UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
Facultad de Ingeniería Civil, Sistemas y de Arquitectura
Escuela Profesional de Ingeniería de Sistemas

TESIS

Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas

Para Obtener el Título Profesional de:

Ingeniero de Sistemas

Montenegro Chore, Israel
Pravia Purihuaman, Manuel Grabiel
Autores

Mg. Ing. Capuñay Uceda, Oscar Efraín
Asesor

Lambayeque - Perú
2023

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

Facultad de Ingeniería Civil, Sistemas y de Arquitectura

Escuela Profesional de Ingeniería de Sistemas

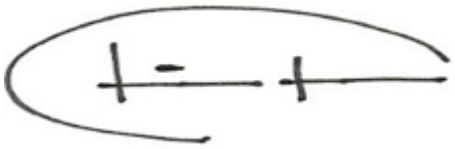
TESIS

Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas

Para Obtener el Título Profesional de:

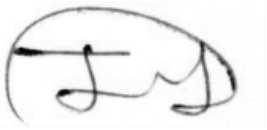
Ingeniero de Sistemas

Aprobado por los Miembros de Jurado:



Dr. Ing. Jacinto Mejía, Pedro Miguel
Presidente del Jurado

Mg. Ing. Arteaga Lora, Roberto Carlos
Secretario



Dr. Ing. Villegas Cubas Juan Elías
Vocal

LAMBAYEQUE - PERÚ
2023

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
Facultad de Ingeniería Civil, Sistemas y de Arquitectura
Escuela Profesional de Ingeniería de Sistemas

TESIS

Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas

Para Obtener el Título Profesional de:

Ingeniero de Sistemas

Montenegro Chore, Israel
Autor

Pravia Purihuaman, Manuel Grabiel
Autor

Mg. Ing. Capuñay Uceda Oscar Efraín
Asesor

Lambayeque – Perú
2023

DEDICATORIAS

Para mis padres y hermanos que
son mi fuente de inspiración y me
dan las fuerzas para superarme,
no rendirme y seguir adelante.
Pravia Purihuaman, Manuel Grabiel

En primer lugar, esta tesis la dedico a Dios
quien me guía en mi camino,
a mis queridos padres Daniel y Betty,
quienes me dan mucho de su amor,
dedicación y esfuerzo, y me motivan
a andar por el buen camino.
A toda mi familia y amigos que
me apoyaron en este trabajo.
Montenegro Chore, Israel

AGRADECIMIENTOS.

Doy gracias a dios quien me ha permitido continuar seguir en este sueño y me ha brindado sabiduría, resiliencia y estabilidad en momentos de dificultad.

Agradezco a mi padre Manuel Pravia Cespedes que desde el cielo es la luz que me guía y me da fuerzas, mientras estuvo vivo me apoyo incondicionalmente y sus consejos siempre los llevo presente. A mi madre Tomasa Purihuaman Reyes que con tanto amor, sus sabios consejos y su apoyo incondicional me inspira a superarme cada dia mas. A mis hermanos y hermanas que siempre están ahí para reír, llorar, solidarizarnos y apoyarnos mutuamente les agradezco profundamente porque siempre están conmigo en los momentos más difíciles.

Agradecer a mi asesor de tesis el Mg. Ing. Oscar Efrain Capuñay Uceda por compartir su experiencia y saberes científico, y así guiarme durante la elaboración de esta investigación. Además agradecer a todos los docentes de la Universidad Nacional Pedro Ruiz Gallo que con sus conocimientos y experiencias han influido en mi formación profesional.

Pravia Purihuaman, Manuel Grabiel

Inmenso agradecimiento a nuestro creador Dios por la vida y salud que prodiga, permitiendome alcanzar esta meta a través de su sabiduría.

Agradezco a mis queridos padres por ayudarme a crecer en lo personal y profesional. Gracias a ellos por cada día apoyarme, aconsejarme y motivarme a no rendirme. Gracias por sus ánimos en momentos tormentosos.

Agradezco a mi asesor Mg. Ing. Oscar Efraín Capuñay Uceda, por su apoyo en permitirme formarme profesionalmente en materia de ingeniería. Además, a todos los docentes de mi casa de estudios, la Universidad Nacional Pedro Ruiz Gallo, que con su abnegación brindaron sus conocimientos y experiencias en mi formación profesional.

Por último, agradezco a todo el que lee la presente tesis, por permitir que mi investigación incurra dentro de su repertorio de información.

Montenegro Chore, Israel

RESUMEN

En la presente tesis “Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas”, el objetivo primordial es construir un algoritmo (conjunto de pasos a seguir para resolver un problema) que permita el reconocimiento de las señas del alfabeto de signos peruano, de esta manera nos permite comunicarnos con personas que padecen de deficiencia auditiva, debido a la gran importancia actual de incluir a más personas en nuestra sociedad.

Las personas sordas utilizan una manera de comunicarse mediante el alfabeto de señas, pero este alfabeto es desconocido por muchas personas. Con el sistema (software) implementado en este proyecto se busca que exista una comunicación fluida entre dos personas: una oyente y otra sorda, aunque esta última no tenga uso o no sepa el alfabeto de señas. La traducción de voz a señas, y de señas a voz es realizada por este software, el cual está basado en inteligencia artificial.

En Perú conviven un número mayor a 232000 personas con problemas de sordera (según la Defensoría del Pueblo, 2020), por esta razón proponemos en el presente proyecto desarrollar un sistema computacional basado en inteligencia artificial que sirva como intérprete entre una persona hablante y otra sorda. Y para este fin proponemos el uso de tecnologías como la visión artificial y también las redes neuronales convolucionales; y emplear Python como lenguaje de programación, el cual está teniendo un mayor auge actualmente en la comunidad de programadores.

ABSTRACT

In this thesis "Computer system based on artificial intelligence that improves communication with a deaf person through the sign alphabet", the primary objective is to build an algorithm (set of steps to follow to solve a problem) that allows the recognition of the signs of the Peruvian sign alphabet, in this way allows us to communicate with people who suffer from hearing impairment, due to the current great importance of including more people in our society, both nationally and internationally.

Deaf people use a way to communicate through the sign alphabet. but this alphabet is unknown by many people. With the system (software) implemented in this project, it is sought that there is fluid communication between two people: a hearing person and a deaf person, even if the latter does not use or does not know the sign alphabet. The translation from speech to signs, and from signs to speech is performed by this software, which is based on artificial intelligence.

In Peru there are more than 232,000 people with deafness problems (according to the Defensoría del Pueblo, 2020), for this reason we propose in this project to develop a computer system based on artificial intelligence that serves as an interpreter between a speaking person and a deaf person. And for this purpose we propose the use of technologies such as artificial vision and convolutional neural networks; and use Python as a programming language, which is currently having a greater boom in the community of programmers.

Palabras clave:

Alfabeto de señas, Redes neuronales convolucionales, Visión artificial, Lenguaje de programación Python, OpenCV

ÍNDICE DE CONTENIDO

RESUMEN 6

ABSTRACT 7

PALABRAS CLAVE **7**

CAPÍTULO 1: INTRODUCCIÓN 14

1. Planteamiento o formulación de lo que se investigará 14

1.1 Resumen del escenario problemático 14

1.2 Planteamiento del problema a investigar 14

1.3 Hipótesis o solución del problema 15

1.4 Objetivo general 15

1.5 Objetivos específicos 15

CAPÍTULO 2: DISEÑO TEÓRICO 16

2.1 Antecedentes de la investigación 16

2.2 Bases teóricas 20

2.2.1 Discapacidad auditiva en el ser humano 20

2.2.2 Lenguaje de señas 20

2.2.2.1 Lengua de señas peruana (LSP) 20

2.2.2.2 Alfabeto de señas peruano 21

2.2.3 Sistemas computacionales 23

2.2.4 Inteligencia artificial (IA) 23

2.2.4.1 Definición de IA 23

2.2.4.2 Breve reseña histórica de la IA 24

2.2.5 Machine Learning 27

2.2.6 Imágenes digitales 27

2.2.7 Visión artificial 35

2.2.7.1 Cámara de tipo digital 38

2.2.7.2 Reconocimiento 40

2.2.7.3 Técnicas de categorización de patrones 41

2.2.7.4 Interpretación 42

2.2.8 Redes neuronales artificiales 42

2.2.8.1 Elementos de la Red Neuronal 45
2.2.8.2 Funciones matemáticas para activación y salida 47
2.2.8.4 Clasificación de las Redes Neuronales Artificiales 48
2.2.8.5 Cómo hacer que aprenda una red neuronal 54
2.2.8.6 Funciones de activación 55
2.2.8.7 Funciones de perdida 57
2.2.8.8 Optimizadores 58
2.2.9 OpenCV 59
2.2.10 TensorFlow 61
2.2.11 Lenguaje de programación Python 61
CAPÍTULO 3: DISEÑO METODOLÓGICO 66
3.1 Tipificación de la investigación 66
3.2 Tabla de operacionalización de variables 68
3.3 Población y muestra 68
3.4 Metodología y recolección de datos 70
3.4.1 Procedimiento y Herramientas para Recolección de datos 71
CAPÍTULO 4: DESARROLLO DEL PROYECTO 73
4.1 Análisis del uso del alfabeto de señas en el colegio Harvest 73
4.2 Determinación de las tecnologías que se emplearán en la construcción del sistema inteligente 75
4.3 Requisitos previos 80
4.4 Entorno de trabajo 80
4.5 Obtención del modelo 85
CAPÍTULO 5: EVALUACIÓN DE LOS RESULTADOS 104
5.2.1 Participantes 106
5.2.2 Especificación del escenario o ambiente 107
5.2.3 Resultados obtenidos: 107
CAPÍTULO 6: CONCLUSIONES 111
BIBLIOGRAFÍA 112
ANEXOS 117
Anexo 1: Librerías de Python usadas en el presente proyecto 117
Anexo 2: Contenido del archivo parameter.json 120

Anexo 3: Código del archive Sign.py 121

Anexo 4: Código del archivo CNN.py 126

Anexo 5: Código del archive Main.py 129

Anexo 6: Interfaz final de usuario del sistema desarrollado en este proyecto 144

Anexo 7: Imágenes de alumnos del colegio Harvest, el día en que se hicieron las pruebas 145

ÍNDICE DE TABLAS

Tabla 1: Fases del tratamiento digital de imágenes 37
Tabla 2: Entradas y salidas del proceso del PDI 38
Tabla 3: Operacionalización de variables 68
Tabla 4: Algunos posibles valores que puede tomar Z 69
Tabla 5: Contrastación de hipótesis 72
Tabla 6: Enumeración de señas por clase. 85
Tabla 7: Verificación de precisión de la CNN 99
Tabla 8: Indicadores a evaluar en la etapa de pre-test 104
Tabla 9: Cantidad de señas interpretadas por el oyente 105
Tabla 10: Tiempo promedio en que un estudiante realiza una seña legible 106
Tabla 11: Indicadores a evaluar en la etapa de post-test 107
Tabla 12: Cantidad de señas reconocidas por el sistema. 108
Tabla 13: Tiempo promedio que tarda el sistema en reconocer una seña. 109
Tabla 14: Tiempo que tarda un estudiante con deficiencia auditiva en comunicarse con una persona oyente. 110

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Alfabeto de Señas Peruano 22
Ilustración 2: Imágenes con distintos bits por pixel. 29
Ilustración 3: Imágenes cuya resolución espacial se va rebajando 30
Ilustración 4: Imágenes con disminución de tonalidad 32
Ilustración 5: Imagen tipo RGB de una mano 33
Ilustración 6: Imagen de una mano a escala de grises 34
Ilustración 7: Imagen de tipo binaria de una mano 35
Ilustración 8: Un esquema de visión artificial de manera práctica empleado en la producción. 36
Ilustración 9: Modelo que ejemplifica el mecanismo de segmentación 40
Ilustración 10: Técnica de categorización de elementos 41
Ilustración 11: Comparación del RNA con una biológica 44
Ilustración 12: Similitudes entre una neurona artificial y una Biológica 45
Ilustración 13: Modelo de una Neurona Artificial 45
Ilustración 14: Composición de un Perceptrón Multicapa 47
Ilustración 15: División de las RNA 49
Ilustración 16: RNA con Conexiones hacia Adelante Monocapa 50
Ilustración 17: RNA conectadas hacia Adelante Multicapa 51
Ilustración 18: RNA con Conexiones hacia Atrás. 51
Ilustración 19: Agrupamiento promedio en una CNN 54
Ilustración 20: Función de activación relu 55
Ilustración 21: Función de activación sigmoide 56
Ilustración 22: Función de activación softmax 57
Ilustración 23: Representación de OpenCV 60
Ilustración 24: Mediapipe haciendo la detección de la mano, y trazando las coordenadas. 64
Ilustración 25: Puntos de detección de Mediapipe 65
Ilustración 26: Esquema que esclarece la metodología para la construcción del sistema 70
Ilustración 27: Representación en señas de un árbol 74
Ilustración 28: Representación en Alfabeto de Señas d la palabra árbol 75
Ilustración 29: Resultado de búsqueda en SCOPUS para Redes Neuronales Convolucionales y Python 75
Ilustración 30: Resultado de búsqueda en SCOPUS para Inteligencia artificial y php .. 76
Ilustración 31: Resultado de búsqueda en SCOPUS para Inteligencia artificial y visual basic 77
Ilustración 32: Resultado de búsqueda en SCOPUS para Inteligencia artificial y javascript 77
Ilustración 33: Uso de lenguajes de programación en IA, según búsquedas en Scopus 78
Ilustración 34: Binarización de la imagen con la ayuda de mediapipe 89
Ilustración 35: Con la ayuda de cv2 conseguimos una imagen de mano totalmente binarizada 90
Ilustración 36: Procesado de imágenes. 90
Ilustración 37: Imágenes binarias de las señas. 92

Ilustración 38: Inicio de conversación de persona sorda. 100
Ilustración 39: Inicio de conversación por persona hablante 101
Ilustración 40: Interfaz de usuario del sistema 103

CAPÍTULO 1: INTRODUCCIÓN

1. Planteamiento o formulación de lo que se investigará

1.1 Resumen del escenario problemático

La inclusión de las personas con habilidades especiales, incluido las personas sordas, ha cobrado una gran importancia en la agenda mundial. La inclusión de aquellas personas, y el saber entenderlas, es fundamental para tener sociedades más igualitarias y desarrolladas. Por ello, la inclusión de las personas sordas es un ideal universal. Conforme a la Federación Mundial de Sordos, a nivel mundial existen alrededor de 72 millones de personas con sordera. Sin embargo, es poco factible comunicarse con las personas sordas ya que existe poco uso o desconocimiento del lenguaje de señas (aquí se incluye el alfabeto de señas) en casi toda la población del planeta. La carencia de inclusión de las personas sordas por no saber comunicarse con ellas es más acentuada en los países subdesarrollados.

En Perú conviven un número mayor a 232 000 de ciudadanos con problemas de sordera. Si queremos comunicarnos con personas sordas y desconocemos el lenguaje de señas, es necesario que haya un intérprete, ya que aprender el lenguaje (incluye alfabeto) de señas nos demandaría cierto tiempo. Las personas con discapacidad auditiva sufren limitaciones para hacer uso de servicios por la carencia de traductores del lenguaje de señas peruano.

Se propone desarrollar un sistema computacional basado en inteligencia artificial que sirva como intérprete entre una persona hablante y otra sorda. Este intérprete virtual nos permitirá interactuar y comunicarnos con personas sordas utilizando el alfabeto de señas.

1.2 Planteamiento del problema a investigar

¿Cómo un sistema computacional basado en inteligencia artificial actuando como intérprete mejorará la comunicación de los alumnos de la IE Bautista para sordos Harvest y una persona oyente?

1.3 Hipótesis o solución del problema

El sistema computacional basado en inteligencia artificial mediante el alfabeto de señas, actuando como un intérprete en tiempo real permitirá disminuir la barrera de comunicación de los alumnos de la IE Bautista para sordos Harvest y un oyente.

1.4 Objetivo general

Desarrollar un sistema computacional basado en inteligencia artificial para mejorar la comunicación en los alumnos de la Institución Educativa para sordos Harvest, haciendo uso del alfabeto de señas.

1.5 Objetivos específicos

1. Analizar el uso del alfabeto de señas en los alumnos del colegio Harvest.
2. Determinar las tecnologías que se emplearán en la construcción del sistema inteligente.
3. Evaluar el porcentaje de precisión en reconocer las señas del sistema computacional basado en inteligencia artificial en los alumnos del colegio Harvest

CAPÍTULO 2: DISEÑO TEÓRICO

2.1 Antecedentes de la investigación

- Prangon Das, Tanvir Ahmed, Firoj Ali. 2020. Bangladesh. Departamento de Ingeniería Mecatrónica, Universidad Rajshahi de Ingeniería y Tecnología. **Reconocimiento de gestos manuales estáticos para el lenguaje de señas estadounidense mediante una red neuronal convolucional profunda.**

  La siguiente investigación consiste en construir un modelo de reconocimiento de imágenes estáticas del lenguaje de señas americano(ASL) empleando herramientas de inteligencia artificial basado en redes neuronales convolucionales. Se utilizaron un conjunto de datos de ASL de 1815 imágenes del alfabeto inglés para entrenar y validar el modelo. al validar el modelo arrojó una precisión de 94,34% , también se hizo la prueba con el algoritmo de SVM arrojando una precisión 69% con lo que se concluye que el algoritmo de CNN proporciona resultados más precisos.

- Virgil Jeny, Anjana, Karnati Monica, Thandu Sumanth, Mamatha. 2021. India. Departamento de Ingeniería Informática, Instituto de Tecnología y Ciencias de Vignan. **Reconocimiento de gestos manuales para lenguaje de señas mediante red neuronal convolucional.**

  En la siguiente investigación se desarrolla un sistema que puede traducir el lenguaje de señas a texto y luego a audio por lo que puede mejorar la comunicación mediante el lenguaje de señas. En la implementación de este sistema se utilizaron tecnologías como python utilizando el algoritmo CNN y la técnica de enmascaramiento para el preprocesamiento de imágenes con la ayuda del módulo openCv y pyttsx3 para pasar de letras a voz.

  El sistema toma las imágenes mediante una cámara web de computadora luego se realiza el preprocesamiento de la imagen usando una técnica de enmascaramiento en la que se enmascara la mano para reconocer el alfabeto firmado. Posteriormente usando un algoritmo de red neuronal convolucional se se mapean las características mediante las tres capas internas del algoritmo y por último las clases se forman a partir de las características y se predice la imagen.

CNN funciona como la corteza visual del cerebro humano. Para lograr esto, se realizan diferentes circunvoluciones mediante el uso de filtros que se utilizan como pesos entrenables, para cada canal se pueden aplicar múltiples filtros y se pueden formar mapas de características. En este modelo se utilizaron 52000 imágenes como el conjunto de datos de entrada los cuales se dividen en datos de prueba(45500) y entrenamiento(6500) arrojando una precisión del 98.55%.

- Nishi Intawala, Arka Banerjee, Meenakshi, Nikhil Gala. 2019. India. Escuela de ingeniería y gestión de tecnología Mukesh Patel. **Conversor de lenguaje de señas indio usando redes neuronales convolucionales.**

  En la actualidad las personas que padecen con discapacidad tanto auditiva y del habla se enfrentan a muchas dificultades para comunicarse con el público en general ya que al ser la minoría, el lenguaje utilizado por ellos no es conocido para la mayoría de las personas.

  En este artículo de investigación no detalla el desarrolló un convertidor de lenguaje de señas indio utilizando un algoritmo de redes neuronales artificiales con el objetivo de poder clasificar 26 letras del alfabeto de señas indio capturando una imagen en tiempo real del signo y convirtiéndolo en su equivalente de texto.

  Primero se creó una base de datos en varios fondos y se utilizaron varias técnicas de preprocesamiento de imágenes para preparar la base de datos para la extracción de características.

  Después de haber extraído las características, las imágenes se introdujeron en la CNN utilizando el software python. Se probaron varias imágenes en tiempo real para encontrar la precisión y la eficiencia. Los resultados arrojaron una precisión del 96% para las imágenes de prueba y una precisión del 87,69% para las imágenes en tiempo real.

- Suat Rojas, Néstor E., Montoya Serna, Brayan S. 2021. Colombia. Universidad de los Llanos. **Reconocimiento del abecedario de la lengua de señas colombiana con Redes Neuronales Convolucionales.**

  El siguiente trabajo de investigación consistió en diseñar un método que involucra tecnologías de IA y visión artificial que identifique las señas estáticas(que constó de 21 letras del abecedario) de la Lengua de Señas Colombiana (LSC). La metodología consiste en aplicar técnicas de procesamiento de imágenes y el

algoritmo de clasificación que combina una arquitectura de Redes Neuronales Convolucionales (CNN). Dicho modelo pudo lograr reconocer las señas del alfabeto estático(sin movimiento) con un 79.2% de precisión(Accuracy). El sistema diseñado fue capaz de reconocer con gran precisión las letras formadas por la mano según la orientación, posición y la forma de los dedos, usando un conjunto de datos desbalanceado por cada clase.

- Facundo Quiroga, Laura Cristina Lanzarini. 2019. Argentina. Facultad de Informática de la Universidad Nacional de La Plata. **Aprendizaje automático. aplicaciones en reconocimiento de gestos, acciones y señas**

  En las ramas de la Inteligencia Artificial tenemos una que se denomina Aprendizaje Automático, la cual estudia sistemas con la habilidad de aprender a hacer una labor específica a partir de datos de entrada. Su esencia es ser inductiva, muy aparte de la inteligencia artificial clásica, y abarca técnicas y métodos para ejecutar clasificación, optimización y predicción; de manera amplia en dominios en donde las problemáticas no se definen de manera fácil, o no hay respuestas concretas. En base a estas razones, las metodologías que propone son correctas para el procesamiento de imágenes y otros signos dactilológicos.
  En los presentes tiempos, el procesamiento de video, sonido, texto y otras señales ha conseguido significativos avances a través de la utilización de una técnica de Aprendizaje Automático que se llama Redes Neuronales Profundas o Aprendizaje Profundo (Deep Learning).

  Los propósitos de estos métodos es conseguir representaciones correctas de la información sin intervención de personas, utilizando modelos correctos y bases de datos con enormes instancias. Haciendo esto se logra una diferenciación de los instrumentos clásicos de Aprendizaje Automático, en donde los versados por lo general deben usar una cantidad suficiente de tiempo en elaborar representaciones apropiadas de los datos.

  De manera práctica, en esta investigación se propone escudriñar las técnicas para comprender qué ocurre en un ámbito desde un video o imagen de la fuente. Se enfatizará en tres subproblemas: acciones, ademanes realizados por individuos y clasificar signos. Todas las problemáticas tienen sus diferencias, pero están muy

conectadas entre sí. El clasificador de señas tiene como objetivo transformar a texto un video donde un individuo hace señas en algún idioma de señas que ya exista, como la Lengua de Señas Argentina (LSA). En el reconocimiento de acciones, se tiene como meta comprender y clasificar una acción que realiza un individuo. Y para terminar, en el reconocimiento de gestos, por lo general, se tiene como meta reconocer un set de gestos previamente determinados.

Es cierto que en los tiempos actuales hemos conseguido avances en esta rama, propulsados mayormente por el avance de recientes tecnologías, todavía hay un tramo ancho por caminar para elaborar softwares potentes y sólidos que logren la interpretación y traducción de los signos dactilológicos.

2.2 Bases teóricas

2.2.1 Discapacidad auditiva en el ser humano

Discapacidad auditiva

(José Luis Aguilar Martínez, 2008) La discapacidad auditiva se define como la carencia o falta de normalidad de la capacidad fisiológica y/o anatómica del oído, esto repercute directamente en una falta de habilidad para escuchar, lo que conlleva a una deficiencia en el procesamiento del lenguaje hablado. Teniendo como origen de que el sentido del oído es la manera primordial mediante la cual se construye el habla y el lenguaje, tengamos en claro que sin importar el déficit en la percepción de sonido de la persona, aun desde la infancia, va a repercutir a su desenvolvimiento comunicativo y lingüístico, a sus capacidades cognitivas y, por obviedad, a su posterior integración a la escuela, con sus amigos y en su trabajo.

Si bien es cierto, la palabra sordera significa un nivel o grado de deficiencia auditiva, éste se ha empleado y se usa comúnmente para referirse indistintamente a la deficiencia leve como severa, englobando su utilización en la designación de cualquier problema para oír.

2.2.2 Lenguaje de señas

Según el Ministerio de Educación de Perú (2015), El lenguaje de señas es un sistema de comunicación producido por el cuerpo, esto incluye extremidades, cabeza y dorso, y que se percibe a través de la vista.

Aunque son las manos los elementos esenciales en la emisión de este tipo de lenguaje, cabe destacar la importancia de las destrezas visuales para su comprensión.

Es un lenguaje innato en el ser humano, que permite la comunicación entre individuos sordos ya que no son capaces de comunicarse de manera verbal. Sin embargo, de acuerdo al país en donde viven, conviven distintos lenguajes de señas. Consecuentemente, hacer uso de esta lengua requiere usar la expresión facial, manos, posturas corporales y el espacio.

2.2.2.1 Lengua de señas peruana (LSP)

Según tal cual lo define el Ministerio de Educación (2015), la Lengua Peruana de Señas (siglas: LSP) es un idioma autóctono del país, la que tiene origen de la misma comunidad sorda en Perú, la que también hace uso de este lenguaje dactilológico. Así mismo, a semejanza de otros lenguajes, tiene sintaxis propia, léxico y gramática. Afortunadamente

el estado peruano ha reconocido este lenguaje mediante la Ley 29535, y lo ha reglamentado apropiadamente. La LSP tiene no pocas influencias de otros idiomas dactilológicos empleados en países latinoamericanos, como por ejemplo el lenguaje de señas americana. El Censo que se hizo en el país en el año 2017, dató que hay más de 10000 personas que emplean el LSP para comunicarse, pero muchos afirman que este número datado no es exacto, y que haya más peruanos que emplean este lenguaje en su día a día.

2.2.2.2 Alfabeto de señas peruano

Es el conjunto de señas usadas en Perú referentes al alfabeto.

Cabe destacar que en este tipo de señas sólo se utiliza una mano, a diferencia de otros países donde se usan ambas manos.

Según el Ministerio de Educación, el alfabeto de Señas Peruano consta de 27 letras (a, b, c, d, e, f, g, h, i, j, k, l, m, n ñ, o, p, q, r, s, t, u, v, w, x, y, z) cada una de las cuales es representada de manera distinta.

En la Ilustración 1 se muestra cuáles son dichas señas según la letra:

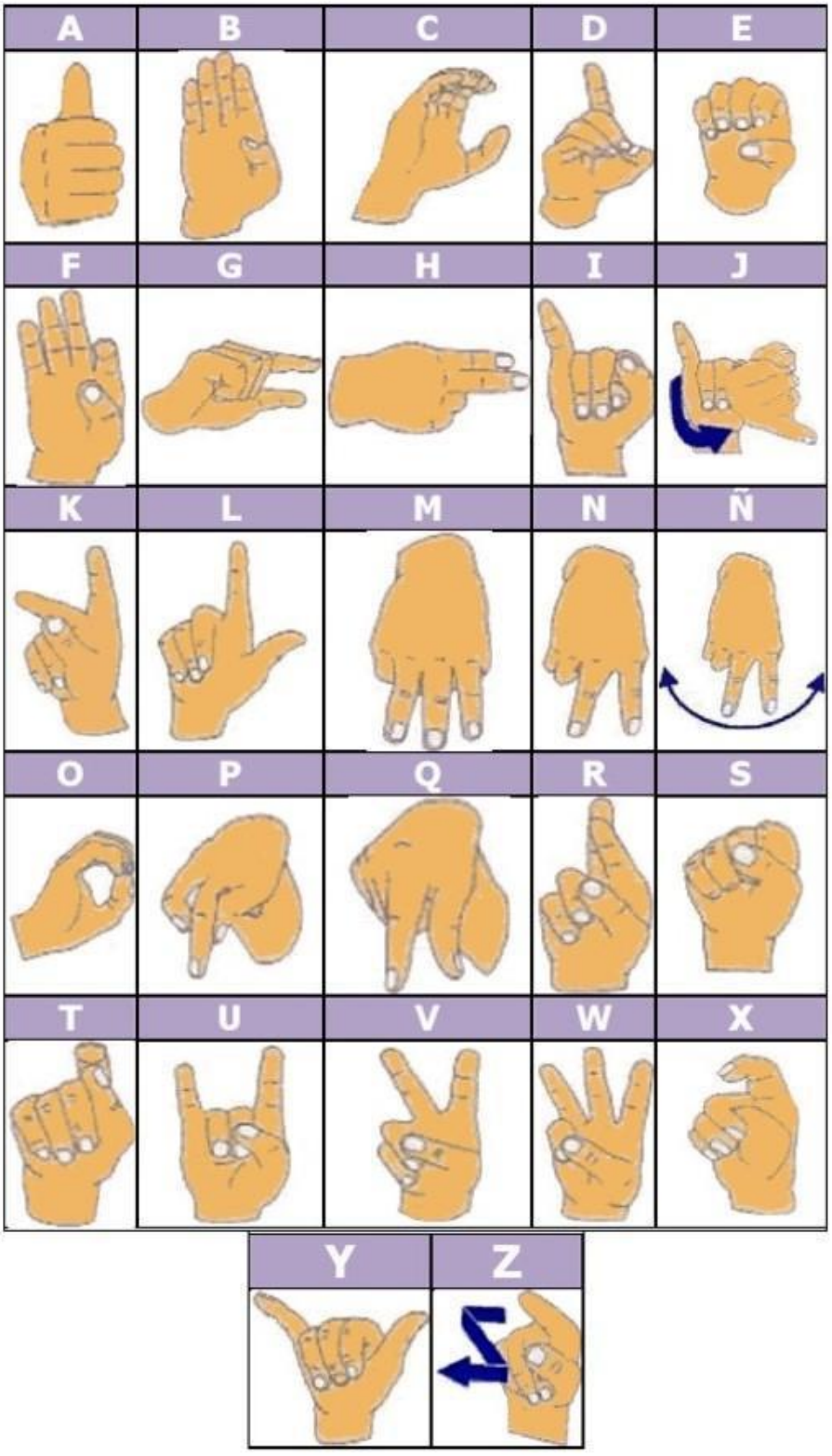


Ilustración 1: Alfabeto de Señas Peruano

(Fuente: Ministerio de Educación 2015)

2.2.3 Sistemas computacionales

Según Marco Alfredo Cedano Olvera (2014 – México), un sistema computacional es aquel sistema que hace uso y aplica disciplinas como mecatrónica, robótica, electrónica, cibernética, inteligencia artificial, matemáticas, lógicas, informática, entre otras, para la elaboración de productos o servicios que les sea indispensable y demanden información De igual modo, proporciona un soporte necesario en el ciclo de la información, mediante el recibimiento, el proceso, la devolución y el empleo de datos.

Estos sistemas computacionales alcanzan mayor cantidad de tecnologías que los ordenadores convencionales, incluso si muchos de estos sistemas emplean dispositivos computacionales o se elaboran en reglas iguales, logrando procesar la información de muchas maneras: audios, textos, videos, valores numéricos, imágenes, sonidos, entre otros.

También, Juan Francisco Esquivel Díaz (2013 – México) afirma que un sistema computacional está constituido por elementos tangibles (hardware) que se relacionan a través de instrucciones previamente definidas (software) con el objetivo de conseguir algo.

2.2.4 Inteligencia artificial (IA)

2.2.4.1 Definición de IA

La inteligencia artificial (IA) es un campo con variadas disciplinas de la computación, apreciada como el entendimiento científico de los procedimientos que dan como origen el comportamiento inteligente y el pensamiento, y su adhesión a los objetos, equipos o máquinas. Como como meta trazar y elaborar máquinas, artefactos y software, los que tienen que comportarse racionalmente, con cierto grado de habilidad de pensar, e inteligencia (AAAI).

Teniendo esto como origen, la IA ha contribuido a desarrollos en una gran cantidad de áreas de investigación, o campos de estudio o ciencias; como, por ejemplo: la lógica, matemática, informática, filosofía, electrónica, psicología, nanotecnología, robótica (Cairo, 2011., p. 1), además de muchas más; consecuentemente, se ha transformado en una fuente para otros aspectos como el arte, especialmente el séptimo arte, mediante la ciencia ficción, y la literatura. El entusiasmo generado por la IA se basa en su mayoría en la posibilidad de dar origen a softwares, máquinas o sistemas que son capaces de simular

la inteligencia del ser humano, hasta parecer de un nivel más potente que la del hombre. Pero, la idea de inteligencia no es aislado, siendo un poco controversial y ambiguo, dando origen a variados significados, interpretaciones, o deducciones.

Etimológicamente, la palabra inteligencia surge de la expresión latina legere que quiere decir recolectar; entonces, intelligere hace referencia a la idea de tener que elegir entre varias opciones. Entonces, esta expresión solo se comprendía como la habilidad de valorar, diferenciar y discernir. Sin embargo, mientras que el conocimiento del ser humano iba en aumento, la noción crecía, adhiriendo de esta manera una variedad de aspectos no repetitivos ni automáticos de la conducta, entrelazándose con mayor intensidad con el proceso creativo e imaginativo, y la resolución de situaciones problemáticas (Grupo de inteligencia Artificial Y Robótica, 2014).
La Inteligencia artificial (IA), no teniendo un concepto exclusivo, determinado y único, decidió tener un actuar práctico, definiéndose como mecanismo con cierto grado de inteligencia a aquello, que, en contextos semejantes, actúa semejante a un ser humano. Para estar seguro de esto se evalúa mediante el Test de Turing; en el cual se hacen preguntas a una máquina a través de una forma que no conlleve contacto físico. La clave estriba en que el individuo que expresa las interrogantes no tiene que saber si el que da la respuesta es una persona o una máquina (Grupo de inteligencia Artificial Y Robótica, 2014).

Como dato adicional pero importante, es ideal explicar la manera en que la IA se ha desarrollado, cómo se originó y su desenlace histórico desde el comienzo. Tenemos este propósito para así comprender su gran potencial en la mejora de innumerables ámbitos de investigación, en los cuales se encuentra la visión artificial

2.2.4.2 Breve reseña histórica de la IA

En 1956, la expresión inteligencia artificial surgió originalmente en la asamblea de Dartmouth en EEUU, en el cual un conjunto de eminentes hombres de ciencia versados en materias de la computación tales como John McCarthy y Marvin Minsky, se congregaron con el propósito de conseguir comprender e intentar investigar cómo construir maquinaria con cierto grado de inteligencia, que tuviera la capacidad de entender un lenguaje para solucionar problemáticas, mediante el aprendizaje y, a través de feedback, hacerse mejor cada vez. Afortunadamente, obtuvieron reducciones

esenciales que conforman la piedra angular de la Inteligencia Artificial en nuestros días: Primero, admitir que nuestro cerebro no es él único que piensa, sino que también puede suceder en el exterior, como en una máquina; la inferencia de que el pensar se le es capaz de entenderlo de manera científica mediante máquinas como el ordenador (Cairo, 2011., p. 7).

Sorprendentemente, tiempos atrás, previo a la noción de la expresión, varias investigaciones fueron hechas al referente. En el año de 1943, se elaboró el informe el cual daba a conocer la estructura primaria y la conducta de las células nerviosas por el investigador Walter Pitts y el médico Warren McCulloch. En el año de 1950 el filósofo y científico Alan Turing dio a conocer Computing Machinery and Intelligence con la cual mostró su popular Test de Turing. El mencionado test se puede resumir en que si una máquina actúa en todos los puntos de vista como dotada de inteligencia, entonces se tiene que aceptar como tal: inteligente. En 1951 se dio la elaboración del primer simulador de redes neuronales (Cairo, 2011., p. 6).

Como dato extra: Aproximadamente 2000 años atrás, Herón de Alejandría,matemático, pensador, diseñador de aparatos de vapor y cajas de engranajes, redactó Autómata, la primera obra conocida que hablaba sobre robots (Sagan, 2010, p. 19).

En 1963, en la General Motors se instaló el primer robot industrial, construido por Joseph Engelberger. En el año de 1965 apareció el software ELIZA, creado por Joseph Weizenbaum, el primero que era capaz sostener una conversación en inglés, prácticamente sobre cualquier tema. Sin mucho tiempo, en 1974 se construyó un sistema experto denominado MYCIN, el cual diagnosticaba infecciones de la sangre. MYCIN era capaz de dar a conocer el proceso de razonamiento que obedecía con elm objetivo de conseguir su diagnóstico, y recetaba medicinas según la necesidad de cada atendido, según su edad, estatura peso, etc. En 1983 se dio a conocer al mundo el primer robot autónomo, el cual era manejado por ordenador. El robot fue capaz de cruzar una sala completa de obstáculos en alrededor de 5 horas, sin la ayuda de ninguna persona.

En 1997, a mediados de año, se llevó a cabo un hecho muy importante para la Inteligencia Artificial: por primera vez en la historia una computadora le ganaba al ganador mundial de ajedrez. La Deep Blue, computadora de IBM capaz de realizar doscientas millones de

jugadas cada segundo, le ganó al ruso Garri Kaspárov en una partida de ajedrez. En el año de 2007 Ernst Dickmanns, elaboró un auto autónomo. El automóvil consiguió andar una distancia de casi 2000 kilómetros, en calles transitadas. En 2008 la universidad de Osaka construyó e hizo una demostración de un robot con apariencia humanoide, teniendo una cantidad de formas en su rostro, y era capaz de realizar una gran cantidad de gestos y ademanes, y constaba a 50 sensores, lo que la hacían inteligente. (Cairo, 2011., pp. 9-12)

Con la ayuda de la IA la NASA produjo CURIOSITY, una máquina que llegó a Marte en 2012, asombrando a la humanidad con las fotografías (360cities, 2013) capturadas, y los distintos tests que ha hecho sobre la constitución de la superficie del planeta. (RT en Español, 2014).

En 2014, en Junio, se consiguió un gran avance para la IA: por primera vez en la historia de la humanidad un programa de computadora consiguió ser mejor que la prueba de Turing, confundiendo a una persona interlocutor, y engañándole con la idea que una persona contestaba sus interrogantes. Lo sucedido tuvo como escenario un concurso hecho por la Universidad de Reading, Londres. El software ruso Eugene, logró fingir que era un joven de 13 años. Este hecho se realizó en la Royal Society de Londres, dando por resultado que una computadora sería mejor que el Test, engañando a la persona interlocutora más del 30 % del cronómetro en una sucesión de diálogos de 5 minutos. Esto es un avance para la Inteligencia Artificial ya que el software consiguió confundir a las personas el 33 % del cronómetro. (El País, 2014).

Adicionalmente, en 2015, Google (el gigante de internet en la actualidad) elaboró una inteligencia artificial que podía aprender, desde cero, Atari 2600. El software mejora de sus errores tras intentar varias veces, incrementando su aprendizaje, hasta conseguir ir más adelante que los expertos humanos. Con esta clase de softwares se percibe que en los años que vienen se podría construir soluciones a problemas del día a día, por ejemplo, la aplicación a automóviles inteligentes (Hipertextual, 2015).

Cabe destacar que los ejemplos citados anteriormente dan a entrever solo algunos de los desarrollos y evidente potencial que la IA ha logrado. Tenemos que considerar que esta rama de la ciencia es compulsivamente cambiante a diario, en el sentido de que va en

aumento, y mejora sin parar, transformando lo que antes solo estaba en nuestra marginación, en cosas que vemos y usamos en nuestro día a día.

2.2.5 Machine Learning

Esta rama de la IA, o en inglés Machine Learning, son un conjunto de pasos o algoritmos capaces de analizar y procesar una enorme cantidad de datos, y localizar relaciones y patrones que las personas no vislumbran (McNulty K., 2018). Esta clase de algoritmos no es algo, tal cual, nuevo, sin embargo, la diferencia con los otros métodos de aprendizaje es su habilidad de aprovechar una superior cantidad de información.

Hoy por hoy lo que engloba la IA son muchas ciencias y ramas. Pero simplificadamente se podría aceptar como máquina con Inteligencia Artificial aquella capaz de resolver problemas como lo haría una persona. En el contexto de la IA, surge el Machine Learning a manera de una subrama. Consecuentemente al Machine Learning se lo podría definir como aquellos algoritmos cuya eficiencia es cada vez mejor en proporción a la cantidad de datos que se brindan a través del tiempo. Cabe recalcar que el Machine Learning se divide en otras ramas más pequeñas, por ejemplo, el Deep Learning. Éste último se basa en Redes de Neuronas con algunos niveles de procesamiento, las que a su vez reciben una enorme cantidad de información.

2.2.6 Imágenes digitales

Se define como imagen a la representación de un objeto (persona, animal, objeto, otros). Para conseguir que las imágenes puedan ser procesadas por computadora, se tienen que digitalizar, ya sea en espacio y en amplitud. Matemáticamente las imágenes digitales se pueden representar como una función f(x,y), en donde “f ” es la amplitud; en otras palabras, la tonalidad; y “x” e “y” son las coordenadas en el espacio de la imagen.

Si los valores de “f ”, “x” e “y” son no continuos, o sea discretos, la representación es llamada imagen digital, la cual se puede conseguir mediante un aparato electrónico. Esta imagen digital se puede transcribir matemáticamente mediante una matriz de M fila por N columnas, lo que se aprecia en la siguiente imagen:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

(Ecuación 1)

Este proceso de digitalización está enmarcado en dos etapas: cuantificación y muestreo. Todas las partes de la imagen tienen tonalidad o amplitud. Si la imagen sólo fuera en blanco y negro, "f " determina la cantidad de gris contenido píxel por pixel. Este mecanismo de digitalización de la función "f " se lo llama por los investigadores cuantificación

El muestreo es la digitalización de "x" e "y"; en otras palabras, a la partición de la imagen análoga en secciones, en donde todas estas pequeñas partes tienen puntos (x,y), entonces aquí "x" e "y" son números discretos. A todos los elementos se les denomina píxel (González, 1996).

La cantidad de bits empleados con el propósito de representar un pixel consigue establecer el número de amplitudes o tonalidades de la imagen. Esto es denominado bits por pixel o profundidad de color. En otras palabras, si la profundidad es de n bits, se conseguirán 2^n tonalidades o valores diferentes.

Se aprecian tres ejemplos de profundidad de color en la siguiente imagen:

Fig. 2a, de 1 bit por píxel: $2^1 = 2$ tonalidades (monocromo).

Fig. 2b, de 8 bits por pixel: $2^9 = 512$ tonalidades (escala de grises).

Fig. 2c, de 24 bits por pixel: $2^{24} = 16777216$ tonalidades (formato RGB).

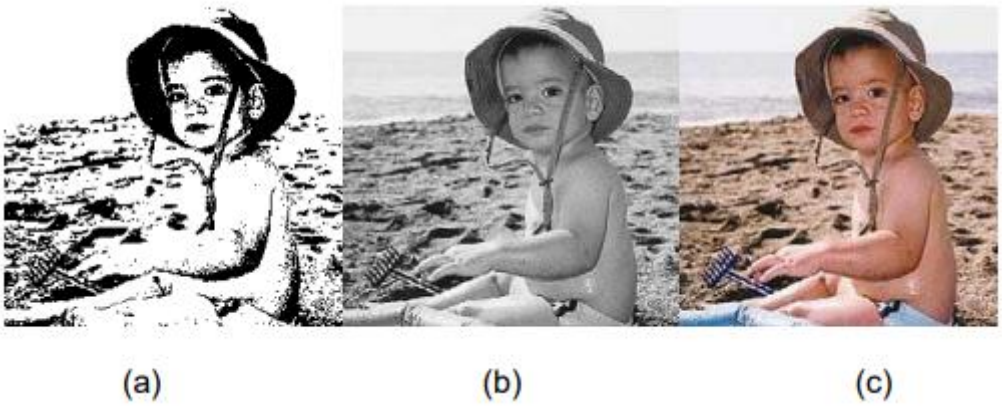


Ilustración 2: Imágenes con distintos bits por pixel.

(a) Imagen monocroma. (b) Imagen en escala de grises. (c) Imagen en formato RGB

(Fuente: González, 1996)

En este proyecto haremos uso de imágenes monocromas, ya que son más fáciles de procesar, ya que solo puede haber un valor en un píxel: 0 o 1. Al acortamiento del valor "n" se le llama *cuantización*.

La cantidad de bits indispensables para guardar en memoria una imagen digital se consigue según la expresión matemática de la ecuación 2:

$$b = M \times N \times n$$

(Ecuación 2)

"M" y "N" connotan la resolución espacial, y "n" determina las tonalidades. La resolución es la claridad que hay en la imagen digital.

En la imagen 3, se percibe el mecanismo para decrementar la resolución espacial. La imagen 3a es una imagen de 500 x 700 píxeles de 8 bits por píxel; en otras palabras, 256 escala de grises. Considerando la anterior ecuación, los "M", "N" y "n" serían 500, 700 y 8 respectivamente. La imagen 3b es la misma figura pero con menos resolución espacial que la anterior imagen, ya que la cantidad de píxeles es la mitad que la anterior, ya sea en filas y columnas (250 x 350 píxeles), sin embargo mantiene 256 escalas de grises (n = 8

bits). De igual forma, se ha rebajado la cantidad de píxeles para las imágenes de 3c a 3f, sin embargo, aun conservan los 256 escalas de grises (González 1996)

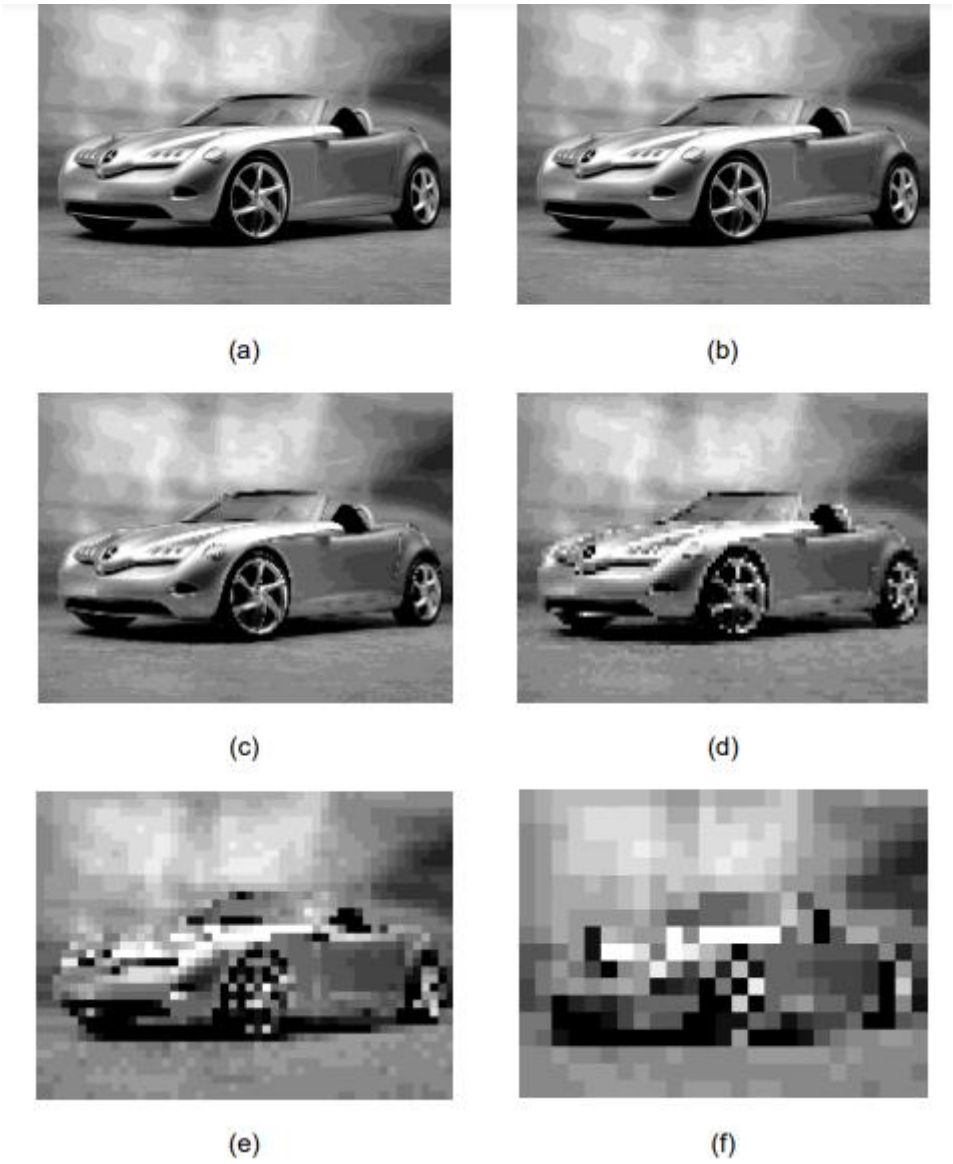


Ilustración 3: Imágenes cuya resolución espacial se va rebajando

Por ilustración, se ha disminuido la cantidad de píxeles a la mitad respecto de la anterior ilustración, ya sea en las columnas y en las filas.

(González 1996)

En la Ilustración 4 se aprecia cómo cambian las imágenes al rebajar la resolución de tonalidades. En la imagen 4a se aprecia una imagen de 500 x 700 píxeles, y en cada píxel

8 bits. En las imágenes que le siguen, la cantidad de bits en cada píxel va reduciéndose una unidad; dicho de otra forma, la escala de grises va menguando a 128, 64, 32, 16, 8, 4 y 2 respectivamente. Pero la resolución espacial se mantiene igual (500 x 700 píxeles). En cada figura con 256, 128 y 64 niveles de gris, no existen diferencias que resaltan, lo que se ve mejor en las ilustraciones de 4a a 4c:

(a)

(b)

(c)

(d)

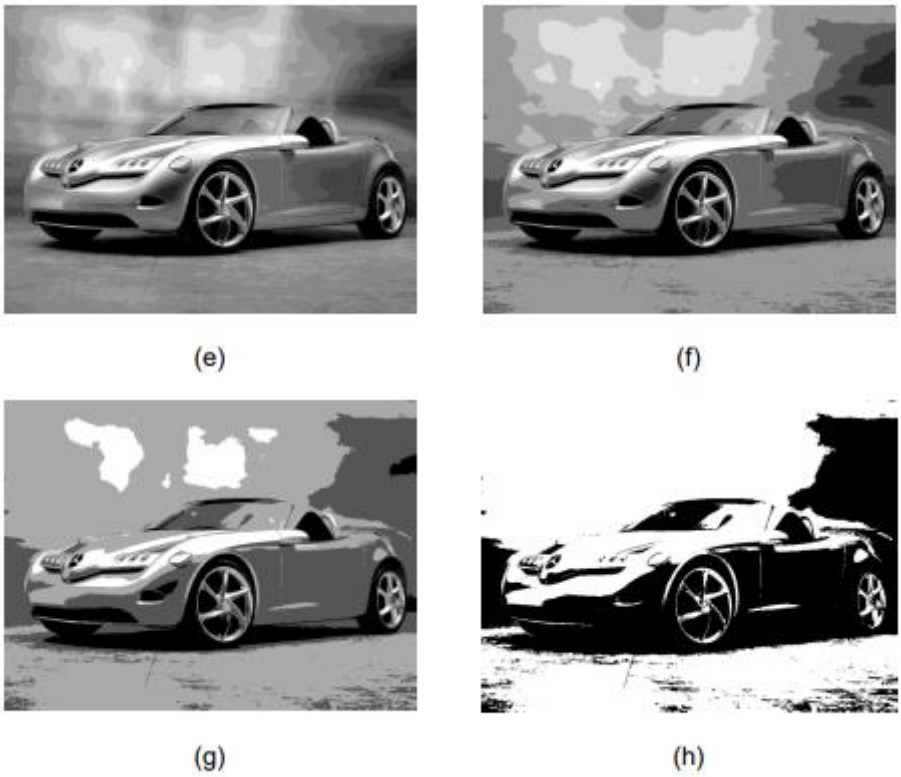


Ilustración 4: Imágenes con disminución de tonalidad

Sin embargo, se sigue conservando la resolución espacial.

Se percibe que desde la imagen 4d, la cual tiene 32 escalas de gris, comenzamos a percibir una diferenciación en la imagen, respecto de las otras anteriores, y surge lo que se conoce como falsos contornos. Lo mencionado se percibe con más fuerza comenzando en la imagen 4e, en donde las diferencias son notablemente superiores. La causa de esto es las escasas tonalidades que hay es estás ilustraciones (González, 1996).

Clases de imágenes digitalizadas

Según García (2008), las imágenes digitalizadas son clasificables en 4 grupos: imágenes binarias, escala de grises, indexadas y RGB (p.21). En este proyecto nos interesa más que otros las imágenes binarias, pero explicaremos algunos aspectos de cada tipo de imágenes, aunque luego nos centraremos solo en las imágenes binarias.

A. Imágenes Red-Green-Blue o, simplificado, RGB

Este tipo de imagen surge de la mezcla de los tres colores primarios, los cuales son el rojo (R), el verde (G) y el azul (B). Este tipo RGB es una presentación de color sumatorio. Dicho de otra forma, que la mezcla de colores es la suma de los componentes por separado teniendo como fundamento el negro como color. (Cuevas et al., 2010, p.441).

Por otra parte, García (2008, p.22) dice las particularidades de una imagen RGB, y las mencionamos a continuación:

- ✓ Emplean 3 canales para mostrar los colores.
- ✓ Hacen uso de 8 bits por canal (8 bits x 3), dicho de otra forma, 24 bits de color por pixel.
- ✓ Se adapta bien a los formatos de imágenes tales como: PNG BMP, JPG, etc.

En la Ilustración 5 podemos ver una instancia de una imagen RGB, constituida por 3 niveles: Rojo, verde y azul.

Ilustración 5: Imagen tipo RGB de una mano

B. Imágenes de tipo indexadas

Una imagen indexada brinda una cantidad limitada de colores, y está constituido por 2 elementos: una matriz RGB y una matriz de datos. La matriz RGB tiene nx3 filas y columnas, respectivamente, en la cual **n** equivale a la cantidad de colores usados en la imagen. (Cuevas et al., 2010, p.446).

Por otro lado, García (2008, p.22) menciona que este tipo de imágenes indexadas se diferencian por las particularidades citadas a continuación:

- ✓ Mengua los colores en la imagen a un límite superior de 256.
- ✓ Soporta formatos tales como PNG-8, GIF y varias aplicaciones multimedia
- ✓ Mengua el peso de archivo a causa de borrar la información sobre el color.

C. Imágenes a escalas de grises

Cuevas (2010) expresa que este tipo de imágenes son definidas mediante un arreglo bidimensional en donde sus valores fueron escalados con el propósito de reproducir una determinada cantidad de intervalos" (p.35).

Por su parte, García (2008, p.22) menciona que una imagen de este tipo porta estas particularidades::

- ✓ Emplea unos límites de tonos grises.
- ✓ Una imagen que tiene 8 bits por pixel, conseguirían 256 tonalidades de grises, como máximo.
- ✓ Un pixel emplea un valores limitados entre 0 (negro) y 255 (blanco).

En la Ilustración 6 percibimos un ejemplar de este tipo de imágenes en los límites de 0 a 255; Sin embargo los mencionados parámetros están en una capa unitaria, contrastando con el tipo de imágen RGB la cual tiene 3 capas.

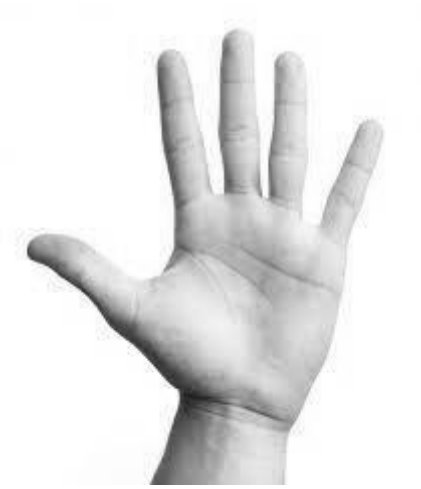


Ilustración 6: Imagen de una mano a escala de grises

D. Imágenes de tipo binarias

Una imagen de este tipo se puede representar matemáticamente como matriz que sólo tiene ceros y unos, dicho de otra forma, los ceros y unos son representaciones que expresan el estado de activo (1) o desactivo (0) (Cuevas et al., 2010, p.35).

Observando la Ilustración 7 se percibe un ejemplo de imagen de tipo binaria. Aquí el 1 representa al color blanco y el 0 al negro. El negro es el background.

Ilustración 7: Imagen de tipo binaria de una mano

2.2.7 Visión artificial

Se la puede definir como la habilidad que tienen las máquinas para poder tratar y analizar imágenes brindadas, o capturadas de su entorno, con el propósito de que esta máquina haga automáticamente una labor en específico. (Roguiguez y Sossa, 2012, p.19).

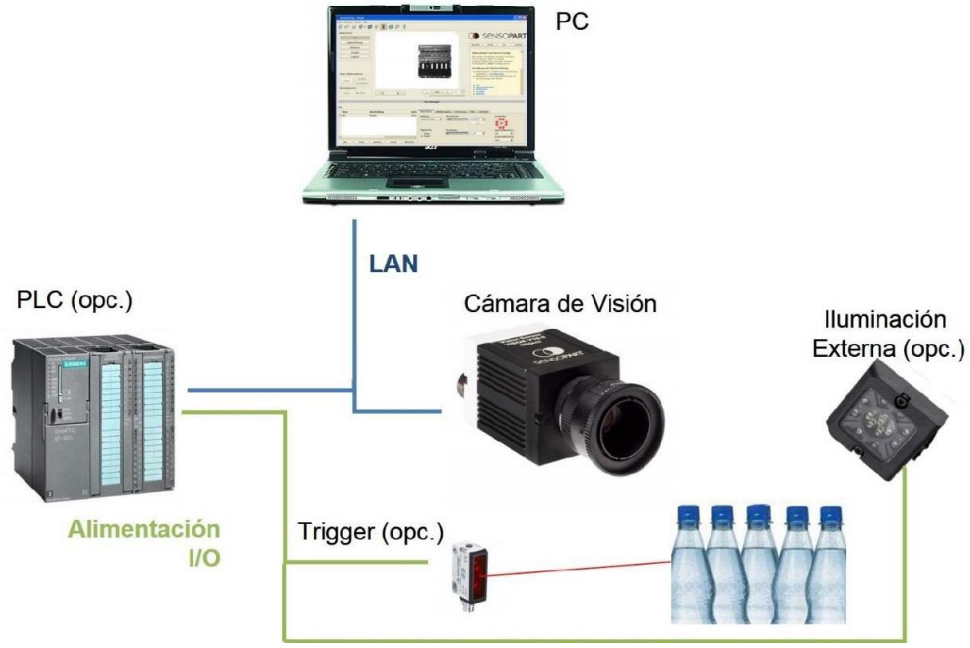


Ilustración 8: Un esquema de visión artificial de manera práctica empleado en la producción.

(Fuente: Contaval, 2016)

Observando la Ilustración 8 percibimos un mecanismo de visión artificial para mantener el nivel de las botellas. Sus elementos indispensables para mantener el nivel serían: controlador PLC, ordenador (computadora), cámara de visión e iluminación.

Fases del procedimiento para conseguir visión artificial

Sus fases se particionan en 6 ramas importantes, las que se asignan según su delicadeza y complicación requeridos para su elaboración. Esto lo vemos en la Tabla 1. Se aprecian 3 escalas categorizadas. (García, 2008, p.27).

Etapas del procesamiento digital de imágenes

Proceso del PDI	Nivel de Visión
Captura/Adquisición	
Preprocesamiento	Bajo
Segmentación	
Descripción	Medio
Reconocimiento	
Interpretación	Alto

Fuente: (García, 2008, p. 27)

Tabla 1: Fases del tratamiento digital de imágenes

Los procesos o métodos del tratamiento de imágenes digitales (PDI) se engloban en un par de grupos (García, 2008, p. 27), lo que son apreciados en la Tabla 2:

- ✓ Procesos que tienen por entrada y salida imágenes
- ✓ Procesos en donde sus entradas son imágenes y lo que resulta o la salidas son características obtenidas de las mismas.

Entradas y salidas de los métodos del PDI

NIVEL	MÉTODOS/PROCESOS	ENTRADA	SALIDA
Bajo	Reducción de ruido Realce de contraste Realce de características	Imagen	Imagen
Medio	Segmentación (regiones, objetos) Descripción de objetos Clasificación o Reconocimiento	Imagen	Atributos de objetos: bordes, contornos, áreas identidades de objetos individuales
Alto	Interpretación Análisis de la imagen Funciones cognitivas	Objetos encontrados	Análisis de la imagen (información, sentido a los objetos.)

Fuente: (García, 2008, p. 27)

Tabla 2: Entradas y salidas del proceso del PDI

Captación de imágenes

Es lo que apertura un tratamiento digital de imágenes. Este tiene que ver con capturar una imagen mediante un equipo de obtención de imágenes, tales como un lente digital y/o análogo, entre otros. (Roguiguez y Sossa, 2012, p.25)

2.2.7.1 Cámara de tipo digital

En este proyecto empleamos una cámara digital, consecuentemente es primordial que sepamos las particularidades técnicas de este tipo de dispositivos. Estas son mencionadas y tratadas a continuación:

- Resolución
- Memoria
- Comprensión de archivos

Resolución

La resolución en un imagen fotográfica es la cantidad de píxeles que tiene la fotografía. Las cámaras digitales más empleadas reproducen imágenes de 640x480 píxeles. Existen cámaras que se pueden personalizar y utilizar diferentes resoluciones. (García, 2008, p. 22).

Memoria

Se define como la habilidad de grabar videos o imágenes. Si tiene mucha memoria es ventajosamente mejor el dispositivo. Las memorias son útiles ya que permite descargar las imágenes en otros dispositivos como en un ordenador. Las cámaras conllevan una memoria integrada, sin embargo se le puede poner tarjetas de memoria externas. (García, 2008, p. 22).

Compresión de archivos

Si una imagen tiene baja comprensión, entonces tiene mejor calidad; y si tiene mucha comprensión, tendrá baja calidad. Esto es una relación inversa. También existe la desventaja de que si la resolución de las imágenes son muy altas, estas tendrán más peso, y el almacenamiento de la cámara se llenará rápidamente. (García, 2008, p.22).

Preprocesamiento

El preprocesamiento es un mecanismo que permite conseguir una imagen digital, empleando las siguientes técnicas: Disminución de ruido existente en la misma, expansión del contraste y algunos elementos presentes. Cabe recalcar que el resultado de este mecanismo es otra imagen (Roguiguez y Sossa, 2012, p.26).

Segmentación

Esto hace referencia al mecanismo de nivel intermedio en la visión artificial, el cual tiene que ver con seleccionar una imagen digital en focos buscados de lo otro que no se desea de la imagen: Estas características buscadas pueden ser el tamaño, la forma, la longitud, el color, el brillo, entre otros (Roguiguez y Sossa, 2012, p.155). Este mecanismo es muy importante cuando hablamos de procesamiento de imágenes ya que se puede conseguir información apropiada o inapropiada de la imagen que está siendo trabajada o procesada.

Algunos ejemplares de segmentación podrían ser: encontrar las manos, ojos, rostro, etc de una persona al posarse frente de una cámara; seccionar las letras de un párrafo en una figura, enfocar las placas de los automóviles en movimiento, determinar dolencias teniendo como estrada imágenes médicas, entre muchos otros. (García, 2008, p.71)

En la Ilustración 9 se aprecia gráficamente el mecanismo de segmentación. Se observa que previamente al iniciar el mecanismo, hay una imagen digital bruta, con diferenciadas escalas de luminosidad, sin embargo, después de la segmentación el resultado son datos de los elementos presentes.

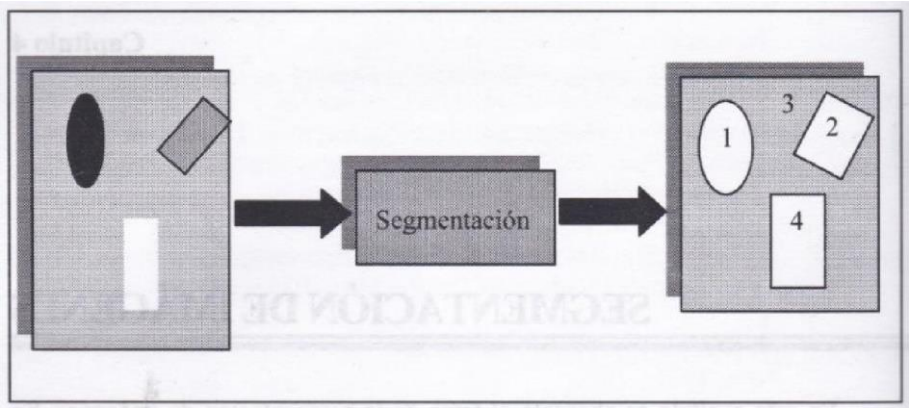


Ilustración 9: Modelo que ejemplifica el mecanismo de segmentación

(Fuente: Roguiguez y Sossa, 2012, p.156)

Descripción

Es un mecanismo de categoría intermedia de visión artificial, el cual se puede definir como el procedimiento para localizar las características importantes de una figura que ha sido segmentada con anterioridad. Este procedimiento permite distinguir un objeto de otro objeto diferente, teniendo como base el tamaño, la forma, el área, el color, entre otras características. (García, 2008, p.29)

2.2.7.2 Reconocimiento

Es una fase de escala intermedia de visión artificial, la cual permite localizar los objetos a la clase respectiva previamente determinada. Antes de ello se clasifica cada objeto o elemento a un grupo o categoría correspondiente. (García, 2008, p.29)

Ilustración 10: Técnica de categorización de elementos

(Fuente: García, 2008)

Observando la Ilustración 10 percibimos la técnica de agrupación de una imagen, clasificando las categorías previamente definidas, que marchan en una cinta transportadora. La entrada es una imagen con llaves y monedas, sin embargo tras el mecanismo o técnica de segmentación conseguimos 4 objetos diferenciados y clasificados. De esta manera, haciendo la descripción y extracción de cada objeto, se agrupa una categoría para así determinar qué tipo de objetos hay en la imagen. Todo es realizado automáticamente.

2.2.7.3 Técnicas de categorización de patrones

Existen varias técnicas para la categorización de patrones. De acuerdo de los parámetros empleados, por ejemplo: la manera de elaborarse, la clase de muestra, de cuantos datos podemos hacer uso, entre otros. (García, 2008, p.80)

- ✓ **Pattern Matching, (adaptación, en español)**

  Expresa cada clase a través de un patrón prototipo. Como ejemplos de este tipo de técnicas tenemos:

  - Categorizador para corto espacio
  - Adecuación bajo contexto (García 2008, p.80)

- ✓ **Clasificadores estadísticamente óptimos**

  Esta categoría de clasificador engloba el Clasificador Bayesiano para clases gaussianas (García 2008, p.81).

✓ **Redes neuronales**

Tiene que ver con un grupo de métodos que brindan posibilidades para obtener un resultado al categorizar.(García, 2008, p.81)

En la presente investigación obtendremos la clasificación de las imágenes mediante este tipo de técnica.

2.2.7.4 Interpretación

García (2008, p. 29) menciona que es un mecanismo de escala mayor de visión. Tiene que ver con categorizar a un grupo de elementos u objetos que han sido categorizados con anterioridad (animales, manos, figuras, llaves, rostros, automóviles, etc.) en donde se tiene que apreciar de una forma práctica los elementos u objetos en una imagen.

2.2.8 Redes neuronales artificiales

El comienzo de estas redes empieza en los años 1900. Sin embargo, no fue sino hasta los años 1940 y 1950 que las investigaciones de las redes neuronales tuvieron un incrementado impacto, y esto se debió a la corriente de pensamiento denominado Conexionista. Esta corriente ideológica apoyaba la idea que la clave para el conocimiento y el aprendizaje estaba en verdades incuestionables o axiomas, sumado a la idea de que el conocimiento no tiene que ver con el diseño que emplee los caracteres. Sumado a que la expresión del saber comienza en el nivel más simple. (Tawfiq, 1999)

Estas redes (llamadas por lo general en español RNA, y en inglés ANN, por Artificial Neural Networks) son redes interrelacionadas masivamente en paralelo y secuencialmente, formada por hardware y software, con la habilidad de realizar una conducta en colectivo consiguiendo relacionarse con otros elementos, tal y cual el aparato nervioso de un ser humano o animal (Blum, 1992).

(Anderson, 1995) La RNA es un modelo computacional semejante a las redes neuronales biológicas. Se la entiende por el mecanismo de tratamiento de datos. Es una arquitectura

seccionada, la cual está constituida de neuronas artificiales (denominadas elementos de procesamiento), relacionadas por una enorme cantidad de conexiones (sinapsis), las que se utilizan para guardar conocimiento que puede ser usado después.

Además, las RNA consiguen imitar al cerebro de una persona, y esto se debe a la interconexión de muchos componentes de procesamiento, y todos ellos indistintamente presentan un comportamiento totalmente focalizado.

Las particularidades esenciales de las RNA las mencionamos a continuación: (Jaramillo, 2005)

- Mejoran continuamente: Las RNA logran cambiar su conducta dependiendo de su entorno. Si tenemos un grupo de llegadas, las RNA pueden acomodarse emitiendo salidas inteligentes.

- Hacen diferencias entre elementos previos a recientes: Cuando la RNA ya haya sido entrenada, sus respuestas son insensibles a insignificantes diferencias en las entradas, capacitándose de una manera muy potente para la clasificación.

- Procesamiento previo a salidas: Las RNA consiguen datos de un grupo de llegadas. Si hablamos del reconocimiento de patrones, la RNA se podría entrenar con una sucesión de patrones distorsionados de un carácter. Pero cuando la red ya está debidamente entrenada, esta podrá emitir un resultado apropiado frente a una entrada distorsionada, implicando que ha podido aprender algo nuevo que no ha visto anteriormente.

Tal como lo menciona Hilera (1995), las RNA son modelos matemático estructurados por un grupo de pequeñas partes de procesamiento, las que denominamos neuronas, relacionados por varios puentes de intercambio de información conocidos como conexiones, con el propósito de admitir entradas, y, tras mecanismos de procesamiento, dar como resultado salidas.

Todas las relaciones se ligan a un ajuste equivalente a los datos empleados por neuronas solucionando así una problemática.

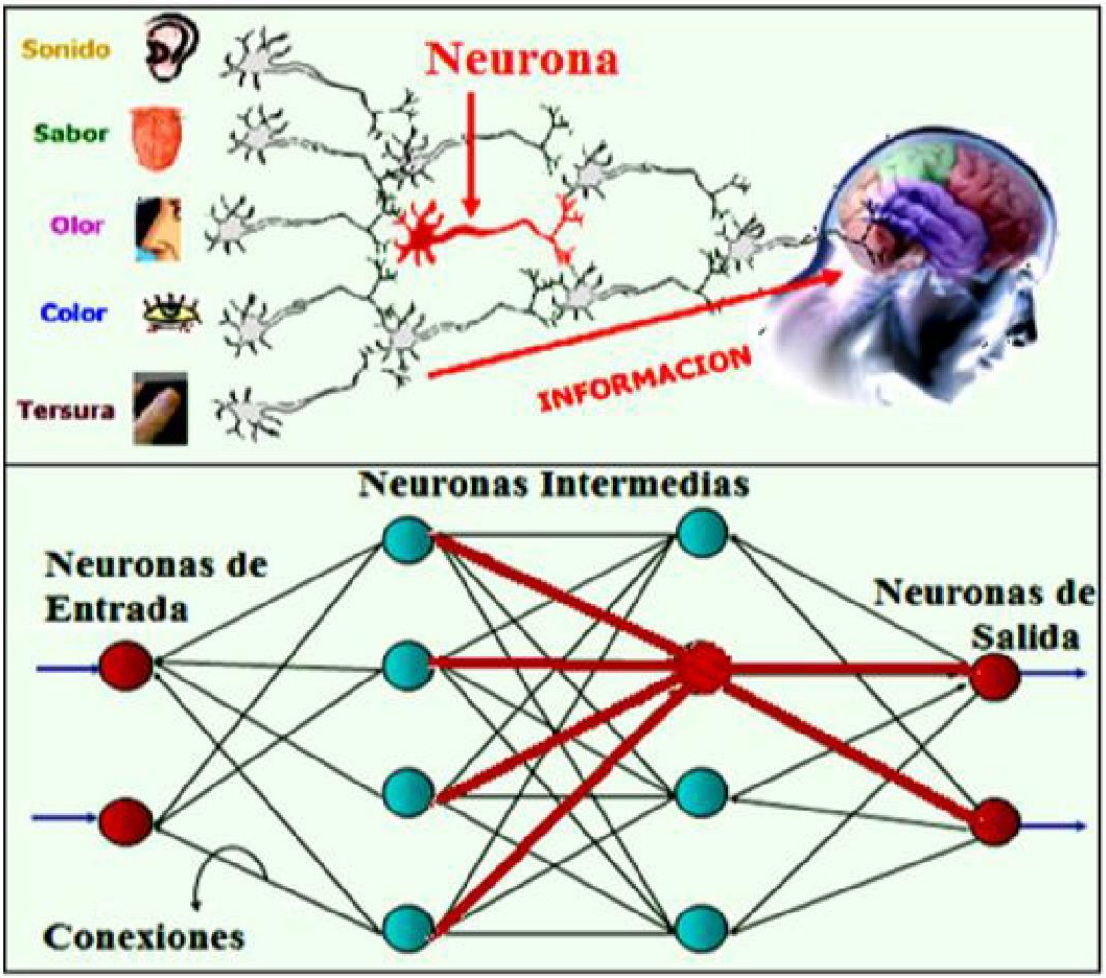


Ilustración 11: Comparación del RNA con una biológica

(Fuente: Caudill, y otros, 1992)

En la Ilustración 11 se observa que tanto las neuronas artificiales como las biológicas tienen entradas, están asociadas a un peso y emiten salidas.

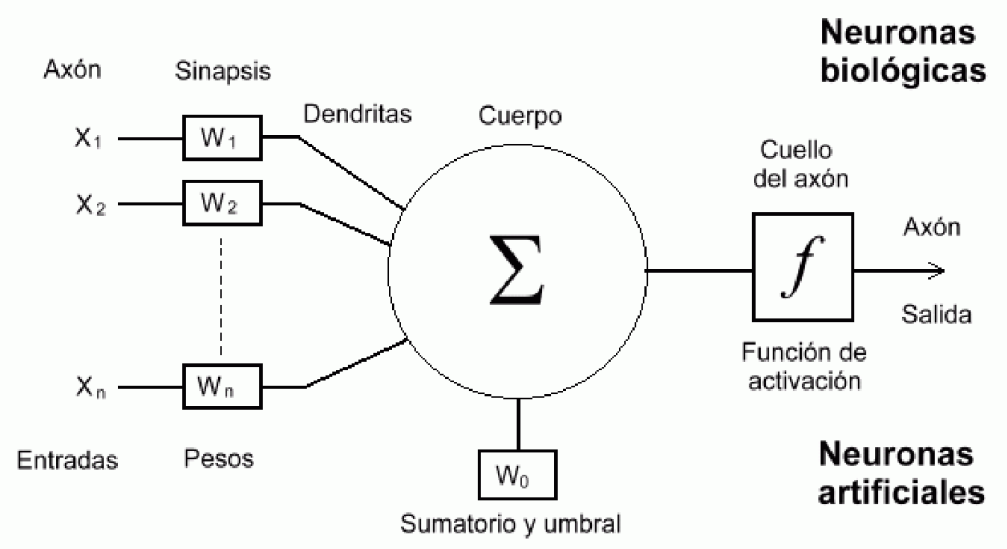


Ilustración 12: Similitudes entre una neurona artificial y una Biológica

(Fuente: Hertz, J. 1991)

2.2.8.1 Elementos de la Red Neuronal

Tal como lo explica Freeman (1991) objetos particulares para el procesamiento que constituyen los diseños para mecanismos artificiales de neuronas, son conocidos como Objetos del tratamiento o Neuronas Artificiales. Todos los elementos ejecutan una labor sencilla: admiten señales y calculan la respuesta para emitir a otros elementos y, también, hace uso de otros factores. El elemento de procesamiento más sencillo tiene la estructura que se ve en la Ilustración 13.

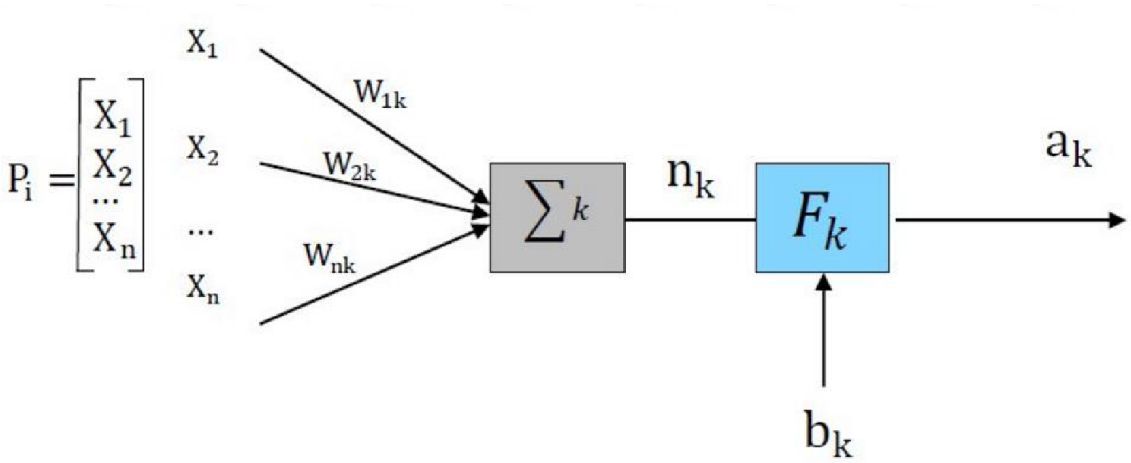


Ilustración 13: Modelo de una Neurona Artificial

(Fuente: Freeman 1991)

$$a = F_k\left(W_{ij}\,P_i + b_k\right)$$

(Ecuación 3)

Tenemos:

- a: es lo que emite la neurona.
- F_k: es la función de transferencia de la neurona.
- Wij: son los pesos.
- Pi: es la secuencia del modelo.
- b_k: valor de procesamiento.

Este modelo porta las características: (Martin, 2007)

- Todas contienen algunas admisiones relacionadas a particularidades distintas.
- Esta llegada de procesado no es obligatoria.
- Podemos clasificar las llegadas en: Reductoras, excitadoras, crecientes, de amortiguamiento o de emisión aleatoria.
- Estas llegadas se ponderan mediante un escalar incrementando o disminuyendo de manera distinta lo que llega.
- Todos los objetos de procesamiento conlleva un índice de ejecución obtenido según la llegada original. En ocasiones, el valor para la activación, varía de acuerdo al valor previo de activación.
- Cuando ya se haya obtenido el índice, se emite otra respuesta haciendo uso de cálculos matemáticos de emisión.

Tal y cual se observa en la Ilustración 14, lo que llega es lo que sale de una neurona de una capa previa. Así mismo en todos los niveles, consiguiendo la emisión de resultados.

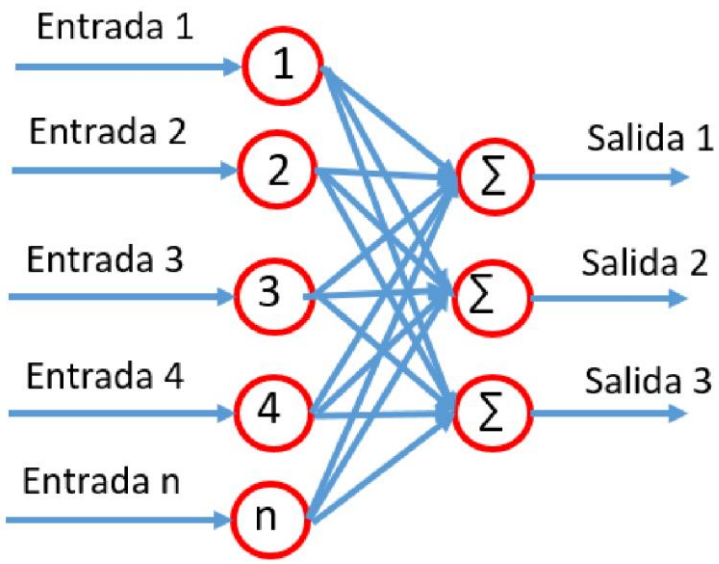


Ilustración 14: Composición de un Perceptrón Multicapa

(Fuente: Martin, 2007)

La cantidad de niveles medios y la cantidad de perceptrón por nivel está determinado por el tipo de tarea que tenga que realizar la RNA.

2.2.8.2 Funciones matemáticas para activación y salida

Sumado sería correcto tener modelos para nuestras funciones de activación (determinan la activación en proporción de lo que llega) y las funciones de salida (determinan lo que se emite en proporción de lo que se activa).

En este contexto la función de activación determina el empleo de la neurona dependiendo de la llegada total y la activación anterior, pero casi siempre en las situaciones es solo una función no menguante de todo lo que llegó. Las clases mayormente utilizadas serían: las funciones tanto lineal, sigmoidal y escalón (Freeman, 1991). Sumado a esto nos encontramos con las funciones de activación: Tangente hiperbólica (tanh), Rectified Linear Unit (relu), softmax, entre otras.

En este proyecto solo se trabajará con las funciones de activación relu y softmax

Función de activación RELU

Esta convierte las entradas cancelando los números menores que cero, admitiendo los mayores que cero:

$$f(x) = \max(0, x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$$

(Ecuación 4)

En este tipo de funciones se aprecia que:

- Activación Sparse – solamente se activará si son mayores o iguales a cero.
- No está acotada.
- Actúa apropiadamente con imágenes.
- Excelente uso en redes convolucionales.

Función de activación SOFTMAX

Esta convierte las emisiones a valores estadísticos, consiguiendo la adición de valores como resultado a la unidad (1).

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

(Ecuación 5)

En este tipo de funciones, percibimos que:

- son empleadas cuando deseamos conseguir expresiones estadísticas
- Lo limita en el rango 0 a 1.
- Extremadamente clasificador.
- Es empleada en estandarizar muchas categorías.
- Eficiencia en las salidas.

2.2.8.4 Clasificación de las Redes Neuronales Artificiales

Las redes neuronales artificiales(RNA) podrían ser agrupadas dependiendo de la manera de aprender, arquitectura, y de qué manera se las emplea. (Charytoniuk. 2000)

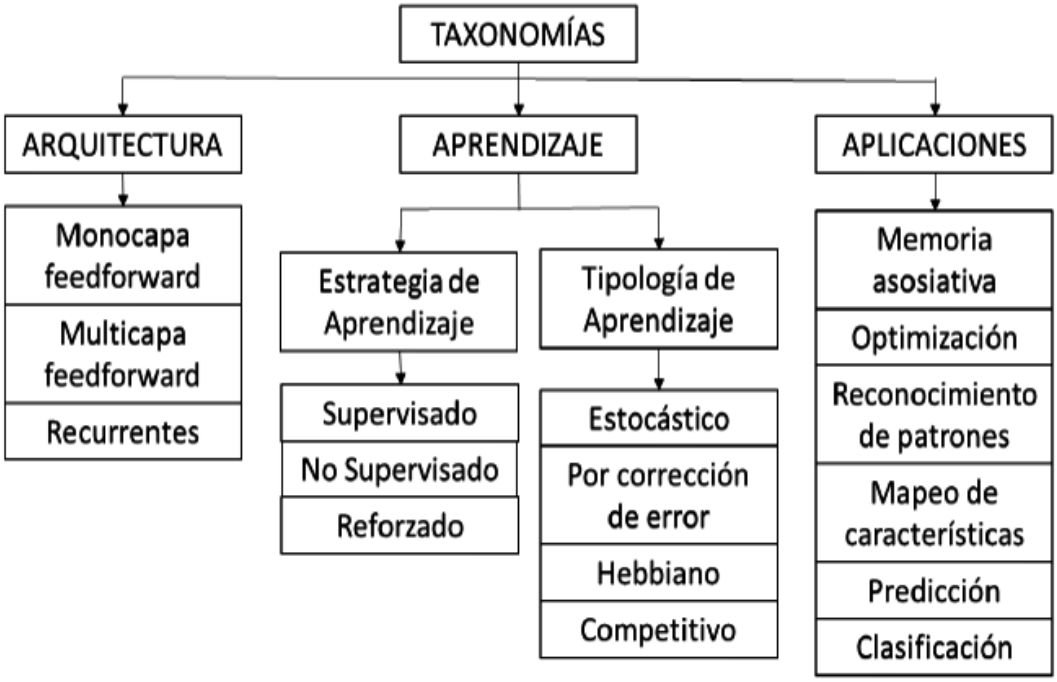


Ilustración 15: División de las RNA

(Fuente: Charytoniuk. 2000)

En el presente proyecto sólo veremos las que se clasifican según su arquitectura, ya que se requiere su explicación porque una de estos tipos de redes es usado en lograr el objetivo de esta investigación. Específicamente las redes neuronales convolucionales (CNN)

Según su arquitectura

En esta arquitectura sus neuronas se encuentran establecidas en capas o niveles ligados por conexiones conocidas como sinapsis. Según cómo se relacionan se agrupan en recurrentes y no recurrentes. (Charytoniuk. 2000)

Cuando cuentan con conexiones hacia Adelante (Feed-forward o No Recurrentes)

Tienen por característica que sus relaciones avanzan crecientemente y en un sentido. Dependiendo de los niveles que tengan se agruparán en Monocapas y Multicapas.

A. Redes Neuronales Monocapa.

Solamente poseen un nivel donde se emiten las llegadas a un nivel emisor, y ahí se ejecutan muchos procesos matemáticos. (Charytoniuk. 2000)

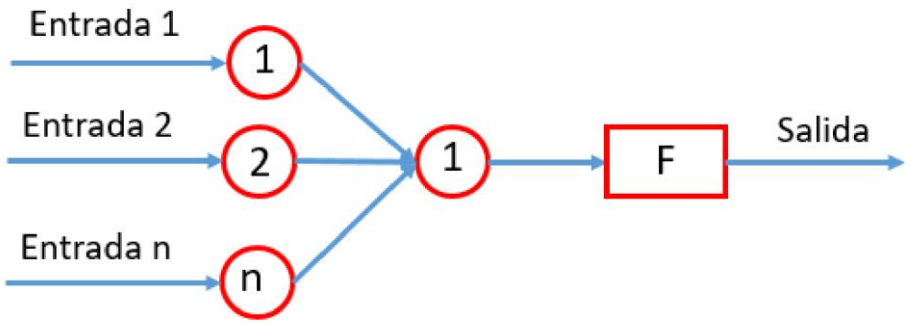


Ilustración 16: RNA con Conexiones hacia Adelante Monocapa

(Fuente: Charytoniuk. 2000)

Para estos casos las RNA con mayor uso en este diseño tenemos: el ADALINE y el Mono Layer Perceptron.

B. Redes Neuronales Multicapa.

Estas son una estandarización de las redes neuronales monocapa, habiendo un grupo contenido entre las llegadas y las emisiones.

Las RNA usadas con este modelo tenemos el Multilayer Perceptron y el MADALINE. (Charytoniuk. 2000). Y también las Redes neuronales convolucionales (CNN)

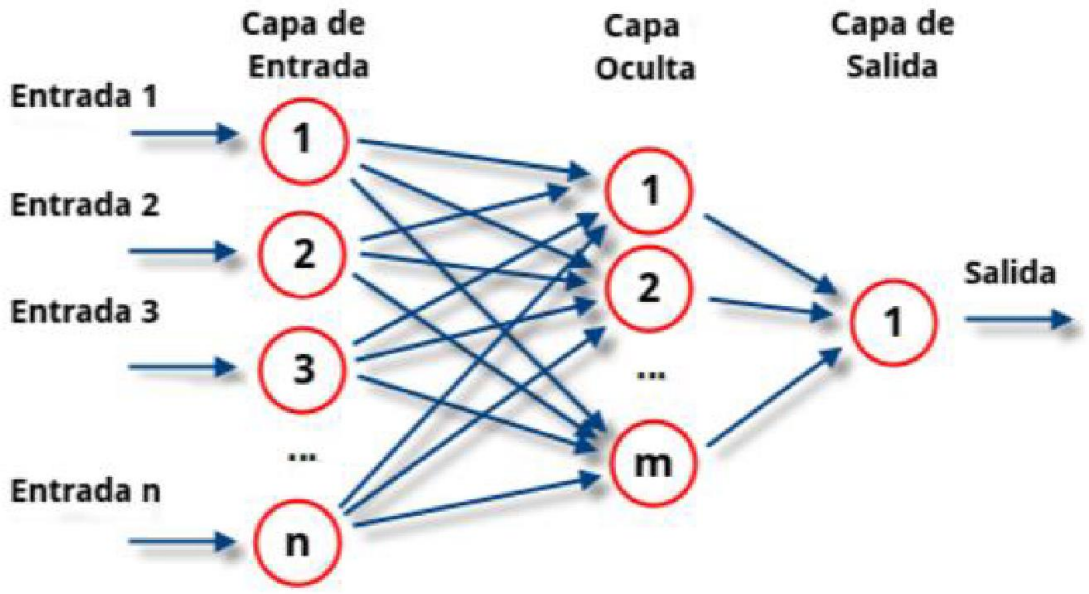


Ilustración 17: RNA conectadas hacia Adelante Multicapa

(Fuente: Charytoniuk. 2000)

Cuando cuentan con conexiones hacia Atrás (Recurrentes o Feedback)

Son diferenciadas por la aparición de relaciones hacia otros niveles previos, relaciones entre un mismo nivel o relaciones mixtas. (Charytoniuk. 2000)

Las conocidas en este diseño tenemos: las redes ART, Hopfield y SOM de Kohonen.

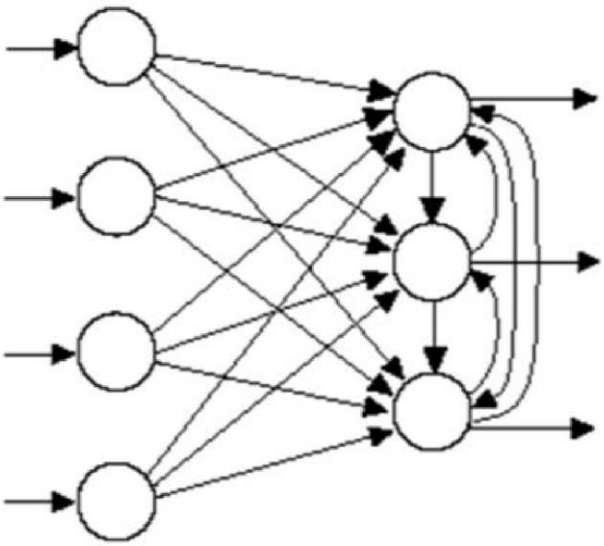


Ilustración 18: RNA con Conexiones hacia Atrás.

(Fuente: Charytoniuk. 2000)

C. Redes neuronales convolucionales (CNN)

Como su nombre lo indica, las CNN provienen de la aplicación de la operación matemática llamada convolución.

Estas Redes Neuronales Convolucionales, o también conocidas por las siglas CNN, son algoritmos de aprendizaje profundo capaz de admitir una imagen, determinar relevancia de valor a algunos detalles y distinguirlos entre sí. El procesado previo necesario en una CNN es bastante bajo que el de otros métodos de categorización.

Una CNN puede conseguir las dependencias temporales y espaciales en una imagen mediante un método de aplicación de filtros. Este esquema hace un mayor ajuste del set de datos e imágenes ya que hace una reducción de la cantidad de parámetros existentes y el volver a usar pesos.

La CNN es una clase de red. Existen tres factores importantes que hacen particular a una CNN de las otras de redes neuronales: Capaz de agrupamiento o submuestreo, campos receptivos locales y pesos compartidos. (D. J. Wu, 2012).

Las arquitecturas CNN, encuentran subcomposiciones focales en la entrada procesada parametrizando todas las neuronas haciendo una dependencia de un set más pequeño dimensionalmente focal de los índices del nivel anterior. En otras palabras, si como valores de ingresos de la CNN tenemos una imagen de 64x64, alguna neurona del inicial nivel oculto dependería solamente de un bloque de 16x16 del todo que es 64 x 64. La agrupación de neuronas en el bloque inicial que tiene que ver con la excitación de una neurona se la conoce como la sección de entrada de la neurona. Mayormente, en las CNN todas las neuronas poseen un área excitatoria focal pero no total. (D. J. Wu, 2012).

La segunda característica que distingue a las CNNs es que sus pesos están enlazados mediante distintas neuronas en las capas que no se ven. Cabe recalcar que todos los elementos de la red inicialmente determinan la relación lineal en las entradas, y este mecanismo es la determinación de un diferenciador lineal acerca de los datos que vienen o entran. Bajo esta idea, intercambiar los índices con varias neuronas del nivel que no se percibe es interpretado como testear el depurador en escena por encima de varias subsecciones de la imagen inicial. Bajo estos criterios se afirma que las CNN calculan un grupo de filtros $F = \{F_i \mid i = 1, ..., n\}$, y todos ellos son aplicados a las subsecciones de lo

que vino. Empleando el grupo de filtros aplicados en la imagen, la red adquiere conocimiento de la estructura o modelo total de la información existente. Limitando los índices para con el objetivo que sean similares mediante distintas neuronas además consigue un evento de normalización sobre la CNN, logrando que la red sea capaz de calcular de una manera adecuada. Se consigue con el compartimiento de pesos que se mengue de manera considerable la cantidad de parámetros sueltos de la CNN, consiguiendo que al entrenarse sea mejor y más eficiente. Además, inspeccionar un depurador F en cada frame de la imagen entrante I es igual a hacer una operación matemática de convolución de la imagen I con el depurador F. En este contexto, la fase convolucional en la CNN, coge el frame entrante y se opera con los depuradores F consiguiendo el grupo de salidas convolucionales (mapa de filtros). (D. J. Wu, 2012)

El elemento característico último de una CNN es la existencia de capas de agrupamiento la cuales tienen dos metas: menguar las dimensiones de las salidas convolucionales y transferir un minúsculo nivel de invarianza traslacional a nuestro modelo. La perspectiva aceptada es mediante categorización espacial (Y. L. Boureau., y otros, 2010). Cuando hacemos la categorización espacial, el grupo de respuestas convolucionales inicialmente se parte en un grupo de mxn conjuntos (mayormente heterogéneos), y entonces se inspecciona una métodos de categorización hacia las salidas emitidas en los bloques. Su mecanismo brinda una respuesta como un conjunto de salidas más chicas de mxn filas y columnas. Sin embargo, en la categorización extrema (max pooling), la salida emitida hacia todos los sets es el índice mayor del grupo de salidas, y si es agrupamiento medio (average pooling), lo que se emite es el índice central del conjunto saliente (D. J. Wu, 2012). En la Ilustración 19 observamos una instancia de convolución. Aquí, las salidas de convolución es un grupo de 4 x 4, hacemos una operación de segmentación media en 4 grupos de 2 x 2, la salida es la media de los índices. Como acto seguido de hacer el mecanismo de agrupamiento medio, la salida es un grupo de 2 x 2, analizándolo al primero que era de 4 x 4, esto es equivalente a una reducción de las longitudes de la salida (D. J. Wu, 2012).

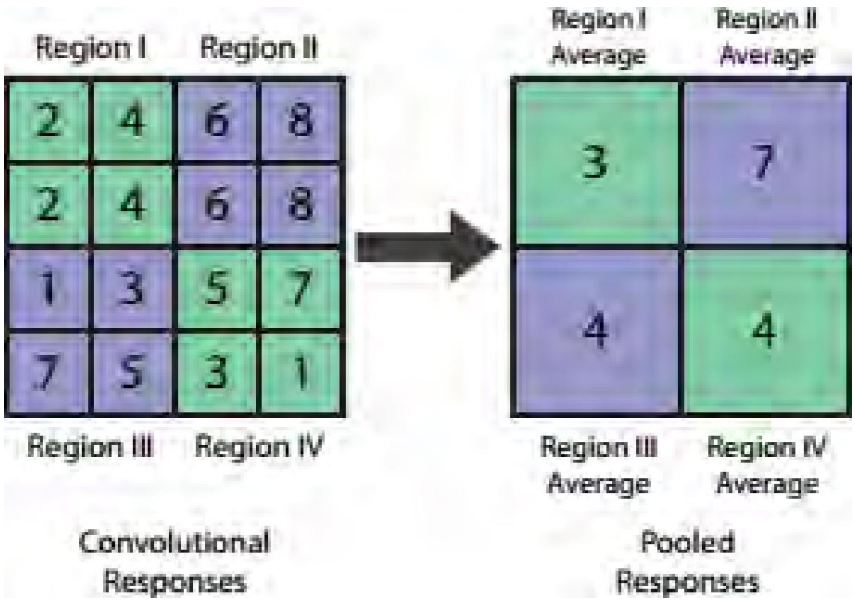


Ilustración 19: Agrupamiento promedio en una CNN

(Fuente: D. J. Wu, 2012)

En las CNN estándares, existen muchas capas, intercambiándose entre capas de agrupamiento y de convolución. De igual forma, podríamos poner una capa más en la sección siguiente de las respuestas del nivel inicial; de esta forma, se procesan las respuestas del grupo inicial siendo esta la llegada al siguiente grupo. Así somos capaces de elaborar una arquitectura con muchas capas o profunda. De esta forma se deduce que los depuradores convolucionales iniciales, brindan una capa inferior de procesamiento de la información que llega. En la situación de la información contenida en una imagen, estos filtros primarios podrían ser depuradores de bordes. Si nos desplazamos a niveles más bajos en la estructura, la red es capaz de manejar crecientemente más información complejas. Si empleamos muchas capas y elevadas cantidades de filtros, la arquitectura CNN es capaz de brindar un potencial de representación de mucha complejidad. Para entrenar a una CNN, usamos el método de propagación del error en reversa. (C. M. Bishop, 2006)

2.2.8.5 Cómo hacer que aprenda una red neuronal

Previo a comenzar la fase de aprendizaje, tenemos que asignar parámetros, por ejemplo, las funciones de activación empleadas, y también el número y el tipo de bloques. Este proceso es un mecanismo de retroalimentación de los valores en la CNN. Lo conseguimos empleando un mecanismo conocido como propagación del error hacia atrás (C. M.

Bishop, 2006), cuyo propósito es reducir una función de pérdida; o también, el objetivo es modificar los valores en la CNN tanto que las predicciones se acercan a las predicciones o clases previamente establecidas.

2.2.8.6 Funciones de activación

Estas funciones presentes en un nodo tiene como objetivo establecer la respuesta de bloque, teniendo previamente un valor entrante. Por lo general, lo que emiten estas funciones se sitúan en un margen calculado entre (0, 1) o (-1, 1)

Función Relu

Esta función está determinada mediante los valores positivos provenientes de las entradas. Se expresa mediante la función:

$$f(x) = x^{+} = max(0, x)$$

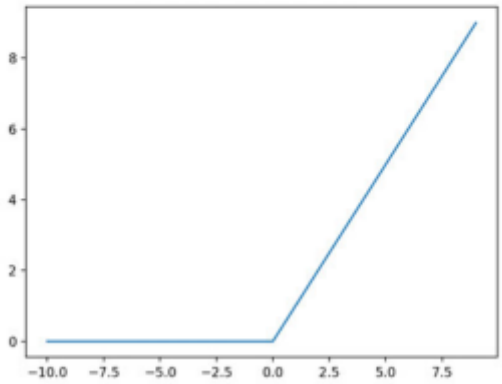


Ilustración 20: Función de activación relu

Función Sigmoide

Se trata de una función matemática la cual proporciona una curva semejante a una «S», y se la conoce como curva sigmoide. Con frecuencia, la función sigmoidea se entiende por la expresión matemática:

$$f(x) = \frac{1}{1 + e^{-x}}$$

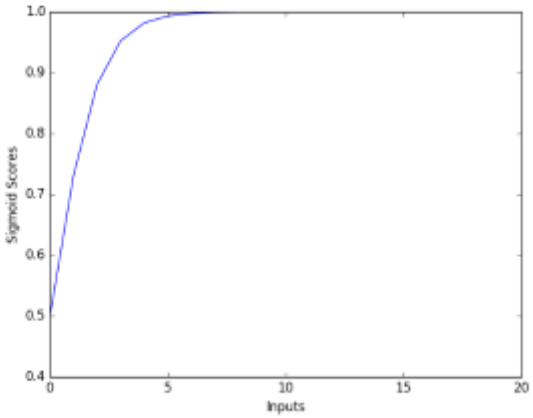


Ilustración 21: Función de activación sigmoide

Función Softmax

Esta función recibe como valor entrante un vector de K números perteneciente al conjunto de los reales, para luego estandarizar en una distribución de probabilidad la cual tiene K probabilidades. En otras palabras, previo a usar softmax, elementos en el vector pueden ser menores que cero o superiores a 1; y puede que la suma no llegue a 1; sin embargo, luego de hacer trabajar a softmax, cada elemento se situará en el rango (0, 1), y los elementos sumarán 1,siendo así capaces de interpretarse como probabilidades. Como dato extra, los elementos entrantes con mayor tamaño se relacionarán a probabilidades mayores.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \qquad i = 0, 1, ..., k$$

(Ecuación 6)

Por lo general, la función sigmoide es empleada para modelos binarios, por ejemplo diferenciar si una imagen es un perro o un gato; y la función softmax es utilizada para

modelos que cuenten con más de dos clases. Y en esta presente investigación estamos en este caso, ya que tenemos por objetivo lograr detectar 29 clases de señas (las 27 letras del alfabeto, 2 señas de espacio y borrar.

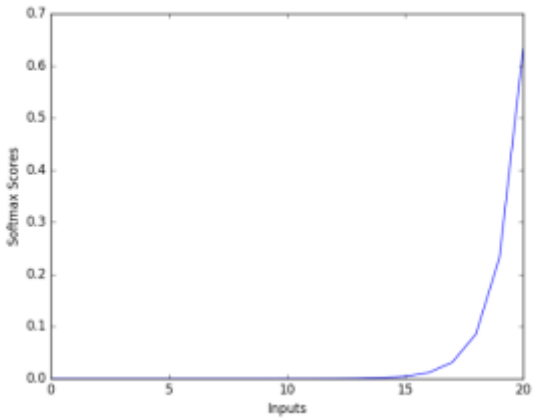


Ilustración 22: Función de activación softmax

2.2.8.7 Funciones de perdida

Hay múltiples de este tipo; sin embargo, en este proyecto analizaremos las tres que nos ayudarán en alcanzar la meta del presente proyecto, ya que sirven en la clasificación de muchas clases.

Multi-Class Cross-Entropy Loss

Cross-entropy, o en español entropía cruzada, calcula el promedio de bits requeridos para clasificar un evento entre dos grupos de probabilidad P y Q. El modelo de procesamiento empleado en el set es mejorado para un cálculo estadístico del bloque Q.

En esta situación, dado que es aplicado a clasificación de múltiples clases, en lo cual los objetivos son un conjunto 0, 1, 2, ..., n, en donde todas las clases poseen un valor distinto.

Determinamos los valores que detallan el margen medio existente en los bloques estadísticos continuos y asignados a la solución. Los valores se reducen, de esta manera obtenemos un índice de entropía cruzada ideal: el cero.

En caso de distribuciones no continuas P y Q, con igual soporte, X el índice de la entropía cruzada estaría expresado matemáticamente:

$$H(P,Q) = -\sum_{x \in X} P(x) \log Q(x)$$

(Ecuación 7)

Kullback–Leibler Divergence Loss

Conocida como entropía relativa, expresa la medida para saber el cómo una distribución de probabilidad es distinta de otra. Sus usos engloban personalizar la entropía referente al sistemas de información, aleatoriedad en sucesiones cronológicas no discretas, y obtención de información al relacionar conceptos probabilísticos de inferencia. A diferencia del cambio de información, esta es una métrica no equivalente, en consecuencia, no es categorizada como la medida dispersión. Un ejemplo sencillo: cuando una divergencia de Kullback-Leibler de 0 se entiende que el par de distribuciones tratadas serían muy iguales.

Para distribuciones no continuas P y Q, la divergencia de Kullback-Leibler estaría expresada por la ecuación:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log(\frac{P(x)}{Q(x)})$$

(Ecuación 8)

2.2.8.8 Optimizadores

Hay mucha cantidad de optimizadores, sin embargo aquí estaremos enfocados en el par con mayor empleabilidad: ADAM y RMSProp. En la presente investigación usaremos ADAM.

ADAM

Según Kingma, ADAM es un algoritmo de optimización estructurado en derivadas lineales de funciones estocásticas, soportándolo en los cálculos adaptativos dinámicos de bajo nivel. Este algoritmo es relativamente fácil de construir, y tiene una eficiencia computacionalmente aceptable. Además posee solo algunos requerimientos de almacenamiento, no varía con la diferencia métrica de los gradientes, y es ideal para obtener soluciones complejos en relación a parámetros y/o datos.

RMSProp

Este es semejante al anterior. Limita el dinamismo en el eje Y. En consecuencia, se puede incrementar la rapidez de aprendizaje, y el algoritmo sería capaz de realizar pasos más amplios en el eje X. Lo que hace distinto a RMSProp con la disminución de gradiente es la manera de determinar los gradientes (Gandhi R., 2018).

Utilidad de una CNN

- Sin tener la necesidad de extraer las características manualmente la hacen directamente de ellas.
- Los resultados de reconocimientos que generan son altamente precisos.
- podemos volver a entrenar para nuevas tareas de reconocimiento, lo que nos permitirá aprovechar las redes.

2.2.9 OpenCV

The Open Source Computer Vision Library (OpenCV), surgió en el año 2000, creado por Intel Corporation. Originalmente elaborado con el objetivo de procesamiento, obtención y reconocimiento de imágenes en zonas interactivas de hombre-máquina, reconocimiento,segmentación, biométrica, robótica y monitorización de objetos, seguridad, entre otros. Además brindar librerías de categorías de datos estáticos y dinámicos. Esta biblioteca es eficiente en múltiples procesadores, de fácil manejo y también multiplataforma, pudiéndose ejecutar en Mac OS, Linux y Windows. (Quispe M., 2013)

Para ser más específicos, las posibilidades brindadas por la biblioteca de software OpenCV se podrían categorizar en los grupos a continuación:

- Diseños y procedimientos simples.
- Tratamiento y manejo de imágenes: filtros, histogramas, remasterizado, entre otros.
- Análisis estructural: tratamiento de bordes, geometría, etc.
- Procesamiento dinámico y reconocimiento de elementos: flujo óptico, seguidores, localizadores, etc.
- Localización de elementos: modelos HMM, segmentación, etc.
- Ajuste de lente óptico: zoom, geometría epipolar, morphing.
- Rediseño en tres ejes (funcionalidad experimental): seguimiento de objetos tridimensionales, detección de elementos presentes, etc.

OpenCV podría categorizarse en 4 grupos de bibliotecas de software distintas esenciales de cómo emplear dicha biblioteca.

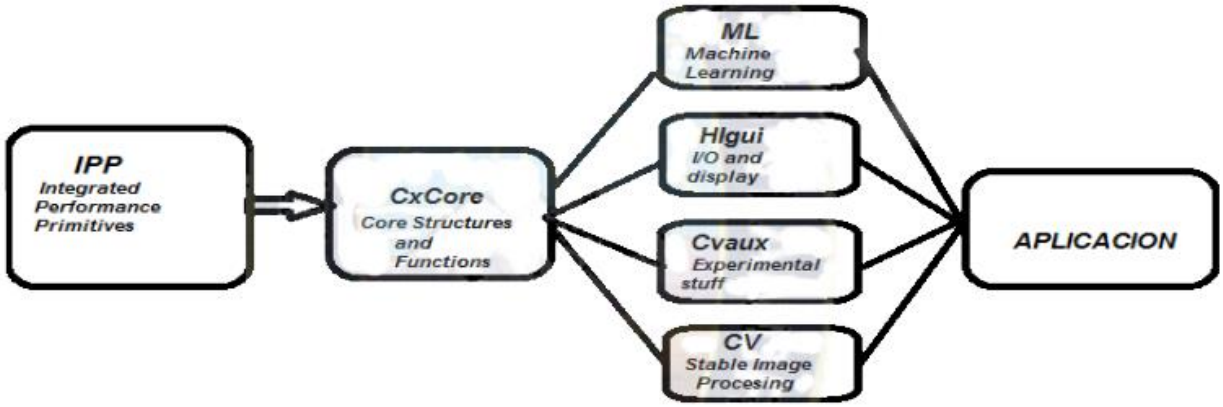


Ilustración 23: Representación de OpenCV

(Fuente: Quispe, 2013)

En el análisis estructural, podemos hallar la función cv, la cual sirve para el procesamiento de contornos y computación geométrica. Adicionalmente hay funciones que cumplen el objetivo de procesar enfoques y análisis dinámico;reconocimiento de patrones, tratamiento de campo visual y estimación de la pose de objetos; además, métodos de calibración en tres dimensiones (localización de señas, y muchos más).

2.2.10 TensorFlow

Esta es una librería de software elaborada por el gigante de internet Google, para el Machine Learning y la Inteligencia artificial. Es gratis y de código abierto (open-source). Es empleada en muchas tareas, sin embargo posee un enfoque especial en el entrenamiento y la inferencia de redes neuronales profundas. Posee un entorno variado de librerías adaptables, instrumentos, y mejoras continuas con lo cual los desarrolladores implementan y construyen de una manera sencilla y fácil softwares más potentes gracias al Machine Learning.

Sus caracterśticas más destacadas de TensorFlow son:

Fácil Implementación de modelos

TensorFlow facilita en gran manera la elaboración y entrenamiento de estructuras de Machine Learning, gracias al uso de APIs potentes como por ejemplo Keras. TensorFlow permite ejecutar de manera eficiente, conllevando a una iteración sin demoras del modelo y una depuración sin complicaciones.

Desarrollo en la nube Machine Learning

Este ecosistema facilita diseñar y construir sencillamente modelos localmente o en línea, con la ayuda de algún navegador web, o en teléfonos móviles, no habiendo problemas por el lenguaje con el que se programe.

Apropiado para la investigación

TensorFlow permite un diseño flexible y simple para plasmar lo que tenemos en mente y llevarlo al código, a estructuras algorítmicas contemporáneas y a la transmisión al público sin tanta dilación.

2.2.11 Lenguaje de programación Python

En la actualidad existe una gran cantidad de lenguajes de programación por los que podemos optar. Python es muy utilizado por su simplicidad de código. Es un lenguaje para programar que facilita la construcción de aplicaciones con pocas líneas. Este lenguaje es bastante utilizado para crear prototipos en tiempos más cortos, pero también

permite el desarrollo de algoritmos complejos. Su código es muy sencillo de aprender, comprender y al usar la indentación en lugar de los paréntesis permite una mayor legibilidad del mismo, logrando así una mayor calidad de sintaxis.
Python posee una curva de aprendizaje sencilla y amigable, su sintaxis de propósito general es muy intuitiva: cualquier desarrollador puede dedicar poco tiempo para aprenderlo y concentrarse más en cómo crear productos innovadores con él, aprovechando sobre todo su gran flexibilidad. Se trata de una herramienta que ahorra mucho tiempo de programación debido a su simplicidad, versatilidad, agilidad de lectura y amplia disponibilidad de librerías de todo tipo, lo que lo ha hecho crecer popularmente en las aplicaciones científicas. (Damiani, 2019)

Es un lenguaje de alto nivel no compilado sino interpretado, por lo que es multiplataforma. Es dinámico y orientado a objetos, se puede agregar nuevas funciones y clases a un objeto que ya existe, incluso en tiempo en la que se está ejecutando. Otra gran ventaja de Python es que se puede utilizar como software libre para combinar distintas aplicaciones que hayamos creado en otras aplicaciones.

El "ecosistema" Python desde sus inicios ha sido concebido como una alternativa de software libre. Cabe destacar que, además de las muchas librerías que posee, existen una gran cantidad de paquetes que son de acceso libre para ser usado por científicos desarrolladores que han sido creados por programadores tanto del ámbito académico y privado.

Python cuenta con una gran cantidad de entornos de trabajo (frameworks) de gran relevancia y utilidad. Un ejemplo sería el framework Django, que permite desarrollar aplicaciones orientadas a la web y además es flexible y se puede poner en línea con poco esfuerzo.

Podemos programar en Python en diversos entornos de desarrollo integrado (IDEs por sus siglas en inglés), desde entornos bastantes sencillos como IDLE, hasta otros más completos y complejos como NetBeans, Spyder o Eclipse. Para el desarrollo de este sistema estaremos utilizando Visual Studio Code, ya que cuenta con variados plugins que facilitan la programación en este lenguaje.

Cuenta con muchas librerías muy avanzadas y de gran capacidad, además son gratuitas por ende potencian su uso en aplicaciones de ciencia e ingeniería. Las que se usan en este proyecto de investigación, son mencionadas y descritas a continuación:

OS

Este módulo nos ayuda a hacer acciones tales como asignar un directorio nuevo, obtener lo que hay en un folder, saber el desarrollo de alguna ejecución, terminar una tarea, entre otras más. Os posee mecanismo para obtener información del Sistema Operativo, y así Python consigue mejorar continuamente. (COVANTEC)

PIL

Esta dependencia llamada Python Imaging Library (PIL) es libre, y facilita la manipulación de imágenes con Python. Trabaja con muchos formatos, tales por ejemplo JPEG, PNG y GIF. En su mayoría está codificado en C, mejorando su eficiencia.

PYTTSX3

Esta es una dependencia de software que transforma texto a voz. No se asemeja a las otras dependencias similares ya que trabaja offline, y se asimila muy bien con las versiones 2 y 3 de Python. Es una biblioteca que no presenta complicaciones en su uso.

MEDIAPIPE

Es una dependencia de software preconstruido en PyPI. Además, brinda instrumentos con el objetivo de que los programadores desarrollen salidas a problemas. La biblioteca MediaPipe de Python se encuentra accesible en PyPI las plataformas de sistemas operativos más usados.

Algunos propósitos que podemos conseguir con la facilidad que nos brinda Mediapipe:

- Detección de malla de rostros y de rostros
- Detección holística y pose
- Detecta y puede seguir a objetos en movimiento
- Reconocimientos de manos (Es para lo que se usa en este proyecto)
- Ajuste de volumen

Esta biblioteca es desarrollada por Google, y permite detectar manos humanas en una imagen de entrada. Y tras hacer la detección, permite trazar 21 puntos y líneas entre estos puntos. Esto se puede apreciar en la Ilustración 24:

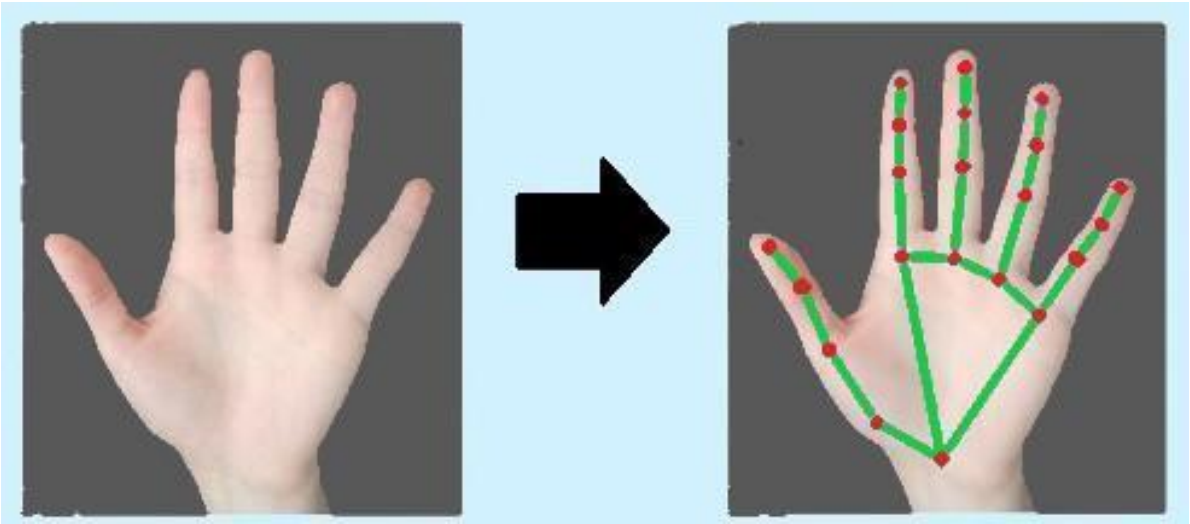


Ilustración 24: Mediapipe haciendo la detección de la mano, y trazando las coordenadas.

Estas coordenadas son denominadas individualmente, y según la documentación oficial de Mediapipe, estos nombres se aprecian en la Ilustración 25

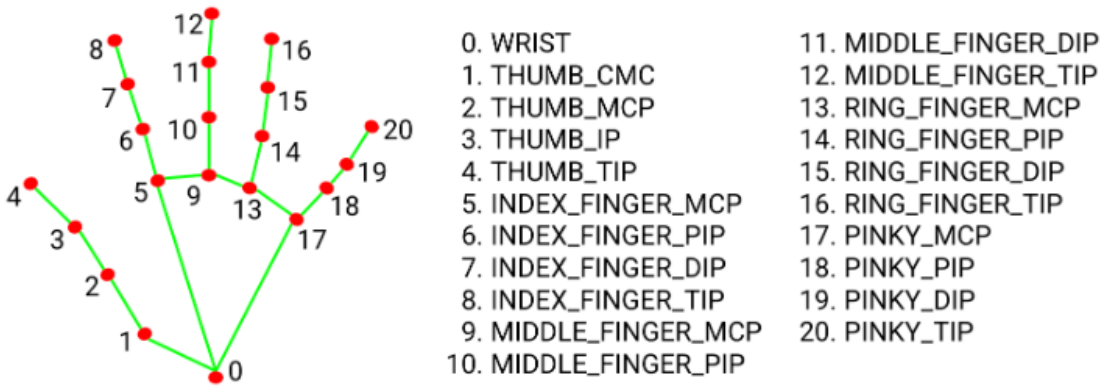


Ilustración 25: Puntos de detección de Mediapipe

NUMPY

Este paquete de software para Python brinda apoyo con el objetivo de generar vectores y matrices de muchas dimensiones, acompañado de muchas colecciones de funciones matemáticas con el objetivo de operar con valores. NumPy es una biblioteca de código abierto poseyendo una comunidad de soporte muy extensa.

SPEECH – RECOGNITION

El reconocimiento de voz es el mecanismo de transformar audio en texto. Y esto se hace generalmente en asistentes de voz como Siri, Cortana, entre otros. Python brinda una API conocida como SpeechRecognition que nos ayuda a transformar audio en texto con el propósito de procesarlo después.

TKINTER

Tkinter es una librería integrada de Python. Proporciona un conjunto de herramientas robustas e independientes para diseñar y administrar interfaces gráficas de usuarios (ventanas).

CAPÍTULO 3: DISEÑO METODOLÓGICO

En esta sección brindaremos detalles de las características metodológicas del presente proyecto. Para empezar, haremos una presentación de la tipificación de la investigación, tabla de operación de las variables de estudio, la población y la muestra; posteriormente, se presentarán los instrumentos y equipos utilizados. Por último, haremos una narración específica de los procesos, recolección de datos y otros elementos del contexto involucrados para el desarrollo del presente proyecto.

3.1 Tipificación de la investigación

Justificación de porqué es investigación cuantitativa

La siguiente investigación tiene un enfoque cuantitativo es sucesivo y probatorio. Fase por fase sigue a la otra y es inevitable omitir etapas. El orden es estricto, sin embargo, se puede establecer una que otra etapa o fase. Se inicia de una hipótesis que va limitándose y, cuando ya se delimita, se determinan interrogantes para investigar, se escudriña la teoría y se elabora un contexto o un enfoque conceptual. De las interrogantes se definen variables e hipótesis; se estructura una manera de probarlas (diseño); se obtienen los valores de las variables en un marco preestablecido; se estudian los valores resultantes empleando métodos probabilísticos, y se obtienen un conjunto de deducciones en relación a la o las hipótesis. (Hernández S., 2014)

Teniendo en cuenta la definición anterior, la presente investigación es cuantitativa, ya que mide y estima el nivel de eficiencia que tiene el sistema computacional basado en inteligencia artificial para mejorar la comunicación con personas con deficiencia para escuchar.

Justificación de porqué es investigación experimental

En este tipo de investigación, es la persona quien controla una o varias variables de estudio, para así manipular el incremento o decremento de dichas variables y su repercusión en los elementos de estudio. En otras palabras, un experimento tiene que ver con realizar una variación en el índice de la variable independiente y cómo repercute en la variable dependiente. Esto es realizado en condiciones estrictamente controladas, con el propósito de detallar de qué manera o por qué razón surge un acontecimiento o

situación buscada. Los métodos experimentales son los correctos para probar la hipótesis de relaciones causales. (Cohen, L., 2002)

Tenemos como uno de los objetivos elaborar un sistema computacional basado en inteligencia artificial que va a permitir la comunicación con personas sordas. Por tal motivo, la presente investigación es experimental.

Justificación de porqué es investigación aplicada

Este tipo de investigación, también conocida como empírica, se diferencia de otro tipo de investigaciones por materializar en la práctica los conocimientos teóricos existentes en un campo de estudio. Propone solucionar rápidamente problemas que necesiten ser tratados de manera específica e inmediata, mirando como meta la investigación de la problemática llevada a la práctica. Esta es capaz de brindar información desconocida, basándose en los datos ya existentes. (Baena, 2014)

Emplea sus recursos para solucionar los problemas que apremian a la humanidad en su conjunto. La presente investigación tiene por objetivo elaborar un sistema computacional basado en inteligencia artificial de reconocimiento del alfabeto dactilológico peruano a través de la ayuda de áreas de estudio tales como aprendizaje profundo y la visión artificial, con el propósito de poder comunicarse entre dos personas: una sorda y otra oyente. El diseño empleado en el presente proyecto es justamente experimental. Esto lo hacemos con el objetivo de determinar una relación causal y determinar si hay una relación de causa a efecto entre nuestras variables dependiente e independiente.

Justificación de porque el alcance de la investigación es correlacional

Según lo menciona Hernández, Fernández & Baptista (2010), si estamos evaluando el alcance de una investigación hay que descartar la idea de una tipología, porque además de una clasificación, el alcance determina el resultado buscado en la investigación.

El diseño empleado en la presente investigación es de tipo correlacional, ya que busca hallar una conexión de causa a efecto y determinar si existe relación entre las variables dependiente e independiente.

3.2 Tabla de operacionalización de variables

Variable independiente

Sistema computacional basado en inteligencia artificial de reconocimiento del alfabeto de señas

Variable dependiente

Barrera de comunicación de las personas con deficiencia auditiva.

	VARIABLE	DEFINICIÓN	DIMENSIÓN	INDICADORES
VARIBALE INDEPENDIENTE	Sistema computacional basado en inteligencia artificial para el reconocimeinto del alfabeto de señas	Sistema inteligente alternativo que ayudará como intermediario entre una persona con deficiencia auditiva y una persona oyente, donde las señas de las personas con discapacidad auditiva se convierten en texto o voz. (Nada, Mazen y Zayed, 2017)	Fiabilidad	Número de señas reconocidas
				Tiempo promedio de reconocimiento de una seña
VARIABLE DEPENDIENTE	Barrera de comunicación de los estudiantes con discapacidad auditiva de la IE para sordos Harvest	Dificultad que presentan los estudiantes para comunicarse y relacionarse con su entorno. (Gómez E., y Posada S., 2012)	Interacción entre un estudiante con discapacidad auditiva y una persona oyente	Cantidad promedio de señas interpretadas correctamente por el oyente
				Tiempo promedio que tarda un estudiante con deficiencia auditiva en hacer una seña legible para el oyente

Tabla 3: Operacionalización de variables

3.3 Población y muestra

Realizamos nuestra investigación en la Institución Educativa Bautista para Sordos Harvest localizado en el distrito de Pimentel, Chiclayo (Región Lambayeque - Perú).

En el presente año, 2022, por circunstancias de la pandemia de la COVID-19, la población estudiantil es de 7 alumnos, los cuales tienen discapacidad auditiva.

Para la muestra, se consideró lo siguiente:

Según Martínez Bencardino, 2012, para obtener el tamaño de la muestra para una población finita, hay que considerar la fórmula:

$$n = \frac{N * Z_{\propto}^2 * p * q}{e^2 * (N-1) + Z_{\propto}^2 * p * q}$$

(Ecuación 9)

Donde:

n: Tamaño de muestra buscado

N: Tamaño de la población o Universo

Z: Parámetro estadístico dependiente del nivel de confianza

e: Error de estimación máximo aceptado

p: Probabilidad de que suceda el evento en estudio

q: Probabilidad de que no suceda el evento en estudio

Y Bercardino (2012) nos proporciona la siguiente tabla para saber el valor de Z a asignar

Nivel de confianza	Z_{alfa}
99.7%	3
99%	2,58
98%	2,33
96%	2,05
95%	1,96
90%	1,645
80%	1,28
50%	0,674

Tabla 4: Algunos posibles valores que puede tomar Z

En base a esto, hemos asignados los siguientes valores:

N=7, por ser la cantidad de alumnos del colegio Harvest

Z=1.645, ya que hemos considerado un nivel de confianza del 90%

e=5%

p=50%

q=50%

p y q se ha considerado al 50% por sugerencia del autor. Es más adecuado considerar que existe un 50% de probabilidad de que ocurra un evento, y un 50% de que no.

Reemplazando todo en la fórmula, se obtiene que n=7

Por lo tanto, la muestra tendrá un tamaño de 7

3.4 Metodología y recolección de datos

A continuación, el organizador visual, se esclarece la metodología que permite la construcción del sistema computacional basado en inteligencia artificial que permite reconocer el lenguaje dactilológico de Perú. (Belavagi y Muniyal, 2016)

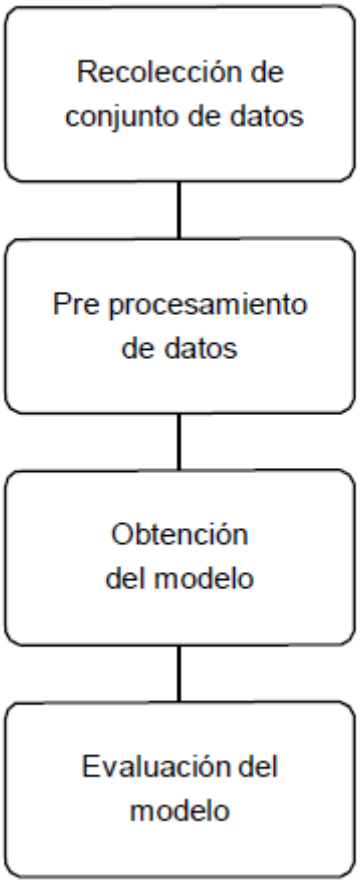


Ilustración 26: Esquema que esclarece la metodología para la construcción del sistema

Como inicio, obtenemos el set de datos requeridos con el fin de la elaboración del software. El conjunto de datos recolectados lo hemos dividido en dos partes: una para entrenamiento y la otra para validación. Pre- procesamos las imágenes con el objetivo de reducir elementos que ocasionan ruido y quitar valores atípicos, consiguiendo datos confiables para el entrenamiento de nuestro modelo. En la etapa de entrenamiento se emplea el primer set de datos para calcular los parámetros y pesos que determinan el modelo de red neuronal. Se obtiene su valor de una forma repetitiva según los datos de entrenamiento establecidos previamente, con el propósito de reducir valores falsos entre la salida resultante y la buscada.

En la etapa previa el modelo podría ajustarse adecuadamente a ciertas características existentes en los datos de entrenamiento. Con el propósito de eliminar esta impertinencia, es recomendable emplear otro grupo secundario diferentes a los datos de entrenamiento, logrando así manipular la fase de aprendizaje. Y para terminar, se evalúa el modelo considerando los parámetros de precisión y rendimiento.

3.4.1 Procedimiento y Herramientas para Recolección de datos

En la fase de la obtención de datos de nuestra investigación, se emplea como herramienta el formulario de observación, a través de este se conseguirá establecer qué se hace, cómo se hace, en cuanto tiempo se hace y la razón del porqué se hace.

Diseño de contrastación de hipótesis

PRETEST	INSTRUMENTO	POSTEST
Se avaluará la cantidad promedio de señas interpretadas correctamente por el oyente; además, se determinará tiempo promedio que tarda un estudiante con deficiencia auditiva del colegio Bautista Harvest en realizar una seña legible para una persona que desconoce el alfabeto de señas, este tiempo se medirá utilizando el cronómetro como instrumento;	Sistema computacional basado en inteligencia artificial de reconocimiento del alfabeto de señas	Se evaluará la cantidad de señas reconocidas por el sistema, asimismo el tiempo promedio en que el estudiante con discapacidad auditiva del colegio Bautista Harvest realiza una seña legible para el oyente a través del sistema

Tabla 5: Contrastación de hipótesis

CAPÍTULO 4: DESARROLLO DEL PROYECTO

4.1 Análisis del uso del alfabeto de señas en el colegio Harvest

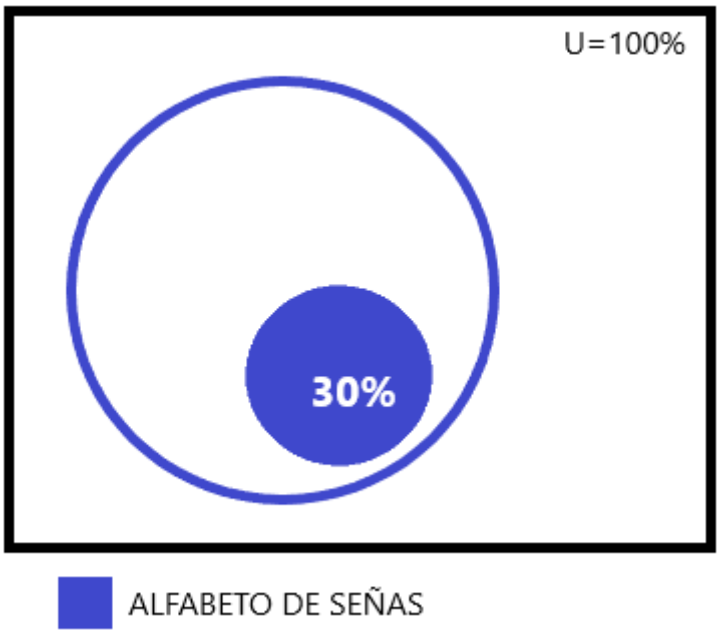
Se realizó una entrevista a tres personas que laboran en el colegio: La directora del colegio, el profesor traductor, y un misionero de la iglesia Bautista que labora ahí como capellán.

Según lo que nos comentó la directora en su entrevista, es que los alumnos usan alrededor del 30% el alfabeto de señas. En otras palabras, de cada 100 expresiones, 30 las realizan con el alfabeto de señas.

De igual manera, el traductor nos indicó que entre el 25% y 35% de las expresiones realizadas por los alumnos del colegio Harvest son con el alfabeto de señas. Y este dato concuerda con lo indicado por el misionero, el cual nos indicó en su entrevista que entre el 30% y el 35%, aproximadamente, de las expresiones de los niños sordos son realizados con el alfabeto manual.

De estas tres entrevista se concluye que el alfabeto de señas sirve como un apoyo para el lenguaje de señas, y que tiene un porcentaje de uso del 30% del total de señas realizados por los estudiantes.

Representación del uso del Alfabeto de Señas



Con el lenguaje de señas se pueden formar oraciones tal cual lo hacen las personas hablantes con la voz. Cada seña tiene su propio significado al igual que las palabras en el lenguaje hablado.

Por ejemplo, para decir árbol, se usaría la seña mostrada a continuación en la **Ilustración 27**, y para deletrear esa misma palabra usaríamos las señas mostradas en la **Ilustración 28**

Ilustración 27: Representación en señas de un árbol

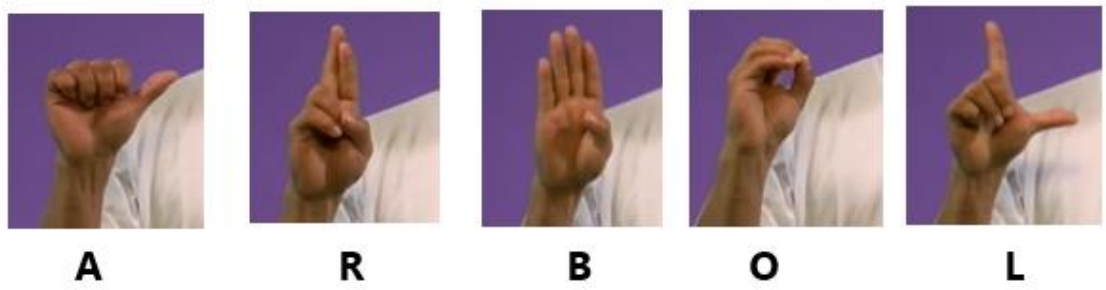


Ilustración 28: Representación en Alfabeto de Señas d la palabra árbol

Todas las palabras existentes tienen su equivalente en alfabeto de señas, convirtiendo así al alfabeto dactilológico en un medio muy potente para comunicarse con las personas sordas. Y esta potencialidad será aun mejorada con el sistema desarrollado en esta investigación.

4.2 Determinación de las tecnologías que se emplearán en la construcción del sistema inteligente

Haciendo una búsqueda en SCOPUS, el cual es una de las bases de datos más grande del mundo respecto a revistas y artículos de investigación, se encontró que existe una relación mayoritaria entre redes neuronales convolucionales y el uso del lenguaje de programación python.

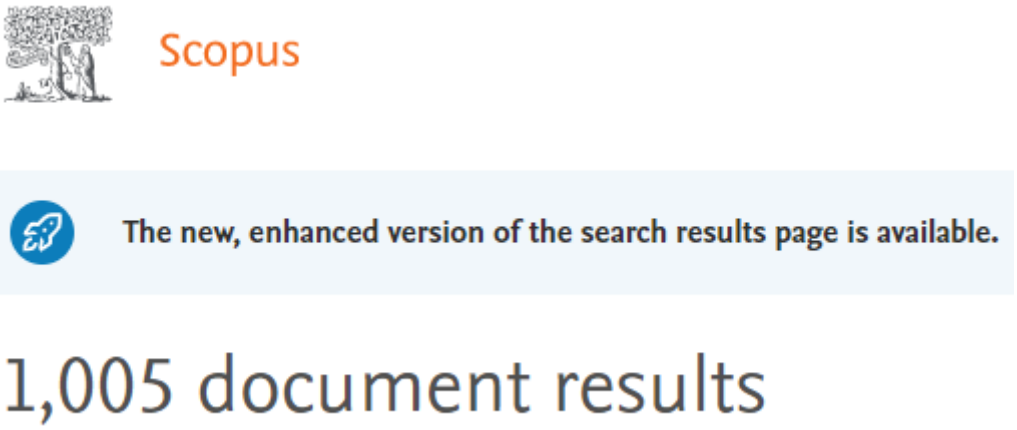


Ilustración 29: Resultado de búsqueda en SCOPUS para Redes Neuronales Convolucionales y Python

Al realizar una búsqueda de “artificial AND intelligence AND Python” se encontraron 1436 resultados

Cuando se realizó otra búsquedas con otros lenguajes de programación como php (Ilustración 30) o visual basic (Ilustración 31), la diferencia fue notablemente marcada, evidenciándose que se puede hacer muy poco en inteligencia artificial con otros lenguajes de programación a diferencia de python.

Ilustración 30: Resultado de búsqueda en SCOPUS para Inteligencia artificial y php

Ilustración 31: Resultado de búsqueda en SCOPUS para Inteligencia artificial y visual basic

Ilustración 32: Resultado de búsqueda en SCOPUS para Inteligencia artificial y javascript

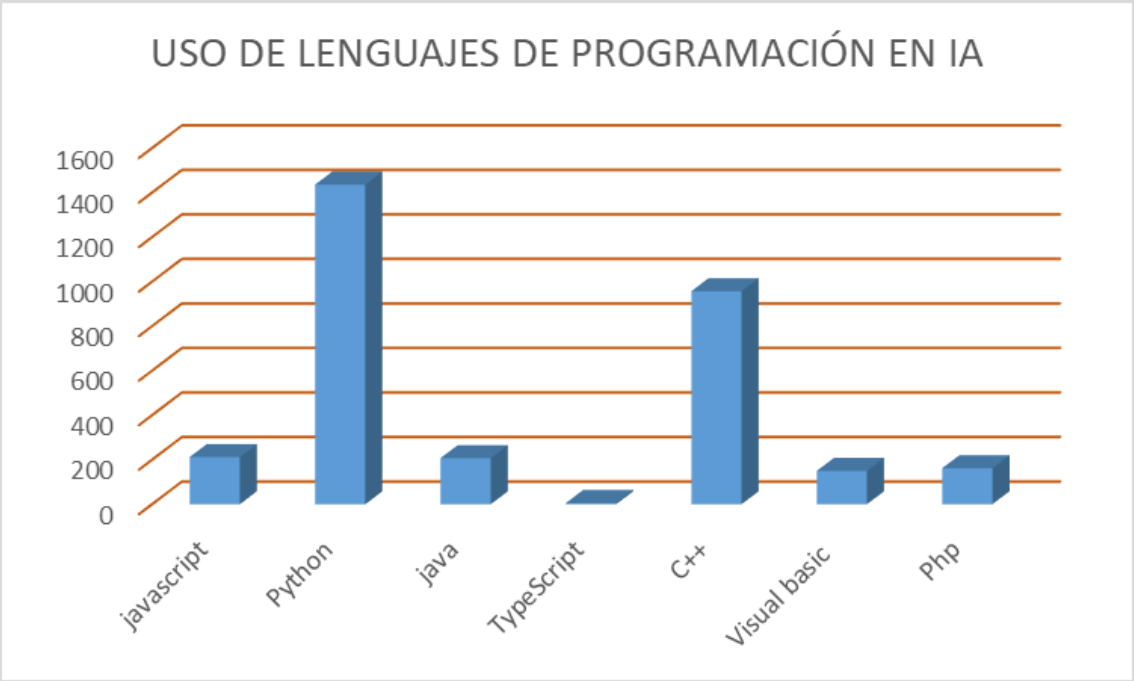


Ilustración 33: Uso de lenguajes de programación en IA, según búsquedas en Scopus

En base a esto, se determinó desarrollar el sistema inteligente (software) haciendo uso del lenguaje de programación python, ya que este cuenta con un enorme set de dependencias que permite implementar y construir algoritmos de aprendizaje automático, o de máquinas, de manera más rápida y sencilla. Por mencionar algunas de estás dependencias tenemos: Tensorflow, Keras, Mediapipe, etc. Estas bibliotecas de código desarrolladas son muy superiores a la existente con Java u otros lenguajes.

Python es un lenguaje de programación de alto nivel, siendo uno de los más populares y usados a nivel mundial por su facilidad de programación y versatilidad.

Además, en la actualidad este lenguaje cuenta con una gran comunidad de desarrolladores que alimentan, respaldan el código y permiten demostrar su solidez ante otros lenguajes como C + + y Java.

Sergio Kaufman, Presidente de Accenture Argentina y Sudamérica Hispana, en la 26° Conferencia Industrial organizada por la Unión Industrial Argentina, señaló que el "57% de los desarrolladores y científicos de datos de aprendizaje automático utilizan Python y el 33% toma como prioridad para sus desarrollos."

Python es muy utilizado en proyectos de inteligencia artificial IA, diseñar videojuegos, para crear sitios web escalables, mantenibles y robustas, realizar cálculos estructurales complejos con elementos finitos, entre otras muchas aplicaciones, que incluso involucran actividades de cómputo en el espacio.

En consecuencia, la NASA utiliza Python y machine learning en sus misiones, para tareas relacionadas con la predicción de condiciones meteorológicas durante el lanzamiento de cohetes y clasificación de rocas espaciales .

Existen proyectos emblemáticos desarrollados con Python como las redes sociales Pinterest y Instagram. Uber también está escrito con este lenguaje de programación, al igual que Dropbox .

En cuanto a la industria del entretenimiento, podemos destacar a: juego Battlefield 2, Netflix, Spotify y el motor para la creación de juegos Panda 3D.

Con Python también ha sido desarrollado Google App Engine, el cual nos permite crear aplicaciones web y móviles.

El objetivo de poder dotar a las computadoras con capacidades propias de la mente humana, como la comunicación, la interpretación de conductas, el aprendizaje y el proceso de estímulos, es un desafío que desvela y entusiasma a muchos científicos y para lo cual se desarrollaron las redes neuronales artificiales.

Los avances que se produjeron al utilizar Python en este tipo de proyectos ha permitido el desarrollo de vehículos autónomos. Por ejemplo, el multimillonario Elon Musk creador de Tesla y SpaceX, mencionó en una conferencia que la red neuronal del Autopilot de Tesla está hecha en Python.

También se usa en redes neuronales convolucionales en sistemas que identifican y clasifican elementos dentro de una imagen y plataformas de traducción de idiomas.

Por todas estas razones, por toda esta comunidad que tiene python detrás, y por todas las librerías que ya ofrece python para desarrollar inteligencia artificial, decidimos en esta investigación hacer uso de este lenguaje de programación para la construcción del sistema computacional inteligente que hará el reconocimiento de las señas del alfabeto dactilológico peruano.

4.3 Requisitos previos

En la elaboración de esta investigación se empleó el software administrador de librerías pip empleado para obtener, instalar y gestionar bibliotecas de software desarrollados en Python. Además, la versión de Python empleada fue la 3.10.4, la librería de TensorFlow fue la versión 2.6.0, y la biblioteca OpenCV fue la versión 3.5.2

Para la elaboración, entrenamiento e interpretación del modelo se necesitó un ordenador Core i5 7th Gen de procesador, conteniendo una GPU integrada Intel ® HD Graphics 620; además de una cámara digital HD integrada al portátil con 27 FPS para obtener y procesar señales contínua de video.

El sistema operativo usado fue Windows 10, y se usó como editor de código a Visual Studio Code, el cual es una herramienta gratuita.

4.4 Entorno de trabajo

Con pip se facilita crear y administrar entornos virtuales con distintas versiones de Python y / o bibliotecas de software instalados ahí mismo en el entorno, y esto se hace con el propósito de aislar las bibliotecas que empleemos para un caso en particular.

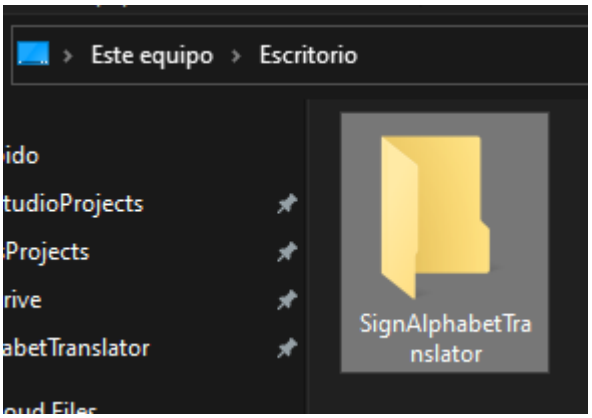
Pip se puede ejecutar en una terminal de comandos del sistema operativo. Ahí se realizan las configuraciones y la personalización del ecosistema de trabajo. Para evitar errores posteriores fue útil cerciorarse que pip esté instalado adecuadamente y esté corriendo en la computadora de trabajo. Para verificar esto escribimos en la terminal de comandos: "pip –version"

Pip posee un entorno por defecto, pero es poco aconsejable instalar bibliotecas directamente ahí. Entonces, creamos un entorno virtual separado.

Para poder crear un entorno virtual con pip, primero necesitamos el módulo "virtualenv", el cual lo podemos instalar en el sistema ejecutando: pip install virtualenv

```
C:\Users\israe>pip install virtualenv
```

Luego creamos una carpeta donde estará nuestro proyecto. En este caso la carpeta se llamará SignAlphabetTranslator. La carpeta puede ser creada en cualquier dirección. Aquí lo haremos en el escritorio.



Luego debemos ir a esa ruta en el terminal. Copiamos la ruta y la pegamos en el terminal anteponiendo cd

```
C:\Users\israe>cd C:\Users\israe\Desktop\SignAlphabetTranslator
```

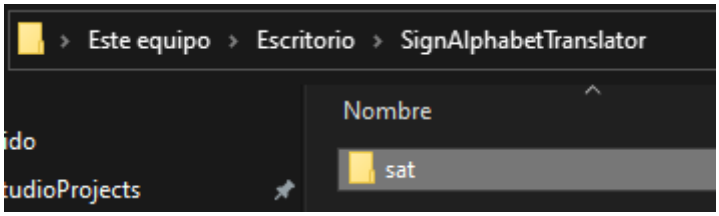
Y tras ubicarse ahí, quedaría:

```
C:\Users\israe\Desktop\SignAlphabetTranslator>
```

Seguidamente, pasamos a crear el entorno virtual. En nuestro caso hemos decidido llamar al entorno sat, siglas de Sign Alphabet Translator, nombre con el que hemos decidido llamar al sistema del presente proyecto. Para crear el entorno, tipeamos en la terminal: virtualenv sat

```
C:\Users\israe\Desktop\SignAlphabetTranslator>virtualenv sat
```

Y en la carpeta principal (SignAlphabetTranslator) aparecerá una nueva carpeta que llevará el nombre sat



Entonces pasamos a activar el entorno virtual para poder trabajar en él, y para instalar las librerías necesarias del proyecto. La forma de activar el entorno es tipeando en la terminal: .\sat\Scripts\activate

```
C:\Users\israe\Desktop\SignAlphabetTranslator>.\sat\Scripts\activate
```

Y se puede verificar que el entorno está activo porque al principio de la línea aparece su nombre (sat):

```
(sat) C:\Users\israe\Desktop\SignAlphabetTranslator>
```

Ahora sí, pasamos a instalar todas las librerías necesarias para nuestro proyecto. En este caso las principales bibliotecas empleadas fueron:

- Tensorflow
- Mediapipe
- Opencv-Contrib-Python
- Pyttsx3
- SpeechRecognition
- Scipy
- PyAudio

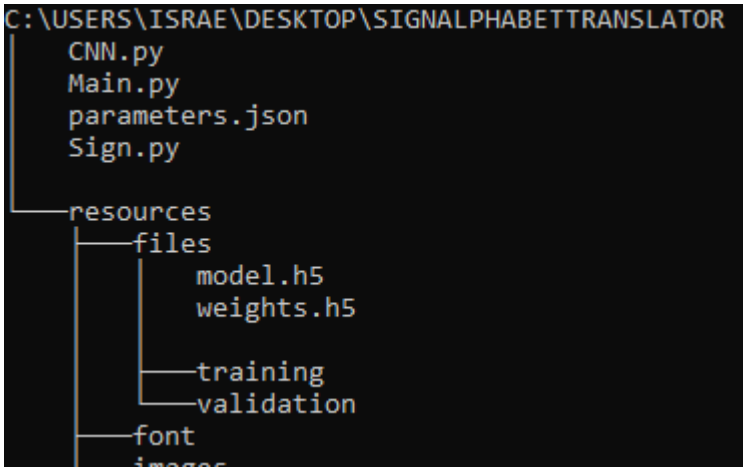
Todas estas bibliotecas, fueran instalados con los comandos mostrados a continuación, pero cabe resaltar que cada línea se ejecutó de manera ordenada y separadamente:

```
pip install tensorflow
pip install mediapipe
pip install opencv-contrib-python
pip install pyttsx3
pip install SpeechRecognition
pip install scipy
pip install PyAudio
```

.

Sin embargo, estas bibliotecas principales necesitan otras librerías para su funcionamiento, es por ello que el conjunto de todas las librerías usadas e instaladas, con sus respectivas versiones, se encuentran en el Anexo 1.

La estructura de directorios para el proyecto quedó de la siguiente manera:



La descripción de la estructura es la siguiente:

- La carpeta raíz que es SignAlphabetTranslator, contiene un subdirectorio y cuatro archivos: tres de extensión .py y otro último de extensión .json
- En el subdirectorio resources, aparecen tres subcarpetas más: files, font, y images.
- Dentro de files, están las subcarpetas training y validation, y los archivos model.h5 y weights.py.
- Dentro de training y validation (en cada una de ambas carpetas) hay 29 subcarpetas, que corresponden a las 27 letras del alfabeto y a dos señas más de espacio y eliminar, haciendo un total de 29 señas.
- La carpeta font contiene una archivo arial.ttf que es usado en darle formato a las letras que se muestran en el sistema
- En la carpeta images se encuentran las imágenes usadas en la interfaz gráfica, tales como el ícono, etc.; y además, se encuentran las imágenes que se muestran tras hacer la conversión de voz a texto, y luego el texto a imagen.
- El archivo Sign.py fue usado en la captación de las señas, antes de entrenar el modelo.
- El archivo CNN.py es donde se encuentra el algoritmo que entrena el modelo

- El archivo parameters.json contiene los parámetros empleados en el sistema. Estos son: el número de clases (en este caso 29), la cantidad de imágenes por cada clase (en este caso 3000), el valor de redimensión de cada imagen antes de ser guardada y procesada (aquí es 50 px de ancho y alto), los canales de cada imagen (aquí corresponde a 1, ya que son imágenes binarias), el número de neuronas usadas para entrenar la CNN, y por último el número de iteraciones (en este caso 5 iteraciones por cada clase, lo que equivale a 145 iteraciones).
- Por último, el archivo Main.py es la clase que contiene la interfaz gráfica y se ejecuta primero al poner a funcionar el sistema.

Con el objetivo de construir el sistema se consideró 29 clases o señas (27 para las letras, 2 señas más de espacio y eliminar). De esta manera se entrenó la red neuronal convolucional. Estas señas o clases las observamos en la tabla 6:

CLASE	ETIQUETA
0	A
1	B
2	C
3	D
4	E
5	ELI
6	ESP
7	F
8	G
9	H
10	I
11	J
12	K
13	L
14	M
15	N
16	Ñ
17	O
18	P
19	Q
20	R
21	S
22	T
23	U
24	V
25	W
26	X
27	Y
28	Z

Tabla 6: Enumeración de señas por clase.

4.5 Obtención del modelo

A. Obtención de las imágenes para entrenar el modelo: Construcción del archivo Python Sign.py

Para poder entrenar la red neuronal convolucional del presente proyecto, primero tuvimos que obtener imágenes de las 29 señas que interpretará el sistema inteligente.

En el Anexo 2 se podrá visualizar todo el código del archivo Sign.py. Pero a continuación haremos una descripción de su construcción.

Primero importamos las librerías necesarias:

```
import cv2
import mediapipe as mp
import os
import numpy as np
import json
```

Paso seguido, leemos los parámetros ya establecidos en el fichero parámeters.json:

```
with open("parameters.json") as data:
    parameters=json.loads(data.read())
```

Luego creamos las siguientes variables:

```
letter='a'
letter=letter.upper()
# folder='resources/files/validation/'+letter
folder='resources/files/training/'+letter
quantity=parameters['quantity']
width, height=parameters['width'], parameters['height']
```

La variable letter contiene el valor de la seña-letra que se quiere capturar. La variable folder especifica la ruta donde se va a guardar las imágenes de entrenamiento y de validación. Por cada letra-seña hay que guardar imágenes en las carpetas validation y training.

Como aconseja Warden 2017; para entrenar de manera recomendada un clasificador se requieren 1000 imágenes por categoría. El motivo de esta cantidad surge del reto de clasificación de ImageNet, en el cual cada categoría tenía alrededor de mil imágenes, y esto resultó ser lo exitosamente eficiente para el entrenamiento las primeras generaciones de clasificadores como AlexNet, y pone en evidencia que es suficiente un millar de imágenes, aproximadamente, para un clasificador de 8 categorías.

Haciendo una regla de tres, para entrenar 29 clases se necesitan 3000 imágenes por clase, lo que se ve en la variable quantiy.

Las variables height y width son los valores de alto y ancho que tendrá cada imagen, que aquí es de 50px por 50px de ancho y alto, respectivamente.

La señal de video continua se capturó mediante el método VideoCapture de OpenCV: Este puede leer una sucesión de frames o imágenes ya sea que provenga de una cámara externa o también de un archivo de video, logrando así el que sea posible realizar diferentes operaciones sobre estos frames o imágenes. Pero previo a emplear el método VideoCapture, fue prioritario importar la biblioteca de OpenCV (cv2) en la clase Sign.py. Posteriormente, creamos un objeto de la clase VideoCapture el cual requiere como parámetro el nombre del archivo de video o el índice del dispositivo.

Comúnmente dispositivo de cámara está integrado al ordenador. En nuestro caso usamos la cámara integrada al computador (laptop), entonces se estableció a 0 el valor del índice. Con esto se le indica a OpenCV que emplee la cámara disponible y única integrada al ordenador. Si hubiera más de una cámara conectada, es posible emplear la segunda modificando el valor de 0 a 1. La variable cap expresa que se emplea el valor de cero en VideoCapture.

```
cap=cv2.VideoCapture(0)
```

Luego pasamos a crear la carpeta dependiendo de cada letra-seña que estemos trabajando. Recordemos que debemos repetir el proceso dos veces por cada seña.

```
if not os.path.exists(folder):
    os.makedirs(folder)
```

Paso siguiente definimos las variables mp_hands=mp.solutions.hands y mp_drawing=mp.solutions.drawing_utils. Con mp_hands podremos acceder al modelo que nos detectará las manos, y con mp_drawing podremos trazar los puntos y lías en la imagen, luego de la detección

```
mp_drawing=mp.solutions.drawing_utils
mp_hands=mp.solutions.hands
```

Luego viene el cuerpo de algoritmo, el cual está delimitado por un bucle while en Python:

```
36  with mp_hands.Hands(
37      static_image_mode=False, max_num_hands=1,min_detection_confidence=0.5
38  ) as hands:
39      cont=1
40      capture=False
41      while True:
42          ret, frame=cap.read()
43          if ret==False:
44              break
45          if cv2.waitKey(1) & 0xFF==27 or cont>quantity:
46              break;
47          if cv2.waitKey(1) & 0xFF==32:
48              capture=True
49          background=cv2.imread('resources/images/background.png')
50          # Working mediapipe
51          frame_rgb=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
52          results=hands.process(frame_rgb)
53          hands_image=results.multi_hand_landmarks
54          if hands_image is not None:
55              for hand_landmarks in hands_image:
56                  mp_drawing.draw_landmarks(
57                      background, hand_landmarks, mp_hands.HAND_CONNECTIONS,
58                      mp_drawing.DrawingSpec(color=(255,255,255), thickness=0, circle_radius=0),
59                      mp_drawing.DrawingSpec(color=(255,255,255), thickness=16)
60                  )
```

```
62                  h, w, _= background.shape
63                  p1x=int((hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MCP].x)*w)
64                  p1y=int((hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MCP].y)*h)
65                  p2x=int((hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_MCP].x)*w)
66                  p2y=int((hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_MCP].y)*h)
67                  p3x=int((hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_MCP].x)*w)
68                  p3y=int((hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_MCP].y)*h)
69                  p4x=int((hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].x)*w)
70                  p4y=int((hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].y)*h)
71                  p5x=int((hand_landmarks.landmark[mp_hands.HandLandmark.WRIST].x)*w)
72                  p5y=int((hand_landmarks.landmark[mp_hands.HandLandmark.WRIST].y)*h)
73                  points = np.array([[p1x, p1y], [p2x, p2y], [p3x, p3y], [p4x, p4y], [p5x, p5y]])
74                  cv2.fillPoly(background, pts=[points], color=(255, 255, 255))
75          x1, y1=220, 140
76          x2, y2=420, 340
77          if capture:
78              cut=background[y1:y2, x1:x2]
79              background=cv2.cvtColor(background, cv2.COLOR_BGR2GRAY)
80              cut=cv2.resize(cut, (height,width))
81              cv2.imwrite(folder+'/'+letter+'_{}.jpg'.format(cont), cut)
82              cont+=1
83          background= cv2.flip(background, 1)
84          cv2.rectangle(img=background, pt1=(x1,y1), pt2=(x2,y2), color=(255, 255, 255), thickness=3)
85          cv2.putText(background, letter, (x1, y1 - 5), 1, 3, (255, 255, 255), 2, cv2.LINE_AA)
86          if capture:
87              cv2.putText(background, str(cont-1), (x2, y2 + 30), 1, 2, (255, 255, 255), 1, cv2.LINE_AA)
88          cv2.imshow("Sign capture", background)
```

En línea 36 aperturamos el modelo para la detección de la mano. Los parámetros son:

- static_image_mode=False, aquí determinamos que la imagen será en moviemiento
- max_num_hands=1, esta línea es para decirle al modelo que solo detecte una mano en la imagen
- min_detection_confidence=0.5, esto es para decirle que la confianza de detección sea del 50%

En la línea 39 la variable cont se inicia en 1, y es la que llevará la contabilidad de las imágenes capturadas y guardadas. El máximo valor posible de cont es 3000.

En la línea 17, la variable capture es un booleano que permite capturar, o no, las imágenes. Se inicia en false, hasta que nosotros, al presionar una tecla, cambiemos su valor a true y entonces empezará a capturar la imágenes.

Dentro del bucle while lo que hacemos es:

- Línea 42: con la función read() hacemos captura de frame con la cámara
- Línea 43: Con esta condición salimos del bucle while en caso no funcione la cámara
- Línea 45: Esperamos hasta que se haya alcanzado el límite de 3000 imágenes capturados, para así salir del bucle "while".
- Línea 49: con cv2, obtenemos el fondo de la imagen que nos permitirá binarizarlo (Fondo blanco con los puntos y líneas de la mano en blanco).
- Línea 51: Convertimos la imagen a formato RGB, ya que mediapipe lo necesita así para poder hacer la detección de la mano
- Línea 54: Validamos si hubo alguna detección de mano
- Línea 56: Ahora sí, si hubo alguna mano detectada, Mediapipe dibujará las líneas en las manos. En las líneas 58 y 59 indicamos que el color del trazo será de (255, 255, 255), esto equivale a blanco. A continuación, en la Ilustración 34 se muestra la obtenido:

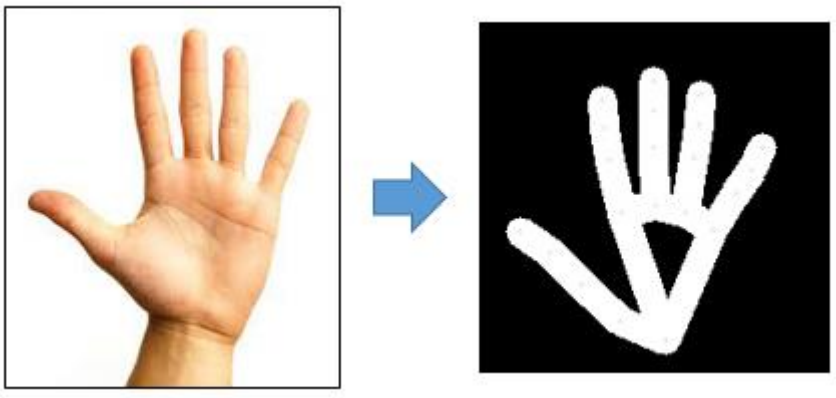


Ilustración 34: Binarización de la imagen con la ayuda de mediapipe

- En las líneas 62 hasta la 74, con la ayuda de cv2 conseguimos cubrir la parte oscura de la palma de la mano, y como resultado tenemos lo de la Ilustración 35

Ilustración 35: Con la ayuda de cv2 conseguimos una imagen de mano totalmente binarizada

- Líneas 75 y 76: Determinamos las coordenadas para poder dibujar un rectángulo de 200px por 200px en el frame. Sabiendo que el frame total tiene un ancho de 640px y 480px de alto, podemos calcular las coordenadas del rectángulo. El frame nos quedaría como lo observamos en la Ilustración 36

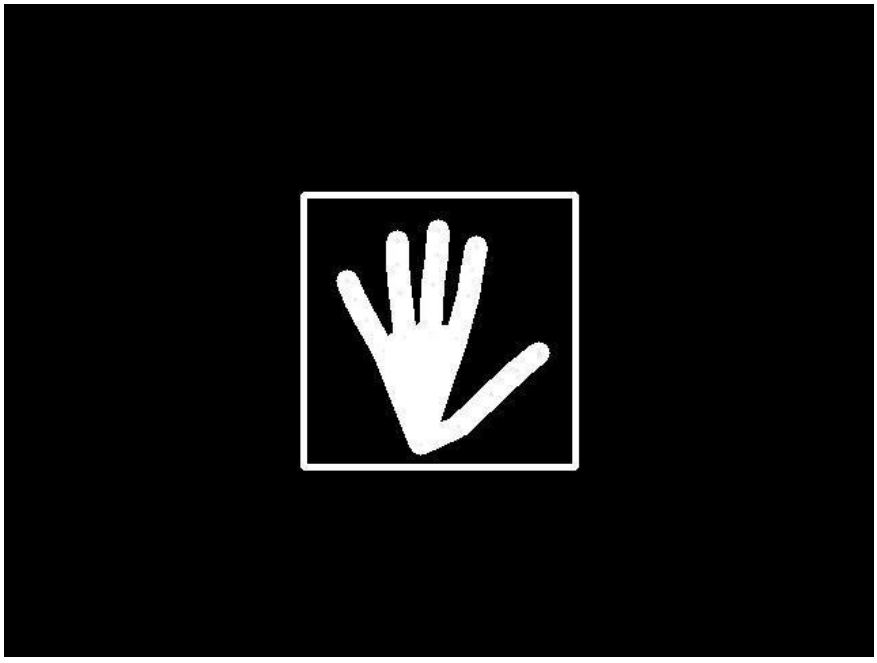


Ilustración 36: Procesado de imágenes.

Luego de la detección, dibujamos un rectángulo en la imagen para indicar la zona que será guardada con la imagen de la mano.

- Línea 80: En nuestro proyecto, las imágenes para entrenar nuestra red neuronal tendrán una dimensión de 50px por 50px, y esto lo conseguimos redimensionando las imágenes con la ayuda de cv2
- Línea 81: pasamos a guardar la imagen.

Luego que salimos del bucle while cerramos la ventana de captura

```
23    cap.release()
24    cv2.destroyAllWindows()
```

A continuación mostramos como quedaría la seña para cada letra, y para las señas especiales de espacio y eliminar

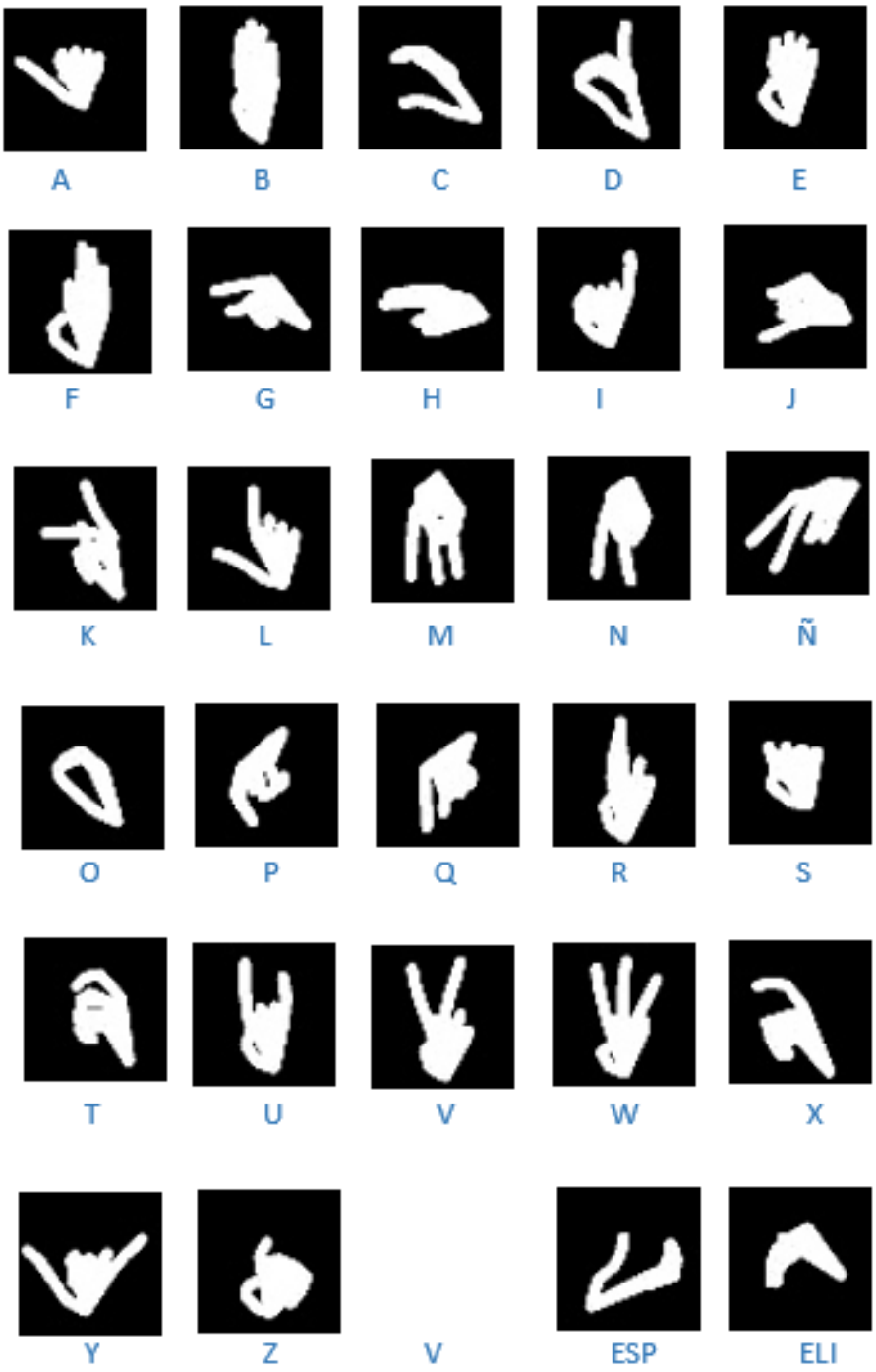


Ilustración 37: Imágenes binarias de las señas.

B. Entrenamiento de la red neuronal convolucional: Construcción del archivo CNN.py

Una vez obtenidas las 3000 imágenes por cada seña, haciendo un total de 87000 imágenes en la carpeta validation, y la misma cantidad en la carpeta training, podremos ahora sí entrenar la red neuronal.

En el Anexo 3 se podrá visualizar todo el código completo del archivo CNN.py. que es el que contiene el algoritmo para entrenar nuestra red neuronal.

Pero a continuación haremos una descripción del código.

```
CNN.py > ...
1   from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
2   from tensorflow.python.keras.models import Sequential
3   from tensorflow.python.keras.layers import Dropout, Flatten, Dense
4   from tensorflow.python.keras.layers import Convolution2D, MaxPooling2D
5   from tensorflow.python.keras import backend as k
6
```

Empezamos importando las librerías necesarias.

Después limpiamos cualquier sesión que haya antes de ejecutar el resto de líneas.

Luego de importar la clase ImageDataGenerator era necesario vaciar la sesión de Keras, con el propósito de resetear todos los valores de modelos creados en la etapa de test y error, consiguiendo de esta forma liberar recursos de almacenamiento y evitar que los modelos creados recientemente necesiten mucho tiempo de entrenamiento o tener como resultado un fallo de almacenamiento.

Lo hacemos con el método clear_session()

```
7   k.clear_session()
```

Luego especificamos las variables globales:

```
9   folder='resources/files/'
10  classes=29
11  iterations=3*classes
12  quantity=3000
13  height, width=50, 50
14  chanels=1
15  neurons=256
```

- Línea 9: indicamos la carpeta de donde va obtener las imágenes de validación y de entrenamiento, y también es la carpeta donde guardará el archivo model.h5 y weights.h5
- Línea 10: especificamos que son 29 clases
- Línea 11: el total de iteraciones que tendrá que hacer el entrenamiento será de 3 por cada clase, haciendo un total de 87 iteraciones
- Línea 12 y 13: quantity especifica que son 3000 imágenes por cada clase, y height con width son las dimensiones de dichas imágenes
- Línea 14: especifica que nuestras imágenes solo tienen 1 canal de color. Esto se debe a que son binarias.
- Línea 15: Es la cantidad de neuronas de nuestra red. En este caso solo usaremos 256 neuronas

Se definieron más variables

```
17  data_training=folder+'training'
18  data_validation=folder+'validation'
19  filtrosconv1=32
20  filtrosconv2=64
21  filtrosconv3=128
22  tam_filtro1=(4,4)
23  tam_filtro2=(3,3)
24  tam_filtro3=(2,2)
25  tam_pool=(2,2)
```

- Líneas 17 y 18: se estableció la ruta de acceso para el conjunto de datos de validación y entrenamiento, respectivamente.
- Líneas 19, 20 y 21: Se especifica que nuestra red neuronal tendrá tres filtros de 32, 64 y 128, respectivamente
- Líneas 22, 23 y 24: Se especifican los tamaños de los filtros.

```
27  preprocesamiento_entre=ImageDataGenerator(
28      rescale=1./255
29  )
30
31  preprocesamiento_vali=ImageDataGenerator(
32      rescale=1./255
33  )
```

Para los datos de de validación y entrenamiento, con la ayuda de clase ImageDataGenerator fueron normalizados mediante la propiedad rescale.

```
image_training=preprocesamiento_entre.flow_from_directory(
    data_training,
    target_size=(height, width),
    batch_size=1,
    class_mode='categorical',
    color_mode='grayscale'

)

image_validation=preprocesamiento_vali.flow_from_directory(
    data_validation,
    target_size=(height, width),
    batch_size=1,
    class_mode='categorical',
    color_mode='grayscale'
)
```

Así también, con la función flow_from_directory, se determinó que emplee la carpeta especificada y use como etiquetas los nombres de las subcarpetas de la misma. También, se determinó la resolución de las imágenes a través de la propiedad target_size. En esta investigación se usó una dimensión de 50 x 50, valores establecidos en las variables globales height y width. Acto seguido se estableció que se trabajaría con más de una clase (clasificación multiclase), determinando el valor de class_mode como categorical. Y ya que se tiene un set de imágenes en blanco y negro, fue preciso asignar valor al parámetro color_mode a grayscale (escala de grises); y por último, la propiedad batch_size que determina la cantidad de imágenes de entrenamiento en cada paso, definiendo un valor a uno

Creación de la red neuronal convolucional.

```
52  #Red neuronal convolucional (CNN)
53  cnn=Sequential()
54  cnn.add(Convolution2D(filtrosconv1,
55                        tam_filtro1,
56                        padding='same',
57                        input_shape=(height, width, chanels),
58                        activation='relu'))
59  cnn.add(MaxPooling2D(pool_size=tam_pool))
60
61  cnn.add(Convolution2D(filtrosconv2, tam_filtro2, padding='same', activation='relu'))
62  cnn.add(MaxPooling2D(pool_size=tam_pool))
63
64  cnn.add(Convolution2D(filtrosconv3, tam_filtro3, padding='same', activation='relu'))
65  cnn.add(MaxPooling2D(pool_size=tam_pool))
66
67  cnn.add(Flatten())
68  cnn.add(Dense(neurons, activation='relu'))
69  cnn.add(Dropout(0.5))
70  cnn.add(Dense(classes, activation='softmax'))
71
72  cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
73  cnn.fit(image_training,
74          steps_per_epoch=quantity,
75          epochs=iterations,
76          validation_data=image_validation,
77          validation_steps=quantity)
78  cnn.save(folder+'model.h5')
79  cnn.save_weights(folder+'weights.h5')
```

Línea 53: Hay dos maneras de elaborar modelos en Keras, funcionales y secuenciales. La API Functional elabora modelos flexibles y complejos, consecuentemente se hace un hecho el generar redes paralelas, en tanto que la API Sequential facilita diseñar modelos por niveles, las que mayormente son ideales para construir modelos de aprendizaje profundo. Basándonos en esto, usamos una red neuronal convolucional en la presente investigación.

Línea 54: Para adicionar una capa a la red se empleó el método add(). Mayormente una red neuronal convolucional tiene una capa de convolución (Convolutional2D) y luego consta de una capa de Pooling (MaxPooling2D).

Líneas 61 y 64: Se le agrega a la red dos capas de filtros más, con el objetivo de que la red sea más precisa.

El modelo requiere conocer qué tipo de entrada va a recibir, por esta causa la capa inicial debe obtener información sobre su tipo de entrada, las otras pueden realizar deducciones

automáticamente. Hay muchas maneras de conseguirlo. Empleamos el parámetro input_shape para determinar una tupla con las dimensiones de la entrada, teniendo como valor a (50, 50, 1), especificando el alto, ancho y la cantidad de canales de la imagen. Se determinó filtrosconv1=32 como la cantidad de filtros de salida en la convolución, con una dimensión del kernel de (2, 2) que determina la altura y el ancho del frame de convolución 2D. Utilizamos como función de activación a relu.

Línea 67: Antes de agregar una capa densa, nivelamos el tensor inicial empleando Flatten(): Este borra todas las medidas menos una, consiguiendo un nuevo tensor con una cantidad de elementos equivalentes a los existentes en el tensor de entrada.

Línea 68: Luego se agregó una capa densa completamente conectada con 256 neuronas que se relacionan con la capa previa; también se empleó la función de activación relu.

Línea 69: A continuación, se agregó la capa Dropout, la que aleatoriamente determina un porcentaje de elementos entrantes a 0 en todos las épocas de entrenamiento, contribuyendo a que ocurra un desbalance. Luego de testear el avance en muchos epochs, se estableció la tasa de Dropout en 0.5

Línea 70: Entonces se agregó una capa densa, pero aquí especificando como cantidad de salidas la cantidad de clases a predecir. Empleamos softmax como función de activación, esta convierte las salidas en una distribución de probabilidades cuyos márgenes están entre 0 a 1.

Línea 72: Con el diseño de capas establecido, ejecutamos la compilación del modelo, este proceso necesitó los parámetros a continuación: una función de pérdida (loss), establecida como categorical_crossentropy ya que estamos en una tarea de clasificación de varias clases; un optimizador, se determinó usar ‘adam’ el cual modifica los pesos de la red de manera iterativa en función de los datos de entrenamiento; y por último, las métricas empleadas para determinar la eficiencia del modelo, se escogió ‘categorical_accuracy’ que valida si el índice del valor superior verdadero es equivalente al índex del mayor valor esperado y ‘accuracy' para hallar la fracción de predicciones que el modelo logró detectar con éxito.

Línea 73: Para ir terminando, se empleó el método fit para entrenar la red. Especificamos que el entrenamiento sea por epochs=145 épocas. Así mismo, indicamos un número de

pasos para los datos de validación especificando un valor de quantity=3000. Y luego se le añadió la data de entrenamiento y validación.

Líneas 78 y 79: Luego de haber entrenado la red neuronal se crearon un par de archivos con extensión .h5: model.h5 y weights.h5. Estos portan la información ordenada del modelo, requeridos para la evaluación y posterior clasificación.

C. Verificación de la CNN

Primero, se asignó el modelo en una variable a través del método de Keras load_model; también, con el objetivo de conseguir las métricas requeridas y evaluar cuán fiable era el clasificador, realizamos una comparación del resultado de las predicciones del modelo obtenido contra el set de datos de prueba, y lo conseguimos con la función predict_classes. Por último, empleamos el método classification_report de la biblioteca sklearn. Entonces pudimos visualizar una tabla estructurada con las métricas obtenidas por las 29 clases de este proyecto.

CLASE	ETIQUETA	PRECISIÓN
0	A	0.99
1	B	0.96
2	C	0.95
3	D	0.98
4	E	0.98
5	ELI	0.99
6	ESP	0.98
7	F	0.97
8	G	0.99
9	H	0.96
10	I	0.98
11	J	0.98
12	K	0.96
13	L	0.97
14	M	0.98
15	N	0.99
16	Ñ	0.98
17	O	0.98
18	P	0.95
19	Q	0.99
20	R	0.97
21	S	0.97
22	T	0.98
23	U	0.99
24	V	0.98
25	W	0.99
26	X	0.98
27	Y	0.99
28	Z	0.97
Average		0.98

Tabla 7: Verificación de precisión de la CNN

Observamos en la tabla 7 que se consiguió una precisión resultante de 98%, con lo que concluimos que el modelo se ajusta notablemente a datos nuevos, siendo un valor considerablemente aceptable al estar por encima del 50%. Se encontró óptimo el modelo en esta investigación.

D. Elaboración de la interfaz de usuario: Construcción del archivo Main.py

El código completo del archivo Main.py estará disponible en el Anexo 4.

Primero se importaron las librerías necesarias

```
Main.py > selected
1   import tkinter as tk
2   from tkinter import Tk, Button, Label, Entry, PhotoImage, Radiobutton, messagebox
3   from PIL import Image, ImageTk, ImageFont, ImageDraw
4   import cv2
5   import pyttsx3
6   import mediapipe as mp
7   import os
8   import numpy as np
9   import time
10  from keras_preprocessing.image import img_to_array
11  from keras.models import load_model
12  import speech_recognition as sr
13  from threading import Thread
14
```

La interfaz gráfica de usuario se elaboró con la libreía tkinter.

El sistema tiene dos formas de iniciar la conversación: lo puede iniciar la persona hablante, o lo puede iniciar la persona con discapacidad auditiva.

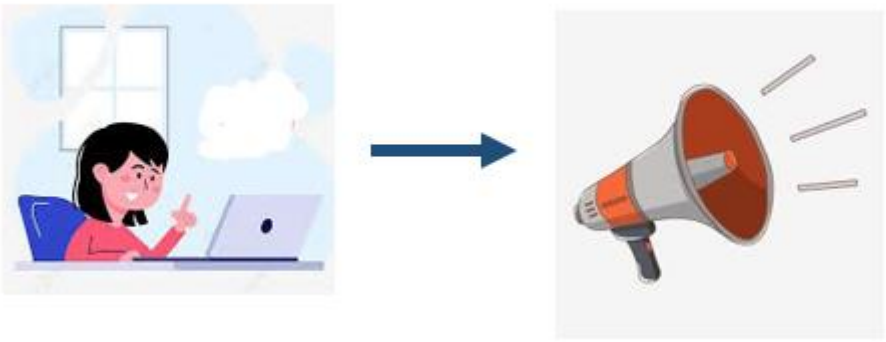


Ilustración 38: Inicio de conversación de persona sorda.

Si el diálogo lo inicia la persona sorda, el sistema estará atento a las señas que haga, para luego convertirla a voz, tal como lo indica la Ilustración 37.

Si el diálogo lo inicia la persona hablante, el sistema estará atento a la voz para convertirlo a señas, tal como lo indica la Ilustración 38

Ilustración 39: Inicio de conversación por persona hablante

Luego de importar lo necesario, creamos las variables globales:

```
#Global variables
listening=False
seeing=False
showing=False
speaking=False
window=Tk()
selected=tk.IntVar()
cameraDisableI = ImageTk.PhotoImage(Image.open("resources/images/camera_disable.png"))
microphoneDisableI = ImageTk.PhotoImage(Image.open("resources/images/microphone_disable.png"))
manos = mp.solutions.hands.Hands(max_num_hands=1)
```

Para que el sistema emita voz, usamos la dependencia pyttsx3, la cual no necesita conexión a internet. Y asignamos el tipo de voz que usará:

```
#Habilitamos el motor de voz de pyttsx3
engine=pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty("voice", voices[2].id)
engine.setProperty("rate", 100)
```

Luego cargamos el modelo y pesos para la predicción:

```
#Cargamos el modelo y pesos para la predicción
folder='resources/files/'
cnn = load_model(folder+'model.h5')
cnn.load_weights(folder+'weights.h5')
dire_img = os.listdir(folder+'validation')
```

Luego creamos el método que puede detener la conversación, en caso sea necesario:

```
38  #Función para terminar la conversación
39  def finish(event):
40      global listening
41      global showing
42      global seeing
43      global speaking
44      seeing=False
45      listening=False
46      showing=False
47      speaking=False
48      textE.delete(0, tk.END)
49      textE['state']='disable'
50      voiceB['state']='normal'
51      signB['state']='normal'
52      voiceL['image']=microphoneDisableI
53      signL['image']=cameraDisableI
54      finishB['state']='disable'
```

La clase Main.py tendrá cuatro métodos principales: El primero lo hemos llamado listen(), el cual escucha si la persona hablante está emitiendo voz; el segundo es show(), el cual muestra las señas a la persona sorda luego de haber convertido la voz a señas; el tercero es el método see(), que es el que se encarga que capturar las señas, pasarlas al modelo y hacer la predicción; y por último, el método speak() emite voz luego de que la persona sorda ha formado las palabras con el alfabeto de señas.

Al final la apariencia de la interfaz gráfica será la siguiente, tal cual la Ilustración 39:

Ilustración 40: Interfaz de usuario del sistema

CAPÍTULO 5: EVALUACIÓN DE LOS RESULTADOS

Para evaluar los resultados tuvimos dos etapas: Pre-test y post-test

5.1 Etapa de post-test

5.1.1 Participantes

El testeo se realizó con 7 alumnos de la Institución Educativa Bautista para sordos Harvest. Se evaluó el número promedio de señas interpretadas correctamente por el oyente y el tiempo promedio que tarda un estudiante con deficiencia auditiva del colegio Bautista Harvest en realizar una seña legible para una persona oyente.

5.1.2 Escenario

Con el objetivo de implementar el escenario, se determinó los indicadores que se detallan en la tabla a continuación:

INDICADOR	TIPO
Cantidad promedio de señas interpretadas correctamente por el oyente	Cuantitativo
Tiempo promedio que tarda un estudiante con deficiencia auditiva del colegio Bautista Harvest en realizar una seña legible para una persona oyente	Cuantitativo

Tabla 8: Indicadores a evaluar en la etapa de pre-test

5.1.3 Resultados obtenidos

A. Cantidad de señas interpretadas por el oyente

Por fines didácticos, el check (✓) significa que si reconoció la seña, y el aspa (x) significa que no reconoció la seña (letra):

CLASES	ESTUDIANTES						
	1	2	3	4	5	6	7
A	x	✓	x	✓	✓	x	✓
B	✓	✓	x	✓	x	✓	x
C	✓	✓	x	x	✓	✓	✓
D	x	✓	x	✓	✓	✓	x
E	x	x	x	x	x	x	x
F	✓	x	x	x	✓	✓	✓
G	x	✓	x	✓	✓	✓	x
H	✓	✓	x	✓	✓	✓	✓
I	x	x	✓	✓	✓	✓	✓
J	✓	✓	x	x	✓	x	x
K	✓	✓	✓	✓	✓	✓	✓
L	✓	x	x	x	x	✓	✓
M	✓	✓	✓	✓	✓	✓	✓
N	x	✓	x	x	✓	✓	✓
Ñ	x	✓	✓	✓	x	✓	✓
O	✓	x	✓	✓	✓	✓	✓
P	✓	✓	x	✓	✓	x	✓
Q	✓	✓	✓	x	✓	✓	x
R	x	x	x	✓	✓	✓	✓
S	✓	✓	✓	✓	x	✓	✓
T	✓	✓	✓	✓	✓	✓	x
U	x	x	x	x	✓	✓	✓
V	x	✓	x	✓	✓	x	✓
W	✓	✓	x	✓	✓	✓	x
X	x	✓	✓	x	✓	✓	✓
Y	✓	✓	x	✓	x	x	x
Z	x	✓	✓	✓	x	✓	✓
ELI	x	x	x	x	x	x	x
ESP	x	x	x	x	x	x	x
TOTAL	15	20	10	18	20	22	18

Tabla 9: Cantidad de señas interpretadas por el oyente.

Se aprecia que en promedio solo se reconoció 17 señas de 29

B. Tiempo promedio en que un estudiante realiza una seña legible

El tiempo fue medido en segundos, y se hizo con un cronómetro.

CLASES	ESTUDIANTES							TOTAL
	1	2	3	4	5	6	7	
A	3.7	1.8	1.4	1.6	3.4	2.1	2.6	2.37
B	1.9	2.7	3.6	1.6	1.3	2.4	2.9	2.34
C	2.3	3.8	3	2.9	3.7	2.1	2.8	2.94
D	1.9	2.2	1	3.7	3.7	2.6	2.9	2.57
E	3.4	1.9	3.7	1	3	3.5	2.7	2.74
F	2.5	2.7	3.4	2.2	2.4	2.2	2.4	2.54
G	1.9	1.6	1.7	1.4	3.6	4	3.2	2.49
H	2.1	3.6	1.9	1.5	3	1.7	3.2	2.43
I	3.1	1.4	2.5	2	2.9	1.6	3.6	2.44
J	1.4	2.8	2.3	2.3	3.6	1	2.4	2.26
K	2.2	1.4	3.8	3.2	2.7	3.4	3.3	2.86
L	1.9	2.2	3.6	1	3.1	2.3	1.4	2.21
M	1.4	2.9	3.6	3	3.4	1.5	1.6	2.49
N	1.5	2.3	3.1	1	1.9	3.4	3	2.31
Ñ	2.7	1.6	2.6	3.3	2.7	1.6	2.6	2.44
O	1.2	3.9	1.5	1.5	3.9	3.2	2.1	2.47
P	2.7	2.3	3.3	1.6	3.1	1.6	2.7	2.47
Q	1.2	2.3	1.9	1.6	2.7	3.1	2.5	2.19
R	2.6	2.8	3.5	2.5	2.7	4	2.5	2.94
S	3.1	2.5	1.9	3.2	3.9	2	1.4	2.57
T	2.1	1.8	1.9	3.1	3.6	2.7	1.7	2.41
U	2.7	1	1.3	2.7	2.3	1.6	2.7	2.04
V	4	3.1	2.3	2.1	1.7	2.8	1.9	2.56
W	3.3	1.4	2.1	3.9	4	2.4	2.4	2.79
X	2.9	2.1	2.1	3.9	1.3	2.8	1.6	2.39
Y	3.7	2.4	2	3.8	3.3	2.2	1.8	2.74
Z	2.1	2.6	1.9	2.5	2.7	2.2	1.5	2.21
ELI	3.8	1.9	3.6	4	1.8	2.3	2.7	2.87
ESP	1.2	1.3	1.4	1.7	2	1.8	1	1.49
							Total	2.47

Tabla 10: Tiempo promedio en que un estudiante realiza una seña legible

Según vemos en la tabla anterior, el promedio de tiempo que tarda un estudiante en realizar una seña legible es de 2.47 segundos

5.2 Etapa de Post-test

2. 5.2.1 Participantes

El testeo del software se realizó con 7 alumnos de la Institución Educativa Bautista para sordos Harvest. La interrelación entre el alumno con problemas de sordera y una persona oyente se hizo a través del sistema computacional basado en inteligencia artificial de reconocimiento del alfabeto de señas peruano. La entrada del software fue una señal continua de video mediante una cámara integrada al ordenador (aquí se usó una laptop). Lo obtenido de la cámara se procesa, logrando de resultado la clase o seña predicha por el clasificador. La contrastación de la hipótesis se hizo conforme al método de realización de test, para decidir si se rechaza o acepta la hipótesis.

3. 5.2.2 Especificación del escenario o ambiente

Se evaluó el tiempo promedio que demora un alumno con problemas de sordera en comunicarse con una persona que desconoce el alfabeto dactilológico,: también evaluamos el número de señas interpretadas y reconocidas por el sistema, y el tiempo promedio que tarda el sistema en reconocer dichas señas. De igual forma, se determinó como tiempo máximo 30 segundos, esto si la seña formada no se reconoce por el sistema. Con el objetivo de implementar el escenario, se determinó los indicadores que se detallan en la tabla 7:

INDICADOR	TIPO
Cantidad de señas reconocidas por el sistema	Cuantitativo
Tiempo promedio que tarda el sistema en reconocer una seña	Cuantitativo

Tabla 11: Indicadores a evaluar en la etapa de post-test

4. 5.2.3 Resultados obtenidos:

A. Cantidad de señas reconocidas por el sistema

Primero mostramos la tabla donde se puede apreciar cuantas señas (letras) fue capaz de reconocer el sistema por cada uno de los siete estudiantes de la muestra.

Por fines didácticos, el check (✓) significa que si reconoció la seña, y el aspa (x) significa que no reconoció la seña (letra):

CLASES	ESTUDIANTES						
	1	2	3	4	5	6	7
A	✓	✓	✓	✓	✓	✓	✓
B	✓	✓	✓	✓	✓	✓	✓
C	✓	✓	✓	✓	✓	✓	✓
D	✓	✓	✓	✓	✓	✓	✓
E	✓	✓	✓	✓	✓	✓	✓
F	✓	✓	✓	✓	✓	✓	✓
G	✓	✓	✓	✓	✓	✓	✓
H	✓	✓	✓	✓	✓	✓	✓
I	✓	✓	✓	✓	✓	✓	✓
J	✓	✓	✓	✓	✓	✓	✓
K	✓	✓	✓	✓	✓	✓	✓
L	✓	✓	✓	✓	✓	✓	✓
M	✓	✓	✓	✓	✓	✓	✓
N	✓	✓	✓	✓	✓	✓	✓
Ñ	✓	✓	✓	✓	✓	✓	✓
O	✓	✓	✓	✓	✓	✓	✓
P	✓	✓	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓	✓	✓
R	✓	✓	✓	✓	✓	✓	✓
S	✓	✓	✓	✓	✓	✓	✓
T	✓	✓	✓	✓	✓	✓	✓
U	✓	✓	✓	✓	✓	✓	✓
V	✓	✓	✓	✓	✓	✓	✓
W	✓	✓	✓	✓	✓	✓	✓
X	✓	✓	✓	✓	✓	✓	✓
Y	✓	✓	✓	✓	✓	✓	✓
Z	✓	✓	✓	✓	✓	✓	✓
ELI	✓	✓	✓	✓	✓	✓	✓
ESP	✓	✓	✓	✓	✓	✓	✓
TOTAL	29	29	29	29	29	29	29

Tabla 12: Cantidad de señas reconocidas por el sistema.

Según vemos en la tabla 8, el sistema reconoció las 29 señas con las que hemos trabajado en el presente proyecto. Este reconocimiento lo logró en los siete estudiantes de nuestra muestra. De esta manera se percibe que el modelo está muy bien entrenado para poder reconocer todas las señas del alfabeto dactilológico.

B. Tiempo promedio que tarda el sistema en reconocer una seña

Ahora mostraremos el tiempo en que demoró el sistema en reconocer cada seña según el estudiante.

El tiempo fue medido en segundos, y se hizo con un cronómetro.

CLASES	ESTUDIANTES							TOTAL
	1	2	3	4	5	6	7	
A	1.3	2	1.9	1.4	1.6	2	1.2	1.63
B	1.9	1.9	1.7	1.6	1.9	2	1.7	1.81
C	1.2	1.1	1.4	1.4	1.5	1	1.3	1.27
D	1.6	1.5	1.4	1.7	1.5	1.9	1.4	1.57
E	1.3	1.2	1.7	1.1	1.2	1.7	1.5	1.39
F	1.7	1.5	1.7	1.6	1.5	2	1.7	1.67
G	1.6	1.6	1.8	1.9	1	1	1.4	1.47
H	1.4	1.6	1.7	1.6	1.9	1.6	1.5	1.61
I	1.2	1.9	1.3	1.7	1.9	1.6	1.3	1.56
J	2	1.7	1.8	1.6	1.8	1.2	1.6	1.67
K	1.7	2	1.8	1	2	1.8	1.7	1.71
L	1.6	1.9	1	1.5	1.8	1	1.7	1.50
M	1	1.2	1.3	1.9	1.7	1.3	1.9	1.47
N	2	1.6	1.3	1.3	1.9	1.7	1.8	1.66
Ñ	1.5	1.4	1	1.2	1.7	1.8	1.8	1.49
O	1.6	2	1.3	2	1.2	1.4	1.2	1.53
P	1.9	1.9	1.4	1.1	1.7	1.8	1.3	1.59
Q	1.4	1.7	1.9	1.9	1.2	1.6	1.1	1.54
R	1.8	1.8	1.1	1.7	1.4	1.5	1.5	1.54
S	1.5	1.6	1.3	1.2	1.4	1.3	1.5	1.40
T	1.7	1.7	1.5	1.8	1.6	1.3	1.9	1.64
U	1.2	2	1.1	1.5	2	1.6	1.8	1.60
V	1.6	1	1	1.1	1.7	1.6	1.2	1.31
W	1.7	1.9	1.3	1.3	1.9	1.2	1.4	1.53
X	1.2	1.4	1.9	1.3	1.9	1	1.7	1.49
Y	1.8	1.4	1.8	1.6	1.6	1.2	1.2	1.51
Z	1.4	1.1	1.3	1.4	1	2	1.8	1.43
ELI	1.3	1.9	1.1	1.5	1.4	1.6	2	1.54
ESP	1.1	2	1.5	1.6	1.4	1.9	1.7	1.60
							Total	1.54

Tabla 13: Tiempo promedio que tarda el sistema en reconocer una seña.

Según vemos en la tabla 9, el promedio de tiempo que tarda el sistema en reconocer una seña es de 1.54 segundos, evidenciando que el sistema no tarda más de dos segundos en poder reconocer una seña. Lo que permite a la persona con discapacidad auditiva, formar oraciones simples en menos de un minuto.

C. Tiempo promedio que tarda un estudiante con deficiencia auditiva en comunicarse con una persona oyente.

En esta medición lo que se calcula es el tiempo de respuesta de parte de la persona oyente, tras haber la persona sorda formado una palabra con señas y el sistema haberlo lanzado

en voz. A este tiempo se adhiere lo que tardó el sistema en convertir la voz de la persona oyente a señas.

Esta medición se hizo en minutos, y fue hecha con un cronómetro.

ESTUDIANTES							TOTAL
1	2	3	4	5	6	7	
2	1.6	1.5	1.8	1.4	1.7	1	1.57
2	1.6	1.4	1.7	1.6	1	1.9	1.60
1.7	1.9	1.7	1	1.5	1.5	2	1.61
1.5	1.4	1.5	2	1.8	1.1	1.9	1.60
1.7	2	1.7	1	1.3	1.5	1.4	1.51
1.2	1.7	1.3	2	1.1	1.9	1.2	1.49
1.8	1.5	1.8	1.3	2	1.3	1.5	1.60
1.6	1	1.4	1.2	2	1.2	1.6	1.43
1.7	2	1.8	1.8	1	1.1	1.7	1.59
1.4	1.5	1.6	1.5	1.3	1.3	1.9	1.50
1.1	1.5	1.5	1.2	1.6	1.6	1.9	1.49
1.1	1.4	1.4	1.2	1.8	1.5	1	1.34
1.1	1.1	1.4	1.2	2	1.4	1.8	1.43
2	1.5	2	1.1	1	1.8	1.8	1.60
1.7	1.7	1.9	1.4	1.8	1.6	2	1.73
1.1	2	1.8	1.4	1.9	1.8	1.8	1.69
1.9	1.2	1.8	1.2	1	1.8	1	1.41
1.2	1.9	1.4	1.3	2	1.7	1.8	1.61
2	1.5	2	2	1.6	1	1.6	1.67
1.2	1.8	1.5	1	1.2	1.4	1.9	1.43
1.2	1.4	1.4	2	1.9	1.3	1.9	1.59
1	1.6	1.3	1.4	2	1.9	1.3	1.50
1.7	1.7	1.2	1	2	1.7	2	1.61
1.7	1.8	1.3	1.8	1.9	1.6	1	1.59
1.7	1.1	1.8	1.6	2	1.8	1.8	1.69
1.8	1.6	1	1.6	2	1.6	1.9	1.64
1.4	1.1	1.1	1	1.9	1.7	1	1.31
1.8	1.9	1.2	1.8	1.7	1.8	1.3	1.64
1.8	1.7	1.3	1	1.8	1.3	1.2	1.44
						Total	1.55

Tabla 14: Tiempo que tarda un estudiante con deficiencia auditiva en comunicarse con una persona oyente.

Según vemos en la tabla 10, para que haya comunicación fluida entre una persona con discapacidad auditiva y otra oyente, y haciendo uso del sistema, apenas se tarda 1.55 minutos de tiempo.

Este tiempo es relativamente corto, pero se comprueba la hipótesis de que el sistema sí permite la comunicación entre un alumno con discapacidad auditiva y otra persona oyente

CAPÍTULO 6: CONCLUSIONES

En esta investigación se analizó el uso del alfabeto de señas en los alumnos del colegio Harvest, determinando que el 30% de las expresiones realizadas por los alumnos del colegio Harvest, las transmiten haciendo uso del alfabeto de señas.

Así mismo, también se determinó hacer uso de Python para desarrollar el sistema inteligente de reconocimiento del alfabeto de señas, ya que más del 60% de los científicos de datos y desarrolladores de inteligencia artificial y aprendizaje automático utilizan este lenguaje, constituyendo así una enorme comunidad con amplio soporte, y facilitando el uso de librerías como Tensorflow, Keras y Mediapipe para la implementación de algoritmos de inteligencia artificial.

Esta investigación demuestra un potencial de reconocer las señas del alfabeto de señas peruano mediante detección automática de imágenes. Se implementó una metodología que recolecta, pre procesa y reconoce las imágenes de las señas utilizando tecnologías de visión artificial e IA. El sistema desarrollado con arquitectura CNN logra una precisión del 98% entrenado con un dataset de prueba de 29 signos.

BIBLIOGRAFÍA

- AAAI ORGANIZATION. Recuperado el 19 de octubre del 2021. Disponible en: http://www.aaai.org/home.html

- Aguilar Martínez José Luis (2008). *Manual de atención al alumnado con necesidades específicas de apoyo educativo derivadas de discapacidad auditiva*. España

- Anderson, J. (1995). Introducción a las Redes Neuronales. USA: MIT Press

- Baena Paz, G. (2014). Metodología de la investigación. México: Patria.

- Belavagi, M. C., & Muniyal, B. (2016). Performance Evaluation of Supervised Machine Learning. India: Manipal Institute of Technology.

- Blum, A. (1992). Redes neuronales: un marco orientado a objetos para construir sistemas conexionistas. New York: John Wiley & Sons.

- C. M. Bishop (2006), Pattern Recognition and Machine Learning. Springer.

- Cairo O. (2011). *El Hombre artificial. El futuro de la Tecnología.* México. Editorial Alfaomega.

- Caudill, M. & Butler, Ch. (1992). Entendiendo Redes Neuronales. USA: Computer Explorations, M.I.T, Press.

- Charytoniuk, W., Box, E.D., Lee, W.J., Chen, M.S., Kotas & P. Van Olinda, P. (2000). Previsión de la demanda basada en redes neuronales en un entorno desregulado, Industry Applications, IEEE Transactions on, Volumen 36, serie 3, (pp. 893-898).

- Cedano Olvera Marco A., Cedano Rodríguez Alfredo, Rubio González José Antonio & Vega Gutiérrez Arlem Carolina (2014). *Fundamentos de Computación para Ingenieros*. México.

- Cohen, L. & Manion, L. (2002). Métodos de investigación educativa. Madrid: La Muralla.

- Contaval. (18 de Febrero de 2016). ¿Qué es la visión artificial y para qué sirve? Obtenido de Contaval: https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/

- COVANTEC, Programación en Python nivel básico. Disponible en: https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion7/archivos.html

- Cuevas E., Zaldívar D. & Pérez M. (2010). Procesamiento digital de imágenes con Matlab y Simulink. México: Alfaomega.

- Cuevas, E., Díaz, M., & Camarena, J. (2017). Tratamiento de imágenes con MATLAB. México, Alfaomega

- D. J. Wu, "End-to-end text recognition with convolutional neural networks," 2012. Disponible en: https://crypto.stanford.edu/~dwu4/papers/ HonorThesis.pdf

- Damiani L. (2019). Optimización Estocástica Acelerada con Aplicación a la Ingeniería de Procesos. Disponible en: https://repositoriodigital.uns.edu.ar/bitstream/handle/123456789/4664/TesisMIPP_Luc%C3%ADa%20Damiani.pdf?sequence=1&isAllowed=y

- Defensoría del Pueblo (2020), Facilitar el aprendizaje de la lengua de señas peruano. Disponible en: https://www.defensoria.gob.pe/defensoria-del-pueblo-debe-facilitarse-el-aprendizaje-de-la-lengua-de-senas-peruana-y-promover-la-identidad-linguistica-y-cultural-de-las-personas-sordas/

- El País (9 de junio del 2014). Un ordenador supera el Test de Turing simulando ser un chico de 13 años. España

- Freeman, J. y Skapura, D. (1991). Redes Neuronales - Algoritmos, Aplicaciones y Técnicas de Programación. Willington: Adison Wesley.

- Gandhi R. (2018). “A look at gradient descent and rms-prop optimizers”. Disponible en: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

- García I. (2008). *Visión Artificial y Procesamiento Digital de Imágenes usando Matlab*. Ibarra

- Gómez Esteban, L. I., & Posada Sepúlveda, I. E. (2012). Barreras comunicativas que influyen en la interacción social entre la población oyente y sorda. Bogotá: Grupo de investigaciones en psicología, ciencia y tecnología.

- González Rafael & Woods Richard (1996). Tratamiento digital de imágenes. Ed. Wilmington, DL: Addison-Wesley Iberoamericana

- Grupo de Inteligencia artificial y Robótica (2014). Grupo de inteligencia Artifical y Robótica. (U. T. Aires). Recuperado el 19 de octubre de 2021. Disponible en: http://www.secyt.frba.utn.edu.ar/gia/inteligencia_artificial.htm

- Hernández, R., Fernández, C., & Baptista, P. (2010). Metodología de le investigación (5ta ed.). México D.F.: McGraw-Hill Interamericana.

- Hernández Sampieri Roberto (2014). Metodología de la investigación. México. McGRAW-HILL

- Hertz, J., Krogh, A. & Palmer, R.G. (1991). Introducción a la teoría de la computación neural. California: Addison-Wesley, Redwood City.

- Hilera, J. y Martínez, V. (1995). Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Madrid: Alfaomega.

- Hipertextual. (27 de febrero del 2015). Recuperado 19 de octubre de 2021, de Hipertextual.com: http://hipertextual.com/2015/02/inteligencia-artificial-de-google

- Jaramillo, M.A., Carmona, D., González, E. y Álvarez, J.A. (2005). Predicción de series temporales con Redes Neuronales. Aplicación a la demanda de energía eléctrica Avances Recientes, (pp 247-251).

- Kingma D. P. & Ba J. L. (2014) "Adam: A method for stochastic optimization"

- Martin, B. y Sanz, A. (2007). Redes Neuronales y Sistemas Borrosos. Madrid: Alfaomega.

- McNulty K., (2018). "What is machine learning?. Disponible en: https://towardsdatascience.com/what-is-machine-learning-891f23e848da

- Ministerio de Educación (2015). *Lengua de Señas Peruana*. Disponible en https://www.conadisperu.gob.pe/observatorio/wpcontent/uploads/2019/03/Lengua-de-se%C3%B1as-peruana-gu%C3%ADa-para-el-aprendizaje-de-la-lengua-de-se%C3%B1as-peruana-vocabulario-b%C3%A1sico-44-87.pdf

- Nada B., I., Mazen M., S., & Zayed, H. (2017). An Automatic Arabic Sign Language Recognition System (ArSLRS). Egypto.

- Quispe Aduviri M. (2013). Reconocimiento de gestos para la interacción por computador, con realidad aumentada. Disponible en: https://repositorio.umsa.bo/bitstream/handle/123456789/7814/T.2768.pdf?sequence=1&isAllowed=y

- Roguiguez R. & Sossa, J. (2012). Procesamiento y Análisis Digital de Imágenes. México: Alfaomega.

- RT en Español. (19 de julio de 2014). Disponible en: http://actualidad.rt.com/ciencias/view/134359-fotos-curiosity-marte-tierra

- Sagan C. (2010). Cosmos (26 ed.). Barcelona, España.

- Tawfiq, A. & Ibrahim, E. (1999). Redes neurales artificiales aplicadas a la predicción de la demanda a largo plazo, Inteligencia Artificial en Ingeniería, Volumen 13, Serie 2, (pp. 189-197)

- Y. L. Boureau, F. Bach, Y. LeCun, & J. Ponce, “Learning mid-level features for recognition,” 2010.

ANEXOS

Anexo 1: Librerías de Python usadas en el presente proyecto

absl-py==1.0.0

astunparse==1.6.3

attrs==21.4.0

beautifulsoup4==4.11.1

cachetools==5.1.0

certifi==2022.5.18.1

charset-normalizer==2.0.12

comtypes==1.1.11

cycler==0.11.0

docopt==0.6.2

flatbuffers==1.12

fonttools==4.33.3

gast==0.4.0

google-auth==2.6.6

google-auth-oauthlib==0.4.6

google-pasta==0.2.0

grpcio==1.46.3

h5py==3.7.0

idna==3.3

Js2Py==0.71

keras==2.9.0

Keras-Preprocessing==1.1.2

kiwisolver==1.4.2

libclang==14.0.1

Markdown==3.3.7

matplotlib==3.5.2

mediapipe==0.8.10

numpy==1.22.4

oauthlib==3.2.0

opencv-contrib-python==4.5.5.64

opt-einsum==3.3.0

packaging==21.3

Pillow==9.1.1

pipwin==0.5.2

protobuf==3.19.4

pyasn1==0.4.8

pyasn1-modules==0.2.8

PyAudio @ file:///C:/Users/israe/Documents/VisualStudioCodeProjects/SignAlphabetTranslatorMediapipe/PyAudio-0.2.11-cp310-cp310-win_amd64.whl

pyjsparser==2.7.1

pyparsing==3.0.9

pypiwin32==223

PyPrind==2.11.3

pySmartDL==1.3.4

python-dateutil==2.8.2

pyttsx3==2.90

pytz-deprecation-shim==0.1.0.post0

pywin32==304

requests==2.27.1

requests-oauthlib==1.3.1

rsa==4.8

scipy==1.8.1

six==1.16.0

soupsieve==2.3.2.post1

SpeechRecognition==3.8.1

tensorboard==2.9.0

tensorboard-data-server==0.6.1

tensorboard-plugin-wit==1.8.1

tensorflow==2.9.1

tensorflow-estimator==2.9.0

tensorflow-io-gcs-filesystem==0.26.0

termcolor==1.1.0

typing_extensions==4.2.0

tzdata==2022.1

tzlocal==4.2

urllib3==1.26.9

Werkzeug==2.1.2

wrapt==1.14.1

Anexo 2: Contenido del archivo parameter.json


```
{
    "classes":29,
    "quantity":3000,
    "width":50,
    "height":50,

    "chanels":1,
    "neurons":256,
    "iterations":5
}
```

Anexo 3: Código del archive Sign.py

```
import cv2
import mediapipe as mp
import os
import numpy as np
import json

# Read parameters ##################################################################
with open("parameters.json") as data:
    parameters=json.loads(data.read())


# Initial conditions ##################################################################
letter='a'
letter=letter.upper()
# folder='resources/files/validation/'+letter
folder='resources/files/training/'+letter
quantity=parameters['quantity']
width, height=parameters['width'], parameters['height']


# Camera ##################################################################
cap=cv2.VideoCapture(0)


# Create folder if it doesn't exists ##################################################################
if not os.path.exists(folder):
    os.makedirs(folder)
```

```
#                                                                    Mediapipe
#####################################################################
mp_hands=mp.solutions.hands
mp_drawing=mp.solutions.drawing_utils



#Loop                                                                    body
#####################################################################
with mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=1,
    min_detection_confidence=0.5
) as hands:
    cont=1
    capture=False
    while True:
        # Read image from camera
        ret, frame=cap.read()
        if ret==False:
            break


        # If ESC key pressed or get quantity imagen, then exit
        if cv2.waitKey(1) & 0xFF==27 or cont>quantity:
            break;


        # If SPACE BAR key pressed, then capture and save image
        if cv2.waitKey(1) & 0xFF==32:
            capture=True


        # Getting backgroun image
```

```
        background=cv2.imread('resources/images/background.png')

        # Working mediapipe
        frame_rgb=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results=hands.process(frame_rgb)
        hands_image=results.multi_hand_landmarks
        if hands_image is not None:
            for hand_landmarks in hands_image:
                # Paintng dots and linies
                mp_drawing.draw_landmarks(
                    background, hand_landmarks, mp_hands.HAND_CONNECTIONS,
                    # Changing color
                    mp_drawing.DrawingSpec(color=(255,255,255), thickness=0,
circle_radius=0),
                    mp_drawing.DrawingSpec(color=(255,255,255), thickness=16)
                )

                # Fill triangle
                h, w, _= background.shape

p1x=int((hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MC
P].x)*w)

p1y=int((hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_MC
P].y)*h)

p2x=int((hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_M
CP].x)*w)

p2y=int((hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_M
CP].y)*h)

p3x=int((hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_MCP].
x)*w)
```

```
p3y=int((hand_landmarks.landmark[mp_hands.HandLandmark.RING_FINGER_MCP].
y)*h)

p4x=int((hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].x)*w)

p4y=int((hand_landmarks.landmark[mp_hands.HandLandmark.PINKY_MCP].y)*h)

p5x=int((hand_landmarks.landmark[mp_hands.HandLandmark.WRIST].x)*w)

p5y=int((hand_landmarks.landmark[mp_hands.HandLandmark.WRIST].y)*h)
            points = np.array([[p1x, p1y], [p2x, p2y], [p3x, p3y], [p4x, p4y], [p5x, p5y]])
            cv2.fillPoly(background, pts=[points], color=(255, 255, 255))



        # Points for cut and rectangle
        x1, y1=220, 140
        x2, y2=420, 340

        # Saving cut image
        if capture:
            cut=background[y1:y2, x1:x2]
            background=cv2.cvtColor(background, cv2.COLOR_BGR2GRAY)
            cut=cv2.resize(cut, (height,width))
            cv2.imwrite(folder+'/'+letter+'_{}.jpg'.format(cont), cut)
            cont+=1

        # Inverse horizontally image before show it
        background= cv2.flip(background, 1)

        # Put rectangle and text
        cv2.rectangle(img=background, pt1=(x1,y1), pt2=(x2,y2), color=(255, 255, 255),
thickness=3)
```

```
        cv2.putText(background, letter, (x1, y1 - 5), 1, 3, (255, 255, 255), 2, cv2.LINE_AA)
        if capture:
            cv2.putText(background, str(cont-1), (x2, y2 + 30), 1, 2, (255, 255, 255), 1, cv2.LINE_AA)

        # Showing image
        cv2.imshow("Sign capture", background)


cap.release()
cv2.destroyAllWindows()
```

Anexo 4: Código del archivo CNN.py

```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dropout, Flatten, Dense
from tensorflow.python.keras.layers import Convolution2D, MaxPooling2D
from tensorflow.python.keras import backend as k
import json

k.clear_session()



# Read parameters
##############################################################
with open("parameters.json") as data:
    parameters=json.loads(data.read())



# Initial conditions
##############################################################
folder='resources/files/'
classes=parameters['classes']
iterations=parameters['iterations']*classes
quantity=parameters['quantity']
height, width=parameters['height'], parameters['width']

chanels=parameters['chanels']
neurons=parameters['neurons']

data_training=folder+'training'
data_validation=folder+'validation'
filtrosconv1=32
filtrosconv2=64
```

```
filtrosconv3=128
tam_filtro1=(4,4)
tam_filtro2=(3,3)
tam_filtro3=(2,2)
tam_pool=(2,2)

preprocesamiento_entre=ImageDataGenerator(
    rescale=1./255
)

preprocesamiento_vali=ImageDataGenerator(
    rescale=1./255
)

image_training=preprocesamiento_entre.flow_from_directory(
    data_training,
    target_size=(height, width),
    class_mode='categorical',
    color_mode='grayscale',
    batch_size=1

)

image_validation=preprocesamiento_vali.flow_from_directory(
    data_validation,
    target_size=(height, width),
    class_mode='categorical',
    color_mode='grayscale',
    batch_size=1
)
```

```
#Red neuronal convolucional (CNN)
cnn=Sequential()
cnn.add(Convolution2D(filtrosconv1,
                      tam_filtro1,
                      padding='same',
                      input_shape=(height, width, chanels),
                      activation='relu'))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Convolution2D(filtrosconv2, tam_filtro2, padding='same', activation='relu'))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Convolution2D(filtrosconv3, tam_filtro3, padding='same', activation='relu'))
cnn.add(MaxPooling2D(pool_size=tam_pool))

cnn.add(Flatten())
cnn.add(Dense(neurons, activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(classes, activation='softmax'))

cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
cnn.fit(image_training,
        steps_per_epoch=quantity,
        epochs=iterations,
        validation_data=image_validation,
        validation_steps=quantity)
cnn.save(folder+'model.h5')
cnn.save_weights(folder+'weights.h5')
```

Anexo 5: Código del archive Main.py

```
import tkinter as tk
from tkinter import Tk, Button, Label, Entry, PhotoImage, Radiobutton, messagebox
from PIL import Image, ImageTk, ImageFont, ImageDraw
import cv2
import pyttsx3
import mediapipe as mp
import os
import numpy as np
import time
from keras_preprocessing.image import img_to_array
from keras.models import load_model
import speech_recognition as sr
from threading import Thread
import json

# Read parameters
#################################################################
with open("parameters.json") as data:
    parameters=json.loads(data.read())

#Global variables
listening=False
seeing=False
showing=False
speaking=False
window=Tk()
selected=tk.IntVar()
cameraDisableI =
ImageTk.PhotoImage(Image.open("resources/images/camera_disable.png"))
microphoneDisableI =
ImageTk.PhotoImage(Image.open("resources/images/microphone_disable.png"))
```

```
# Mediapipe
mp_drawing=mp.solutions.drawing_utils
mp_hands=mp.solutions.hands

#Habilitamos el motor de voz de pyttsx3
engine=pyttsx3.init()
# voices = engine.getProperty('voices')
# engine.setProperty("voice", voices[2].id)
engine.setProperty("rate", 100)

#Cargamos el modelo y pesos para la predicción
folder='resources/files/'
cnn = load_model(folder+'model.h5')
cnn.load_weights(folder+'weights.h5')
dire_img = os.listdir(folder+'validation')

#Función para terminar la conversación
def finish(event):
    global listening
    global showing
    global seeing
    global speaking
    seeing=False
    listening=False
    showing=False
    speaking=False
    textE.delete(0, tk.END)
    textE['state']='disable'
    voiceB['state']='normal'
    signB['state']='normal'
```

```
    voiceL['image']=microphoneDisableI
    signL['image']=cameraDisableI
    finishB['state']='disable'

#Función para quitar acentos
def normalize(s):
    replacements = (
        ("Á", "A"),
        ("É", "E"),
        ("Í", "I"),
        ("Ó", "O"),
        ("Ú", "U"),
    )
    for a, b in replacements:
        s = s.replace(a, b)
    return s

#Función para escuchar al hablante
def listen(event):
    finish(event)
    global listening
    global showing
    listening=True
    showing=True
    textE['state']='normal'
    voiceB['state']='disable'
    finishB['state']='normal'
    ThreadGif(daemon=True).start()
    ThreadListen(daemon=True).start()
```

```
#Función para mostrar palabras en lengiaje de señas
def show(text):
    global showing
    if showing:
        m=list(text)
        size=np.size(m)
        cont=0
        while cont<size:
            l=str(m[cont])
            timer=True
            if l=="J" or l=="Ñ" or l=="Z":
                movingLetters(l)
                timer=False
            elif l=="L" and str(m[cont+1])=="L":
                movingLetters("LL")
                cont =cont+2
                timer=False
            elif l=="R" and str(m[cont+1])=="R":
                movingLetters("RR")
                cont =cont+2
                timer=False
            elif l=="Á" or l=="É" or l=="Í" or l=="Ó" or l=="Ú":

img=ImageTk.PhotoImage(Image.open("resources/images/alphabet/{}.png".format(nor
malize(l))))
            elif l==" ":

img=ImageTk.PhotoImage(Image.open("resources/images/alphabet/ESP.png"))
            elif l.isdigit():
                messagebox.showinfo("ALERTA", "Los carácteres de números no están
soportados")
                signL["image"]=cameraDisableI
```

```
            textE.delete(0, tk.END)
            break
        else:

img=ImageTk.PhotoImage(Image.open("resources/images/alphabet/{}.png".format(l)))

        if timer: #timer indica si hay que controlar el tiempo
            signL["image"]=img
            window.update()
            initial=time.time()
            final=initial
            while (final-initial)<1: # 1 means de delay time to show next letter
                final=time.time()
                if not showing:
                    break
            cont +=1
        if cont==size:
            signL["image"]=cameraDisableI
            textE.delete(0, tk.END)
            break

        if not showing:
            break

    if showing:
        if selected.get()==0:
            see(None)
        elif selected.get()==1:
            listen(None)

#Función para capturar las señas de las letras
```

```
def see(event):
    finish(event)
    global seeing
    global speaking
    seeing=True
    speaking=True
    cap = cv2.VideoCapture(0)
    signB['state']='disable'
    textE['state']='normal'
    finishB['state']='normal'
    textE.focus()
    salio=False
    memory=""
    inicio=0
    #
    inverse=True #delete after
    #
    with mp_hands.Hands(
        static_image_mode=False,
        max_num_hands=1,
        min_detection_confidence=0.5
    ) as hands:
        while seeing:
            ret, frame=cap.read()

            # Getting backgroun image
            background=cv2.imread('resources/images/background.png')

            #Process with mediapipe
            frame_rgb=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
        result=hands.process(frame_rgb)
        hands_image=result.multi_hand_landmarks
        if hands_image is not None:
            for hand_landmarks in hands_image:
                # Pintando los puntos y líneas
                mp_drawing.draw_landmarks(
                    background, hand_landmarks, mp_hands.HAND_CONNECTIONS,
                    # To change color
                    mp_drawing.DrawingSpec(color=(255,255,255), thickness=4,
circle_radius=3),
                    mp_drawing.DrawingSpec(color=(255,255,255), thickness=4)
                )

            x1, y1=220, 140
            x2, y2=420, 340
            cut=background[y1:y2, x1:x2]
            cut=cv2.cvtColor(cut, cv2.COLOR_BGR2GRAY)
            cut=cv2.resize(cut, (parameters['width'], parameters['height']))
            vector = cnn.predict(np.expand_dims(img_to_array(cut), axis=0))
            r=vector[0]
            val=np.amax(r)
            letter=dire_img[np.argmax(vector[0])]
            tiempo = round(time.time() - inicio, 0)
            if letter != memory:
                memory=letter
                inicio=time.time()
            elif tiempo>1: # 1 means time to wait to see next sign
                if letter=="ESP":
                    textE.insert('insert', " ")
                elif letter=="ELI":
                    textE.delete(len(textE.get())-1, tk.END)
```

```
                elif letter=='Nh':
                    textE.insert('insert', 'Ñ')
                else:
                    textE.insert('insert', letter)
                inicio=time.time()


            #
            frame = cv2.flip(frame, 1) #delete after
            inverse=False #delete after
            #
            cv2.rectangle(frame, (x1,y1), (x2,y2), (255, 255, 255), 3)
            if letter=="Nh":
                pil_im = Image.fromarray(frame)
                draw = ImageDraw.Draw(pil_im)
                draw.text((x1, y1-55), "Ñ",
font=ImageFont.truetype("resources/font/arial.ttf", 50))
                frame = cv2.cvtColor(np.array(pil_im), cv2.COLOR_RGB2BGR)
            else:
                cv2.putText(frame, letter, (x1, y1 - 5), 1, 4, (255,255,255), 2,
cv2.LINE_AA)


            salio=True
        else:
            if salio:
                tiempo = round(time.time() - inicio, 0)
                if tiempo>2: #2 means time to wait to speak
                    speak(textE.get())
                    memory=""
                    salio=False


        if inverse:
```

```
            frame = cv2.flip(frame, 1)
            inverse=False
        img=Image.fromarray(frame)
        img=ImageTk.PhotoImage(image=img)
        signL['image']=img
        window.update()

    cap.release()
    signL['image']=cameraDisableI

#Función para hablar lo que ha mostrado en señas la persona sorda
def speak(text):
    global speaking
    if speaking:
        engine.say(text)
        engine.runAndWait()
        textE.delete(0, tk.END)
        window.update()
    if selected.get()==0 and seeing:
        listen(None)

#Clase para mostrar la imagegen gif de escucha
class ThreadGif(Thread):
    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        super().__init__(group=group, target=target, name=name,
                         daemon=daemon)
    def run(self):
        global listening
        frames_of_gif = 49 # Número de frames que tiene el gif
```

```
        frames = [PhotoImage(file="resources/images/listening.gif", format='gif -index %i' %(i)) for i in range(frames_of_gif)]
        cont=0
        while listening:
            frame = frames[cont]
            voiceL['image']=frame
            window.update()
            initial=time.time()
            final=initial
            while (final-initial)<0.1:
                final=time.time()
            if not listening:
                voiceL["image"]=microphoneDisableI
                cont=0
                break
            cont +=1
            if cont == frames_of_gif:
                cont= 0



#Clase para escuchar al hablante
class ThreadListen(Thread):
    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        super().__init__(group=group, target=target, name=name,
                         daemon=daemon)
    def run(self):
        global listening
        while listening:
            with sr.Microphone() as source:
                try:
```

```
                r=sr.Recognizer()
                audio=r.listen(source)
                text=r.recognize_google(audio, language="es-ES")
                text=text.upper()
                if listening:
                    listening=False
                    textE['state']='normal'
                    textE.focus()
                    textE.insert('insert', text)
                    window.update()
                    show(text)
            except:
                print('repeat')

def movingLetters(letter):
    global showing
    frames_of_gif=0 #Número de frames que tiene el gif
    file=""
    if letter=="J":
        frames_of_gif = 31
        file="resources/images/alphabet/J.gif"
    elif letter=="LL":
        frames_of_gif=48
        file="resources/images/alphabet/LL.gif"
    elif letter=="Ñ":
        frames_of_gif=45
        file="resources/images/alphabet/Ñ.gif"
    elif letter=="RR":
        frames_of_gif=50
        file="resources/images/alphabet/RR.gif"
```

```
    elif letter=="Z":
        frames_of_gif=44
        file="resources/images/alphabet/Z.gif"

    frames = [PhotoImage(file=file, format='gif -index %i' %(i)) for i in
range(frames_of_gif)]
    cont=0
    while showing and cont<frames_of_gif:
        frame = frames[cont]
        signL['image']=frame
        window.update()
        initial=time.time()
        final=initial
        while (final-initial)<0.1:
            final=time.time()
        cont +=1

#Función que se ejecuta al terminar o cerrar la ventana main
def windows_on_closing():
    global listening
    listening=False
    window.destroy()

#Diseñamos la GUI
window.title("Traductor del alfabeto de señas")
window.state("zoomed")
window.anchor("c")
window.iconbitmap('resources/images/icon.ico')
window.protocol("WM_DELETE_WINDOW", windows_on_closing)
window.bind('<Control-v>', listen)
window.bind('<Control-V>', listen)
```

```
window.bind('<Control-s>', see)
window.bind('<Control-S>', see)
window.bind('<Control-e>', finish)
window.bind('<Control-E>', finish)
window.bind('<Escape>', finish)

titleL=Label(window)
titleL['font']=("Arial",24, "bold")
titleL['text']="TRADUCTOR A VOZ DEL ALFABETO DE SEÑAS"
titleL.grid(column=0, row=0, columnspan=2)

automaticallyRB=Radiobutton(window)
automaticallyRB.grid(column=0, row=1, sticky='nw')
automaticallyRB['text']='Intercambiar diálogo automáticamente'
automaticallyRB['font']=(12)
automaticallyRB['value']=0
automaticallyRB['variable']=selected

manuallyRB=Radiobutton(window)
manuallyRB.grid(column=0, row=2, sticky='sw')
manuallyRB['text']='Intercambiar diálogo manualmente'
manuallyRB['font']=(12)
manuallyRB['value']=1
manuallyRB['variable']=selected

selected.set(0) #Por defecto se selecciona el primer radiobutton

voiceB=Button(window)
voiceB.grid(column=0, row=3)
voiceB['text']='INICIAR POR VOZ'
```

```
voiceB['bg']='#004c99'
voiceB['fg']='#FFFFFF'
voiceB['command']=lambda:listen(None)

signB=Button(window)
signB.grid(column=1, row=3)
signB['text']="INICIAR POR SEÑAS"
signB['bg']='#00994c'
signB['fg']='white'
signB['command']=lambda: see(None)

textE=Entry(window)
textE.grid(column=0, row=4, columnspan=2, pady=10)
textE['state']='disable'
textE['width']=40
textE['bg']='#000000'
textE['font']=("Arial",20, "bold")
textE['fg']='#FFFFFF'
textE['insertbackground']='#FFFFFF'

signL=Label(window)
signL.grid(column=1, row=5, padx=10)
signL['image']=cameraDisableI

voiceL = Label(window)
voiceL.grid(column=0, row=5, padx=10)
voiceL['image']=microphoneDisableI

finishB=Button(window)
finishB.grid(column=0, row=6, columnspan=2)
```

```
finishB['text']="FINALIZAR CONVERSACIÓN"
finishB['state']='disable'
finishB['command']=lambda:finish(None)

#Función principal que arranca el main
if __name__=='__main__':
    window.mainloop()
```

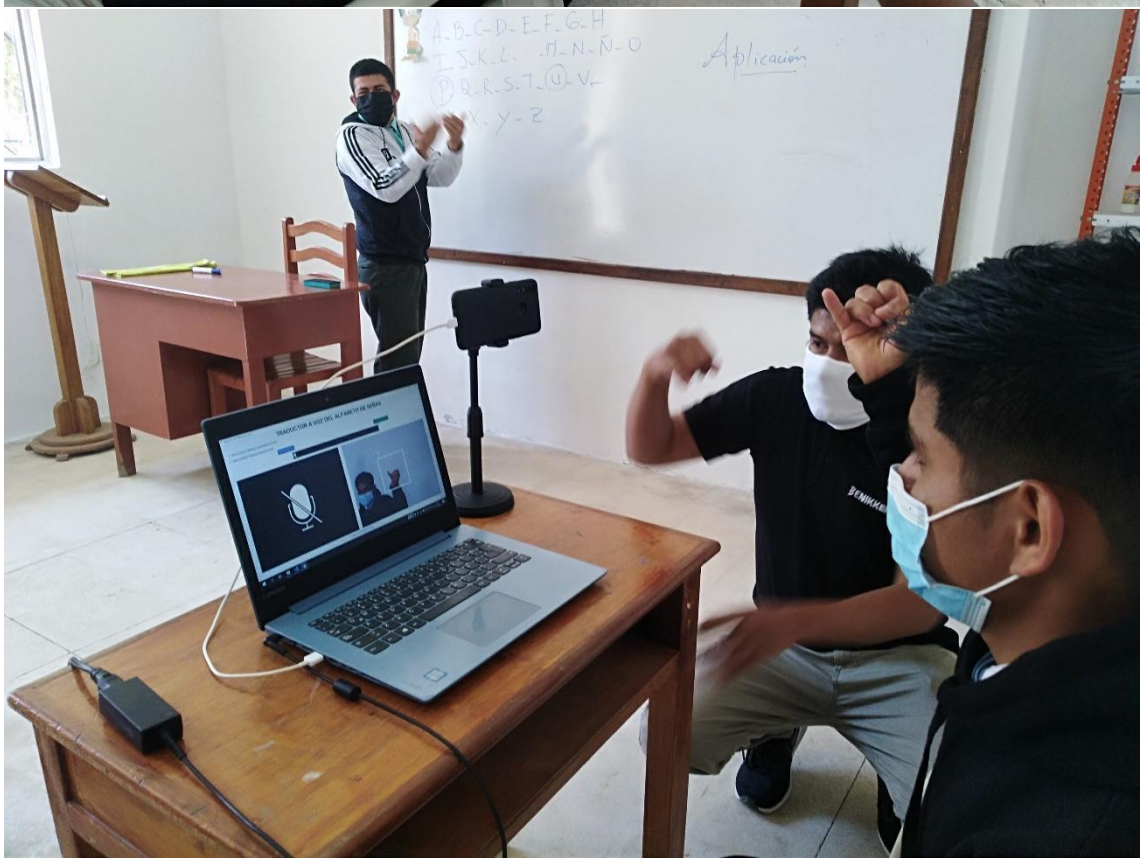
Anexo 6: Interfaz final de usuario del sistema desarrollado en este proyecto

Anexo 7: Imágenes de alumnos del colegio Harvest, el día en que se hicieron las pruebas

Mi compañero y yo. En la parte del medio se encuentra la directora de la IE bautista para sordos Harvest

Salón de clase, junto a los 7 participantes de las pruebas del sistema.

Las fotos a continuación fueran las se hicieron cuando hacíamos las pruebas:


A-B-C-D-E-F-G-H
I-J-K-L- M-N-Ñ-O
P-Q-R-S-T-U-V-
W-X-Y-Z
Aplicación


TRADUCTOR A VOZ DEL ALFABETO DE SEÑAS


TRADUCTOR A VOZ DEL ALFABETO DE SEÑAS


TRADUCTOR A VOZ DEL ALFABETO DE SEÑAS

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO FACULTAD DE INGENIERÍA CIVIL DE SISTEMAS Y DE ARQUITECTURA
DECANATO

ACTA DE SUSTENTACIÓN VIRTUAL N°001-2023-FICSA-D

Siendo las **9:00 am** horas del día 10 de enero del 2023, se reunieron vía plataforma virtual, https://meet.google.com/rzh-bwdv-jye, los miembros de jurado de la tesis titulada: **"SISTEMA COMPUTACIONAL BASADO EN INTELIGENCIA ARTIFICIAL QUE MEJORA LA COMUNICACIÓN CON UNA PERSONA SORDOMUDA MEDIANTE EL ALFABETO DE SEÑAS" con código de proyecto IS_V_2021_022,** designado por Resolución Decanal Virtual N° **270-2021-UNPRG-FICSA** con la finalidad de Evaluar y Calificar la sustentación de la tesis antes mencionada, conformado por los siguientes docentes:

DR. ING. PEDRO MIGUEL JACINTO MEJÍA	**PRESIDENTE**
MG. ING. ROBERTO CARLOS ARTEAGA LORA	**SECRETARIO**
DR. ING. JUAN ELÍAS VILLEGAS CUBAS	**VOCAL**

Asesorado por MG. ING. OSCAR EFRAIN CAPUÑAY UCEDA

El acto de sustentación fue autorizado por **OFICIO VIRTUAL N° 011-2023-UIFICSA**, la tesis fue presentada y sustentada por los Bachilleres: **ISRAEL MONTENEGRO CHORE y MANUEL GRABIEL PRAVIA PURIHUAMAN,** tuvo una duración de 1 hora 40 minutos Después de la sustentación, y absueltas las preguntas y observaciones de los miembros del jurado; se procedió a la calificación respectiva:

ISRAEL MONTENEGRO CHORE	**15**	**QUINCE**	**REGULAR**
MANUEL GRABIELPRAVIA PURIHUAMAN	**15**	**QUINCE**	**REGULAR**

Por lo que quedan **APTOS** para obtener el Título Profesional de **INGENIERO DE SISTEMAS** de acuerdo con la Ley Universitaria 30220 y la normatividad vigente de la Facultad de Ingeniería Civil De Sistemas y de Arquitectura de la Universidad Nacional Pedro Ruiz Gallo.

Siendo las 10: 40 am; se dio por concluido el presente acto académico, dándose conformidad al presente acto, con la firma de los miembros del jurado.

DR. ING. PEDRO MIGUEL JACINTO MEJÍA
PRESIDENTE

MG. ING. ROBERTO CARLOS ARTEAGA LORA
SECRETARIO

DR. ING. JUAN ELÍAS VILLEGAS CUBAS
VOCAL

MG. ING. OSCAR EFRAIN CAPUÑAY UCEDA
ASESOR

DR. ING. SERGIO BRAVO IDROGO
DECANO

REGISTRO VIRTUAL SUSTENTACIONES INGENIERÍA DE SISTEMAS
IS_L001_F001

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE INGENIERIA CIVIL DE SISTEMAS Y ARQUITECTURA
UNIDAD DE INVESTIGACION

"Año de la universalización de la salud".

CONSTANCIA DE APROBACION DE ORIGINALIDAD DE TESIS

Yo Oscar Efraín Capuñay Uceda**, Asesor de Tesis,** de los Integrantes:

Bach. Montenegro Chore, Israel

Bach. Pravia Purihuaman, Manuel Grabiel

- **DE LA TESIS TITULADA:** "Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas"

Luego de la revisión exhaustiva del documento constato que la misma tiene un índice de similitud de 16% verificable en el reporte de similitud del programa TURNITIN.

El suscrito analizó dicho reporte y concluyo que cada una de las coincidencias detectadas NO CONSTITUYEN PLAGIO. A mi leal saber y entender la tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad Nacional Pedro Ruiz Gallo.

Se expide la presente según lo dispuesto en la Resolución N° 659-2020-R, de fecha 8 de setiembre de 2020 formativa para la obtención de Grados y Títulos de la UNPRG:

Lambayeque, 08 de mayo del 2023

ATENTAMENTE,

MG. ING. OSCAR EFRAÍN CAPUÑAY UCEDA
DNI. : 16717150

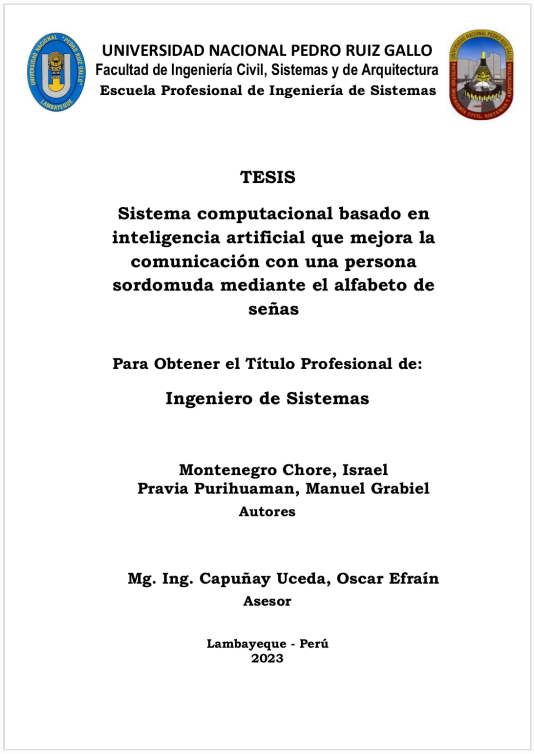
Se Adjunta lo Siguiente:

Recibo digital

Este recibo confirma quesu trabajo ha sido recibido por Turnitin. A continuación podrá ver la información del recibo con respecto a su entrega.

La primera página de tus entregas se muestra abajo.

Autor de la entrega:	Israel Montenegro Chore
Título del ejercicio:	Tesis pregrado
Título de la entrega:	Tesis - versión final
Nombre del archivo:	Tesis_firmada-final.pdf
Tamaño del archivo:	4.69M
Total páginas:	150
Total de palabras:	23,364
Total de caracteres:	129,371
Fecha de entrega:	23-may.-2023 03:51p. m. (UTC-0500)
Identificador de la entre...	2100335560



UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
Facultad de Ingeniería Civil, Sistemas y de Arquitectura
Escuela Profesional de Ingeniería de Sistemas

TESIS

Sistema computacional basado en inteligencia artificial que mejora la comunicación con una persona sordomuda mediante el alfabeto de señas

Para Obtener el Título Profesional de:

Ingeniero de Sistemas

Montenegro Chore, Israel
Pravia Purihuaman, Manuel Grabiel
Autores

Mg. Ing. Capuñay Uceda, Oscar Efraín
Asesor

Lambayeque - Perú
2023

MG. ING. OSCAR EFRAÍN CAPUÑAY UCEDA

Tesis - versión final

INFORME DE ORIGINALIDAD

17%	16%	4%	7%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	4%
2	blog.centrodeelearning.com Fuente de Internet	1%
3	1library.co Fuente de Internet	1%
4	Submitted to Universidad de Piura Trabajo del estudiante	1%
5	github.com Fuente de Internet	1%
6	docplayer.es Fuente de Internet	1%
7	omes-va.com Fuente de Internet	1%
8	repositorio.unc.edu.pe Fuente de Internet	1%
9	orinoquia.unillanos.edu.co Fuente de Internet	<1%

MG. ING. OSCAR EFRAÍN CAPUÑAY UCEDA