthropomorphic robotic hand, which was designed to interpret sign language consisting of a robotic hand that has 8DoF in the fingers and a 3DoF spherical wrist. The robotic hand was made by 3D printing with PLA, ABS, nylon, and rubber for the different parts of the robotic hand. It also used analog servo motors controlled by Arduino One and ROS (Robot Operating System). The cost of the creation of this robotic arm was around 280 euros. The database consists of several gestures using the manual alphabet of the Italian sign language. It was also developed using C/C++ software through ROS, where the movements of the robot were programmed to reproduce the gestures of the sign language. Finally, the system achieved a 90% success rate in the reproduction of visually evaluated gestures. In conclusion, the project works with a pre-designed arm model in which improvements are made by adding additional degrees of freedom, which will help with the interpretation of sign language [184].

In the research of Kenshimov, C. (2021), a 50-DoF humanoid robot with various parts such as hands, wrist, head, waist, and mobile base was created. This robot was designed using 3D printing and a movable iron base with a combination of 27 motors, 25 servos, and Arduino Mega 2560 as the main controller, and is powered by a 24V lithium battery. The software used is called Python, and a pre-trained network, ResNet 18, contains 18 layers for gesture recognition. The database consists of 42 gestures of the Kazakh alphabet, 8000 gestures of which 5000 were used for training and 3000 for evaluation. Finally, the average recognition accuracy of the system was over 98%. In conclusion, a hand gesture recognition system is developed for a robot that can interpret and reproduce Kazakh sign language [185].

In Verma's (2017) research, a robotic hand was made, which is responsible for generating gestures according to ASL. The robotic arm was built using Raspberry Pi 4, PCX9685 servo controller, servo motors, and an external microphone to receive voice inputs. The focus of the research is based on an algorithm created using Python software to perform speech recognition, which generates sign language. The database is composed of the 26 letters of the ASL alphabet. Finally, the system was tested by ten people (5 men and 5 women) and achieved an accuracy of 90% [186].

In the research of Al-Khulaidi, R. (2018), a robotic arm called SignBot was created, which has two robotic hands to perform Malay sign language. They worked using Microsoft Visual Studio with C# for speech recognition; it processes speech input signals, segments them, and converts them into numerical values. This robot is designed to perform signs of letters and numbers of the alphabet in Malay Sign Language, in addition to some gestures. It is composed of Arduino Uno and 12 servo motors, which are responsible for the movement of the hand and wrist. Finally, the voice recognition system achieved 93% accuracy in recognizing gestures [187].

In Islam's research (2017), Matlab R1016a software was used for image processing. The image preprocessing consists of image resizing to 260x260 pixels, conversion from RGB to binary using the Otsu method, cropping the image to focus on the hand from wrist to fingers, and a filter was used to remove noise and preserve edges. The database consists of 37 signs, which include letters and numbers of ASL, so we have 1850 images (50 samples for each sign). The neural network consists of 3 layers: an input layer, a hidden layer of 20 neurons, and an output layer. Finally, we have an accuracy of 99.5%, and in a real-time environment, the system achieved a recognition rate of 94.32% [188].

According to the research, Luo, R. (2012) created a robot designed for sign language detection. The robot is composed of a head and body with an approximate height of 136cm with a touch screen, CCD sensors, LRF (Laser Range Finder), ultrasound sensor, sensory circuits, etc. A PC with an Intel T9400 CPU and 2G RAM was used. For the hand gesture recognition process, the first step is to detect the hand, so the skin color is an important feature. Also, the local binary pattern was combined with a feature selected for training and a hand skeleton method, which comes from the fingertips and the center of the palm, which is drawn by a yellow line. A Bosphorus Hand Database was used and combined with created images. Finally, the recognition rate is over 85%, and some signs are around 92% [189].

Table 12. *Main Characteristics of each author*

#	Author	Network Used	Language	Preprocessing	Robotic Arm	Accuracy	Dataset	Software Used
1	Lim et al. (2019) [174].	CNN (HEI)	ASL	Simple background, Otsu binarization	No	89.33%	RWTH BOSTON 50, ASLLVD	N/A
2	Nihal et al. (2021) [175]	CNN, RNN (DenseNet, DarkNet, ResNet)	BdSL	Uniform background, brightness	Yes (15 DOF)	87.50%	12,581 images, 38 classes	MATLAB
3	Chavan et al. (2021) [176].	LeNet-5, Vgg 16, MobileNet	ASL	Otsu thresholding, Canny edge detection	No	91.37%	ASL (0-9 digits)	Python
4	Rastgoo et al. (2020) [177]	3D ResNet 50 + LSTM	Persian	3D hand skeleton	No	99.80%	RKS-PERSIAN SIGN (10,000 videos)	N/A
5	Zhi et al. (2018) [178].	SVM, ND-DTW	ASL	Leap Motion controller	Yes	98.25%	ASL digits	N/A

6	Dong et al. (2015) [179].	Random Forest	ASL	Color-based segmentation	No	90%	3,000 images with color glove	N/A
7	Barbhuiya et al. (2020) [180].	AlexNet, VGG16	ASL	Resizing, augmentation	No	70%	36 characters from 5 people	N/A
8	Johnson et al. (2021) [181].	Cloud CNN	ASL	None	Yes (15 DOF)	94.70%	N/A	C++
9	Mazhar et al. (2019) [182].	Inception V3 (adapted)	ASL	Histogram equalization, Gaussian noise	Yes	98.90%	20,950 images	N/A
10	Meghdari et al. (2018) [183].	N/A	PSL	N/A	Yes (32 DOF)	N/A	N/A	N/A
11	Bulgarelli et al. (2016) [184].	N/A	Italian	N/A	Yes	90%	N/A	N/A
12	Kenshimov et al. (2021) [185].	ResNet-18	N/A	N/A	Yes	98%	8,000 samples from 4 people	N/A

13	Verma et al. (2017) [186].	N/A	ASL	N/A	Yes (7 DOF)	90%	26 ASL alphabets	Python
14	Al-Khulaidi et al. (2018) [187].	N/A	Malay	Speech signal converted to numeric	Yes	93%	80 tested phrases	C#, Microsoft Visual Studio
15	Islam et al. (2017) [188].	N/A	ASL	Otzu, median filter, rotation	No	94.32%	1,850 images, 37 signs	MATLAB R1016a
16	Luo, R. C., & Wu, Y.-C. (2012) [189].	N/A	N/A	Both method support vector machine (SVM) and hand skeleton recognizer (HSR)	Yes	92%	Bosphorus Hand Database	N/A

2.10. MAIN CHALLENGES

2.10.1. Hardware Limitations.

The biggest limitation we obtained when making the robotic arm was printing the 3D mode. The printer had certain failures when printing because the model had very small holes, which the printer filled with material. For this purpose, the holes were highlighted with a small diameter metal tube, removing the excess material. This point is fundamental so that the tensioning cables do not present inconveniences when performing the function of flexion and extension of the joints.

On the other hand, it is not included as a limiting factor. Still, the assembly of the tensile wires is a time-consuming process that includes implementing three flexible wires and a rope wire that will provide the flexion and extension effect. Three flexible threads were chosen because using only one lacked strength and would break, and when using two, if one broke, we had the same problem. Therefore, using three increased the movement force and prevented the threads from breaking easily.

2.10.2. Software Limitations.

The main limitations occurred in the neural networks phase, both in the configuration of the parameters for training and in the detection stage.

For the initial stage, we used a computer with an Intel Core I7-1065G7 processor at 1.30 GHz and an Nvidia Mx230 graphics card. It took approximately one hour and 30 minutes to train the AlexNet and GoogleNet models for each of the neural networks. When the visualization of neural network detection was implemented, it took a long time to process the images and did not give us a smooth visualization.

To improve performance and take less time in training, a laptop with an AMD Ryzen 5 5000 series processor and an RTX 3050 Ti graphics card was used. An improvement was obtained when training the models from 1 hour and 30 minutes to 20 minutes, in addition to fluidity in the detection of gestures, which is appreciated.

CHAPTER III

METHODOLOGY

3.1. METHODOLOGY OF WORK

IBM's Rational Unified Process (RUP) methodology will be used for this process. This methodology, together with the Unified Modeling Language (UML), is the most widely used model for analyzing, implementing, and documenting oriented systems [190].

In Figure 26, we can see the life of the unified process. We can see the five workflows: requirements, analysis, design, implementation, and testing. Each takes place in four phases: initiation, elaboration, construction, and transition [191].

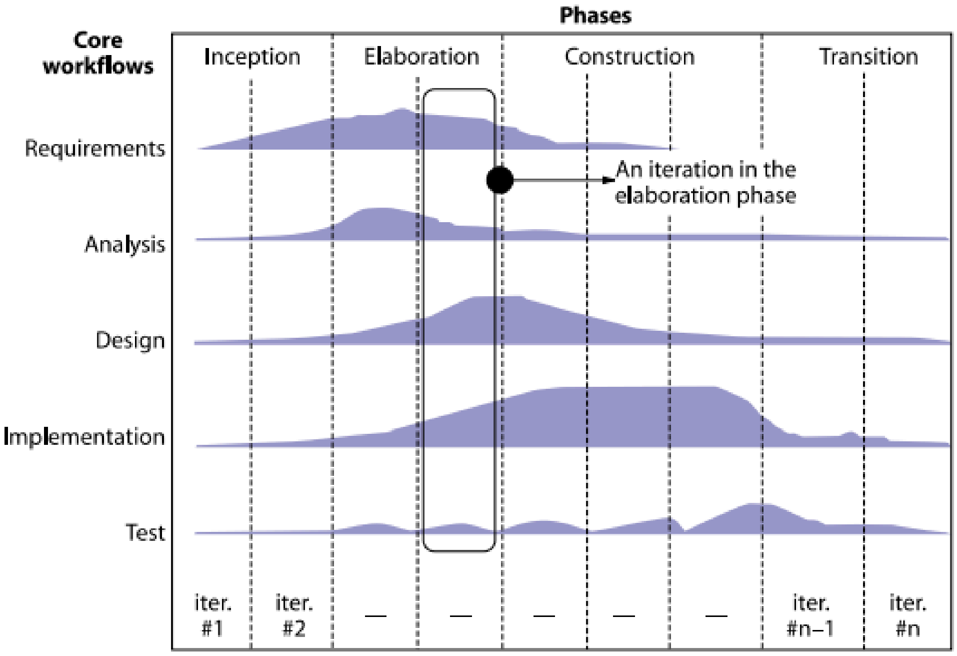


Figure 26. *Main phases and workflows of the Unified process [191].*

3.2. FUNDAMENTAL WORKFLOWS

3.2.1. Requirements

We will analyze the optimal materials, the most appropriate and low-cost ones, for the development and assembly of the curricular integration word.

3.2.2. Analysis

According to the corresponding objectives, we proceeded to determine the functional materials for implementing the curricular integration work, which are specified in the upper item.

3.2.3. Design

We will detail the architecture of the CNN used to interpret sign language and its correct movement from a robotic card.

Implementation: The robotic arm will be assembled, and then the code for interpreting ASL will be programmed. Then, the database will be created with its images by means of movements and positions of the sign language emulated by the robotic arm. Finally, the connection between the AV and the mechanical movement of the arm will be made from a microcontroller and servo motors.

3.2.4. Testing

After the subsequent stages, we will proceed to perform several tests, which will help us verify the prototype's correct operation. In addition, several factors will be implemented as key points to evaluate the model and ensure its performance.

3.3. LIFE CYCLE PHASES

3.3.1. Start

During this phase, a final prototype description is developed from an idea proposed as needed. This need focuses on learning and implementing sign language in everyday life due to the increasing number of people born with or developing some degree of hearing impairment. As a development tool, periodic analysis will be conducted to obtain various models of robotic arms and detail key points with which the system is planned to be implemented, thus achieving a better understanding of the scope of the project. The final product of this phase will be the base model of the arm with which the final prototype is intended to be made upon completion of the project.

3.3.2. Elaboration

During this stage, the project begins to take shape as most of the pending requirements are gathered. In addition, a solid foundation is laid that will guide the project through the different stages.

3.3.3. Construction

In this stage, the system acquires a more detailed perspective since models, designs, their programming, and the tests that must be passed for the project's validation are included. Furthermore, once this phase is completed, we will have the functional prototype with most of the implemented features. However, the prototype may still present flaws that will need to be adjusted in later stages.

3.3.4. Transition

In this section, the prototype will undergo rigorous tests to verify its correct operation. If failures and inconsistencies are found, the necessary adjustments will be made so that the prototype works optimally. Finally, after making the necessary adjustments, the prototype will be released as its final version.

In this section, a description is developed according to the work structure based on the established activities.

3.4. SOFTWARE DEVELOPMENT

In this part of the curricular integration work, the methodology for designing the programs and algorithms used to increase the database, move the arm, train the neural network, and develop AV is proposed.

3.4.1. Database

The code for capturing images and saving them in defined folders was developed with the objective of creating our own database for training neural networks.

For this process, we took into account the size of the pixels that the image has to have, in this case, 200 x 200 pixels. On the other hand, the number of images is also important for correct training; this amount is defined as 600 images for each class. The classes are based on the number of movements or letters that can simulate the robotic arm to emulate the sign language. Therefore, we will have a total of 16 classes for the various letters that will be interpreted, in addition to adding three more classes that are: Nothing, which is when the image of an arm is not found in the camera, Table 13, and Base, which is the representation of a fully extended robotic hand, and Error, which is the representation of a non-hand image

Table 13. *Characteristics of each class of Database.*

Class	Letter	Quantity	Size (px)	Color mode	Weight (Kb) each image
1	A	600	200 x 200	RGB	4.00
2	B	600	200 x 200	RGB	4.00
3	C	600	200 x 200	RGB	4.00
4	D	600	200 x 200	RGB	4.00
5	E	600	200 x 200	RGB	4.00
6	F	600	200 x 200	RGB	4.00
7	I	600	200 x 200	RGB	4.00
8	L	600	200 x 200	RGB	4.00
9	O	600	200 x 200	RGB	4.00
10	S	600	200 x 200	RGB	4.00
11	U	600	200 x 200	RGB	4.00
12	W	600	200 x 200	RGB	4.00
13	Y	600	200 x 200	RGB	4.00
14	Error	600	200 x 200	RGB	4.00
15	Nothing	600	200 x 200	RGB	4.00
16	Base	600	200 x 200	RGB	4.00

Once we defined the parameters that we would use for the database, we implemented a code to speed up this process in Matlab. The key points are the resizing of the image, the creation of a storage folder, the capture of the 600 images, and the saving of these images in each of the folders created for each class. It is important to note that the saving directory for each class must be changed manually, as shown in Figure 27.

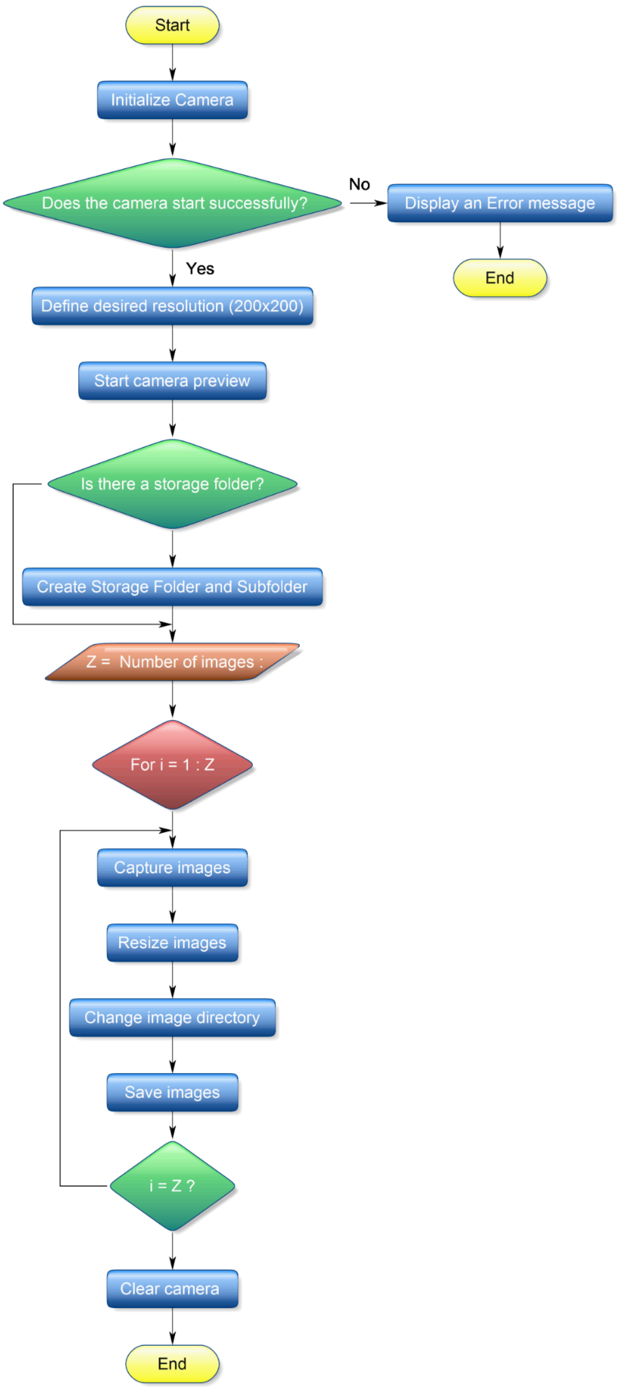


Figure 27. *Flowchart to make our database.*

3.4.2. Convolutional Neural Network

The neural network design will be carried out using Matlab software and the Deep Network Analyzer, which allows the use of transfer learning in different neural network models through a friendly and intuitive interface. For this section, the AlexNet and GoogleNet models will be used due to their processing capabilities. To perform the correct training of the different neural networks, we apply transfer learning to use these models with our database and avoid processing errors.

In this section, it is important to review the theory about the architecture of each one of them. In fact, similar changes have been made, but they differ in small details in the use of each one of them. Both have a fully connected layer at the end of their architecture, which must change its value depending on the number of classes we have. However, the AlexNet model has two other fully connected layers.

The first step in using these neural networks is creating an Image DataStore, which creates a variable with all the images inside the Matlab WorkSpace to avoid overloading the system's memory.

Then, data augmentation is performed to provide more variety in the training images. This process adds translation, resizing, and reflection to the images. Finally, the training options are configured, the most important ones being the use of the sgdm optimizer, an InitialLearnRate of 0.001, a DropOut factor of 0.3 per 2 epoch to avoid overfitting, a maximum number of epochs of 10, a MiniBatchSize of 64 to speed up the process and a validation frequency of 20.

This process can be visualized in Figure 28.

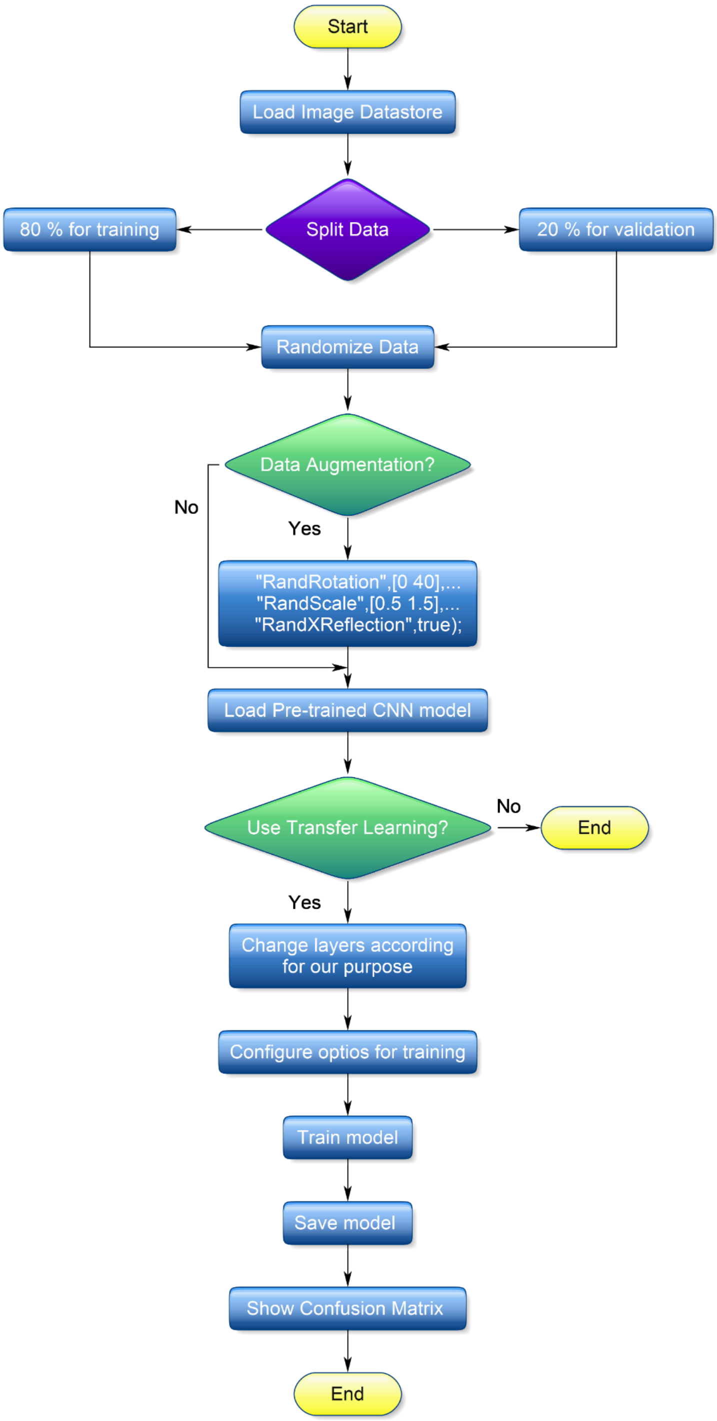


Figure 28. *A flowchart to use in transfer learning on Pre Trained Neural Networks.*

3.4.3. Arm Motion

This section is fundamental for the curricular integration project because it is the basis for the movement of the robotic arm, which emulates sign language.

The language that will be used to perform this process is Arduino ID because we have an Arduino MEGA 2560 microcontroller and servo motors to adjust the position of each phalanx of our arm.

- **Servomotor Controller**

There are several ways to control servo motors. These motors contain internal parts that allow their speed and position to be precisely regulated. According to Autsou (2024), a potentiometer is used inside these devices, and its function is to track the rotation angle or shaft speed, thus creating a closed-loop control system with feedback.

One way to control this system is through pulse width modulation (PWM), which is implemented by sending a pulse of a specific length at a specific frequency. In addition, Autsou notes that this method is essential when immediate response and precise controls are required. Arduino IDE already includes a library for controlling servo motors using PWM, and it is only necessary to configure the angle at which the 180° servo motors should be positioned [192].

First, variables must be created to define each of the fingers that we will use. These fingers must be placed at an initial position of 90 °, which serves as point 0 of our robotic arm, so it can move 90 degrees positively and 90 degrees negatively.

The "Base" movement of our arm corresponds to the extension of all the fingers. Therefore, the servo motors are set to values greater than 90 degrees.

Then, the user can determine which letter he wants the robotic arm to form by means of an input. Then, it is evaluated that the response entered by the user is valid depending on the cases that are created based on the allowed movements; it is also evaluated that the user has not entered invalid characters such as #$%! etc. Finally, it goes through a system of cases that evaluates the response. After that, the information of the requested response is sent to the servomotors, culminating in the movement of the arm.

This section is described more technically in Figure 29.

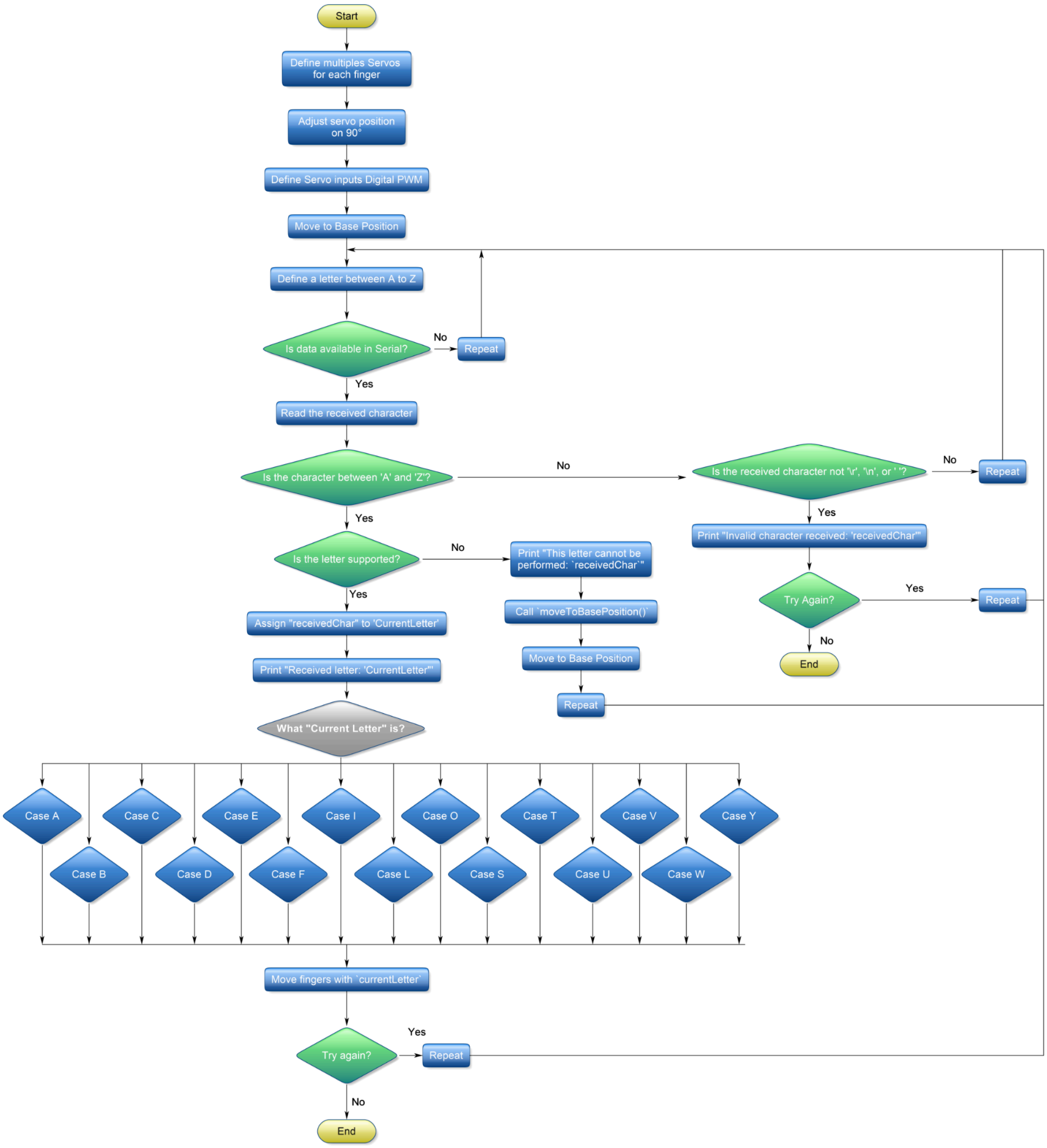

Start
Define multiples Servos for each finger
Adjust servo position on 90°
Define Servo inputs Digital PWM
Move to Base Position
Define a letter between A to Z
Is data available in Serial?
No
Repeat
Yes
Read the received character
Is the character between 'A' and 'Z'?
No
Is the received character not '\r', '\n', or ' '?
No
Repeat
Yes
Print "Invalid character received: 'receivedChar'"
Try Again?
Yes
Repeat
No
End
Yes
Is the letter supported?
No
Print "This letter cannot be performed: `receivedChar`"
Call `moveToBasePosition()`
Move to Base Position
Repeat
Yes
Assign "receivedChar" to 'CurrentLetter'
Print "Received letter: 'CurrentLetter'"
What "Current Letter" is?
Case A
Case B
Case C
Case D
Case E
Case F
Case I
Case L
Case O
Case S
Case T
Case U
Case V
Case W
Case Y
Move fingers with `currentLetter`
Try again?
Yes
Repeat
No
End

Figure 29. *Flowchart for arm movement.*

3.4.4. Artificial Vision

Another key point of the project is the use of AV to validate the movement of the robotic arm and determine that the letter is correct. This section takes several aspects of an image into account, including essential characteristics such as the channel in which the image is located (R, G, B), the background of an image, the region to be analyzed in the image, the calibration of the pixels obtained by the image, the segmentation of an image to highlight the evaluated are, a and finally the prediction on this image.

First, we evaluate the use of the webcam or an auxiliary camera available for data acquisition. If we do not have one, the code will show an error informing the users. Then, if we do have a camera available, an area defined by a rectangle is segmented where the images will be taken. This point is important because unnecessary information in each image can be eliminated efficiently.

Then we proceed to load the neural network model focused on the detection of sign language for a robotic arm, in addition to creating an error variable. This is to adjust the binarization of the image from the range calculated by graythresh, which calculates the global threshold from the grayscale image using Otsu's method (a threshold is chosen that minimizes the interclass variance of the black and white pixels). This step is important to avoid taking unnecessary pixels that do not match the pixels of the color of the human skin or the robotic arm. So, a loop is generated with which the user can visualize if there are errors in an image that is taken and, in case of errors, adjust it automatically according to the user's inputs.

A time range is defined within which a picture of the target image will be taken, and this resulting image will undergo a prediction process.

Finally, the segmented image is obtained with the resulting prediction from the neural network.

It is a rather long process that can be visualized in Figure 30.

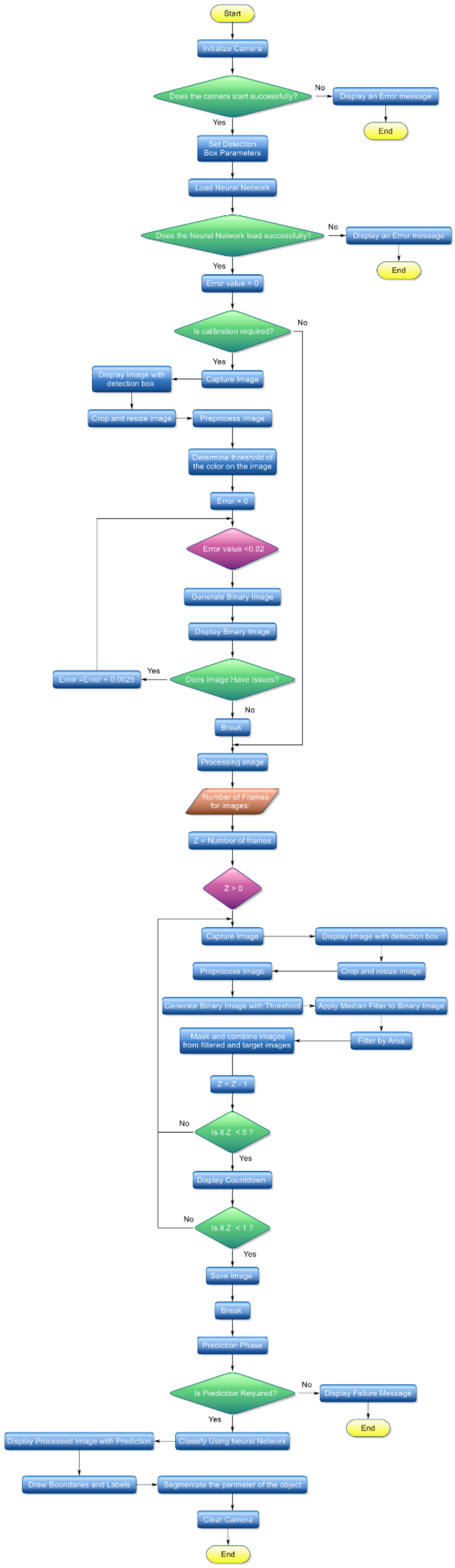


Figure 30. *Flowchart of Artificial vision.*

The result of all these processes is the segmentation of the image and its application in a real-time viewer and predictor. These processes can be summarized in Figure 31, which is based on AV since the algorithm for the movement of the robotic arm goes together with the prototype made.

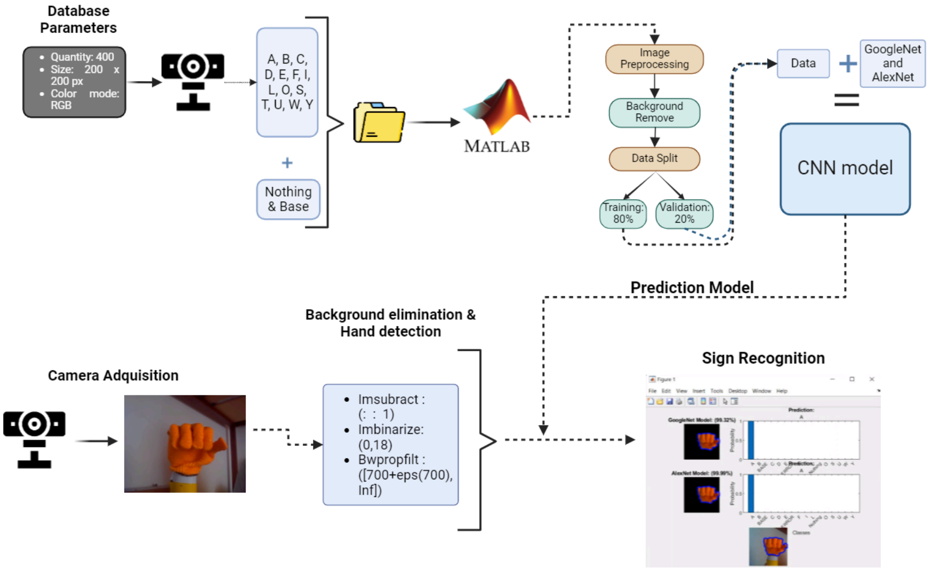


Figure 31. *Scheme of Artificial vision.*

3.5. PRACTICAL DEVELOPMENT – PROTOTYPE

In this part of the curricular integration work, the methodology for concurrent design for the prototype of the robotic hand is proposed.

To begin with, we will identify the need related to the requirements proposed for the robot's development. One of the main objectives of this project is to simulate the movement of the human hand for the interpretation of ASL, so we need the robotic arm model to have the following characteristics: Table 14.

Table 14. *Necessity for the robotic arm.*

#	Necessity
1	An anthropomorphic robot must have 5 fingers to simulate a human hand
2	Movable wrist (Required for certain letters)
3	Design for 6 servo motors
4	Finger flexibility
5	Available for 3D printing

Next, we searched for several models on Thingiverse, which is an online repository that allows the user to acquire and mix 3D printing models and allows the user to upload, share, and download 3D printing models [193]. The first model consists of a claw that allows one to move certain objects from one place to another [194]. The second model is a model much more similar to the human hand as it consists of 5 fingers which have flexibility when printing but no wrist movement [195]. The third model also has five fingers, and it is not possible to place several servos, and it does not have wrist movement [196]. Finally, the last model has 5 fingers, wrist movement, flexibility in the fingers, and space for 5 servos [197].

The different models are classified in the range of 0 to 1, where 0, we place one if the design is better than the other arm designs, 0.5 if the design is equivalent to the others, and 0 if the design is inferior or worse than the others named [198], Table 15.

Table 15. *Concurrent design on an as-needed basis for the robotic arm.*

Model	5 fingers	Movable wrist	Servo Motors for each finger	Finger flexibility	Available for 3D printing	Sum	%
Model 1 [194]	0	1	0	0	1	2	0.142
Model 2 [195]	1	0	1	1	1	4	0.285
Model 3 [196]	1	0	0	1	1	3	0.214
Model 4 [197]	1	1	1	1	1	5	0.357
					Sum	14	1

According to Table 15, the robotic arm model that best fits what we are looking for in this project is model 4, so we use that model.

Then the materials we need to carry out this project are:

- Screws
- 3D printed parts
    - PLA
- Durable rubber
- Arduino Mega 2560

- Arduino Shield V5
- 5V & 10A power supply
- Male-male and female-male cables
- Tension cable
- 6 MG996R servo motors
- Elastic thread
- MatLab
- Arduino IDE

Once the model and materials had been chosen, we proceeded with 3D printing. This method was used due to its low cost. The materials used for printing were PLA due to their light and flexible properties, such as Table 16.

Table 16. *Properties of material (ABS and PLA) [199-200].*

Properties	PLA
Chemical formula	PLA [(C3H4O2)n]
Technical name	Polylactic Acid
Modulus of elasticity	37 MPa
Melting temperature	173 C
Density	1.3 g/cm3
Biodegradability	Yes

Then, for the 3D printing process, we used PrusaSlicer software, which helped us place all the necessary features, see Figure 32. The process of printing the parts was exhausting since it took more than 90 hours to print the robotic arm completely. In addition, the printing characteristics of the robotic arm are detailed in Table 17.

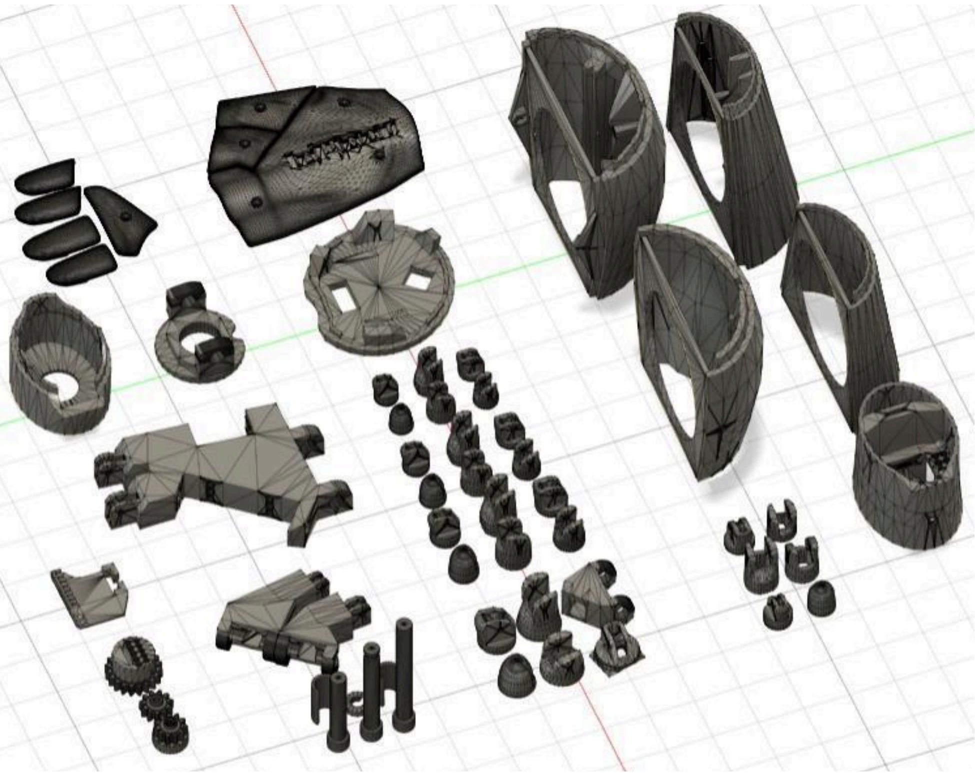


Figure 32. *Robotic arm exploded view using Fusion 360.*

Table 17. *Features of arm parts.*

Parts of the arm	Infill	Wall thickness	Other parameters	Time
Wristarge	30%	2mm	no support, no raft.	6 hours
wrist small	30%	2mm	no support, no raft.	4 hours
Thumb	30%	2mm	no support, no raft.	2 hours & 45 minutes
Index3	30%	1.5mm	no support, no raft, no brim.	2 hours & 15 minutes
majeure 3	30%	1.5mm	no support, no raft, no brim.	2 hours & 19 minutes
ringfinger 3	30%	1.5mm	no support, no raft, no brim.	2 hours & 55 minutes
auriculaire 3	30%	2mm	no support, no raft, no brim.	2 hours & 43 minutes
rob part2	30%	2mm	with brim, no raft, no support.	10 hours
rob part3	30%	2mm	with brim, no raft, no support.	10 hours & 30 minutes

rob part4	30%	2mm	with brim, no raft, no support.	15 hours & 15 minutes
rob part5	30%	2mm	with brim, no raft, no support.	16 hours
cover finger	30%	2mm	with support	1 hour
rota wrist 2	30%	2mm	with support	3 hours & 47 minutes
rota wrist 1V3	30%	2mm	with support	4 hours & 30 minutes
rota wrist 3V2	30%	2mm	with support	1 hour & 22 minutes
Cable Holder wrist	30%	2mm	without support	1 hour
Top Surface	30%	2mm	without support	1 hour & 40 minutes
Ribcap	30%	2mm	without support	2 hours & 23 minutes

The arm was assembled using silicone to secure each part. We proceeded to make the joints of the smaller parts (fingers and hand) and then the larger parts, such as the forearm. Once the parts were sectioned, we proceeded to the incorporation of the threads for the movement. In the extension part, which is the posterior area, elastic threads were incorporated. For the frontal area, slightly stiffer and non-elastic threads were applied so that the elastic threads could be tightened and the fingers could return to the base position of the hand. On the other hand, multiple tests were carried out to ensure that the threads worked correctly.

In this section, we secured these threads to the servomotors, which were previously configured at an angle of 90° so that they could have 90° to perform the flexion movement and 90° to perform the extension movement but in the opposite direction, working the initial 90° angle as our point of origin. Once each of the threads was tensioned, we proceeded to perform a small test of the movement to determine whether it was necessary to adjust the threads in the servomotors.

On the other hand, the wrist's movement was configured at an angle of 180° so that it could perform the expected movement to position letters that require the arm to be in that position.

In addition, we proceeded to perform a motion experiment by energizing all the servo motors using the 5 volts and 10 amps power supply to obtain joint movement of all the servomotors. It was determined that the arm worked properly, but additional tests are still needed to improve the fluidity of the movement. Finally, the code was coded using the Arduino software for the simulation of ASL; see the scheme in Figure 33.

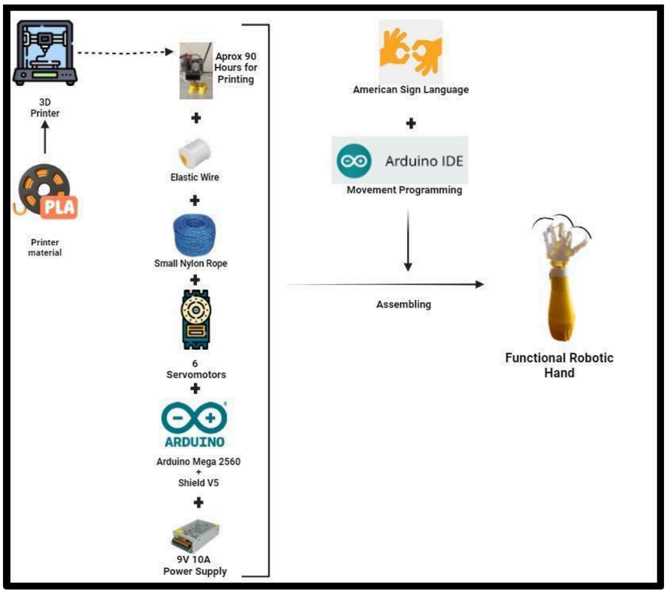


Figure 33. *Scheme of prototype creation.*

On the other hand, the materials needed are specified in Table 18, where we can visualize the individual prices of each material, adding up to a total of 163.67 USD. The materials were selected to fit the user's limited budgets without compromising their quality.

Table 18. *Total cost and unit price of material costs for the construction of the robotic arm.*

Quantity	**Material**	**Cost(Unitary) USD**	**Total Cost USD**
1	PLA	25	25
3	MG996R Servo Motors	11.99	35.97
3	MG946R Servo Motors	12.99	38.97
1	Power supply	12.99	12.99
1	Connector cable	3	3
5	Rubber	1.30	6.5
1	Elastic roll	4	4
1	Rope or string	1	1
1	Arduino Mega 2560	24.99	24.99
1	Arduino Mega shield	5	5
25	Screws	0.15	3.75
25	Nuts	0.10	2.5
Total			**163.67 USD**

3.6. SYSTEM INTEGRATION

Once the machine vision system, the neural network, and the robotic arm prototype were implemented, we integrated the complete system,see Figure 34.

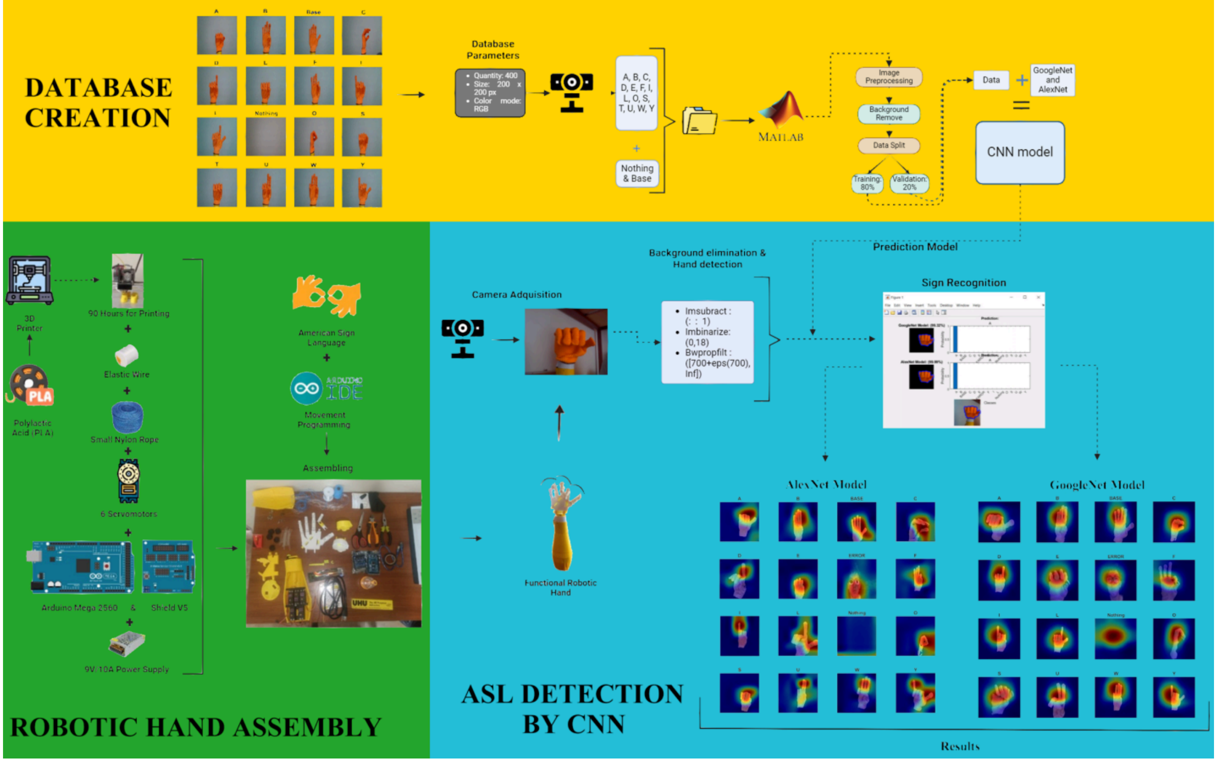


Figure 34. *Global Scheme.*

The project was structured in 3 main stages: Creation of the database, construction and movement of the robotic arm and detection by neural networks of the ASL. The database was created using models of human hand and robotic arm with 16 and 29 classes respectively, the images were captured with black background from 3 different angles to improve the training. For the creation of the robotic arm a 3D printer was used where the main material was PLA due to its malleability and resistance, also used Arduino Mega 2560, 180º Servomotor with torque of 12 kg, Arduino Shield V5 and a power supply which was necessary for the arm to perform the movement of the fingers. Finally, for the detection of the ASL, the AlexNet and GoogleNet models were used, having an accuracy higher than 98% in both cases confirming that the neural network learned as can be visualized in the Grad-Cam.

The system integration is intended to perform tests of the different sign language letters made by the robotic arm to identify whether the CNN correctly predicts the letters created with the arm, see Figure 35.

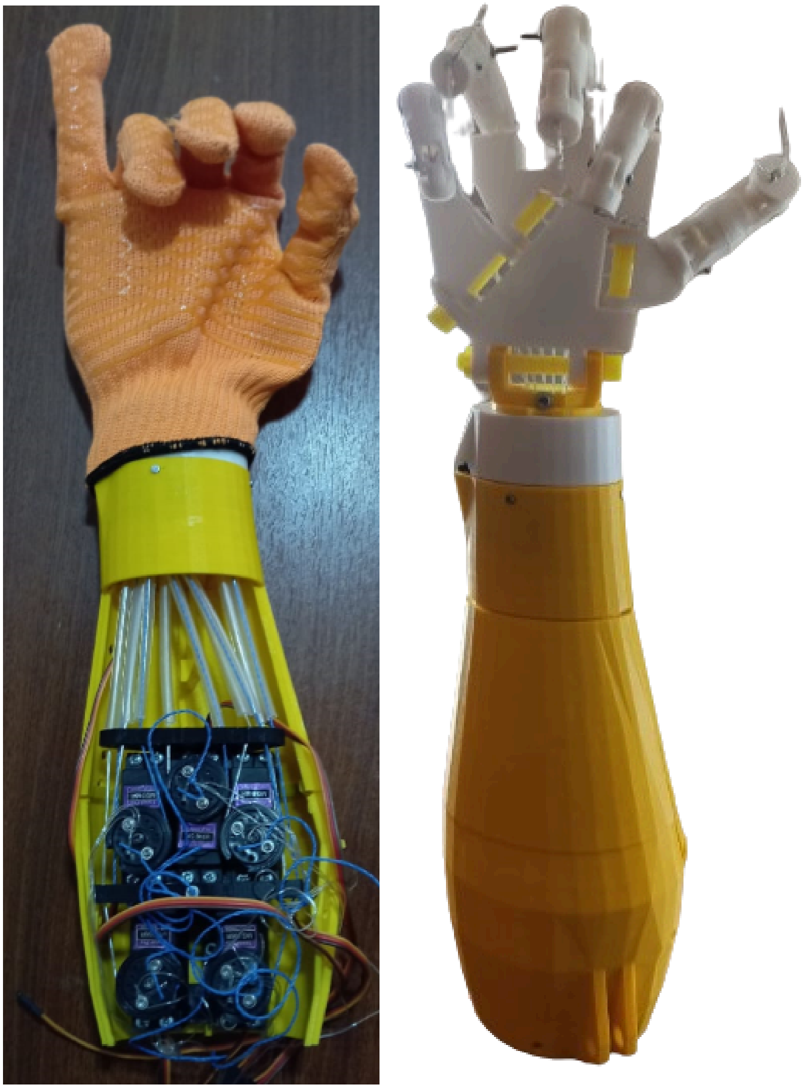


Figure 35. *Robotic hand prototype.*

Afterward, tests were carried out to demonstrate the functionality of the created model, and several errors were revealed.

The first error we noticed was due to the environment in which the database was created. Because of this, the code must be run in a controlled environment because it presents errors when working in an environment that does not have a black background. For this reason, a mockup was made with a black background to place the robotic arm. This is being used to ensure the correct functioning of the network with AV.

Another error we had was when interpreting the results by the CNN since it did not generate the correct shape of the letters of the sign language, so we proceeded to adjust the parameters, such as the color adjustment in which the RGB image was divided into its 3 basic channels in order to separate the red channel since this is the closest to the skin color. After adjusting the parameters, CNN training was carried out.

In addition, an error was detected after making several movements with the robotic arm because it did not perform the movements satisfactorily since the tension threads used lost elasticity due to them; several tension threads were placed in each of the arms. Fingers, which allowed correct movements, the parameters in the finger movement code were also adjusted for the realization of sign language.

3.7. EXPERIMENT PHASE

To determine the validity of our results, several experiments were carried out to create our database, with which we will proceed to train the neural networks to classify and predict new images using human and robotic hands.

3.7.1. Experiment 1: Database

For this section, we must determine both the characteristics of the images we need (valid letters, necessary size, cases of events).

We also use several processes to maintain the main characteristics and suppress unnecessary features. The median filter is indispensable in convolutional neural networks since it eliminates noise and maintains the edges simultaneously. We have opted for this method over a more advanced one. Although the CLAHE filtering model highlights better features in the image, when extracting the background, it also removed important parts of the Han since increasing the contrast changed the colors.

On the other hand, bilateral filtering blurs image values in specific sections or the whole image. By applying this method, we eliminated important features, so this was discarded.When performing the median filter, first, the highlight color of the image must be extracted. In this case, it was re. Then, a binarization is performed to eliminate unnecessary pixels with median filtering. This results in the total elimination of the background, as shown in Figure 36.

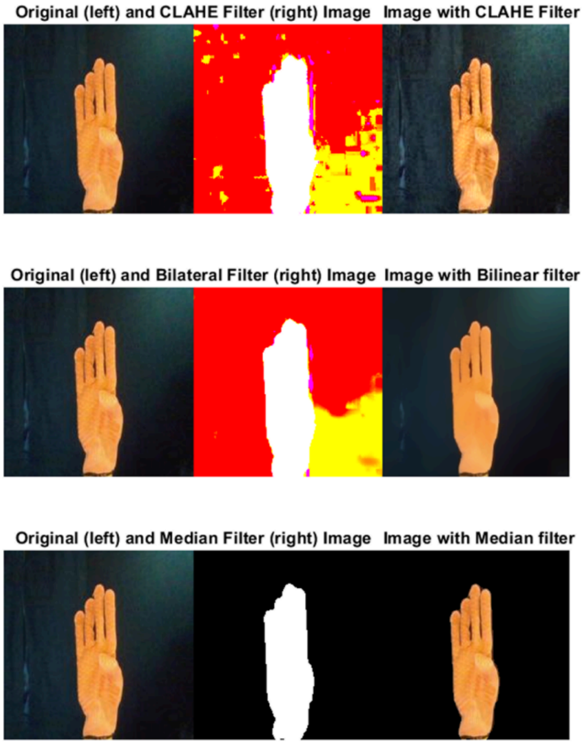

Original (left) and CLAHE Filter (right) Image
Image with CLAHE Filter
Original (left) and Bilateral Filter (right) Image
Image with Bilinear filter
Original (left) and Median Filter (right) Image
Image with Median filter

Figure 36. *Different filters to make our DataBase.*

A. Human Hand

As a first experiment, we will use a method to generate our sign language database. For this purpose, we will use the Matlab program to create a program capable of taking 600 images and storing them in the various classes along with their corresponding labels in different folders.

This database will consist of 29 classes: 26 classes corresponding to the letters of the American alphabet and 3 special classes focused on analyzing cases: Nothing, where there are no objects on the screen; Base, where the image of the hand with the fingers extended and Error, being these images of the face so that the program can determine if it is making an incorrect shot of the human hand, see Table 19.

Table 19. *Human Database characteristics.*

Class	Letter	Quantity (Images)	Size (px)	Color mode	Weight (Kb) per image
1	A	600	200 x 200	RGB	4.00

2	B	600	200 x 200	RGB	4.00
3	C	600	200 x 200	RGB	4.00
4	D	600	200 x 200	RGB	4.00
5	E	600	200 x 200	RGB	4.00
6	F	600	200 x 200	RGB	4.00
7	G	600	200 x 200	RGB	4.00
8	H	600	200 x 200	RGB	4.00
9	I	600	200 x 200	RGB	4.00
10	J	600	200 x 200	RGB	4.00
11	K	600	200 x 200	RGB	4.00
12	L	600	200 x 200	RGB	4.00
13	M	600	200 x 200	RGB	4.00
14	N	600	200 x 200	RGB	4.00
15	O	600	200 x 200	RGB	4.00
16	P	600	200 x 200	RGB	4.00
17	Q	600	200 x 200	RGB	4.00
18	R	600	200 x 200	RGB	4.00
19	S	600	200 x 200	RGB	4.00
20	T	600	200 x 200	RGB	4.00
21	U	600	200 x 200	RGB	4.00
22	V	600	200 x 200	RGB	4.00
23	W	600	200 x 200	RGB	4.00
24	X	600	200 x 200	RGB	4.00
25	Y	600	200 x 200	RGB	4.00
26	Z	600	200 x 200	RGB	4.00
27	Nothing	600	200 x 200	RGB	4.00
28	Base	600	200 x 200	RGB	4.00
29	Error	600	200 x 200	RGB	4.00

B. Robotic Arm

For this section, we will use the same method for taking images and storing them using Matlab. In this experiment, the use of a glove is essential because the color of the robotic hand is white, which ensures its detection.

In the case of the robotic arm, we will use limited-class models because the model is limited in its articulation. So, in this database, we will take into account 16 classes in which we have letters that have a simple movement, plus two special classes that would be a base movement with fully extended fingers and a class that detects when there are no objects to analyze.

The same characteristics are presented in size, number of images, and color. These are 200 x 200 pixels, 600 images, and RGB images, see Table 20.

Table 20. *Robot Database characteristics.*

Class	Letter	Quantity (Images)	Size (px)	Color mode	Weight (Kb) per image
1	A	600	200 x 200	RGB	4.00
2	B	600	200 x 200	RGB	4.00
3	C	600	200 x 200	RGB	4.00
4	D	600	200 x 200	RGB	4.00
5	E	600	200 x 200	RGB	4.00
6	F	600	200 x 200	RGB	4.00
7	I	600	200 x 200	RGB	4.00
8	L	600	200 x 200	RGB	4.00
9	O	600	200 x 200	RGB	4.00
10	S	600	200 x 200	RGB	4.00
11	U	600	200 x 200	RGB	4.00
12	W	600	200 x 200	RGB	4.00
13	Y	600	200 x 200	RGB	4.00
14	Error	600	200 x 200	RGB	4.00
15	Nothing	600	200 x 200	RGB	4.00
16	Base	600	200 x 200	RGB	4.00

3.7.2. Experiment 2 : Convolutional Neural Network

In this section, we will perform several analyses focused on the validation of the neural network models (GoogleNet and Alexnet) in order to determine which one we can obtain the best results from and apply to the established objectives. To assess the validity of this experiment, confusion matrices will be made, and the models will be tested in different epochs since Matlab incorporates the option to save the models as they are trained depending on the configurations that we place. In our case, a total of 10 epochs will be performed, and the model will be saved every 2 minutes.

Below, we can visualize the parameters used for the training of GoogleNet and AlexNet, see Table 21.

Table 21. *CNN configuration options.*

Configuration of training	Operation	Values
Data Processing		
Imds (ImageDataStore)	Training	80 %
	Validation	20 %
	Resize	227 x 227 px AlexNet 224 x 224 px GoogleNet
Data Augmentation	RandScale	[0.3 1.3]
	RandXTranslation	[0 10]
	RandYTranslation	[-50 20]
	RandXReflection	True
Training Options		
Frequently Used	Solver	SGDM
	InitialLearnRate	0.001
	MiniBatchSize	64
	MaxEpoch	10
	ValidationFrequency	20
Learn Rate	LearnRateSchedule	Piecewise
	LearnRateDropFactor	3
	LearnRateDropPeriod	2
Normalization and Regularization	L2Regularization	0.0001

A) Human hand

Once the databases were created, we proceeded to develop an ImageDataStore, with the objective of storing all our images with their corresponding classes and labels so that we would have the necessary data to train our CNNs.

When using pre-trained models such as AlexNet and GoogleNet, it is necessary to perform learning transfer because these models are already trained with data from various images. For this, according to the Matlab guide [162], two methods can be used: the first consists of resizing the images to the input ones and adjusting the classification values to the desired ones, and the second can be done by using images at any type of scale smaller than the base (for these models their base is 224 x224, and 227x227 pixels), this can cause limitations in the prediction or even in several layers.

B) Robotic Arm

On the other hand, the images of the robotic arm were made following the previous parameters but with a smaller number of classes, greatly accelerating this process. We also used AlexNet and GoogleNet models for training. Therefore, the parameters of the neural networks were changed to adjust to the number of classes, in this case, 16. However, in this experiment, an error category was not added because it was planned that there would be no additional objects besides the robotic arm.

C) Grad-CAM

After performing this analysis, the Gradient-weighted class activation technique (Grad-CAM) was used. This technique visualizes the values that the neural network considers important to make its prediction.

3.7.3. Experiment 3: Artificial Vision

This section analyzes the different backgrounds with which the detection of the objects to be classified can work depending on the environment. Changes such as contrast, brightness, and hue can vary the result, as the neural networks were focused on training using RGB images. Also, we applied several factors to keep the detection environment acceptable; those configurations can be shown in Table 22.

Table 22. *CNN configuration options.*

Processing images for detection	Description	Configuration
Detection Sector	It isolates a specific area of the main image to focus only on sign language detection.	[100 0 550 600]
Resized	Images were resized according to the input size of the neural network models.	**CNN**
		AlexNet (227 x 227) px
		GoogleNet (224 x 224) px
Subtract Red Color	It is divided in the red channel of the image, highlighting glove features (orange).	imsubtract(img(:,:,1)

RGB2Gray	It is transformed from grayscale to binary, with an adjustment of 0.086 to extract the glove characteristics, and the rest is removed.	(0.086)
Median Filter	A median filter is applied to remove noise and maintain edges at the same time.	[3 3]
Delete small characteristics	If the filter does not remove all unnecessary small pixels, objects with an area of 700 pixels or less are removed, leaving only objects with a larger area.	'Área', [700 + eps(700), Inf])
Create a mask and the detection image	A mask of the processed image is created on the original image by removing regions that do not belong to the main features.	bsxfun(@times, originalimage, uint8(processedimg));

A) Human hand

The human hand was evaluated on a colored background, and a corresponding pre-processing was performed to eliminate the background of the image, leaving only the area of interest.

For this purpose, the values of the red channel of the image were extracted, and applying binarization values and filters was necessary to extract the most important features and eliminate the background.

B) Robotic Arm

In the case of the robotic arm, we have an orange glove, so the process that was performed for the human hand is similar since the Red channel is extracted from the image to focus on the important features.

3.7.4. Experiment 4: Prediction Applications

In this experiment, the idea is to evaluate two types of applications of CNN models. To obtain prediction results, the combination of the models obtained by AV and CNN models must be carried out.

A) Real-Time Analysis

The first application proposed is using the camera to predict sign language images in real-time. For this section, it is essential to have a suitable environment for pattern recognition and to be able to classify correctly, in addition to having a computer with good features to maintain performance and quality.

B) Single Image

On the other hand, the detection by image acquisition is done by capturing an image with a counter so that the user has enough time to get into position to obtain the image. Once this process is done, IP is applied to reduce errors or unnecessary objects. Finally, the appropriate prediction and segmentation must be performed. For this section, any computer can perform it because it does not consume many resources.

Chapter IV

Results and Discussion

4.1. Experiment 1: DataBase

The main characteristic of each class is the number of images, which totals 600 color images for each class, each 200 x 200 pixels in size and weighing approximately 4 kilobytes per image.

4.1.1. Human hand

The results of the database creation were obtained in a folder called "Photos," where the images were stored in color and with the other previously established parameters. It took approximately 30 minutes to make the 600 images (200 per three angles) for each of the classes (29) previously established; see Figure 37. This is equivalent to 580 images per minute, making their respective resizing. All images had 4 kilobytes of storage; this means 2.4 megabytes of images per class and a total of 69.6 megabytes for this database.

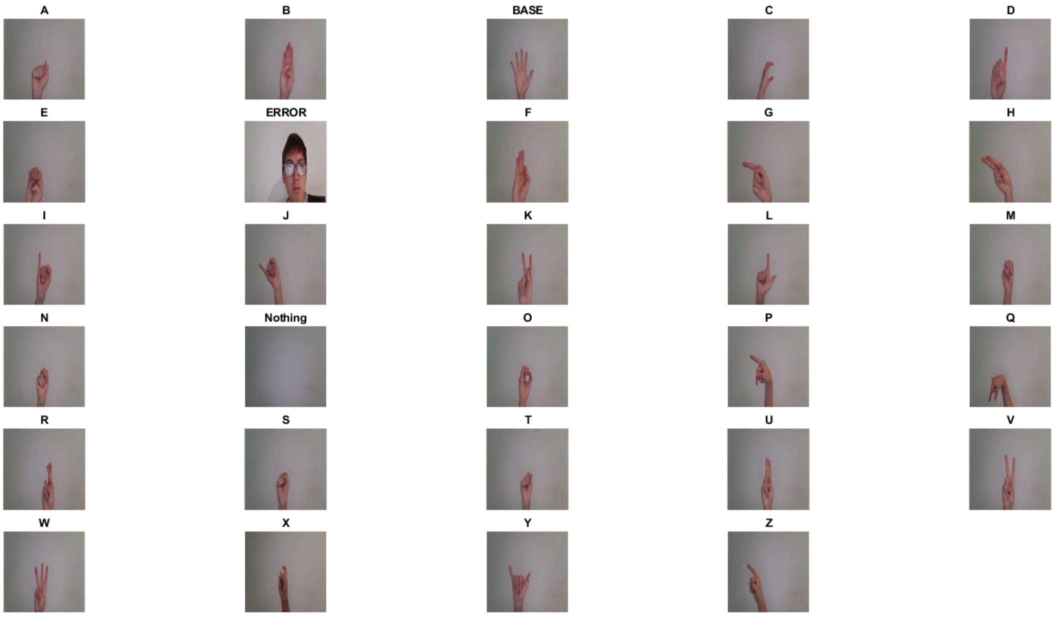


Figure 37. *ASL is made by the Human hand.*

4.1.2. Robotic Hand

The same image acquisition process was carried out for the robotic hand; however, the values that varied were the number of folders obtained because this database was made with 16 classes, as shown in Figure 38. This means that the image acquisition time was reduced to almost half of the time used in the previous database, which had a total size of 38.4 megabytes.

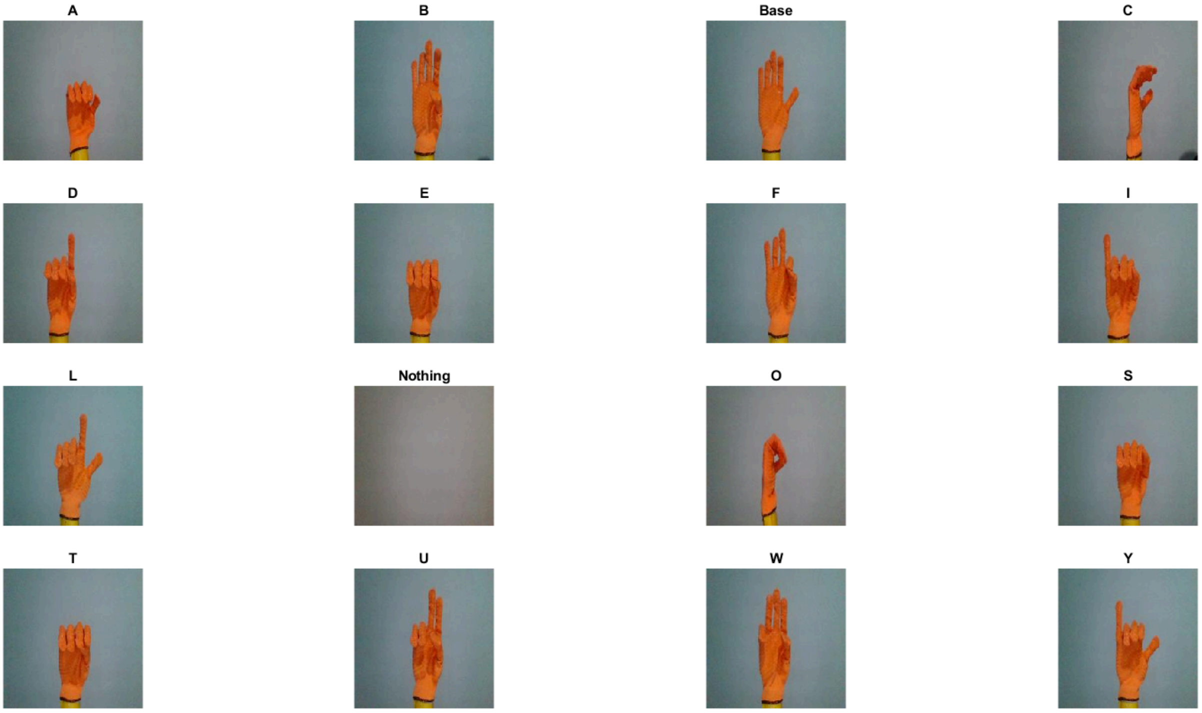
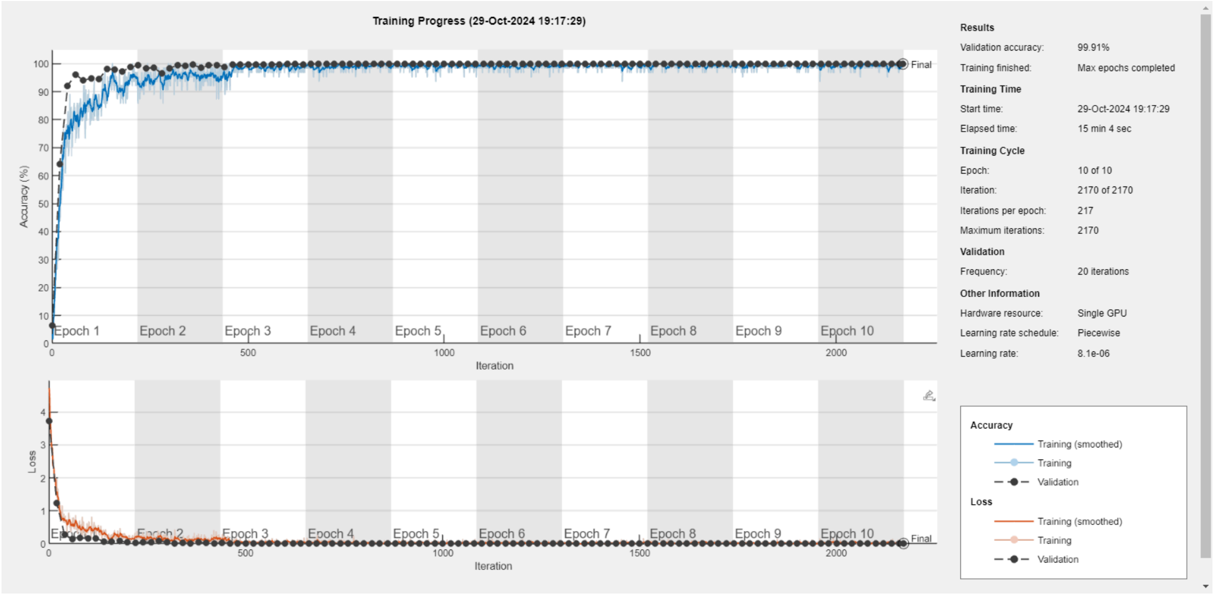


Figure 38. *ASL is made by robotic hand.*

4.2. EXPERIMENT 2: CONVOLUTIONAL NEURAL NETWORK

4.2.1. Human

A) AlexNet



Results
Validation accuracy: 99.91%
Training finished: Max epochs completed

Training Time
Start time: 29-Oct-2024 19:17:29
Elapsed time: 15 min 4 sec

Training Cycle
Epoch: 10 of 10
Iteration: 2170 of 2170
Iterations per epoch: 217
Maximum iterations: 2170

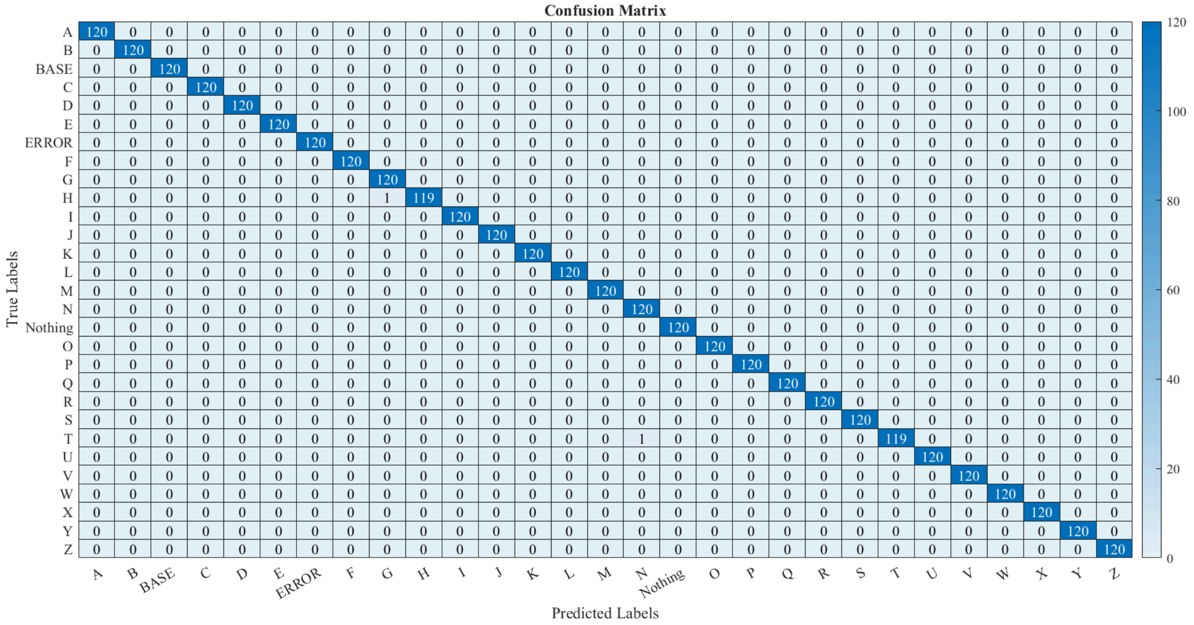
Validation
Frequency: 20 iterations

Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Learning rate: 8.1e-06

Figure 39. Training progress for Alexnet with Human DataBase.

The Alexnet model applying the human hand database presents interesting training because it agrees with normal training. The accuracy values, since they increase as the iterations progress, are also validation values, which represent the correct predictions that are processed together with the validation data.

On the other hand, the loss values show a decrease in the graph, which demonstrates a reduction in the errors that the model can make when making a prediction. In addition to the reduction in the validation loss values, which are also characterized by decreasing because they represent the error of the validation data, the validation accuracy value is 99.91. This may occur because the amount of data is very small, which implies that overfitting may occur, see Figure 39.



	A	B	BASE	C	D	E	ERROR	F	G	H	I	J	K	L	M	N	Nothing	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BASE	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ERROR	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	1	119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0
Nothing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	119	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120

Figure 40. *Confusion Matrix for Alexnet with Human DataBase.*

This confusion matrix shows the performance of the AlexNet pre-trained neural network with the human hand database. Each class is represented by a letter ("A" to "Z") and additional classes labeled "Base", "Error" and "Nothing". As can be seen in Figure 40, each diagonal cell contains values of 120, which indicates that for each class, the model classifies correctly in its great majority. In contrast, in classes such as "H" and "T", there is a readback to 119 in the number of correct predictions, which indicates a slight confusion between these classes. Moreover, no significant off-diagonal values are observed, so we can point out that the model is quite accurate.

The confusion matrix performed by the trained model indicates that there are no false positive or false negative values but an excellent predictive ability, similar to that of the training data, shown in Figure 40. However, as detailed above, this may be due to the number of epochs and also to a very small amount of data.

Confusion matrix is a fundamental key to analyze more specific values that will help us to evaluate our model to determine if we have a good prediction. In Table 23 those values: true positive (TP), false positive (FP), false negative (FN) and true negative (TN) were calculated. These values are obtained from the confusion matrix by making the relationship between each row and column. In the Diagonal values of the matrix we will have the TP values which are the values predicted by the neural network and classified as true and in the opposite case the TN. In the FP and FN values the relationship that exists is that the model predicted erroneously depending on whether they were true or negative and predicted them with their homonyms.

Table 23. *TP, FP, FN and TN values for the AlexNet model with Human Database.*

Classes	**TP**	**FP**	**FN**	**TN**
A, B, C, D, E, F, G, I, J, K, L, M, N, O, P, Q, R, S, U, V, W, X, Y, Z	120	0	0	3360
H and T	119	0	1	

These data allow us to calculate the Accuracy, Recall, Accuracy, F1-Score, in the different models. For our sign language detection model these parameters indicate:

Precision: Ensures that the model does not misinterpret gestures.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Ensures that relevant signs are not omitted.

$$Recall = \frac{TP}{TP + FN}$$

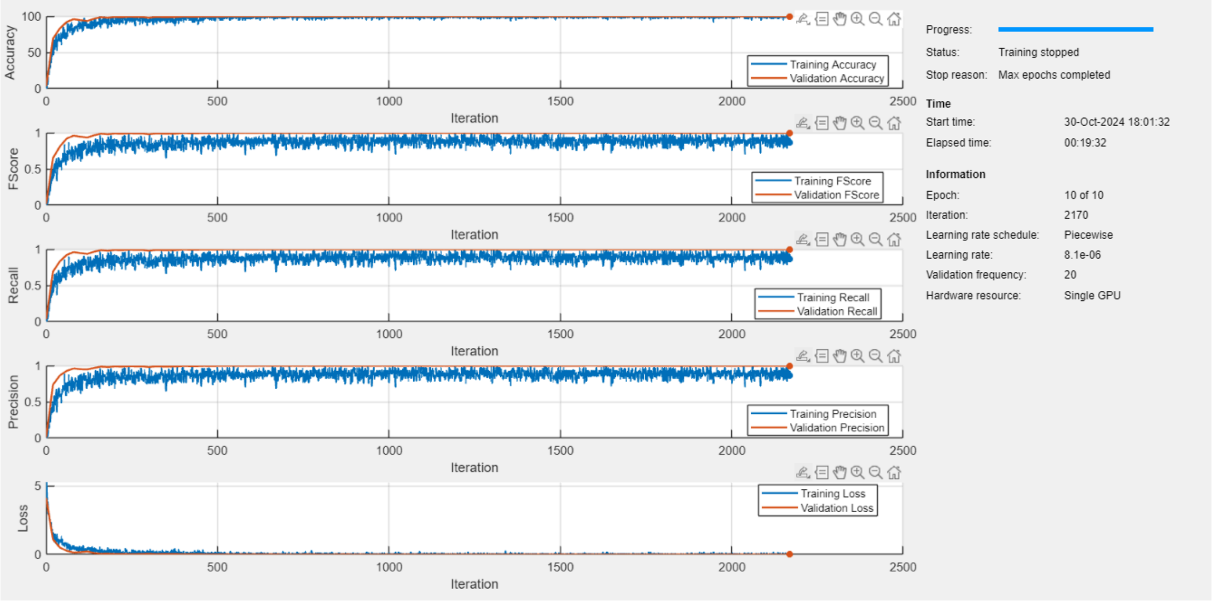
Accuracy: It is the performance of the model and how the data is balanced.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

F1-Score: It is the relationship between accuracy and recall, fundamental to determine if our model presents a considerable performance when relating false positive and false negative values.

$$F1 - Score = 2 * \frac{precision * recall}{precision + recall}$$

These values are obtained through each iteration and epoch, in addition to being able to visualize them in Figure 41 indicating a good training.



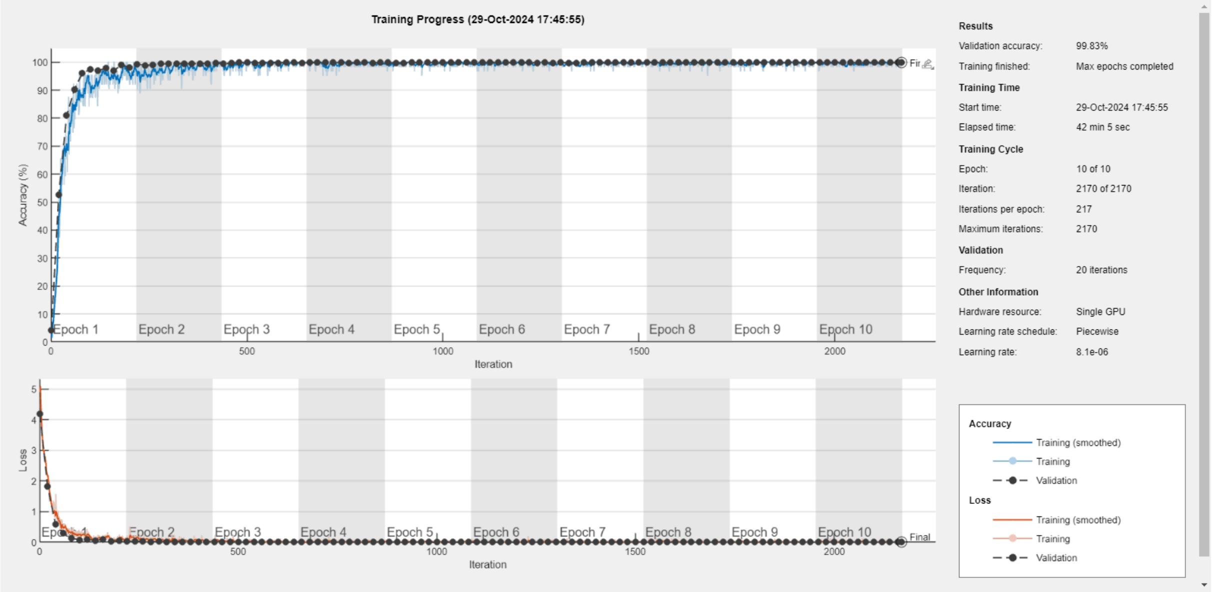
Progress:
Status: Training stopped
Stop reason: Max epochs completed

Time
Start time: 30-Oct-2024 18:01:32
Elapsed time: 00:19:32

Information
Epoch: 10 of 10
Iteration: 2170
Learning rate schedule: Piecewise
Learning rate: 8.1e-06
Validation frequency: 20
Hardware resource: Single GPU

Figure 41. *Important Metrics for AlexNet with Human DataBase.*

B) GoogleNet



Results
Validation accuracy: 99.83%
Training finished: Max epochs completed

Training Time
Start time: 29-Oct-2024 17:45:55
Elapsed time: 42 min 5 sec

Training Cycle
Epoch: 10 of 10
Iteration: 2170 of 2170
Iterations per epoch: 217
Maximum iterations: 2170

Validation
Frequency: 20 iterations

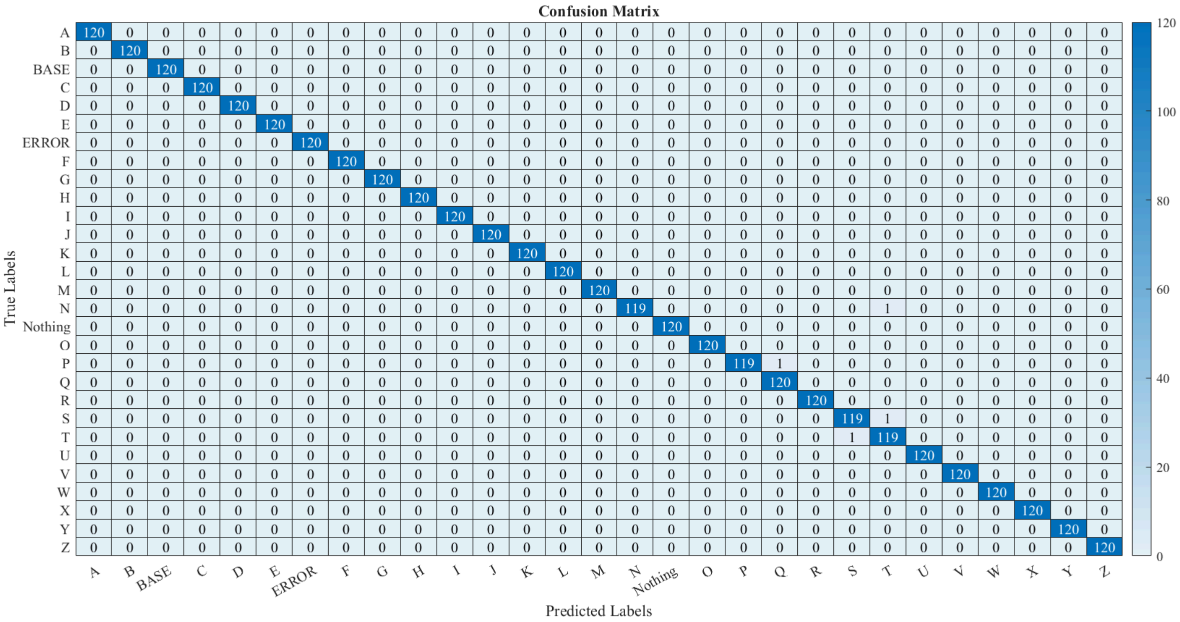
Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Learning rate: 8.1e-06

Figure 42. *Training progress for GoogleNet with Human DataBase.*

The GoogleNet model applying the human hand database presents training similar to the previous model. We also have positive training since the accuracy values increase as the iterations progress, and the validation values remain positive as the training is carried out.

On the other hand, the loss values show a decrease in the model errors. In addition, the errors of the validation values also decrease.

The validation accuracy value is 99.83. The initial values and data are also proposed to be very small for the number of iterations and epochs performed, see Figure 42.



	A	B	BASE	C	D	E	ERROR	F	G	H	I	J	K	L	M	N	Nothing	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BASE	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ERROR	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	0	0	0	0	0	0	1	0	0	0	0	0	0
Nothing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	119	1	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	119	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120

Figure 43. *Confusion Matrix for GoogleNet with Human DataBase.*

This confusion matrix shows the performance of the pre-trained GoogleNet neural network with the human hands database. Each class is represented by a letter ("A" to "Z") and additional classes labeled "Base", "Error" and "Nothing". In Figure 43, it is observed that the majority of cells on the diagonal contain values of 120, indicating that, for each class, the model classifies correctly for the vast majority. However, in classes such as "N", "P" , "S" and "T", there is a regression to 119 in the number of correct predictions, indicating a slight confusion between these classes. For example, the values that produced the minimum error were the predictions of a letter N mistaken for an M. In addition, no off-diagonal values are observed indicating that the model is quite accurate.

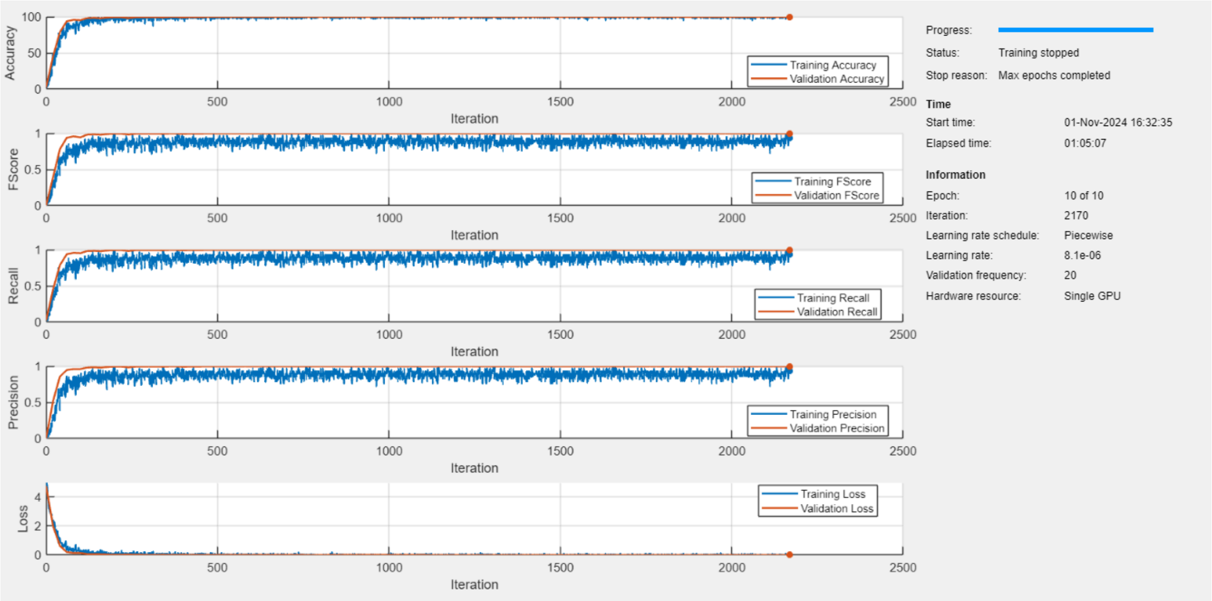
In the same way as the previous experiment here we also have to determine the values of *TP, FP, FN and TN* seen in Table 24 by means of the confusion matrix Figure. 43.

Table 24. *TP, FP, FN and TN values for the GoogleNet model with Human Database.*

Classes	**TP**	**FP**	**FN**	**TN**
A, B, C, D, E, F, G, I, H, J, K, L, M, O, Q, R, U, V, W, X, Y, Z, Base,	120	0	0	3360

Nothing, Error.				
N, P, S, T	119	0	1	

Finally, we have the graph of the accuracy, recall and FScore analysis along with the training values of accuracy and loss, see Figure 44.



Progress:
Status: Training stopped
Stop reason: Max epochs completed

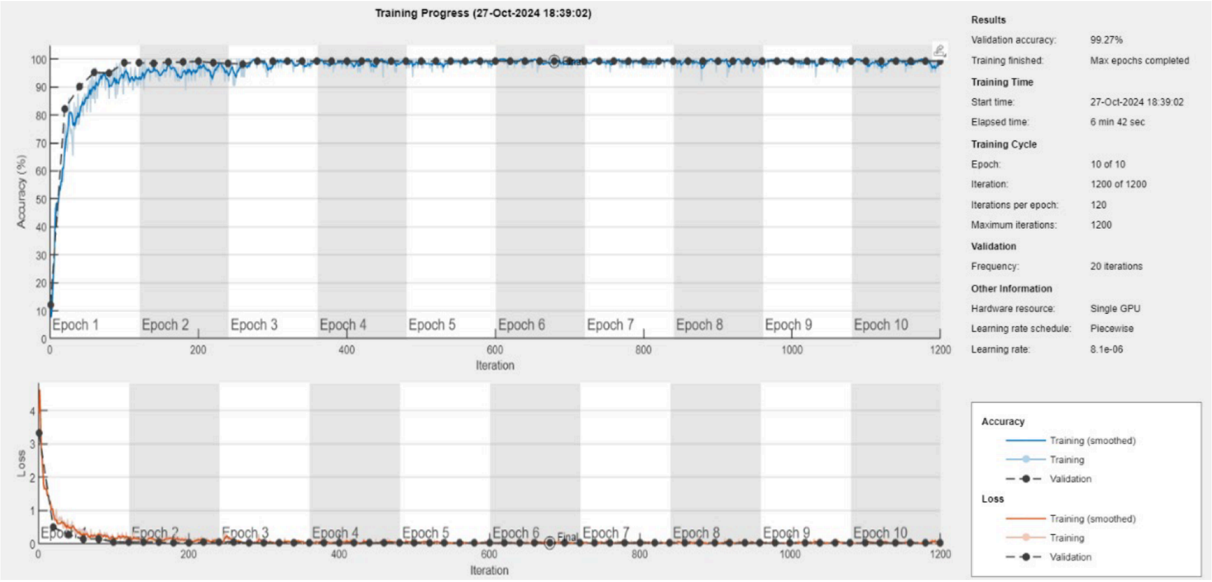
Time
Start time: 01-Nov-2024 16:32:35
Elapsed time: 01:05:07

Information
Epoch: 10 of 10
Iteration: 2170
Learning rate schedule: Piecewise
Learning rate: 8.1e-06
Validation frequency: 20
Hardware resource: Single GPU

Figure 44. *Important Metrics for GoogleNet with Human DataBase.*

4.2.2. Robotic Hand

A) AlexNet



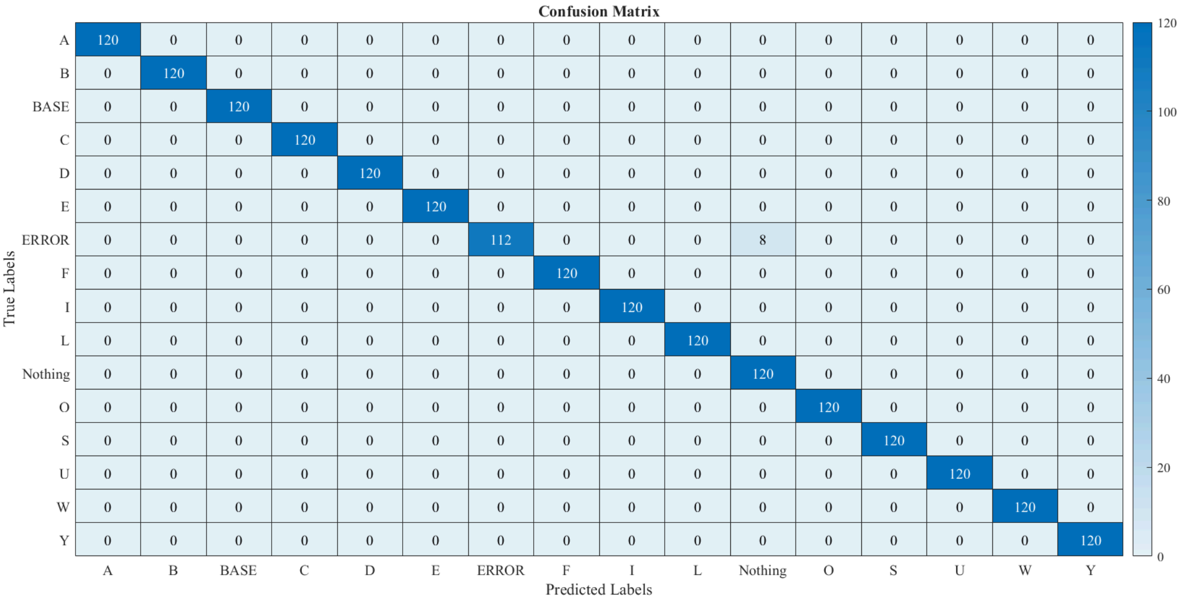
Results
Validation accuracy: 99.27%
Training finished: Max epochs completed
Training Time
Start time: 27-Oct-2024 18:39:02
Elapsed time: 6 min 42 sec
Training Cycle
Epoch: 10 of 10
Iteration: 1200 of 1200
Iterations per epoch: 120
Maximum iterations: 1200
Validation
Frequency: 20 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Learning rate: 8.1e-06

Figure 45. *Training progress for Alexnet with Robotic DataBase.*

The AlexNet model applying the Robotics hand database presents normal training; however, the training values present some irregularities, which are thought to be due to the dropout layer. On the other hand, we have had positive training since the accuracy values increased, and the validation values also showed a significant increase.

Finally, the loss values show a decrease in the model errors. In addition, the errors of the validation values also decrease.

The validation accuracy value is 99.27%, which is interesting due to the different classes in this model compared to those of the human hand, see Figure 45.



	A	B	BASE	C	D	E	ERROR	F	I	L	Nothing	O	S	U	W	Y
A	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BASE	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0
ERROR	0	0	0	0	0	0	112	0	0	0	8	0	0	0	0	0
F	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0
Nothing	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120

Figure 46. *Confusion Matrix for Alexnet with Robotic DataBase.*

This confusion matrix shows the performance of the AlexNet pre-trained neural network with the database of a robotic hand. Each class is represented by a letter ("A", "B", "C", "D", "E", "F", "I", "L", "O", "S" , "U", "W", "Y") and additional classes labeled "Base", "Error" and "Nothing". In Figure 46, it is observed that the majority of cells in the diagonal contains values of 120, which indicates that, for each class, the model classifies correctly in its vast majority. On the contrary, in the class "Error", there is a regression to 112 in the number of correct predictions, indicating a slight confusion between these classes.

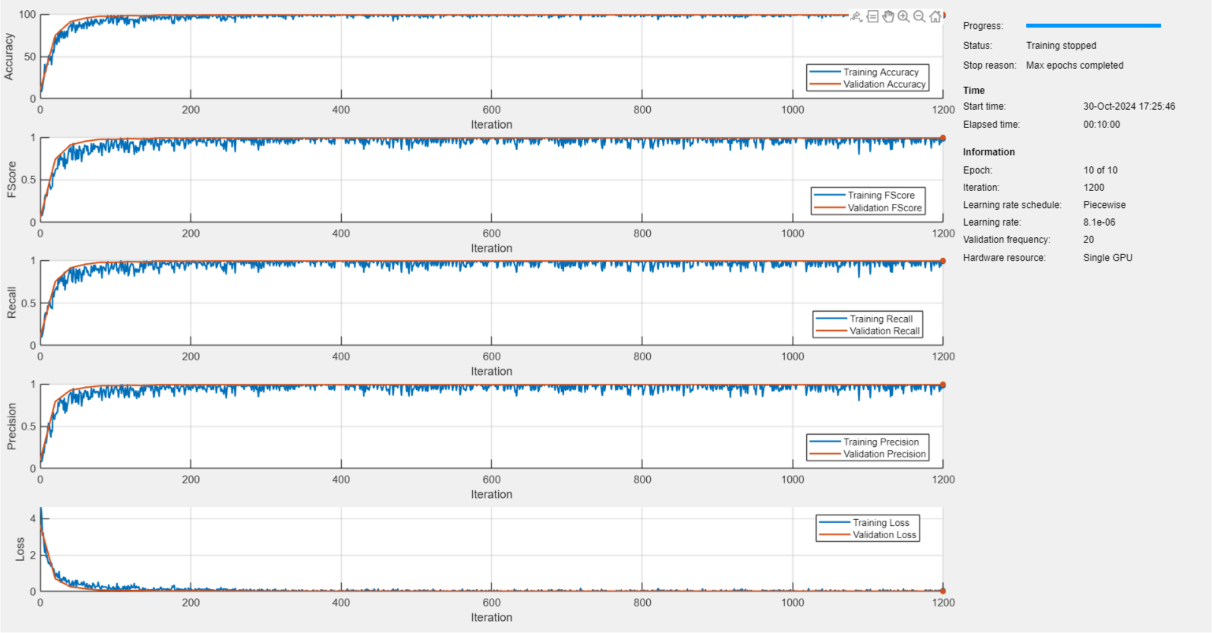
In Table 25 the *TP, FP, FN and TN* values were calculated.

Table 25. *TP, FP, FN and TN values for the AlexNet model with robotic Database.*

Classes	**TP**	**FP**	**FN**	**TN**
A, B, C, D, E, F, I, L, O,S , U, W, Y, Base, Nothing	120	0	0	1800
Error	112	0	8	

The process with which we obtain the recall, precision and Fscore values is the same, using the confusion matrix to obtain their values. However, in this case the number of classes decreased from 29 to 16, it is also reflected in the True Negative values, from 3360 to 1800.

In the same way, the training process, Recall, FScore and accuracy is similar to the previous experiments. Obtaining curves with high values, see Figure 47.



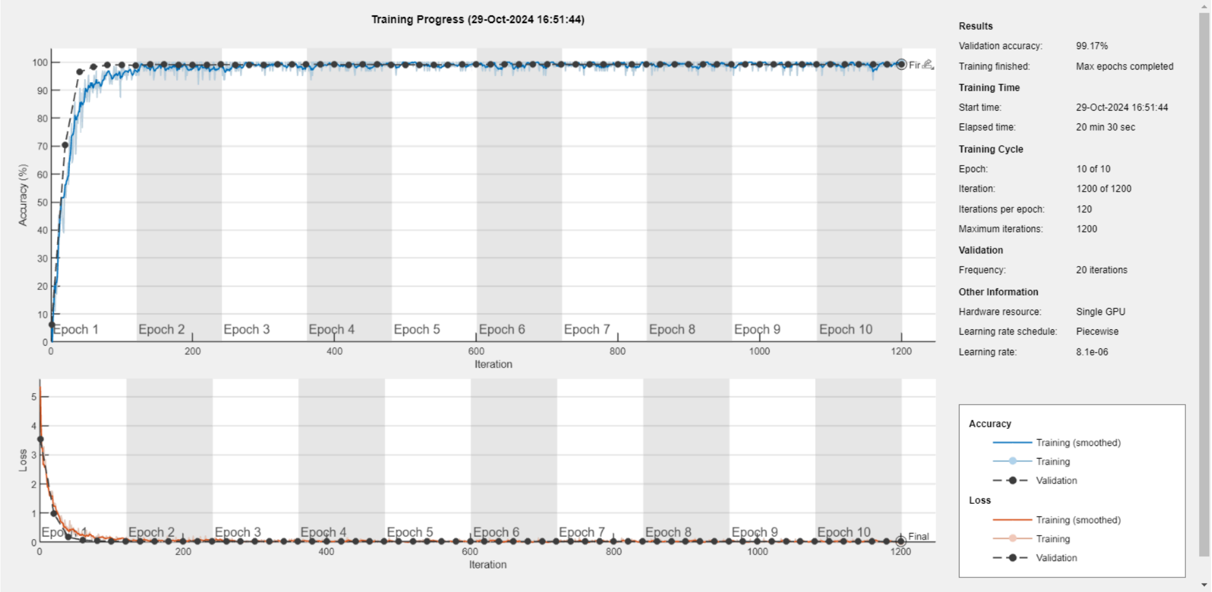
Progress:
Status: Training stopped
Stop reason: Max epochs completed

Time
Start time: 30-Oct-2024 17:25:46
Elapsed time: 00:10:00

Information
Epoch: 10 of 10
Iteration: 1200
Learning rate schedule: Piecewise
Learning rate: 8.1e-06
Validation frequency: 20
Hardware resource: Single GPU

Figure 47. *Important Metrics for AlexNet with Robotic DataBase.*

B) GoogleNet



Results
Validation accuracy: 99.17%
Training finished: Max epochs completed

Training Time
Start time: 29-Oct-2024 16:51:44
Elapsed time: 20 min 30 sec

Training Cycle
Epoch: 10 of 10
Iteration: 1200 of 1200
Iterations per epoch: 120
Maximum iterations: 1200

Validation
Frequency: 20 iterations

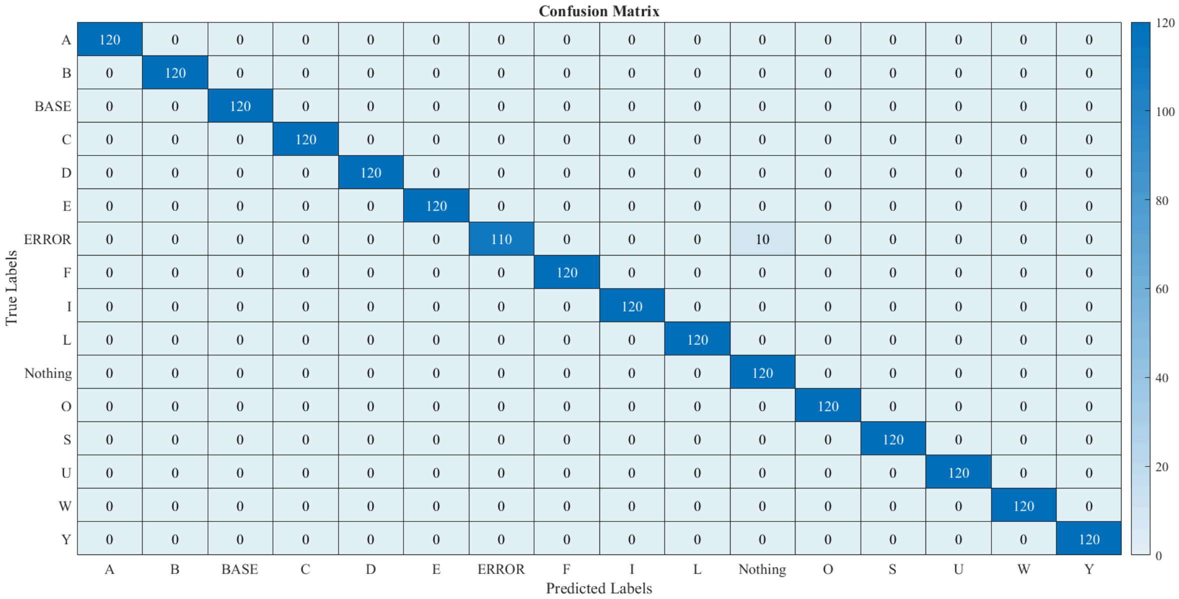
Other Information
Hardware resource: Single GPU
Learning rate schedule: Piecewise
Learning rate: 8.1e-06

Figure 48. *Training progress for GoogleNet with Robotic DataBase.*

The GoogleNet model applying the Robotic Hand database presents normal training reaching high training values at epoch two, as the previous model presents smaller irregularities than the previous model.

Finally, the loss values show a decrease in the model's errors. In addition, the errors of the validation values also decrease, as shown in Figure 48.

The validation accuracy value is 99.17%, which is lower than the previous model's, so tests are proposed to determine which model is better.



	A	B	BASE	C	D	E	ERROR	F	I	L	Nothing	O	S	U	W	Y
A	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BASE	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	120	0	0	0	0	0	0	0	0	0	0
ERROR	0	0	0	0	0	0	110	0	0	0	10	0	0	0	0	0
F	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0	0
Nothing	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120

Figure 49. *Confusion Matrix for GoogleNet with Robotic DataBase.*

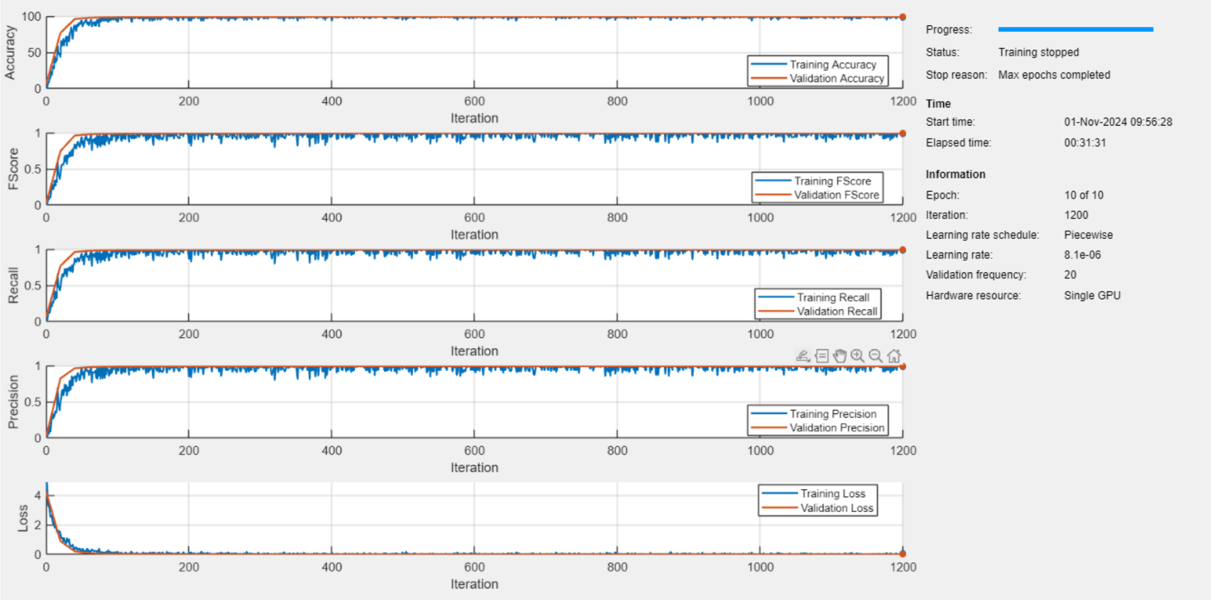
This confusion matrix shows the performance of the pre-trained GoogleNet neural network with the database of a robotic hand. Each class is represented by a letter ("A", "B", "C", "D", "E", "F", "I", "L", "O", "S" , "U", "W", "Y") and additional classes labeled "Base", "Error" and "Nothing". Figure 49 shows that the diagonal contains values of 120, which indicates that for each class the model classifies correctly in its great majority.

On the other hand, the values of *TP, FP, FN and TN* were calculated as shown in Table 26.

Table 26. *TP, FP, FN and TN values for the GoogleNet model with robotic Database.*

Classes	**TP**	**FP**	**FN**	**TN**
A, B, C, D, E, F, I, L, O,S , U, W, Y, Base, Nothing, Error	120	0	0	1800

The method remains similar to the previous experiment only changing the Neural Network model.



Status: Training stopped
Stop reason: Max epochs completed

Time
Start time: 01-Nov-2024 09:56:28
Elapsed time: 00:31:31

Information
Epoch: 10 of 10
Iteration: 1200
Learning rate schedule: Piecewise
Learning rate: 8.1e-06
Validation frequency: 20
Hardware resource: Single GPU

Figure 50. *Important Metrics for GoogleNetNet with Robotic DataBase.*

Finally, once each experiment was performed, we proceeded to compile the important values of Accuracy, Recall, F1-score and accuracy and put them together in Table 27, and Figure 50. Most of the data present high values that indicate an accurate training, however it is important to consider if an overfitting did not occur.

It is important to emphasize that although the values are very close to each other, when determining the model with a visual validation (GRAD-CAM) it is possible to appreciate the characteristics that each model highlights in its prediction.

Table 27. *All important data obtained of each model.*

CNN / Data	**Precision**	**Recall**	**F1-score**	**Accuracy**
Human GoogleNet	0.9952	0.9948	0.9950	99.83 %
Human AlexNet	0.9994	0.9994	0.9994	99.91 %
Robot GoogleNet	0.9989	0.9989	0.9989	99.17 %
Robot AlexNet	0.9961	0.9958	0.9960	99.27 %

4.2.3. Grad-CAM

A) Human DataBase

To evaluate the accuracy of the detections made by our models, we applied analysis using Grad-CAM. This method uses the gradient of the score with respect to the layers of the convolutional networks.

Grad-CAM highlights the key features in the image that the neural network models consider important to highlight their characteristics and make their predictions. This scheme can be seen in Figures 51-54. In addition, for better understanding, a color map is displayed where various shades are presented: intense blue indicates low values, and red indicates high values.

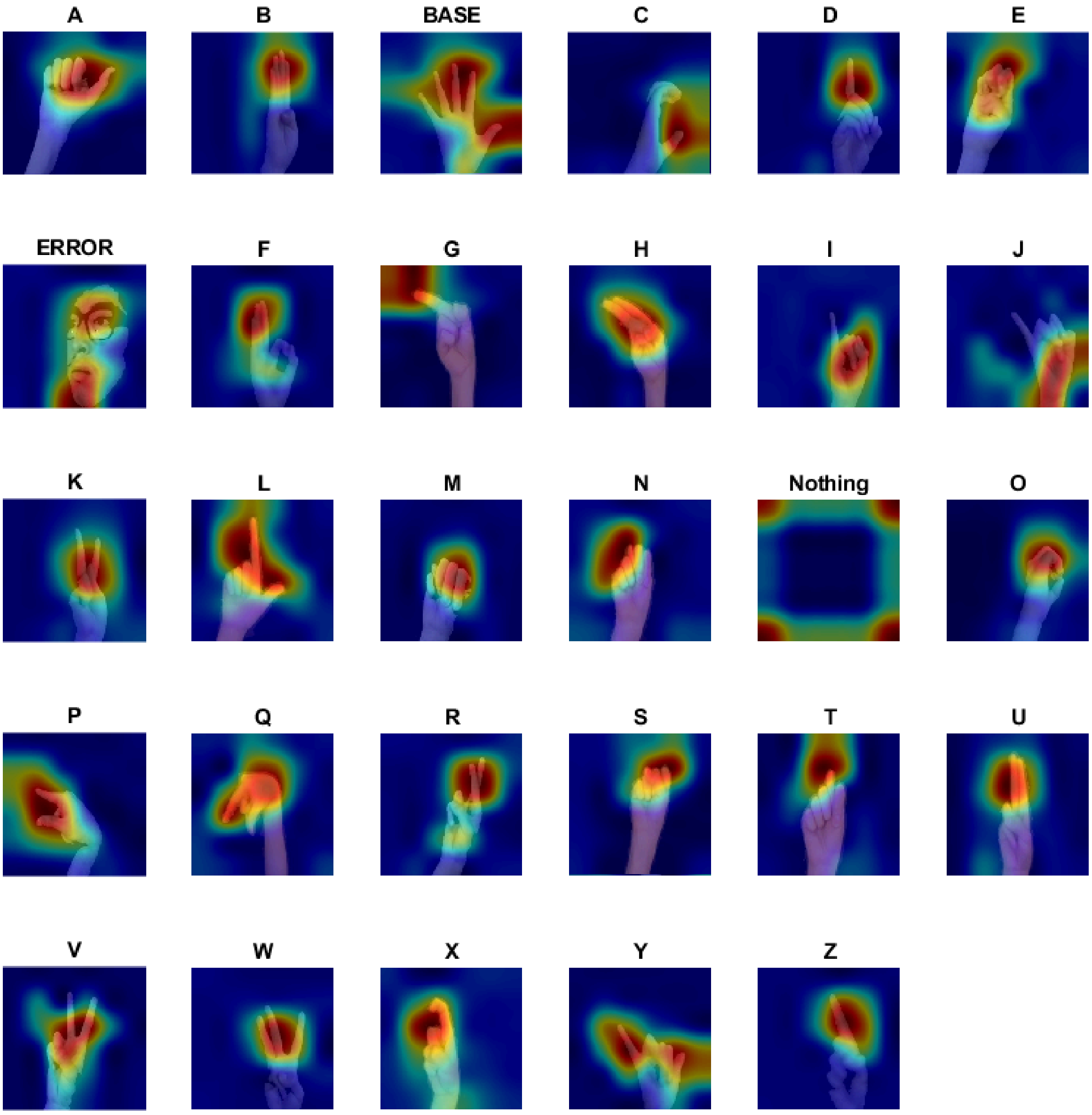


Figure 51. *Grad-CAM for AlexNet model in Human DataBase.*

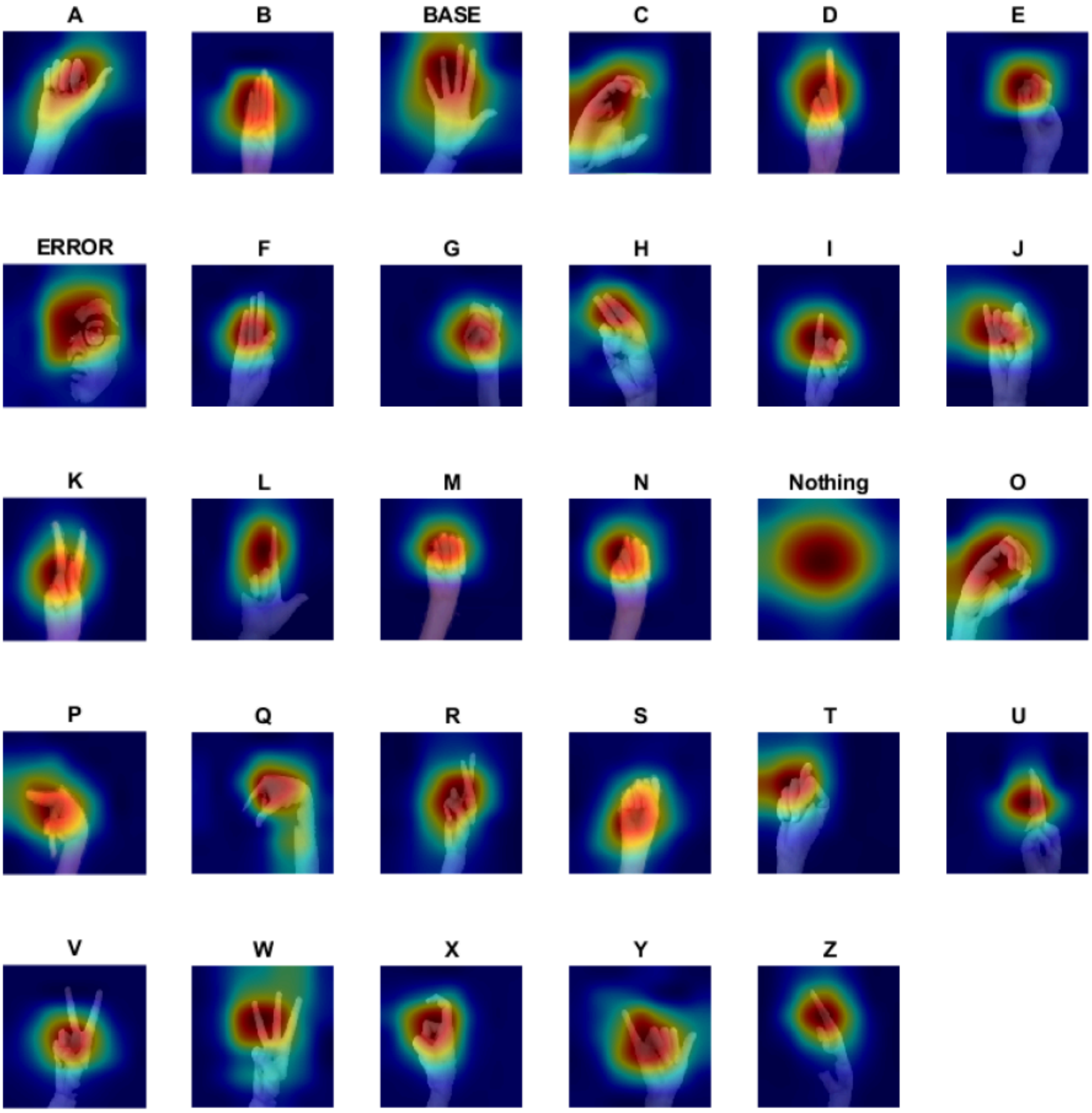
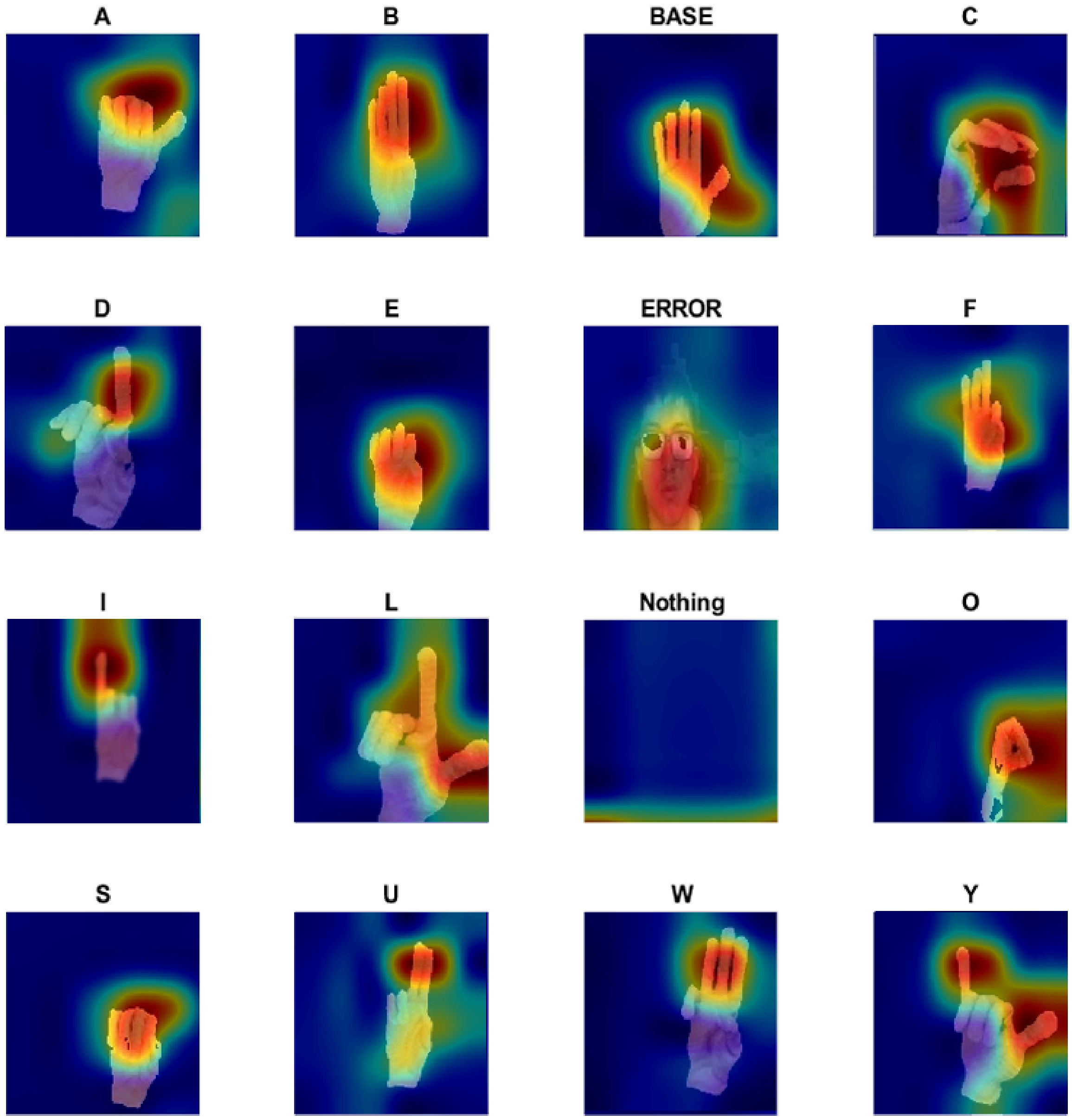


Figure 52. *Grad-CAM for GoogleNet model in Human DataBase.*

B) Robotic DataBase

The important features that can be visualized in the AlexNet and GoogleNet models are areas of the human hand, mainly in the palm and phalanges. The AlexNet-based model tends to highlight incomplete features, such as the letters A, D, I, L, W, and Y, focusing on the finger area. On the other hand, GoogleNet shows higher accuracy in the results, as it covers the hand area and characteristic spaces, indicating the sign language more completely.



A B BASE C

D E ERROR F

I L Nothing O

S U W Y

Figure 53. *Grad-CAM for AlexNet model in Robotic Hand DataBase.*

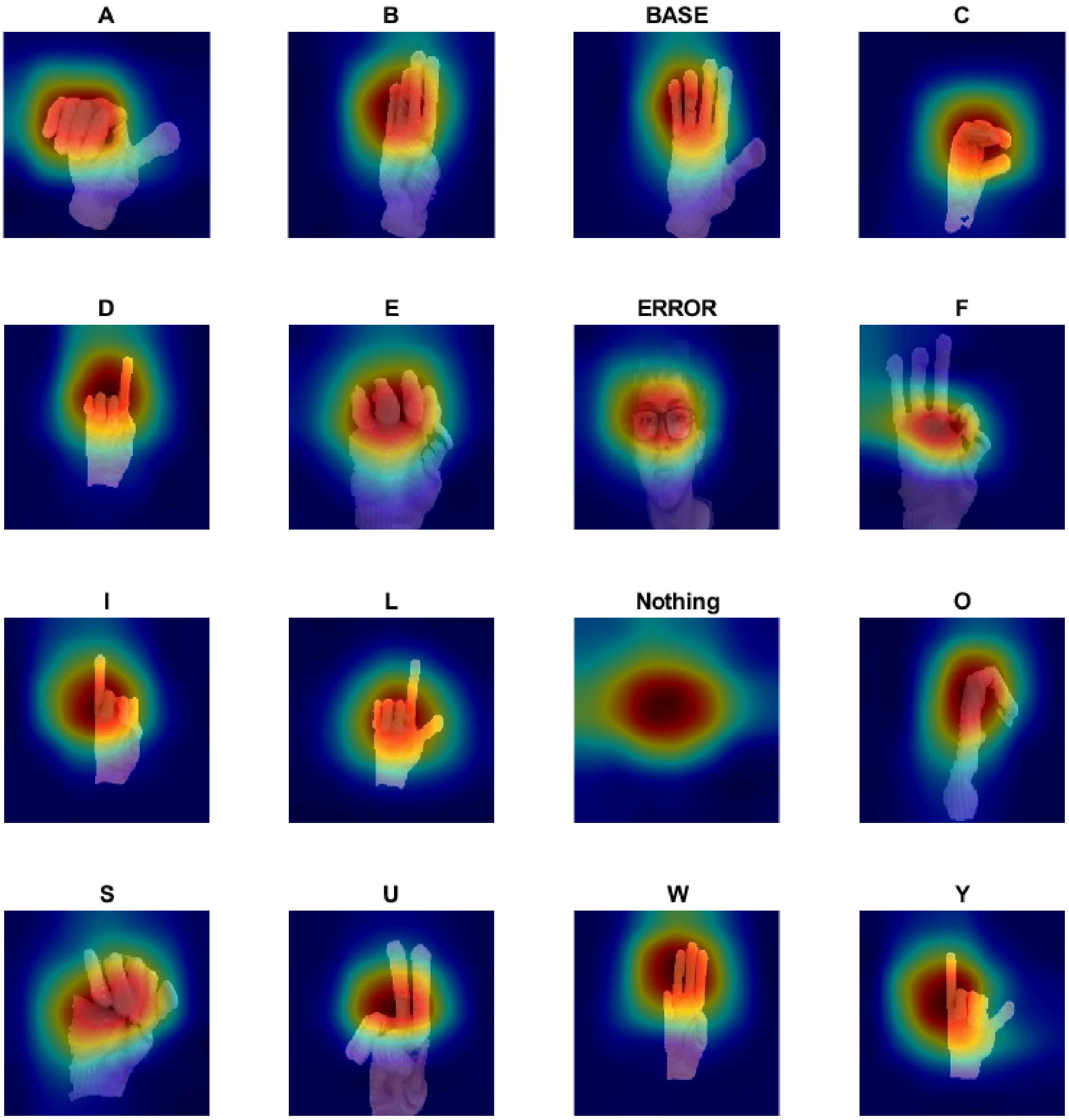


Figure 54. *Grad-CAM for GoogleNet model in Robotic DataBase*

Similarly, the use of this method allows us to visualize the important features of the robotic hand model. It is important to emphasize that in this section, it is clearly observed that the features visualized by GoogleNet are better than the features that can be presented by the AlexNet model. Because the latter presents particularly small areas or incomplete information compared to the GoogleNet model, speculating that the latter has a better predictive ability but is not 100% accurate because certain letters are not detected efficiently.

4.3. EXPERIMENT 3: ARTIFICIAL VISION

4.3.1. Human

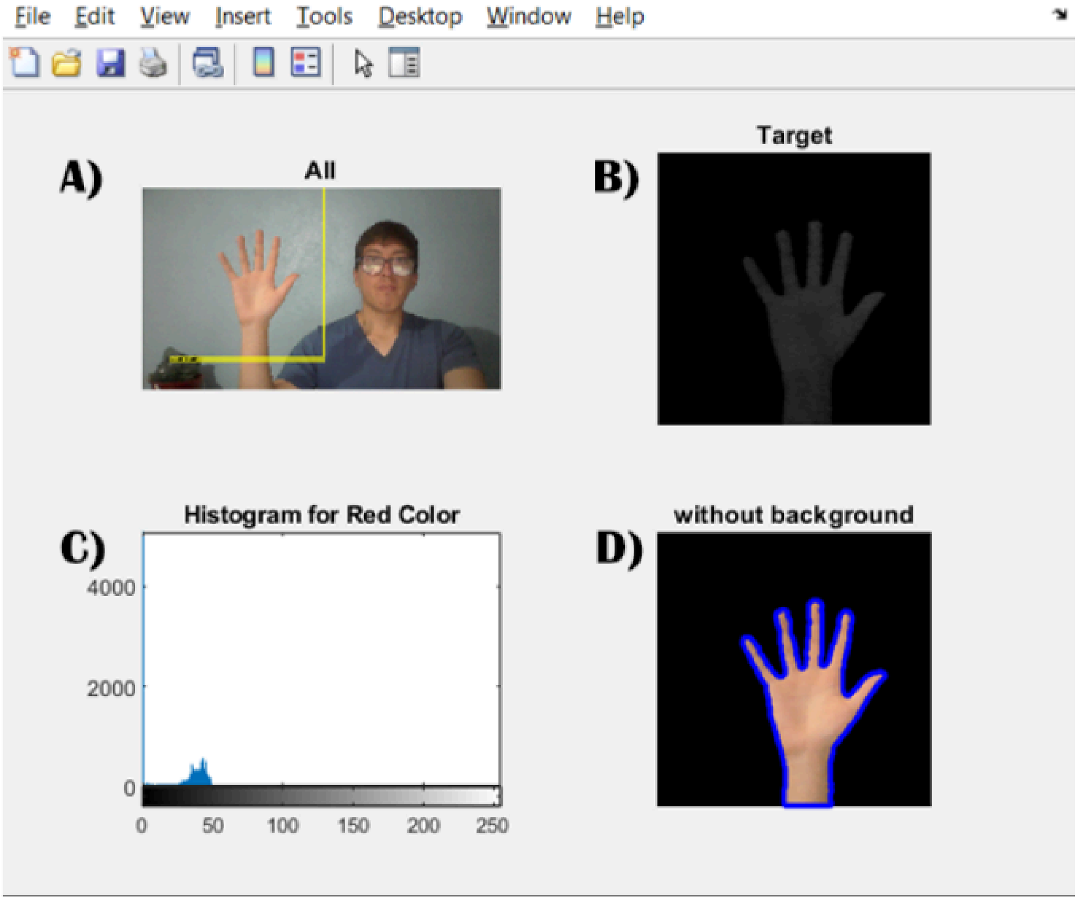


Figure 55. *Multiple steps to apply Artificial Vision for Background elimination.*

In this experiment, different parameters were evaluated to obtain an analysis of the specific area to be analyzed.

As a first step, the image was taken, and sectioning of the area in which the image is going to be analyzed was applied, this can be represented with a yellow rectangle in Figure 55 (A). As a next step, a red color channel extractor was applied to the area previously taken to extract the color characteristics of the hand and differentiate it from other objects that may be present and thus identify their pigments (0.086), in addition to this median filter and a pixel limiter were applied to avoid areas of small pixels or non-conformities in the image, Figure 55 (B).

Then a binary mask is created depending on the color intensity zone of the image; this can be visualized in a better way by a histogram showing the features and pixels that are part of an image and object that does not correspond to the hand we want to visualize. For this purpose, a small set or relief can be observed in the histogram area where the pixels of the color that are part of the hand are determined, Figure 55 (C). This data can be modified either automatically or manually to determine what type of feature is being extracted.

Finally, the binary mask is superimposed on the image obtained in section A) and the background is removed, resulting in our image of the hand without the background color, even if we encounter objects that do not correspond to the color range. In addition, we implemented a system of segmentation of the image to which we are visualizing, and we applied a blue line, Figure 55 (D).

4.3.2. Robotic Hand.

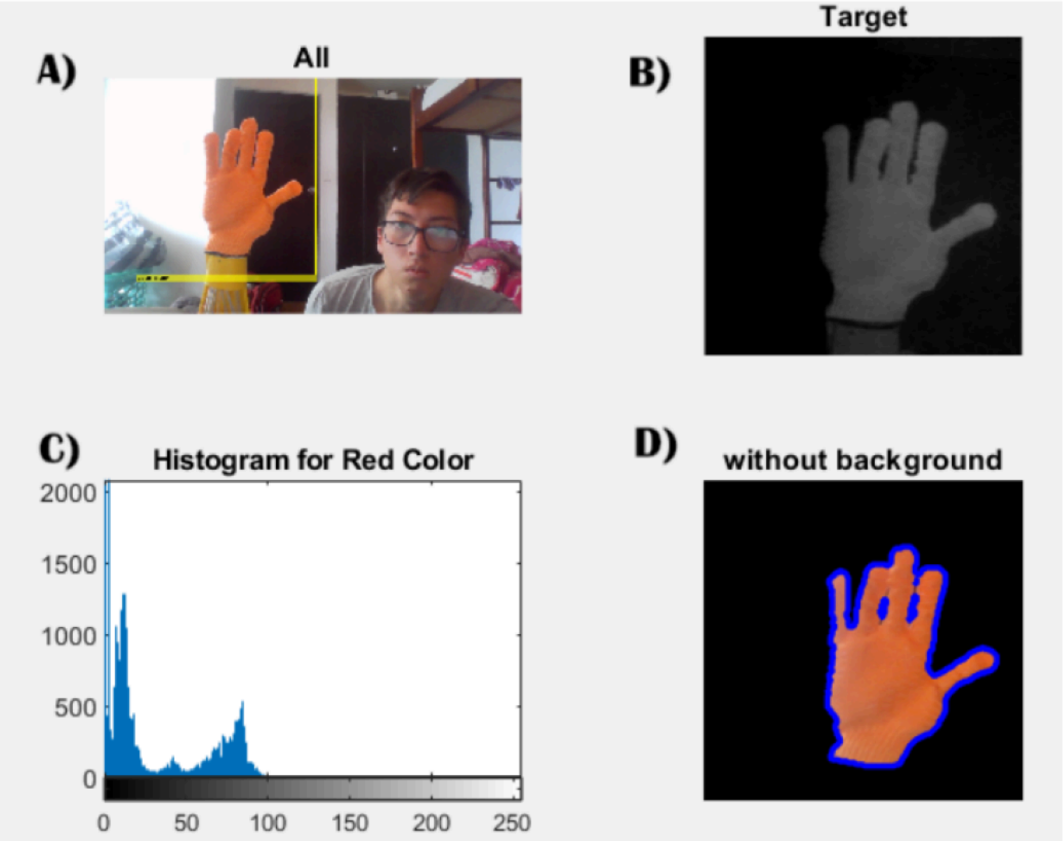


Figure 56. *Multiple steps to apply Artificial Vision for Background elimination on Robotic Hand.*

In the case of the robotic arm, the same principle can be applied; however, to guarantee the correct detection, an orange glove was placed, and the same procedures of the previous experiment were followed with the variations of the color spectrum in the human model it presented a binary mask of 0.086 with respect to the range of pixels. For the robotic arm, it was applied to 0.21 because the range that is visualized in Figure 56 (C) presents a range of color with a tendency to the white zone of the spectrum. This may represent a way to segment the image more efficiently since the color zones are more noticeable, and we can avoid taking other colors that do not correspond. Finally, we can obtain in Figure 56 (D) the same result as the previous experiment, an image of the robotic arm with the perimeter section highlighted and with its correct removal from the background.

4.4. EXPERIMENT 4: PREDICTION APPLICATIONS

4.4.1. Real-Time

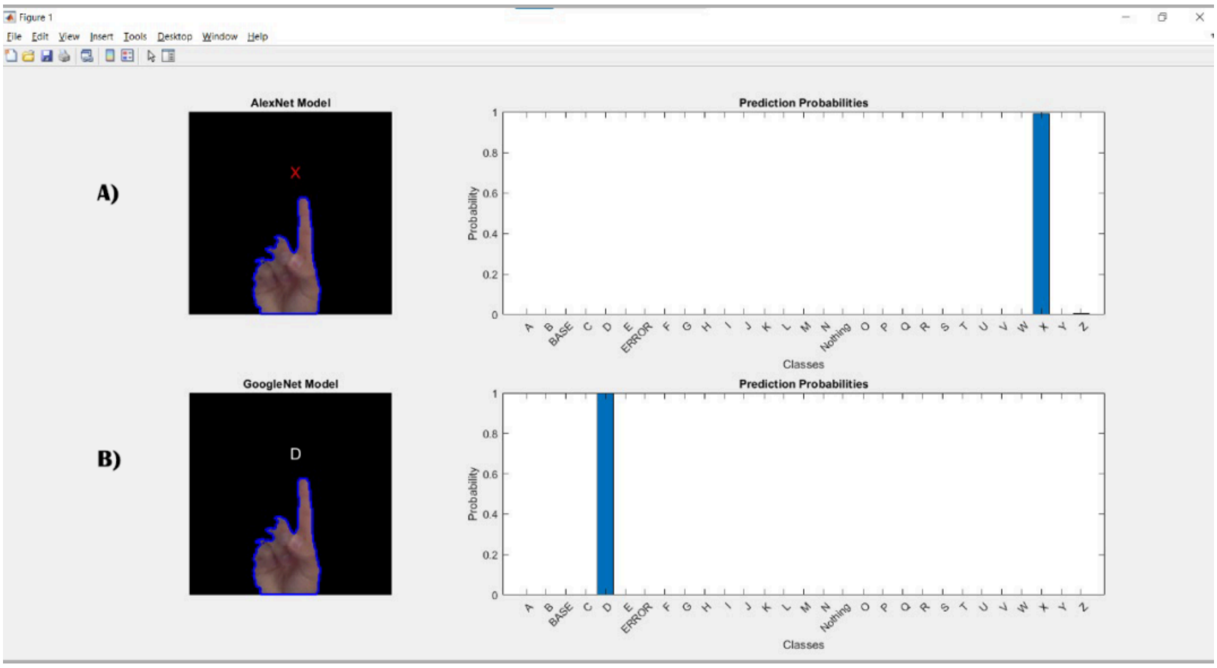


Figure 1
File Edit View Insert Tools Desktop Window Help
A)
AlexNet Model
X
Prediction Probabilities
Probability
1
0.8
0.6
0.4
0.2
0
A B BASE C D E ERROR F G H I J K L M N Nothing O P Q R S T U V W X Y Z
Classes
B)
GoogleNet Model
D
Prediction Probabilities
Probability
1
0.8
0.6
0.4
0.2
0
A B BASE C D E ERROR F G H I J K L M N Nothing O P Q R S T U V W X Y Z
Classes

Figure 57. *Comparison of prediction in Real Time between AlexNet and GoogleNet models.*

The use of image segmentation with AV turned out to be a double-edged sword because several tests were carried out. When using segmentation, the AlexNet network suffered certain inconsistencies in its data. Mainly when there is no hand to analyze. The network with the best performance, depending on the tests performed, was GoogleNet, mainly by detecting various patterns or features that confused the other model. However, some predictions of GoogleNet were not 100% accurate, which is thought to be due to the fact that the database of images was very small, and it is proposed to increase the number of images in the database to improve the model in the future.

Figure 57 shows the interpretation of the letter D, both from the GoogleNet and AlexNet networks. A bar chart was applied to detect the range of predictions and determine which prediction is the highest, thus delivering the respective prediction within the image capture in real-time.

4.4.2. Single Image.

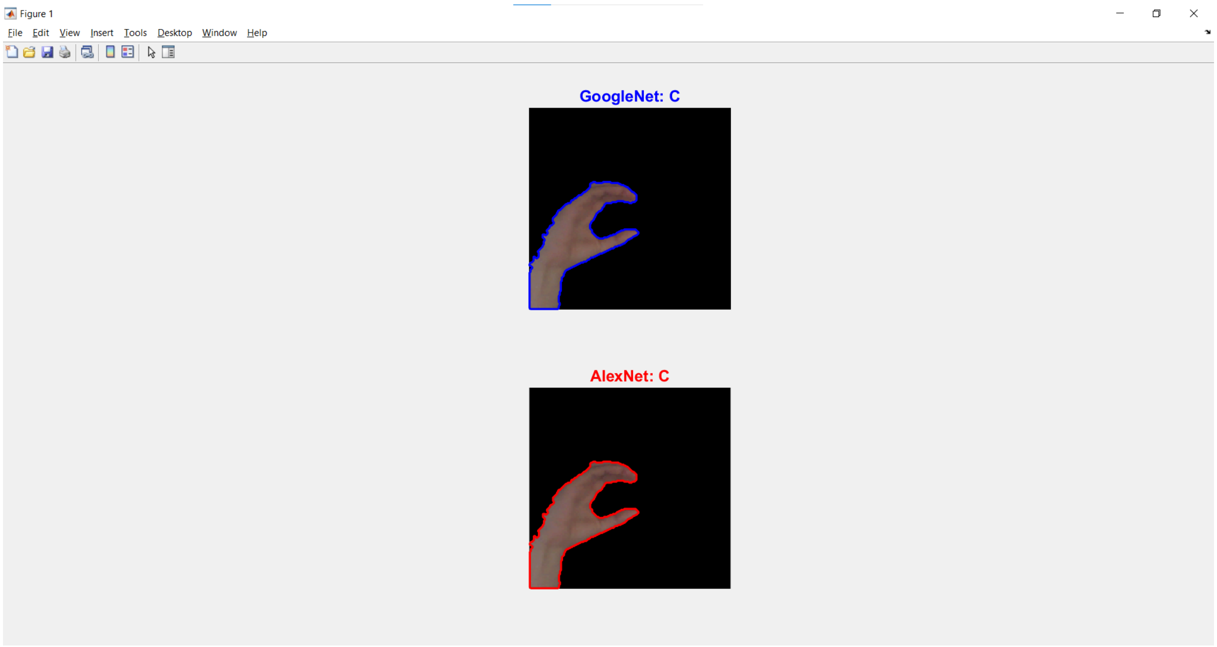


Figure 58. *Comparison of prediction on Single Image between AlexNet and GoogleNet models.*

The results obtained with the single image method showed favorable results. So, the pre-calibration methods and the adjustment of the hand in the time set by the user are important when performing the letters in their correct position, in addition to the established image calibration method presented in Figure 58. However, making a significant increase in the error that can be used in the binarization layer can cause an incorrect prediction if the error is increased too much, which confuses the networks and gives an erroneous result.

4.5. DISCUSSION

This section will cover the analysis of the integration work and explore and compare the model developed previously with similar models based on the literature.

For the human hand database in our curriculum integration work began with the creation of a human hand database which was used to train a model prior to the final model, the images contained in the database consist of 400 images for each class and are stored in 29 classes, where 26 classes correspond to the letters of the American alphabet and 3 special classes focused to analyze classes also used RGB images without background while in another model [201], it uses a data set of 300 images distributed 15 images for each sign language class plus this neural network was trained for 6 different sign language classes resulting in an accuracy of 92. 3%. Finally, in another model [122], they used grayscale images in which data images for 26 signs of 3 different people each person contains 120 images for each class so that there are 9360 images in total, giving an accuracy of 94. 57%.

Similarly, there are previous works that used models of robotic arms for the interpretation of ASL in this project [203] created a data set for the different signs of the alphabet for the robotic arm to interpret the sign language trained the model by the images taken so that the arm performs the movement, another work [182], uses a code written in C++ with libraries of Arduino this model is called TATUM, the robotic arm is made by 3D printing are moved by servomotors, in addition, it can perform 26 letters of ASL this model is designed for people with hearing and vision impairment to place their hand on the robot to facilitate communication of deaf and blind people compared to our model which is made by 3D printing can only interpret 15 different signs of ASL which are made by the movement of the servomotors with a code programmed in Arduino.

In the present study on ASL gesture and image recognition, a robotic arm was designed and assembled to interpret ASL at a cost of $163.67. The system, using GoogleNet and AlexNet models, achieved accuracy rates of 99.27% and 99.17%, respectively. As presented in Table 28, we can note that several studies focus on sign language interpretation by a robotic arm or detection by a neural network. The sign languages studied range from ASL to Bengali Sign Language (BdSL), Persian (PSL), Italian (ISL), Kazakh (KSL), Malay (MSL) and specific medical signs. This diversity highlights the importance of adapting sign-interpreting technology to different cultural and linguistic contexts. When compared to other types of sign languages, such as Bengali (BdSL) and Kasajo Sign Language (KSL), which show high accuracy but use more computational resources. When comparing our system performance with previous studies where several pre-trained neural network models such as DenseNet 201, ResNet 50, DarkNet 53, etc. are employed, giving a percentage of equal or lower 98.19% in contrast to our model in which we use AlexNet and GoogleNet whose accuracy rate is higher than 99% employing as optimizers Sgdm and Adam respectively with data preprocessing techniques such as data resizing and denoising.

In addition, a key aspect of our study is the significantly lower cost compared to other research. Most studies show costs between $280 to $10,000. For example, the humanoid robot RASA, which cost approximately $ 10,000 with an initial accuracy of 77%, required additional adjustments to reach 100%. At the same time, our model highlights an affordable alternative due to its cost of $163.67. In conclusion, our research provides a cost-effective ASL recognition system highlighted by its accuracy economy and simplicity in data processing, making it a promising option for low-cost and affordable applications.

Table 28. *Comparison of other studies with our model.*

Language	Model	Robotic hand	Price	Dataset	Preprocessing	Techniques to avoid overfitting	Optimizers	Accuracy
Bangla sign language (BdSL) and Medical signs interpretation of ASL[175].	ASL: Inception V3 BdSL: DenseNet 201, ResNet 50 y DarkNet 53	N/A	$ 800	ASL: 166 videos(10 kinds of medical signs) BdSL: Dataset 1: 950 images of 27 BdSL signs). Dataset 2: 12,581 images of 38 signs.	Image resizing and normalization Feature extraction	Validation Holdout Dropout Pruning de pesos	Adam: Learning rate of 0.0001 and 20 consecutive iterations. Stochastic Gradient Descent: 0.0001 learning rate and 50 epochs.	ASL: 87.50% BdSL: Dataset 1: 98.19% Dataset 2: 93.8%
American Sign Language (ALS) [178]	Multiclass-SVM classifier and N-Dimensional DTW (ND-DTW)	Ada robotic hand	N/A	10 digits in ASL for static postures and 10 self-defined dynamic gestures	Extraction of velocity and hand position features; data segmentation	Probability filter	SVM parameters: nu = 0.05, C = 1; ND-DTW with radius adjusted and smoothing factor	98.25% in static postures (SVM) and 95.5% in dynamic gestures (ND-DTW).
Persian Sign	---------	A humanoid robot	Approx $ 10000	70 standard PSL signs, including 60 Persian	N/A	N/A	N/A	77% in the first test phase but

Language (PSL) [183]		called RASA		words and 10 signs of the Baghcheban alphabet				improved to 100%.
Italian Sign Language (ISL) [184]	-----------	Robotic hand with 8 degrees of freedom and wrist with 3 degrees of freedom	$280	15 letters of the manual alphabet of Italian Sign Language (LIS)	N/A	N/A	N/A	90% recognition rate in LIS manual alphabet gestures
Kazakh sign language [185]	ResNet-18	Inmoov humanoid robot design	N/A	8.000 Different gestures	Image Normalización	N/A	N/A	>98%
American sign language [186]	It is based on a Raspberry Pi 4 programmed in Python	N/A	N/A	26 letters from the ASL alphabet, with real vs generated gesture comparisons	N/A	N/A	N/A	90%

	that converts voice commands into gestures.							
Malaysian Sign Language (MSL) [187]	Speech-to-text processing using Microsoft Visual Studio in C	SignBot	N/A	Alphabet, numbers, and emergency phrases in MSL.	Speech-to-text translation	N/A	N/A	93%
Our model: American sign languages (ASL)	Google Net AlexNet	Inmoov	$ 159.67	Alphabet letters and the other two classes.	Image resizing, image filtering, and background removal.	Data augmentation, dropout, frequency validation.	AlexNet: Sgdm GoogleNet: Adam	AlexNet:99.27% GoogleNet: 99.17%

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1. CONCLUSIONS

The use of neural networks and a robotic arm for the detection and interpretation of ASL was a project that had several details to take into account. Mainly the software limitations and multiple problems we had in the assembly of each of the pieces of the robotic arm. However, its design made it viable since it was an Open Source model, which, despite not presenting characteristics of large movements, could lay the foundations of our project. The coding was first done on a laptop with simple features, then moved to a powerful laptop with integrated graphics RTX 3050 Ti to speed up the process, perform more tests, and get results in the shortest possible time. However, we not only worked with the detection of the robotic arm but also extended this idea and used detection in a human hand, providing the option to expand these analyses to different languages and sectors.

All this began with the idea of highlighting the importance of the impact that deaf people have today, expanding awareness in the educational sector, and providing a way for young people to interact better with the deaf community.

As main points, we opted for a five-step idea: the creation and assembly of the robotic arm, the generation of the Database, detection coding, movement of the arm, and final assembly, which is characterized by joining all previous steps and testing.

The material chosen for the arm was PLA due to its mechanical properties, which are characterized by high malleability at high temperatures and significant strength. However, if the focus had been on a prosthesis or something that would have to be subjected to loads, we would have opted for ABS. The databases were made with a model of the human hand and a model of the robotic arm, respectively, for 16 and 29 classes. It is important to highlight that the images were obtained with a black background and with 3 cameras in different positions to capture the most important details and thus provide more variety for training.

For the detection of ASL, models such as AlexNet and GoogleNet were used since these models share important characteristics, such as the extraction of important characteristics from the Database. AlexNet (99.27 %) is simpler and faster to process, while GoogleNet (99.17 %) provides exceptional performance, which could be seen visually using Grad-Cam.For the arm's movement, 180° servomotors with a torque force of 12 kg were used to exert the necessary force for flexion and extension. These were coded using ArduinoIDE, which allowed us to easily perform the movement through PWM.

The final assembly was an interesting task since we had to opt for a 5 V to 10 A power supply so that all the servomotors could work efficiently. If we used only the connections that the Arduino had, the amount of amperage was inefficient. On the other hand, for the final detection tests, we created a "Controlled Environment," which consists of a black box in which unnecessary pixels can be avoided. In case of any anomaly, these will be eliminated by code. Finally, for the final visualization, we proceeded to perform two methods: the first is the detection in real-time, and the other is through the detection with image capture.

To conclude, the results obtained show that our model is superior to others in sign language detection; in addition, it presents a model of a robotic arm of lower cost with an approximate value of \$ 164. However, the compared models presented a wide variety of methods with which they varied their detection. In our case, it is necessary to use a "controlled environment" to obtain favorable results, as this is the main limitation of our project.

5.2. FUTURE WORKS

For future work, it is planned to apply this design in education, as the machine vision system can help the education system to teach sign language to hearing-impaired people. Additionally, it is planned to extend the system's scalability to support other sign languages, which would increase its applicability and facilitate the learning of sign language for the hearing impaired. Furthermore, this system could be integrated with specialized robots for teaching foreign languages, making sign language education more accessible.

5.3. LIMITATIONS

The main limitations we presented were due to hardware and software. First, not having found a model with greater mobility, we were limited in the emulation of sign language, so we only had 13 main classes focused on ASL letters. On the other hand, when applying image processing based on Otsu binarization, we obtained inconsistencies when removing the background of the images. We only had the regions of interest, so we opted for a manual configuration model, which is limited by time. Also, the elastic that we use to generate the movement in conjunction with the servo motors wears out over time. Therefore, for future experiments, it is necessary to evaluate different models that comply with the characteristics of good elasticity over a long time.

Finally, the database model can be extended by obtaining a larger number of images and adding more variety to the people from which the images are obtained, thus increasing the detection accuracy of the neural networks.

REFERENCE

[1] World Health Organization. (2024). Deafness and hearing loss. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[2] Abou-Abdallah, M., & Lamyman, A. (2021). Exploring communication difficulties with deaf patients. *Clinical Medicine*, *21*(4), https://doi.org/10.7861/clinmed.2021-0111

[3] Naranjo-Zeledón, L., et. al (2019). A Systematic Mapping of Translation-Enabling Technologies for Sign Languages. Electronics, 8(9), 1047. https://doi.org/10.3390/electronics8091047

[4] World Health Organization. (2021). World report on hearing. World Health Organization.

[5] NIDCD. (2024). Quick Statistics About Hearing, Balance, & Dizziness. National Institute on Deafness and Other Communication Disorders. https://n9.cl/hslnpo

[6] Santos, A. & Protes, A. (2019). Perceptions of deaf subjects about communication in Primary Health Care-*. Rev Lat Am Enfermagem, 27, 31-27. 10.1590/1518-8345.2612.3127

[7] Fellinger, J. et al. (2005). Mental distress and quality of life in a deaf population. Social Psychiatry and Psychiatric Epidemiology, 40, 737-742. https://doi.org/10.1007/s00127-005-0936-8

[8] Shin, J., et al. (2021). American Sign Language Alphabet Recognition by Extracting Feature from Hand Pose Estimation. Sensors. 21(17). 10.3390/s21175856

[9] Instituto Nacional para Sordos (2023). Retrieved from: https://n9.cl/ph1jh

[10] Fang, W., et al. (2019). Computer vision applications in construction safety assurance.Automation in Construction, 110. https://doi.org/10.1016/j.autcon.2019.103013

[11] Krenker, A., Bešter, J., & Kos, A. (2011). Introduction to artificial neural networks. Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech, 1-18.DOI: 10.5772/15751

[12] Keating, S. J. (2012). Renaissance Robotics: Novel applications of Multipurpose Robotic Arms spanning Design Fabrication, Utility, and Art. Massachusetts Institute of Technology.http://hdl.handle.net/1721.1/78184

[13] Krogh, A. (2008). What are artificial neural networks?. *Nature Biotechnology,* 26(2), 195–197. https://doi.org/10.1038/nbt1386

[14] Oxford Dictionaries. (2010). "Robot". Oxford Dictionaries. https://n9.cl/tf1vz

[15] Almeida, G. (2009). Unidad I: Fundamentos generales de la robótica.https://n9.cl/lz1r8

[16] Surati, S., Hedaoo, S., Rotti, T., Ahuja, V., & Patel, N. (2021). Pick and place robotic arm: a review paper. Int. Res. J. Eng. Technol, 8(2), 2121-2129.https://n9.cl/rty8a

[17] Rosales, E. M., & Gan, Q. (2004). Forward and inverse kinematics models for a 5-dof pioneer 2 robot arm. *University of Essex, UK, Tech. Rep. CSM-413*.https://n9.cl/xj66mw

[18] Kawasaki, H. (2021). Robot Kinematics and Dynamics. Human‐Robot Interaction Control Using Reinforcement Learning.https://doi.org/10.1002/9781119782773.app1

[19] Serna C., L., Rodríguez de S., A., & Albán A., F. (2003). Ácido Poliláctico (PLA): Propiedades y Aplicaciones. Ingeniería y Competitividad, 5(1), 16. https://doi.org/10.25100/iyc.v5i1.2301

[20] Arockiam, A. et al. (2022). A review on PLA with different fillers used as a filament in 3D printing. Materials Today: Proceedings, 50 , 2057-2064.https://doi.org/10.1016/j.matpr.2021.09.413

[21] Kristiawan, R., et al. (2021). A review on the fused deposition modeling (FDM) 3D printing: Filament processing, materials, and printing parameters. *Open Engineering*, *11*(1), 639-649. https://doi.org/10.1515/eng-2021-0063

[22] Wojtyła, S., Klama, P., & Baran, T. (2017). Is 3D printing safe? Analysis of the thermal treatment of thermoplastics: ABS, PLA, PET, and nylon. *Journal of Occupational and Environmental Hygiene*, *14*(6), D80–D85. https://doi.org/10.1080/15459624.2017.1285489

[23] Meinsa. (n.d). Historia de los sistemas de control industrial. https://n9.cl/hr3gqz

[24] Kurdila, A. J., & Ben-Tzvi, P. (2019). Dynamics and control of robotic systems.https://n9.cl/enxcdi

[25] Joy, A. (2023). What Is An Actuator-Types and Applications. Tameson. https://n9.cl/rk0yr3

[26] Saha, S. K. (2000). *Introducción a la Robótica*. McGraw-Hill España.https://n9.cl/w6pxt

[27] Unox. (n.d). Pneumatic Actuators. https://n9.cl/cuo6tb

[28] Martin, S. (2021). Actuadores neumáticos Festo: una elección de primer nivel.EPIDOR Technical Distribution. https://n9.cl/hjqlp

[29] Emerson (n.d). Hydraulic Actuators. https://n9.cl/jkokt4

[30] Cowan Dynamics. (n.d). Actuadores hidráulicos.Cowandynamics. https://n9.cl/sxvng

[31] Oriental Motor. (n.d).Rack and Pinion System L Series - AZ Series Equipped. Oriental Motor. https://n9.cl/22gb4

[32] Redekar, A., Deb, D., & Ozana, S. (2022). Functionality analysis of electric actuators in renewable energy systems—a review. Sensors, 22(11), 4273. doi: 10.3390/s22114273.

[33] Industrial Quick Search (2024).Electrical Actuators. Editorial by Industrial Quick Search. https://n9.cl/2icfrs

[34] TME. (2020). Motor paso a paso – tipos y ejemplos del uso de motores paso a paso. TME Electronic Component. https://n9.cl/0u46y

[35] ISL. (n.d).Stepper motor fundamentals.ISL.https://n9.cl/amnp1

[36] Fiore, C. (2020). Stepper Motors Basics: Types, Uses, and Working Principles.MPS. https://n9.cl/3yqy9

[37] OrientalMotors (n.d). Structure of Stepper Motors. https://n9.cl/uuxk7

[38] Solectro. (2020). Introducción a los servomotores. https://n9.cl/3meqg

[39] Gao, D. et. al (2024). An Intelligent Control Method for Servo Motor Based on Reinforcement Learning. Algorithms , 17(1),14. https://doi.org/10.3390/a17010014

[40] Fraile García, J.C. (2021). Introducción al control remoto de servomotores industriales. (Trabajo Fin de Máster Inédito). Universidad de Sevilla, Sevilla. https://idus.us.es/handle/11441/127920

[41] Dejan. (n.d). How to Control Servo Motors with Arduino – Complete Guide.Arduino Tutorials, How It Works
https://n9.cl/14xqy

[42] Festo. (2024). Músculo neumático, DMSP-5-50N-RM-CM, 28/10/24. https://n9.cl/0ogxb

[43] Festo. (2024). Músculo neumático, DMSP-40-120N-RM-CM, 28/10/24. https://n9.cl/c2hg8

[44] Festo. (2024). Músculo neumático, DMSP-20-100N-RM-CM, 28/10/24. https://n9.cl/836yf

[45] Emerson. (2024). Bettis G-Series Hydraulic Valve Actuator. Data Sheets: G-Series Hydraulic Dimensions Data Metric, Bettis-EN. https://n9.cl/qtxyfe

[46] Emerson. (2024). Bettis GVO Linear Pneumatic Valve Actuator. Brochure: Bettis Actuation Solutions Portfolio. https://n9.cl/3murt

[47] Oriental Motor. (2024). LM2B-AZ Horizontal Rack and Pinion Systems (AC Input). LM2B500AZAC-1. https://n9.cl/60bfl9

[48] Oriental Motor. (2024). LM2F-AZ Vertical Rack and Pinion Systems (AC Input). LM2F500AZAC-1 https://n9.cl/zg1t3

[49] ElectroStore. (2024). MOTOR A PASOS POLOLU PAP UNIPOLAR / BIPOLAR 5.7V, 1 A NEMA 23 POLOLU. https://n9.cl/yyjsj8

[50] ElectroStore. (2024). MOTOR A PASOS PAP BIPOLAR 200 PASOS 7.4V, 0.28 A NEMA 14 POLOLU. https://n9.cl/t77ama

[51] ElectroStore. (2024). SERVOMOTOR TOWER PRO MG996R 13 KG.CM STANDARD 180°.https://n9.cl/1zpt6

[52] ElectroStore. (2024). SERVOMOTOR TOWER PRO MG995 11 KG.CM STANDARD 360°. https://n9.cl/fyg2v

[53] Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades. (2022). Gob. ec. https://n9.cl/r0f9

[54] Roizen, N. J. (2003). Nongenetic causes of hearing loss. *Mental Retardation and Developmental Disabilities Research Reviews*, *9*(2), 120–127. https://doi.org/10.1002/mrdd.10068

[55] Department of Health (DOH). (2023). *Hearing disability assessment: report of the Expert Hearing Group*. Lenus.Ie. Retrieved December 13, 2023, from https://n9.cl/voswku

[56] O'Connell, N. (2023). Problematising the problem: Exploring how hearing privilege fosters employment inequality for deaf people. *International Journal of Disability and Social Justice*, *3*(2). https://doi.org/10.13169/intljofdissocjus.3.2.0071

[57] Wadhawan, A., & Kumar, P. (2019). Sign Language Recognition Systems: A Decade Systematic Literature Review. Archives of Computational Methods in Engineering. doi:10.1007/s11831-019-09384-2

[58] Agrawal, A., Kundalia, H., et al. (2020). Talking Fingers-Sign Language Translator. International Journal for Research in Engineering Application & Management (IJREAM). doi: 10.35291/2454-9150.2020.0250

[59] Atkinson, J., et. al (2016). Synesthesia for manual alphabet letters and numeral signs in second-language users of signed languages. Neurocase.22(4),1-8. 10.1080/13554794.2016.1198489

[60] Kuruvilla, J., Sukumaran, D., et al. (2016). A review on image processing and image segmentation. 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE).10.1109/sapience.2016.7684170

[61] The MathWorks Inc. (2023). Medfilt2, Natick, Massachusetts: The MathWorks Inc. https://la.mathworks.com/help/images/ref/medfilt2.html

[62] The MathWorks Inc. (2023). imopen, Natick, Massachusetts: The MathWorks Inc. https://la.mathworks.com/help/images/ref/imopen.html?s_tid=doc_ta

[63] The MathWorks Inc. (2023). Image Processing Toolbox version: 11.7 (R2023a), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com

[64] The MathWorks Inc. (2023). Computer Vision Toolbox version: 10.4 (R2023a), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com

[65] Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[66] Mukhamediev, R., et. al. (2021). From classical machine learning to deep neural networks: A simplified scientometric review. *Applied Sciences*, *11*(12), 5541.

[67] Brady, M. (1985). Artificial intelligence and robotics. Artificial Intelligence, 26(1), 79–121. doi:10.1016/0004-3702(85)90013-x

[68] Chowdhary, K. R. (2020). Fundamentals of Artificial Intelligence. doi:10.1007/978-81-322-3972-7

[69] Nikolaos, P. et al. (2011).Industrial applications with cooperating robots for the flexible assembly, International Journal of Computer Integrated Manufacturing, 24:7, 650-660, DOI: 10.1080/0951192X.2011.570790

[70] Villavicencio, H. (2005). Technology of future: Da Vinci robotic surgery. Actas Urológicas Españolas, 29(10), 919-921. https://n9.cl/r5lg4w

[71] Chopra, A., Prashar, A., & Sain, C. (2013). Natural language processing. *International journal of technology enhancements and emerging engineering research*, *1*(4), 131-134.

[72] Liddy, E.D. (2001). Natural Language Processing. In Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc. https://n9.cl/jopmv

[73] Hirschberg, J., & Manning, C. D. (2015). *Advances in natural language processing. Science, 349(6245), 261–266.* doi:10.1126/science.aaa8685

[74] Zong, Z., & Hong, C. (2018). On Application of Natural Language Processing in Machine Translation. 2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE). doi:10.1109/icmcce.2018.00112

[75] Wang, D., Su, J., & Yu, H. (2020). Feature Extraction and Analysis of Natural Language Processing for Deep Learning English Language. IEEE Access, 1–1. doi:10.1109/access.2020.2974101

[76] Hirschan, L., & Gaizauskas, R. (2001). Natural language question answering: the view from here. Natural Language Engineering, 7(04). doi:10.1017/s1351324901002807

[77] Simmons, R. F. (1970). Natural language question-answering systems: 1969. Communications of the ACM, 13(1), 15–30. doi:10.1145/361953.361963

[78] Voorhees, E. M. (1999). Natural language processing and information retrieval. In *International summer school on information extraction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 32-48. https://doi.org/10.1007/3-540-48089-7_3

[79] Brants, T. (2003). Natural Language Processing in Information Retrieval. *CLIN*, *111*, 1-13.https://n9.cl/465qs

[80] Strzalkowski, T. (Ed.). (1999). *Natural language information retrieval.* Springer Science & Business Media, 7. https://doi.org/10.1007/978-94-017-2388-6

[81] Rajput, A. (2020). Natural language processing, sentiment analysis, and clinical analytics. In *Innovation in health informatics.* Academic Press, 79-97. https://doi.org/10.1016/B978-0-12-819043-2.00003-4

[82] Nasukawa, T., & Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, 70-77.DOI:10.1145/945645.945658

[83] Badler, N., et. al (1997). *Natural language text generation from Task networks*. Technical Report, CIS, University of Pennsylvania, Philadelphia, USA. https://n9.cl/b97jf

[84] Iqbal, T., & Qureshi, S. (2020). The survey: Text generation models in deep learning. Journal of King Saud University - Computer and Information Sciences. doi:10.1016/j.jksuci.2020.04.001

[85] Albalawi, R., Yeap, T. H., & Benyoucef, M. (2020). Using topic modeling methods for short-text data: A comparative analysis. *Frontiers in artificial intelligence*, *3*, 42.DOI:10.3389/frai.2020.00042

[86] Jelodar, H., Wang, Y., Rabbani, M., & Ayobi, S. (2019). Natural language processing via LDA topic model in recommendation systems. *arXiv preprint arXiv:1909.09551*.

[87] Kastner, J. K., & Hong, S. J. (1984). A review of expert systems. European Journal of Operational Research, 18(3), 285–292. doi:10.1016/0377-2217(84)90150-4

[88] Gupta, I., & Nagpal, G. (2020). *Artificial intelligence and expert systems*. Mercury Learning and Information. https://n9.cl/hwd1c

[89] Gollapudi, S. (2019). Artificial intelligence and computer vision. *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*, 1-29.https://doi.org/10.1007/978-1-4842-4261-2

[90] Li, X., & Shi, Y. (2018). Computer vision imaging based on artificial intelligence. In *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*. IEEE, 22-25. doi: 10.1109/ICVRIS.2018.00014.

[91] R. Saravanan & P. Sujatha. (2018). "A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification,".Second International Conference on Intelligent Computing and Control Systems (ICICCS). (8),945-949, doi: 10.1109/ICCONS.2018.8663155.

[92] Cunningham, P., Cord, M., Delany, S.J. (2008). Supervised Learning. In.Machine Learning Techniques for Multimedia. Cognitive Technologies. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-75171-7_2

[93] Ghahramani, Z. (2004). Unsupervised Learning. Lecture Notes in Computer Science, 72–112. doi:10.1007/978-3-540-28650-9_5

[94] Dike, H. U., Zhou, Y., Deveerasetty, K. K., & Wu, Q. (2018). Unsupervised Learning Based On Artificial Neural Network: A Review. 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS). doi:10.1109/cbs.2018.8612259

[95] Berry, M. W., Mohamed, A., & Yap, B. W. (Eds.). (2020). Supervised and Unsupervised Learning for Data Science. Unsupervised and Semi-Supervised Learning. doi:10.1007/978-3-030-22475-2

[96] Hady, M. F. A., & Schwenker, F. (2013). Semi-supervised Learning. Handbook on Neural Information Processing, 215–239. doi:10.1007/978-3-642-36657-4_7

[97] Van Engelen, J. E., & Hoos, H. H. (2019). A survey on semi-supervised learning. Machine Learning. doi:10.1007/s10994-019-05855-6

[98] Sutton, R. S. (1992). Introduction: The Challenge of Reinforcement Learning. Reinforcement Learning, 1–3. doi:10.1007/978-1-4615-3618-5_1

[99] Kaelbling, L., et. al (1996). Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research. 4(1996). https://doi.org/10.1613/jair.301

[100] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An Introduction to Deep Reinforcement Learning. Foundations and Trends® in Machine Learning, 11(3-4), 219–354. doi:10.1561/2200000071

[101] Ketkar, N., & Moolayil, J. (2021). Deep Learning with Python. doi:10.1007/978-1-4842-5364-9

[102] Baldi, P., & Vershynin, R. (2019). The Capacity of feedforward neural networks. Neural Networks. doi:10.1016/j.neunet.2019.04.009

[103] Huang, K., Wang, Y., et al. (2020). Why Do Deep Residual Networks Generalize Better than Deep Feedforward Networks?---A Neural Tangent Kernel Perspective. *Advances in neural information processing systems*, *33*, 2698-2709.https://doi.org/10.48550/arXiv.2002.06262

[104] Gulcehre, C., Cho, K.,et al. (2014). Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks. Lecture Notes in Computer Science, 530–546. doi:10.1007/978-3-662-44848-9_34

[105] Grossberg, S. (2013). Recurrent neural networks. *Scholarpedia*, *8*(2), 1888. doi:10.4249/scholarpedia.1888

[106] Medsker, L., & Jain, L. C. (1999). *Recurrent neural networks: design and applications*. CRC press. https://n9.cl/muvtvk

[107] Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 2673–2681. doi:10.1109/78.650093

[108] Nketiah, E. A., Chenlong, L., et al. (2023). Recurrent neural network modeling of multivariate time series and its application in temperature forecasting. *Plos one*, *18*(5), https://doi.org/10.1371/journal.pone.0285713

[109] Alam, M., et.al.(2023) "A Review of Recurrent Neural Network Based Camera Localization for Indoor Environments, (11) , 43985-44009.doi: 10.1109/ACCESS.2023.3272479.,

[110] Hayale, W., Negi, P. S., & Mahoor, M. H. (2021). Deep siamese neural networks for facial expression recognition in the wild. *IEEE Transactions on Affective Computing*, *14*(2), 1148-1158.https://doi.org/10.1109/TAFFC.2021.3077248

[111] Putra, A. A. R., & Setumin, S. (2021). The performance of Siamese neural network for face recognition using different activation functions. *International Conference of Technology, Science and Administration (ICTSA)*, 1-5.doi: 10.1109/ICTSA52017.2021.9406549.

[112] Ilina, O., Ziyadinov, V., et al. (2022). A survey on symmetrical neural network architectures and applications. *Symmetry*, *14*(7),1391.https://doi.org/10.3390/sym14071391

[113] Zhang, C., Liu, W., et al. (2016). Siamese neural network based gait recognition for human identification.*ieee international conference on acoustics, speech and signal processing (ICASSP)*,2832-2836.doi: 10.1109/ICASSP.2016.7472194.

[114] Mishra, M. (2020). Convolutional neural networks, explained. Towards Data Science. https://n9.cl/8l6pqk

[115] Yamashita, R. et al (2018). Convolutional neural networks: an overview and application in radiology. Insights into Imaging, 9(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

[116] G. Huang, Z. Liu, L. Van Der Maaten & K. Q. Weinberger (2017)."Densely Connected Convolutional Networks,".IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2261-2269, doi: 10.1109/CVPR.2017.243.

[117] Kurama, V. (2020). A guide to ResNet, Inception v3, and SqueezeNet. Paperspace Blog. https://n9.cl/oyyiy

[118] Iandola, F. et al (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size.https://n9.cl/u0mru

[119] Szegedy, C., et al (2014). Going deeper with convolutions. Arxiv.org. https://n9.cl/o9i4n

[120] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, faster, stronger. In arXiv [cs.CV]. https://n9.cl/g1jbe

[121] Szegedy, C., et al. (2015). Going deeper with convolutions. Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2015.7298594

[122] Pawara, P., et, al. (2017). Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition. Proceedings of the 6th

International Conference on Pattern Recognition Applications and Methods.DOI:10.5220/0006196204790486

[123] Alom, M., et. al. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *DOI:10.48550/arXiv.1803.01164*

[124] Adam, K., Mohd, I. I., & Younis, Y. M. (2021). The impact of the soft errors in convolutional neural network on GPUs: Alexnet as case study. Procedia Computer Science, 182, 89–94. https://doi.org/10.1016/j.procs.2021.02.012

[125] Saggie. (2017). What is Object Detection?. Saggie. https://n9.cl/0b3a1

[126] Alqushaibi, A., Abdulkadir, et al. (2020). *A Review of Weight Optimization Techniques in Recurrent Neural Networks. 2020 International Conference on Computational Intelligence (ICCI).* doi:10.1109/icci51257.2020.9247757

[127] Desai, C. (2020). Comparative analysis of optimizers in deep neural networks. *International Journal of Innovative Science and Research Technology*, *5*(10), 959-962.https://n9.cl/dssm55

[128] Ying, X. (2019, February). An overview of overfitting and its solutions. In *Journal of physics: Conference series* (Vol. 1168, p. 022022). IOP Publishing.DOI 10.1088/1742-6596/1168/2/022022

[129] Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.https://n9.cl/616slf

[130] Berrar, D. (2019). Cross-validation.DOI:10.1016/B978-0-12-809633-8.20349-X

[131] Singh, S., & Krishnan, S. (2020). Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11237-11246).https://doi.org/10.48550/arXiv.1911.09737

[132] Chun, W. (2001). *Programación central de Python* (1). Profesional de Prentice Hall.https://n9.cl/6gfaq2

[133] Van Rossum, G. (2007). Lenguaje de programación Python. En *la conferencia técnica anual de USENIX* (41), 1-36.https://n9.cl/i9bae

[134] Sharma, A., Khan, F., et al. (2020). Python: el lenguaje de programación del futuro. *En t. J. Res innovadora. Technol* , *6* (2), 115-118.

[135] Sarkar, D., Bali, R., et al. (2018). The Python machine learning ecosystem. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*, 67-118.

[136] Oliva, A. (2020).Ecosistema Python. MAT281 - Aplicaciones de la Matemática en la Ingeniería – 2020. https://n9.cl/9zctb

[137] Sakai, A., Ingram, D., et al. (2018). Pythonrobotics: a python code collection of robotics algorithms. *arXiv preprint arXiv:1808.10703*.

[138] Avelar, E., Castillo, O., & Soria, J. (2020). Fuzzy logic controller with fuzzylab python library and the robot operating system for autonomous mobile robot navigation. *Journal of Automation Mobile Robotics and Intelligent Systems*, *14*(1), 48-54.

[139] Yumbla, F., Yumbla, E., & Moon, H. (2020l). The bioloid gp robot with different configurations for simulation in v-rep controlled by the robot operating system (ros). *6th International Conference on Control, Automation and Robotics (ICCAR)*, 54-58.

[140] Ardianto, D., & Widiyatmoko, A. T. (2024). Color Detector in an Image using Python and Computer Vision Library. *Journal of Intelligent Systems and Information Technology*, *1*(1), 25-30. https://doi.org/10.61971/jisit.v1i1.27

[141] Asaad, R., Ali, R., et al. (2023). Image Processing with Python Libraries. *Academic Journal of Nawroz University*, *12*(2), 410-416. https://doi.org/10.25007/ajnu.v12n2a1754

[142] Kalmutskyy Kalmutskyy, G. (2021). Simulación de un sistema de clasificación robotizado de propósito general utilizando técnicas de Deep-Learning y visión artificial en Python. http://hdl.handle.net/10835/13455

[143] Mínguez, T. (2021). *Visión artificial: aplicaciones prácticas con OpenCV-Python*. Marcombo. (1th Edition).Marcombo.https://n9.cl/61ten3

[144] Wang, S. C., & Wang, S. C. (2003). Artificial neural network. *Interdisciplinary computing in java programming*, 81-100.

[145] Dhruv, A. J., Patel, R., & Doshi, N. (2021). Python: the most advanced programming language for computer science applications. *Science and Technology Publications, Lda*, 292-299.

[146] Vaferi, K., et. al. (2023). Modelado y optimización del rendimiento hidráulico y térmico de una válvula tesla mediante método numérico y red neuronal artificial. *Entropía* , *25* (7), 967.

[147] Plaza, P., et, al (2018). Arduino is an educational tool that introduces robotics. *IEEE international conference on teaching, assessment, and learning for engineering (TALE),* 1-8.

[148] Peña, C. (2020). Arduino IDE: Domina la programación y controla la placa. RedUsers.

[149] Mistry., S & Pajak., D. (2024). Get Started With Machine Learning on Arduino. Arduino.

[150] Lajara, J., & Pelegrí, J. (2011). *LabView: entorno gráfico de programación*. Marcombo

[151] Kodosky, J. (2020). LabVIEW. *Proceedings of the ACM on Programming Languages*, *4*(HOPL), 1-54.https://doi.org/10.1145/3386328

[152] De Francisco Ortiz, Ó., Estrems Amestoy, M., & Carrero-Blanco, J. (2020). Positioning measurement using a new artificial vision algorithm in LabVIEW based on the analysis of images on an LCD screen. The International Journal of Advanced Manufacturing Technology, 109(1), 155-170.. doi:10.1007/s00170-020-05497-2

[153] Posada-Gómez, R.,et. al. (2011). Digital image processing using LabVIEW. *Practical Applications and Solutions Using LabVIEW Software, InTech*, 297-316. https://n9.cl/y8ewr

[154] Mezher, L. S. (2016). Digital image processing filtering with LABVIEW. *International Journal of Computer Science Trends and Technology (I JCS T)*, 278.

[155] NI. (n.d).LabVIEW Analytics and Machine Learning Toolkit. NI. https://n9.cl/phbx6

[156] Ramirez, J. M., Gomez-Gil, P., & Larios, F. L. (2007). A Robot-vision System for Autonomous Vehicle Navigation with Fuzzy-logic Control using Lab-View. Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007). doi:10.1109/cerma.2007.4367702

[157] DİRİK, M.(n.d). Anomaly detection using LabView-Based machine learning algorithms. International research in engineering sciences, 366. 10.5281/zenodo.7744436

[158] The MathWorks Inc. (2023). MATLAB, Natick, Massachusetts: The MathWorks Inc. https://la.mathworks.com/help/matlab/index.html.

[159] The MathWorks Inc. (2023). Deep Network Designer, Natick, Massachusetts: The MathWorks Inc. https://la.mathworks.com/help/deeplearning/ref/deepnetworkdesigner-app.html

[160] The MathWorks Inc. (2023). Statistics and Machine Learning Toolbox, Natick, Massachusetts: The MathWorks Inc.https://la.mathworks.com/help/stats/index.html

[161] The MathWorks Inc. (2023). Text Analytics Toolbox, Natick, Massachusetts: la.mathworks.com/help/textanalytics/index.html

[162] The MathWorks Inc. (2024). AI, Data Science, and Statistics — Apps: The MathWorks Inc. AI, Data Science, and Statistics — Apps (mathworks.com)

[163] The MathWorks Inc. (2024). Image Processing and Computer Vision : The MathWorks Inc. Image Processing and Computer Vision - MATLAB & Simulink (mathworks.com)

[164] The MathWorks Inc. (2023). Image Region Analyzer, Natick, Massachusetts: The MathWorks Inc. https://la.mathworks.com/help/images/ref/imageregionanalyzer-app.html

[165] The MathWorks Inc. (2023). Image Segmenter, Natick, Massachusetts. https://la.mathworks.com/help/images/ref/imagesegmenter-app.html

[166] The MathWorks Inc. (2023). Create Semantic Segmentation using volume Segmenter, Natick, Massachusetts: https://n9.cl/x9lkp3

[167] The MathWorks Inc. (2023). Volume Segmenter, Natick, Massachusetts. https://la.mathworks.com/help/images/ref/volumesegmenter-app.html

[168] Pololu. (n.d). 1.b. Mini Maestro Pinout and Components. Pololu Robotics & Electronics. https://n9.cl/ushjx

[169] Pereira, V., Fernandes, V. A., & Sequeira, J. (2014). Low cost object sorting robotic arm using Raspberry Pi. *IEEE global humanitarian technology conference-South Asia Satellite (GHTC-SAS)*, 1-6.

[170] RaspBerry PI (n.d). Raspberry Pi 4. RaspberryPI. https://n9.cl/9fo7u

[171] Arduino Mega 2560 Rev3. (2024). Arduino Official Store. https://n9.cl/xmm38

[172] Song, T. L., Lu, Y. P., & Liu, H. Q. (2013). The Controlling Research of the 3D Bionic Snake-Like Robot Based on the Arduino. Applied Mechanics and Materials, 302, 570–573. doi:10.4028/www.scientific.net/amm.302.570

[173] Cheng, H. C., Chiu, M. C., et al. (2017). A design of toxic gas detecting security robot car based on wireless path-patrol. In *MATEC Web of Conferences*. . EDP Sciences, 123, 00035.

[174] Lim, K. M., et. al (2019). Isolated sign language recognition using Convolutional Neural Network hand modelling and Hand Energy Image. Multimedia Tools and Applications, 78(14), 19917–19944. doi:10.1007/s11042-019-7263-7

[175] Nihal, R. A., Broti, N. M., Deowan, S. A., & Rahman, S. (2021). Design and Development of a Humanoid Robot for Sign Language Interpretation. SN Computer Science, 2(3). doi:10.1007/s42979-021-00627-3

[176] Chavan, S., Yu, X., & Saniie, J. (2021). Convolutional Neural Network Hand Gesture Recognition for American Sign Language. 2021 IEEE International Conference on Electro Information Technology (EIT). doi:10.1109/eit51626.2021.9491897

[177] Rastgoo, R., Kiani, K., & Escalera, S. (2020). Hand sign language recognition using multi-view hand skeleton. Expert Systems with Applications, 113336. doi:10.1016/j.eswa.2020.113336 3

[178] Zhi, D., de Oliveira, T. et al. (2018). Teaching a Robot Sign Language using Vision-Based Hand Gesture Recognition. 2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA). doi:10.1109/civemsa.2018.8439952

[179] Cao Dong, M. C. Leu and Z. Yin. (2015). "American Sign Language alphabet recognition using Microsoft Kinect," IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, 44-52, doi: 10.1109/CVPRW.2015.7301347

[180] Barbhuiya, A. A., Karsh, R. K., & Jain, R. (2020). CNN based feature extraction and classification for sign language. Multimedia Tools and Applications, 80(2), 3051–3069. doi:10.1007/s11042-020-09829-y

[181] Johnson, S., et. al (2021). An Adaptive, Affordable, Open-Source Robotic Hand for Deaf and Deaf-Blind Communication Using Tactile American Sign Language. Conference: 43rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society. 4732-4737. doi: 10.1109/EMBC46164.2021.9629994.

[182] Mazhar, O., Navarro, B.,et al. (2019). A real-time human-robot interaction framework with robust background invariant hand gesture detection. Robotics and Computer-Integrated Manufacturing, 60, 34–48. doi:10.1016/j.rcim.2019.05.008

[183] Meghdari, A., Alemi, M., Zakipour, M., & Kashanian, S. A. (2018). Design and Realization of a Sign Language Educational Humanoid Robot. Journal of Intelligent & Robotic Systems. doi:10.1007/s10846-018-0860-2

[184] Bulgarelli, A., Toscana, G., et al. (2016). A low-cost open source 3D-printable dexterous anthropomorphic robotic hand with a parallel spherical joint wrist for sign languages reproduction. International Journal of Advanced Robotic Systems, 13(3), 126. doi:10.5772/64113

[185] Kenshimov, C., & et. al (2021). "Development of a Verbal Robot Hand Gesture Recognition System,". WSEAS Transactions on Systems and Control, (16), 573-583. DOI:10.37394/23203.2021.16.53

[186] Verma, Y., & Anand, R. S. (2017). Gesture Generation by the Robotic Hand for Aiding Speech and Hearing-Impaired Persons Based on American Sign Language. Available at SSRN 4608468

[187] Al-Khulaidi, R. A., Akmeliawati, R., et al. (2018). Development of robotic hands of signbot, advanced Malaysian sign-language performing robot. Advances in robotics research, 2(3), 183

[188] Islam, M. M., Siddiqua, S., & Afnan, J. (2017). Real time Hand Gesture Recognition using different algorithms based on American Sign Language. 2017 IEEE

International Conference on Imaging, Vision & Pattern Recognition (icIVPR). doi:10.1109/icivpr.2017.7890854

[189] Luo, R. C., & Wu, Y.-C. (2012). Hand gesture recognition for Human-Robot Interaction for service robot. 2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). doi:10.1109/mfi.2012.6343059

[190] Lea Plaza Chávez, R., & Gutiérrez Espada, H. (2016). Computing platform proposal for roads building management projects case study: building company "Serco S.R.L.". Journal Boliviano de Ciencias, 12, 30. https://doi.org/10.52428/20758944.v12i36.681

[191] Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The unified process. Ieee Software, 16(3), 96.

[192] S. Autsou, et. al (2024). Principles and Methods of Servomotor Control: Comparative Analysis and Applications. *Applied Sciences,* 14(6).2579. https://doi.org/10.3390/app14062579

[193] Baumann, F. W., & Roller, D. (2018). Thingiverse: review and analysis of available files. International Journal of Rapid Manufacturing, 7(1), 83. doi:10.1504/ijrapidm.2018.089731

[194] Larrañaga, A. (2023). Thor – An Open Source 3D Printable 6DOF Robotic Arm. https://n9.cl/gczif

[195] Gross, R. (2017). Humanoid Robotic Hand. Ultimaker thingiverse. https://n9.cl/txhv8

[196] Gross, R. (2016). Robotic Prosthetic Hand. Ultimaker Thingiverse. https://n9.cl/ab6zi

[197] Langevin, G. (2012) InMoov body parts library : Right-Hand. Inmoov: Open source 3D printed life size robot. https://www.thingiverse.com/thing:17773

[198] Riba Romeva, C. (2002). Diseño concurrente.DOI:10.5821/ebook-9788498800746

[199] Swetham, T., Reddy, K. et al. (2017). A Critical Review of 3D Printing Materials and Details of Materials used in FDM. *Int. J. Sci. Res. Sci. Eng. Technol*, *3*, 353-361.

[200] Roy, R., & Mukhopadhyay, A. (2021). Tribological studies of 3D printed ABS and PLA plastic parts. Materials Today: Proceedings, 41, 856–862. doi:10.1016/j.matpr.2020.09.235

[201] Munib, Q., et. al (2007). American sign language (ASL) recognition based on Hough transform and neural networks. Expert Systems with Applications, 32(1), 24–37. doi:10.1016/j.eswa.2005.11.018

[202] Islam, M., et. al (2018). Hand Gesture Feature Extraction Using Deep Convolutional Neural Network for Recognizing American Sign Language. 2018 4th International Conference on Frontiers of Signal Processing (ICFSP). doi:10.1109/icfsp.2018.8552044

[203] Kolli, Y., et. al (2023). ASL Detection and Gesture Based Control of Robotic Hand Using Image Processing.Research Square.https://doi.org/10.21203/rs.3.rs-2897029/v1

ATTACHMENTS

We will now include a series of attachments containing figure of the 3D printing process of the robotic arm, datasheets, and codes used for this work.

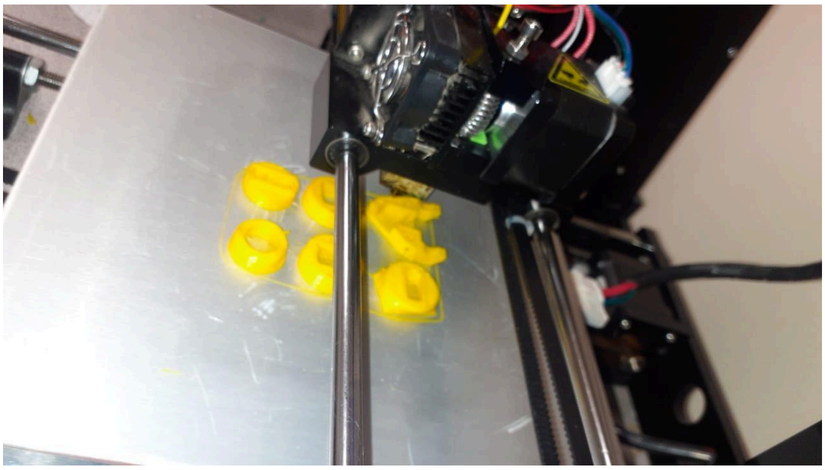


Figure 59. 3D printing of the robotic arm thumb.

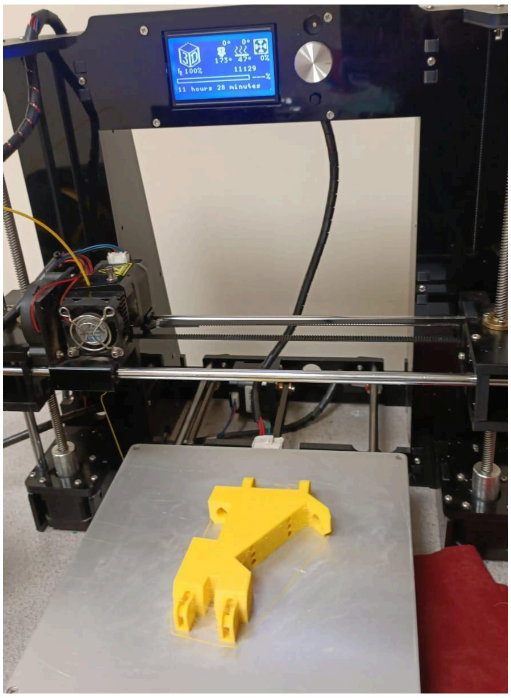


Figure 60. 3D printing of a part of the robotic hand.

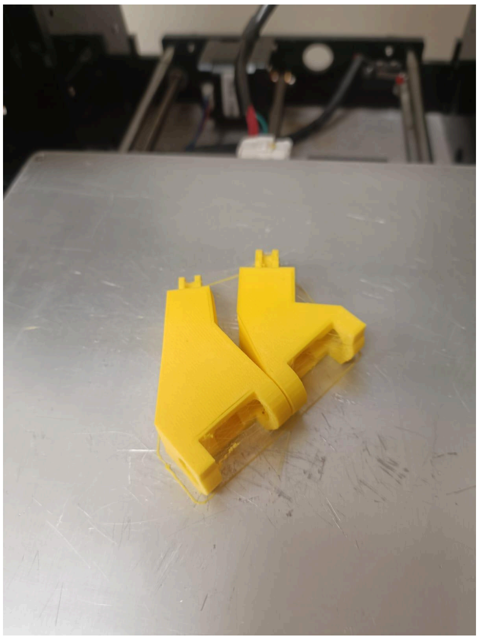


Figure 61. 3D printing of a part of the robotic hand.

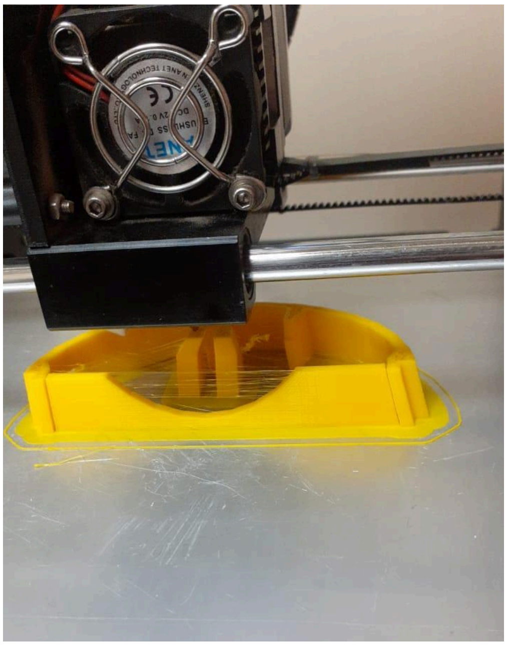


Figure 62. 3D printing of a part of a robotic hand.

Code

- Matlab

Image Capture

```
clc
clear all
close all
warning off
try
   % Inicializar cámara
   disp('Phase 1: Initializing camera...');
   cam = webcam(1);
catch
   error('Error initializing the camera. Make sure the camera is
connected.');
end
% Parámetros de la caja de detección
x = 100;
y = 0;
height = 550;
width = 600;
bboxes = [x y height width];
try
   % Cargar red neuronal
   disp('Phase 2: Loading neural network...');
   load("REDESCOLORXEPOCAS\Guante\GoogleNet\GoogleNet20epoch.mat");
   load("REDESCOLORXEPOCAS\Guante\Alexnet\AlexNetTotal.mat");
   net=trainedNetwork_1;
catch
   error('Error loading the neural network.');
end
error=0;
try
   %Calibrar imagen
   disp("Phase 3: Calibration...")
   calibration=input("Do you want to calibrate the image?1/0:");
   if calibration ==1

       % Capturar imagen de la cámara
       snapshotImg = snapshot(cam);

       % Mostrar imagen con caja de detección
       IFaces =
insertObjectAnnotation(snapshotImg,'rectangle',bboxes,'Detection Sector');

       % Recortar y redimensionar imagen
       croppedImg = imresize(imcrop(snapshotImg, bboxes), [200 200]);

       % Preprocesamiento de imagen
       targetImg = imsubtract(croppedImg(:,:,1), rgb2gray(croppedImg));
       % Determinar el umbral utilizando el método de Otsu
       threshold = graythresh(targetImg);
       error_threshold = 0;
       while error_threshold < 0.02
           binaryImg = imbinarize(targetImg, threshold - error_threshold);
           subplot(1, 1, 1);
           imshow(binaryImg);
           x = input("Is the image flawed? 1/0:");
           if x == 1
               error_threshold = error_threshold + 0.0025;
           else
               % Si no hay errores, se actualiza el umbral y se sale del
bucle
               threshold = threshold - error_threshold;
               break;
           end
       end
   else
       if calibration == 0
           % Capturar imagen de la cámara
           snapshotImg = snapshot(cam);

           % Mostrar imagen con caja de detección
           IFaces =
insertObjectAnnotation(snapshotImg,'rectangle',bboxes,'Detection Sector');

           % Recortar y redimensionar imagen
           croppedImg = imresize(imcrop(snapshotImg, bboxes), [200 200]);

           % Preprocesamiento de imagen
           targetImg = imsubtract(croppedImg(:,:,1), rgb2gray(croppedImg));
           % Determinar el umbral utilizando el método de Otsu
           threshold = graythresh(targetImg);
       end
   end
catch
   error("Correct calibration could not be performed");
end
try
   disp("Phase 4: Image proccesing...")
   z = input("How many frames do you want?:");
   while z > 0
       % Capturar imagen de la cámara
       snapshotImg = snapshot(cam);

       % Mostrar imagen con caja de detección
       IFaces =
insertObjectAnnotation(snapshotImg,'rectangle',bboxes,'Detection Sector');

       % Recortar y redimensionar imagen
       croppedImg = imresize(imcrop(snapshotImg, bboxes), [200 200]);

       % Preprocesamiento de imagen
       targetImg = imsubtract(croppedImg(:,:,1), rgb2gray(croppedImg));
       % Determinar el umbral utilizando el método de Otsu
       binaryImg = imbinarize(targetImg, threshold);
       filteredImg = medfilt2(binaryImg);
       filteredImg = bwpropfilt(filteredImg, 'Area', [700 + eps(700),
Inf]);
       processedImg = bsxfun(@times, croppedImg, uint8(filteredImg));

       subplot(1,1,1);
       imshow(processedImg);
       z = z - 1;
       if z > 0
           if z < 5
               fprintf('%d seconds missing\n', z);
               pause(1);
           end
           %%pause(0.1);
       else
           fprintf("Capturing...\n");
           break;
       end
   end
catch
   error("Image processing failure")
end
try
   disp("Phase 5: Prediction...")
   Predict=input("Do you wish to perform a prediction?1/0:");
   if Predict==1
       disp("Predicting...")
       pause(0.5)
       % Realizar la predicción usando la red neuronal cargada
       prediction = classify(net20epoch, processedImg);
       prediction1 = classify(net, processedImg);
       subplot(2,1,1);
       imshow(processedImg);
       title(sprintf('GoogleNet: %s', char(prediction)), 'Color', 'b',
'FontSize', 15);
       % Obtener propiedades de las regiones
       [B,~] = bwboundaries(filteredImg);
       hold on
       % Dibujar contornos y etiquetas de predicción
       for k = 1:length(B)
           boundary = B{k};
           plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 2);
       end
       hold off

       subplot(2,1,2);
       imshow(processedImg);
       title(sprintf('AlexNet: %s', char(prediction1)), 'Color', 'r',
'FontSize', 15);
       % Obtener propiedades de las regiones
       [B,~] = bwboundaries(filteredImg);
       hold on
       % Dibujar contornos y etiquetas de predicción
       for k = 1:length(B)
           boundary = B{k};
           plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2);
       end
       hold off

   else
      disp("Prediction failure...")
   end
catch
   error("Prediction failure")
end
clear cam;
disp('Camera released. End of program.');
```

```
clc
clear all
close all
warning off
cam = webcam(1);
x = 100;
y = 0;
height = 550;
width = 600;
bboxes = [x y height width];
load("REDESCOLORXEPOCAS\Humano\Alexnet\Alex20epoch.mat");
load("REDESCOLORXEPOCAS\Humano\GoogleNet\GoogleNet20epochCNN.mat");
while true
   % Capturar imagen de la cámara
   snapshotImg = snapshot(cam);
   % Mostrar imagen con caja de detección
   IFaces =
insertObjectAnnotation(snapshotImg,'rectangle',bboxes,'Detection Sector');
   % Recortar y redimensionar imagen
   croppedImg = imresize(imcrop(snapshotImg, bboxes), [200 200]);
   % Preprocesamiento de imagen
   targetImg = imsubtract(croppedImg(:,:,1), rgb2gray(croppedImg));
   binaryImg = imbinarize(targetImg, 0.06);
   filteredImg = medfilt2(binaryImg);
   filteredImg = bwpropfilt(filteredImg, 'Area', [700 + eps(700), Inf]);
   processedImg = bsxfun(@times, croppedImg, uint8(filteredImg));
   % Realizar la predicción usando la red neuronal cargada
   [prediction, probability] = classify(net, croppedImg);
   %[prediction10, probability10] = classify(net10epoch, croppedImg);
   %[prediction15, probability15] = classify(trainedNetwork_1, croppedImg);
   [prediction20, probability20] = classify(net20epoch, croppedImg);
   % Obtener el porcentaje de predicción
   [maxProb, idx] = max(probability);
   [maxProb20, idx20] = max(probability20);

   % Obtener propiedades de las regiones
   [L, N] = bwlabel(filteredImg);
   props = regionprops(L);
   [B, ~] = bwboundaries(filteredImg);

   subplot(2,3,1);
   imshow(processedImg);
   title(['AlexNet Model: (', num2str(maxProb*100, '%.2f'), '%)']);
   hold on
   % Dibujar contornos y etiquetas de predicción
   for k = 1:length(B)
       boundary = B{k};
       plot(boundary(:,2), boundary(:,1),'b','LineWidth',2)
   end
   for n = 1:N
       c = round(props(n).Centroid);
       %text(c(1), c(2)-100, strcat(char(prediction)), 'Color', 'red',
'FontSize', 15);
   end
   hold off

   subplot(2,3,2:3);
   bar(probability);
   title('Prediction:',char(prediction));
   xlabel('Classes');
   ylabel('Probability');
   xticks(1:length(probability));
   xticklabels(categories(prediction));

   subplot(2,3,4);
   imshow(processedImg);
   title(['GoogleNet Model: (', num2str(maxProb20*100, '%.2f'), '%)']);
   hold on
   % Dibujar contornos y etiquetas de predicción
   for k = 1:length(B)
       boundary = B{k};
       plot(boundary(:,2), boundary(:,1),'b','LineWidth',2)
   end
   for n = 1:N
       c = round(props(n).Centroid);
       %text(c(1), c(2)-100, strcat(char(prediction20)), 'Color', 'white',
'FontSize', 15);
   end
   hold off

   subplot(2,3,5:6);
   bar(probability20);
   title('Prediction:',char(prediction20));
   xlabel('Classes');
   ylabel('Probability');
   xticks(1:length(probability20));
   xticklabels(categories(prediction20));
end
clear cam;
```

Confusion Matrix

```
load("REDESCOLORXEPOCAS\Humano\Alexnet\Alextotalcnn.mat")
trainedNetwork_1=net20epoch;
% Evaluar el modelo en el conjunto de validación
predLabels = classify(trainedNetwork_1, imdsValidation);
trueLabels = imdsValidation.Labels;
% Calcular la matriz de confusión
confusionMatrix = confusionmat(trueLabels, predLabels);
% Visualizar la matriz de confusión
figure;
confusionchart(confusionMatrix, unique(trueLabels));
% Realizar predicciones en los datos de validación
YPredValidation = classify(trainedNetwork_1, augimdsValidation);
YValidation = imdsValidation.Labels;
% Crear y mostrar la matriz de confusión para los datos de validación
figure;
plotconfusion(YValidation, YPredValidation);
title('Confusion Matrix');
% Calcular y mostrar el accuracy de validación
%accuracyValidation = sum(YPredValidation == YValidation) /
numel(YValidation);
%disp("Validation Accuracy: ");
```

Grad-Cam

```
% Cargar las redes entrenadas
```

```
load("REDESCOLORXEPOCAS\Humano\GoogleNet\GoogleNet20epochCNN.mat");
load("REDESCOLORXEPOCAS\Humano\Alexnet\Alex20epoch.mat");
net1=net20epoch;
classNames= net.Layers(25,1).Classes;
classNames1= net1.Layers(144,1).Classes;
inputSize = net.Layers(1).InputSize(1:2);
inputSize1 = net1.Layers(1).InputSize(1:2);
img = imread("Fotos\Fotos\Nothing\Nothing0001.jpg");
img = imresize(img,inputSize);
if canUseGPU
   X = gpuArray(img);
end
scores = predict(net,single(img));
scores1 = predict(net1,single(img));
% Convierte las puntuaciones a etiquetas de clase
Y = scores2label(scores, classNames);
imshow(img);
title(Y);
% Convierte las puntuaciones a etiquetas de clase
Y1 = scores2label(scores1, classNames1);
imshow(img);
title( Y1 );
channel = find(Y == categorical(classNames));
map = gradCAM(net,img,channel);
channel1 = find(Y1 == categorical(classNames1));
map1 = gradCAM(net1,img,channel1);
subplot(2,1,1)
imshow(img);
hold on;
imagesc(map,'AlphaData',0.5);
colormap jet
hold off;
title("Grad-CAM AlexNet");
subplot(2,1,2)
imshow(img);
hold on;
imagesc(map1,'AlphaData',0.5);
colormap jet
hold off;
title("Grad-CAM GoogleNet");
```

Artificial Vision

```
clc
clear all
close all
warning off
cam = webcam(1);
x = 100;
y = 0;
height = 550;
width = 600;
bboxes = [x y height width];
load("REDESCOLORXEPOCAS\Guante\Alexnet\AlexNet20epoch.mat")
while true
   % Capturar imagen de la cámara
   snapshotImg = snapshot(cam);
   % Mostrar imagen con caja de detección
   IFaces =
insertObjectAnnotation(snapshotImg,'rectangle',bboxes,'Detection Sector');
   % Recortar y redimensionar imagen
   croppedImg = imresize(imcrop(snapshotImg, bboxes), [200 200]);
   % Preprocesamiento de imagen
   targetImg = imsubtract(croppedImg(:,:,1), rgb2gray(croppedImg));
   binaryImg = imbinarize(targetImg, 0.26);
   filteredImg = medfilt2(binaryImg);
   filteredImg = bwpropfilt(filteredImg, 'Area', [700 + eps(700), Inf]);
   processedImg = bsxfun(@times, croppedImg, uint8(filteredImg));
   % Mostrar imágenes
   subplot(2,2,1);
   imshow(IFaces);
   title("All");
   subplot(2,2,2);
   imshow(targetImg);
   title("Target")
   subplot(2,2,3);
   imhist(targetImg);
   title("Histogram for Red Color")
   subplot(2,2,4);
   imshow(processedImg);
   title("without background")
   % Realizar la predicción usando la red neuronal cargada
   prediction = classify(net20epoch, processedImg);
   % Obtener propiedades de las regiones
   [L, N] = bwlabel(filteredImg);
   props = regionprops(L);
   [B,~] = bwboundaries(filteredImg);
   hold on
   % Dibujar contornos y etiquetas de predicción
   for k = 1:length(B)
       boundary = B{k};
       plot(boundary(:,2), boundary(:,1),'b','LineWidth',2)
   end
   for n = 1:N
       c = round(props(n).Centroid);
       text(c(1), c(2)-100, strcat(char(prediction)), 'Color', 'white',
'FontSize', 15);
   end
   hold off
end
clear cam;
```

DataBase

```
clear all
clc
close all
% Get the list of available cameras
camList = webcamlist;
% Define desired resolution
desiredWidth = 200;
desiredHeight = 200;
% Set camera resolution
cam.Resolution = sprintf('%dx%d', desiredWidth, desiredHeight);
% Connect to the camera.
cam = webcam(1);
% Start the camera preview.
preview(cam);
```

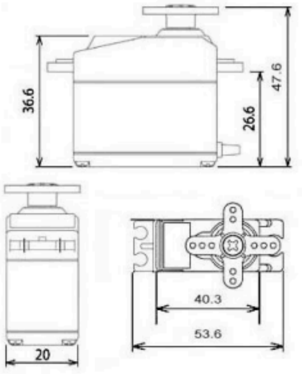
```
% Create a folder to store images
if ~exist('Fotos/', 'dir')
   mkdir('Fotos/A');
   mkdir('Fotos/Nothing');
   mkdir('Fotos/B');
   mkdir('Fotos/C');mkdir('Fotos/D');
   mkdir('Fotos/E');mkdir('Fotos/F');
   mkdir('Fotos/G');mkdir('Fotos/H');
   mkdir('Fotos/I');mkdir('Fotos/J');
   mkdir('Fotos/K');mkdir('Fotos/L');
   mkdir('Fotos/M');mkdir('Fotos/N');
   mkdir('Fotos/O');mkdir('Fotos/P');
   mkdir('Fotos/Q');
   mkdir('Fotos/R');mkdir('Fotos/S');
   mkdir('Fotos/T');mkdir('Fotos/U');
   mkdir('Fotos/V');mkdir('Fotos/W');
   mkdir('Fotos/X');mkdir('Fotos/Y');
   mkdir('Fotos/Z');
end
% Capture and save multiple frames
for idx = 1:400
   img = snapshot(cam);

   % Resize image to desired resolution
   resizedImg = imresize(img, [desiredHeight, desiredWidth]);

   image(resizedImg);
   % Generate a unique name for each image based on the capture index
   filename = sprintf('Fotos/X/X%04d.jpg', idx);

   % Save the resized image in the following folder
   imwrite(resizedImg, filename);
end
% Close the camera connection
clear cam;
```

- Motor Controller with Arduino

```
//Add servo Library
#include <Servo.h>
//Define multiples Servos
Servo servoInd;
Servo servoMed;
Servo servoAnu;
Servo servoPul;
Servo servoMeq;
//Servo postion in degrees
int ServoPosInd = 90;
int ServoPosMed = 90;
int ServoPosAnu = 90;
int ServoPosPul = 90;
int ServoPosMeq = 90;
void setup() {
 Serial.begin(9600);
 //Define servo inputs Digital PWM
 servoInd.attach(3);
 servoMed.attach(4);
 servoAnu.attach(5);
 servoPul.attach(6);
 servoMeq.attach(7);
}
void loop() {
 moveFingers();
}
void moveFingers() {
 // Base movement
 // Finger Movement Extension
 // Thumb
 moveServo(servoPul, ServoPosPul, 90, 180);
 // Little finger
 moveServo(servoMeq, ServoPosMeq, 90, 175);
 // Ring finger
 moveServo(servoAnu, ServoPosAnu, 90, 180);
 // Index finger
 moveServo(servoInd, ServoPosInd, 90, 160);
 // Middle finger
 moveServo(servoMed, ServoPosMed, 90, 140);
}
void moveServo(Servo servo, int &pos, int from, int to) {
 if (pos != to) {
   if (pos < to) {
     pos++;
   } else {
     pos--;
   }
   servo.write(pos);
   delay(15);
 }
}
```

Servomotors datasheet.

MG996R High Torque Metal Gear Dual Ball Bearing Servo



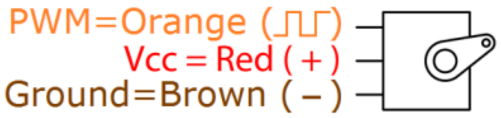
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC



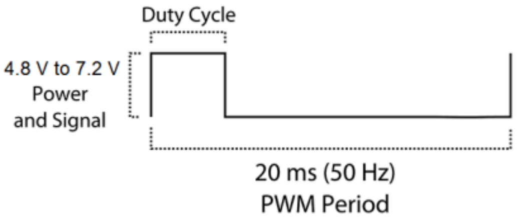


Figure 63. Power source 10A/5V.