UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

PROGRAMA DE ESTUDIO DE INGENIERÍA ELECTRÓNICA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICO

Sistema traductor de comunicación bidireccional para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad (2023).

Línea de investigación: Sistemas Inteligentes

Autores:

Millán Bermejo, Jordan Luiggi
Panta Rivera, Kristhian Martin

Jurado evaluador:

Presidente : Azabache Fernández, Filiberto Melchor

Secretario : Ramos Rojas, Ovidio Hildebrando

Vocal : de la Cruz Rodríguez, Oscar Miguel

Asesor:

Alva Alarcón, Jorge Luis
Código Orcid: https://orcid.org/0000-0003-1288-933X

Trujillo–Perú
2024

Fecha de Sustentación: 2024/07/19

Jurado de sustentación Oral

Ms. Ing. AZABACHE FERNANDÉZ FILIBERTO MELCHOR

N° CIP 97916

Presidente

Ms. Ing. RAMOS ROJAS OVIDIO HILDEBRANDO

N° CIP 92622

Secretario

Ms. Ing. DE LA CRUZ RODRIGUEZ OSCAR MIGUEL

N° CIP 97916

Vocal

Entregado el:

PANTA RIVERA KRISTHIAN MARTIN

DNI 73007751

Aprobado por:

MILLAN BERMEJO JORDAN LUIGGI

DNI 70937235

Ms. Ing. ALVA ALARCÓN JORGE LUIS

Asesor de Tesis

UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

PROGRAMA DE ESTUDIO DE INGENIERÍA ELECTRÓNICA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICO

Sistema traductor de comunicación bidireccional para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad (2023).

Línea de investigación: Sistemas Inteligentes

Autores:

Millán Bermejo, Jordan Luiggi
Panta Rivera, Kristhian Martin

Jurado evaluador:

Presidente : Azabache Fernández, Filiberto Melchor

Secretario : Ramos Rojas, Ovidio Hildebrando

Vocal : de la Cruz Rodríguez, Oscar Miguel

Asesor:

Alva Alarcón, Jorge Luis
Código Orcid: https://orcid.org/0000-0003-1288-933X

Trujillo–Perú
2024

Fecha de Sustentación: 2024/07/19

Informe_de_Tesis_Panta_Kristhian_Millan_Bermejo_Jordan.docx

por Kristhian Martin Panta Rivera & Jordan Luiggi Millán Bermejo

Fecha de entrega: 16-jul-2024 09:12a.m. (UTC-0500)
Identificador de la entrega: 2417762472
Nombre del archivo: Informe_de_Tesis_Panta_Kristhian_Millan_Bermejo_Jordan.docx (42.35M)
Total de palabras: 36829
Total de caracteres: 210997

Informe_de_Tesis_Panta_Kristhian_Millan_Bermejo_Jordan....

INFORME DE ORIGINALIDAD

2%	2%	0%	2%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	2%

Excluir citas	Activo	Excluir coincidencias	< 2%
Excluir bibliografía	Activo		

DECLARACION DE ORIGINALIDAD

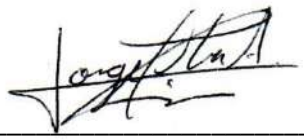
Yo, Alva Alarcón Jorge Luis, docente del Programa de Estudio de Pregrado de la Universidad Privada Antenor Orrego, asesor de la tesis titulada “Sistema traductor de comunicación para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad (2023)”, de los autores Millán Bermejo Jordan Luiggi y Panta Rivera Kristhian Martin

• El mencionado documento tiene un índice de puntuación de similitud del 2%. Así lo consigna el reporte de similitud emitido por el software Turnitin el día 16 de julio del 2024

• He revisado con detalle dicho reporte de la tesis “Sistema traductor de comunicación para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad (2023)” y no se advierte indicios de plagio.

• Las citas a otros autores y sus respectivas referencias cumplen con las normas establecidas por la Universidad.

Ciudad y fecha: Trujillo 16 de julio del 2024



ALVA ALARCÓN JORGE LUIS

DNI: 40294924

ORCID: https://orcid.org/0000-0003-1288-933X

MILLÁN BERMEJO JORDAN LUIGGI

DNI: 70937235

PANTA RIVERA KRISTHIAN MARTIN

DNI: 73007751

Dedicatorias

Dedico este proyecto a mi madre Eresbita Bermejo Carrera, cuyo amor incondicional y constante apoyo han sido pilares fundamentales en mi vida. Sus sacrificios y dedicación han sido una fuente de inspiración continua y sin su presencia, este logro académico no habría sido posible.

A mi padre, Lucidoro Millán López, cuya guía y ejemplo han inculcado en mí los valores del esfuerzo y la perseverancia. Sus palabras de aliento y su apoyo constante han sido esenciales para alcanzar mis metas.

Y a Dios, por concederme la vida, la sabiduría y la fortaleza necesarias para superar cada desafío.

Jordan Luiggi Millán Bermejo.

A dios

Por haberme concedido la fortaleza y la sabiduría necesaria para completar este logro académico*.* Con gratitud infinita, entrego este logro que es fruto de su gracia.

A mi madre, Isabel Rivera

Cuyo amor incondicional, apoyo constante y sacrificio desinteresado han sido mi mayor fortaleza. Este logro también es gracias a ella.

A mi padre, Fernando Panta

Quien con su sabiduría, paciencia y ejemplo de perseverancia ha sido mi guía en cada paso de este viaje académico. Este logro también es gracias a él.

A mi hermano, Paul Panta

Por su apoyo incondicional en cada paso de este camino. Sus palabras de aliento y su inquebrantable fe en mí han sido fundamentales para alcanzar este logro. Este éxito también es gracias a él.

Kristhian Martin Panta Rivera.

Agradecimientos

A Dios, por darnos la fortaleza, la sabiduría y la perseverancia necesarias para culminar este proyecto. Sin Su guía y bendiciones, no habríamos podido alcanzar este logro tan significativo en nuestras vidas.

A nuestros progenitores, les expresamos nuestro más profundo agradecimiento. Su amor incondicional, apoyo constante y sacrificios han sido fundamentales para nuestra formación y éxito académico. Nos han brindado las herramientas y el ánimo necesarios para seguir adelante, incluso en los momentos más difíciles.

A la Universidad Privada Antenor Orrego, y en particular a la Facultad de Ingeniería y la Escuela Profesional de Ingeniería Electrónica, por proporcionarnos una educación de calidad y un ambiente propicio para el aprendizaje y el desarrollo personal. Su compromiso con la excelencia académica y el desarrollo integral de sus estudiantes ha sido una fuente constante de inspiración para nosotros.

A los profesores de la Escuela Profesional de Ingeniería Electrónica. Su dedicación, conocimiento y paciencia han sido invaluables para nuestra formación. Cada clase, cada consejo y cada corrección han contribuido de manera significativa a nuestra preparación como futuros ingenieros electrónicos.

A nuestro asesor de Tesis, Ing. Jorge Luis Alva Alarcón, su orientación, apoyo y experticia han sido cruciales para la realización de este trabajo. Su compromiso y paciencia han sido un faro de guía durante todo el proceso, permitiéndonos alcanzar los objetivos planteados y superar los desafíos encontrados en el camino.

Resumen

La presente investigación estuvo enfocada en desarrollar un sistema traductor de comunicación bidireccional para mejorar la comunicación entre una persona sorda de la Asociación De Sordos De La Libertad y una persona oyente de La Libertad, a partir de la reducción de los tiempos de reconocimiento de las palabras. El sistema se desarrolló para el uso mediante un escritorio en una aplicación web y se programó en Pycharm mediante el lenguaje de programación Python. Para su programación y desarrollo, se abordaron dos áreas principales: la traducción de señas a texto en la comunicación de persona sorda a oyente, y la traducción de voz a vídeo en la comunicación de persona oyente a sorda. En el primer caso, se utilizó visión artificial mediante bibliotecas como Cvzone, Keras, Numpy, Mediapipe y Cv2. Estas herramientas permitieron la detección, seguimiento y reconocimiento de gestos estáticos del alfabeto dactilológico del Lenguaje de Señas Peruanas realizado por una persona sorda frente a una cámara web. En el segundo caso, se emplearon las bibliotecas: speech_recognition para el reconocimiento de voz y moviepy para la manipulación de videos. Esto facilitó la traducción de la entrada de voz a texto y la generación de un video que concatenaba videos correspondientes a cada letra de la palabra reconocida. Con respecto al desarrollo de la aplicación web se utilizó Flask, para poder manejar desde el lado del servidor, las solicitudes HTTP del cliente (persona sorda u oyente desde el navegador web). El tipo de investigación que se siguió fue explicativa y pre-experimental. Se utilizó como método de análisis de datos para pruebas paramétricas la prueba t de Student (para muestras emparejadas). Los resultados de esta investigación demostraron que con el uso del sistema se reduce la media de los tiempos de reconocimiento de las palabras en un 50.14% para las personas oyentes de La Libertad y en un 80.61% para las personas sordas de la Asociación De Sordos. Asimismo, mediante la prueba t de Student se concluyó que estas reducciones en los tiempos de reconocimiento de las palabras son estadísticamente significativas.

Palabras Clave: Sistema traductor de comunicación bidireccional, visión artificial, speech_recognition, moviepy, Flask, Lenguaje de Señas Peruana, personas sordas, personas oyentes.

Abstract

The present research focused on developing a bidirectional communication translator system to improve communication between a deaf person from the Association of Deaf People of La Libertad and a hearing person from La Libertad, by reducing word recognition times. The system was developed for use through a desktop web application and was programmed in PyCharm using the Python programming language. For its programming and development, two main areas were addressed: the translation of signs to text in communication from a deaf person to a hearing person, and the translation of voice to video in communication from a hearing person to a deaf person. In the first case, computer vision was used through libraries such as Cvzone, Keras, Numpy, Mediapipe, and Cv2. These tools allowed the detection, tracking, and recognition of static gestures of the dactylological alphabet of the Peruvian Sign Language performed by a deaf person in front of a webcam. In the second case, the libraries speech_recognition for voice recognition and moviepy for video manipulation were used. This facilitated the translation of voice input to text and the generation of a video that concatenated videos corresponding to each letter of the recognized word. For the development of the web application, Flask was used to handle client HTTP requests (deaf or hearing person from the web browser) from the server side. The type of research followed was explanatory and pre-experimental. Student's t-test (for paired samples) was used as a method of data analysis for parametric tests. The results of this research showed that with the use of the system, the average word recognition times were reduced by 50.14% for hearing people from La Libertad and by 80.61% for deaf people from the Association of Deaf People. Likewise, through Student's t-test, it was concluded that these reductions in word recognition times are statistically significant.

Keywords: Bidirectional communication translator system, computer vision, speech_recognition, moviepy, Flask, Peruvian Sign Language, deaf people, hearing people.

Presentación

Señores miembros del Jurado:

De conformidad con lo estipulado en el Reglamento de Grados y Títulos de la Universidad Privada Antenor Orrego, ponemos a su disposición el informe de tesis titulado “Sistema traductor de comunicación bidireccional para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad (2023)” para que sea revisado y evaluado y de ser aprobado pueda ser defendido oralmente para optar el título profesional de Ingeniero Electrónico.

De antemano, nos excusamos de los errores involuntarios en que se hubiera incurrido en el desarrollo y redacción del misma, esperando del honorable jurado un justo dictamen.

Millán Bermejo Jordan Luiggi

Panta Rivera Kristhian Martin

Tabla de Contenido

Índice de Tablas

Índice de Figuras

I. INTRODUCCIÓN

1.1. Problema de investigación

a) Descripción de la realidad problemática

Según la Organización Mundial de la Salud (2023), una persona que es considerada sorda tiene una disminución significativa de la capacidad auditiva, lo que implica una capacidad limitada o nula para escuchar. En consecuencia, estas personas utilizan el lenguaje de señas o de signos para comunicarse.

La Federación de Personas Sordas de La Comunidad de Madrid (2022) define al lenguaje de señas como la lengua natural de las personas sordas. Además, menciona que es de naturaleza visual, gestual y espacial y tiene una gramática propia al igual que cualquier otra lengua.

Hoy en día, a través de la federación Mundial de Sordos sabemos que en el mundo existen 72 millones de personas con limitación permanente para oír. Y según el último censo realizado en Perú (INEI-2017), existen más de 3 millones de personas que presentan al menos algún tipo de discapacidad, en donde el 7.6% representa a las personas con limitación permanente para oír. (Arevalo Miró Quesada, 2022)

Del mismo modo Quijada (2021) en su artículo “La importancia del emprendimiento en la comunidad sorda peruana: Un reto para la inclusión” señala que de acuerdo al Consejo Nacional para la Integración de la Persona con Discapacidad (2020), en el Perú existe un 4.9% de personas con discapacidad auditiva que acceden a la educación básica regular en instituciones tanto públicas como privadas. Además, señala que un 17.5% de las personas con discapacidad sensorial se benefician del Programa Nacional Jóvenes Productivos del Ministerio de Trabajo y Promoción del Empleo. Sin embargo, también menciona que existe una insuficiencia de empleos dignos y equitativos para las personas sordas, a pesar de la existencia de la Ley General de la Persona con Discapacidad (N°29973), la

cual establece la obligación para empresas públicas y privadas de contratar trabajadores con discapacidad.

Lo que mencionó Quijada (2021) en lo citado de su artículo acerca de la existencia de una insuficiencia de empleos dignos y equitativos para las personas sordas, fue respaldado mediante la encuesta (*ver Anexo 15*) que se realizó hacia las personas sordas de la Asociación De Sordos De La Libertad (*ver Anexo 7*), en la que se observó que Renan Miguel Begazo Torres (persona sorda) enfrenta dificultades para encontrar empleos equitativos.

Aunque no se tenga una cantidad precisa de personas que actualmente se comuniquen por medio de Lenguaje de Señas Peruana, la Defensoría del Pueblo recibe de manera habitual quejas y solicitudes que destacan las dificultades que enfrentan estas personas en los servicios públicos (como salud, educación, entre otros) debido a la insuficiencia de intérpretes capacitados en LSP. (Defensoría del Pueblo, 2020)

Si bien es cierto, en el Perú existen universidades en donde promueven el conocimiento del lenguaje de señas, estas no tienen un gran impacto en los estudiantes debido a que es un taller electivo en el que no se le genera un mayor interés. (Otiniano, 2022)

Asimismo, es importante señalar que, en el Perú, el número de personas con discapacidad auditiva que terminan por abandonar los estudios educativos al sentirse excluidos, es bastante alta. Esto es debido a la falta de intérpretes de lenguaje de señas en las escuelas y de un material pedagógico adaptado para ellos, lo cual trae como consecuencia que las personas sordas tengan muchas dificultades para comprender, leer y escribir. (Saavedra, 2022)

Lo citado por Saavedra (2022) en el Diario La República, fue corroborado mediante la encuesta (*ver Anexo 15*) que se realizó hacia las personas sordas de la Asociación De Sordos De La Libertad, en la cual se

observó que dichas personas tienen muchas dificultades para leer y escribir, asimismo con respecto a sus experiencias en el ámbito educativo, Renan Miguel Begazo Torres mencionaba en la entrevista que en la mayoría de las ocasiones tenía dificultades para comprender las clases de sus profesores (personas oyentes) y que por ello se atrasaba siempre, por lo que era común en él “pedir prestado el cuaderno de su compañero (persona oyente)”.

En la actualidad, según Wang et al. (2022), existen muchos artículos e investigaciones que se han enfocado en cómo ayudar a una persona oyente en comprender lo que dice una persona con discapacidad auditiva mediante sistemas inteligentes que convierten el lenguaje de señas en texto o audio. Sin embargo, también señala que son muy pocas las investigaciones que se han enfocado en cómo ayudar a las personas con discapacidad auditiva a comprender lo que dice las personas oyentes.

Finalmente es importante mencionar que a través de la encuesta (*ver Anexo 15*) que se realizó hacia las personas sordas de la Asociación De Sordos De La Libertad, se observó que dichas personas no tienen conocimiento de alguna tecnología, sistema o aplicación que les ayude a mejorar la comunicación con una persona oyente.

b) Descripción del problema

Con la problemática mencionada y respaldada mediante la encuesta realizada hacia las personas sordas de la Asociación De Sordos De La Libertad (*ver Anexo 15*). Se pudo determinar que la principal causa de los desafíos que enfrentan las personas sordas en sectores como el educativo, social y laboral radica en la escasez de intérpretes debidamente capacitados en Lengua de Señas Peruanas. Lo cual trae como consecuencia una barrera comunicativa entre una persona sorda que utiliza para comunicarse el Lenguaje de Señas Peruanas y una persona oyente que desconoce dicha lengua. Además, mediante el artículo de Wang et al (2022), se entendió que muchas investigaciones no le están dando mucha importancia a cómo ayudar a las personas sordas a entender a las personas oyentes. Esto

sumado a la falta de conocimiento que tienen las personas sordas de la Asociación De Sordos De La Libertad en tecnologías que les ayude a mejorar la comunicación con una persona oyente.

c) Enunciado del problema

¿Cuál es el efecto del uso de un sistema traductor de comunicación bidireccional en los tiempos de reconocimiento de las palabras durante la comunicación entre una persona sorda de la Asociación De Sordos De La Libertad y una persona oyente de La Libertad?

1.2. Objetivos

a) Objetivo General

Determinar el efecto del uso de un sistema traductor de comunicación bidireccional en los tiempos de reconocimiento de las palabras durante la comunicación entre una persona sorda de la Asociación De Sordos De La Libertad y una persona oyente de La Libertad.

b) Objetivos Específicos

- Evaluar el efecto del uso de un sistema traductor de comunicación bidireccional en los tiempos que tarda cada persona oyente de La Libertad en reconocer cada palabra expresada en el Lenguaje de Señas Peruana por una persona sorda de la Asociación De Sordos De La Libertad.

- Evaluar el efecto del uso de un sistema traductor de comunicación bidireccional en los tiempos que tarda cada persona sorda de la Asociación De Sordos De La Libertad en reconocer cada palabra pronunciada por una persona oyente de La Libertad

1.3. Justificación del estudio:

a) Práctica:

La presente investigación se justifica por su implicancia práctica, ya que se busca desarrollar un sistema traductor de comunicación bidireccional, con el objetivo de mejorar la comunicación entre personas sordas de la Asociación De Sordos De La Libertad y personas oyentes, permitiendo así una comunicación en donde ambos puedan comprenderse mutuamente. Lo que contribuirá la inclusión de las personas sordas en el plano educativo, social y laboral.

II. MARCO DE REFERENCIA

2.1. Antecedentes del estudio

a) Antecedentes Internacionales

Nurfirdausi et al. (2021), en su investigación “Implementación del detector de disparo único (SSD) MobileNet V2 en el reconocimiento de gestos con las manos de pacientes discapacitados como sistema de notificación”, desarrollaron un sistema sanitario inteligente el cual tiene como objetivo mejorar la calidad de vida de pacientes discapacitados, de manera que puedan expresar sus necesidades de forma no verbal a personas de su confianza (familiares o médicos). Para desarrollar el reconocimiento de gestos utilizaron una cámara web como sistema de notificación. Con lo que respecta al conjunto de datos, tomaron como muestra de imágenes a 12 sujetos de distintos géneros y edades. Ellos se basaron en las 5 necesidades básicas del ser humano: necesidad de comer, beber, ir al baño, ayuda y medicamentos. Las imágenes que recopilaron lo entrenaron utilizando el algoritmo Single Shot Detector (SSD) en la arquitectura MobileNet V2. Entre las diversas técnicas de aprendizaje profundo, eligieron SSD MobileNet V2, porque tiene buena capacidad de detección de objetos y muy poca computación. El estudio que realizaron dio como resultado una tasa de detección del 85%, lo que significa de 85 de 100 imágenes fueron detectadas correctamente. Adicionalmente, implementaron una notificación en Telegram por cada gesto detectado, para avisar a las personas de

confianza de dicho paciente discapacitado. El principal aporte al trabajo de investigación se haya en conocer las ventajas que tiene una técnica de aprendizaje profundo como MobileNet V2, debido a que ofrece buenos resultados y con muy poca computación, lo cual se ve reflejado en dicha investigación. Esto nos da un punto de partida para poder entender cómo utilizar MobileNet V2 en el entrenamiento y generación de nuestro modelo de clasificación de señas.

Deep et al. (2022), en su investigación “Realtime Sign Language Detection and Recognition”, desarrollaron un sistema de reconocimiento de lenguaje de señas indio (ISL) con la finalidad de ayudar a las personas sordomudas a conectarse con el mundo. Para reconocer las señas, primero identificaron y rastrearon las regiones de interés (ROI) mediante la función de segmentación de piel de OpenCV “cv2.inRange()”. Luego usaron Media Pipe para capturar lo puntos de referencia de las manos y poder almacenarlos en una matriz NumPy. Finalmente entrenaron al modelo usando y apoyándose de TensorFlow, Keras y una red neuronal recurrente como LSTM (Red de memoria a corto plazo largo). El principal aporte al trabajo de investigación fue la recolección de la data, desde las regiones de interés (ROI) hasta la captura de los puntos de los puntos de referencia de las manos. Esto nos da un punto de partida, ya que, al capturar la mano y sus puntos de referencia, podemos tener un mejor control de la data que se utilizará para entrenar a nuestro modelo.

Wang et al. (2022), en su investigación “A Web-Based Audio-to-Sign Language Converter for Chinese National Sign Language”, desarrollaron un método para convertir audio en video en lenguaje de señas y reproducirlo en una página web, esto con el objetivo de ayudar a las personas con discapacidad auditiva a entender lo que dice la gente oyente. Para ello construyeron una base de datos basada en el lenguaje de señas chino y recopilaron audio a través de micrófonos de computadora. El método que utilizaron para la segmentación de las palabras chinas fue el de HanLP, esto lo hicieron con la finalidad de obtener frases para procesarlas y cotejarlas con la base de datos. Una vez que obtienen la salida de la base de datos,

los videos se ordenan y se empalman para hacer coincidir con la frase u oración. Eventualmente, el video se estaría reproduciendo en la página web. Los resultados experimentales de dicha investigación señalan que la precisión de este método es del 90%. El principal aporte al trabajo de investigación es conocer el código y la librería “moviepy” que desarrollaron y utilizaron para ordenar y empalmar los videos, así como el código para reproducir dicho video en una página web. Esto nos da un punto de partida para la creación de una función el cual pueda adaptar y combinar varios videos en lenguaje de señas con la finalidad de hacer coincidir una frase u oración en específico, así como también poder retornar dicho video combinado a una página web para su visualización.

b) Antecedentes Nacionales

Montenegro y Villa (2019), en su tesis para obtener el título profesional de Ingeniero de Sistemas, titulado: “Sistema de reconocimiento de lenguaje de señas peruano para mejorar la comunicación entre las personas sordomudas de la Institución Educativa Bautista para sordos Harvest en Chiclayo”, desarrollaron un sistema de reconocimiento de Lenguaje de señas peruano (LSP) mediante el lenguaje de programación Python. Además, emplearon Visión Artificial para procesar, analizar y predecir los gestos mediante el uso de redes convolucionales. Antes de realizar el procesamiento de la imagen obtenida por la cámara, decidieron hacer uso de un guante color único (verde), eso con la finalidad de segmentar ambas manos del resto de la imagen mediante la función bitwise_and. Luego, realizaron un proceso de suavizado y dilatación de la imagen, y utilizaron estos datos para construir la red neuronal convolucional. En este proceso, aprovecharon herramientas como TensorFlow para la creación del modelo. Los resultados experimentales del estudio indican que el sistema logró interpretar un promedio de 8 señales en un tiempo promedio de 0.57 minutos. El principal aporte al trabajo de investigación se haya en la influencia de la precisión de un modelo de aprendizaje profundo cuando logran aislar mano del resto en una señal de video, obteniendo una precisión total del 91% en los resultados obtenidos.

Aranda y Vargas (2020) en su trabajo de investigación para optar por el grado de bachiller en Ingeniería de Software, titulado “Propuesta de solución para la gestión remota de terapias del habla de pacientes postoperatorios de labio leporino utilizando Speech Recognition y Gamificación”, plantearon el desarrollo de una app móvil la cual permita abordar las dificultades que tiene un paciente postoperatorio de labio leporino (niños) al querer acceder a realizar sus terapias de manera presencialmente en centros especializados de dicha afección, de manera que se pueda gestionar de manera remota. Para ello proponen tecnologías como Speech Recognition y Text-to-Speech para facilitar el aprendizaje. El primero para calificar la pronunciación de los pacientes y el otro para que dichos pacientes puedan aprender y escuchar la correcta pronunciación de diversas palabras u oraciones. El principal aporte al trabajo de investigación se hayan en el benchmarking de las tecnologías de Speech Recognition que ellos realizaron en su investigación. Esto nos da un punto de partida para poder analizar y seleccionar la tecnología más adecuada para el reconocimiento de voz en la comunicación de persona oyente a sordo.

c) Antecedentes Locales

Vilchez & Rommel (2015), en su investigación “Sistema Intérprete de Lenguaje Alternativo para mejorar la comunicación de las personas sordas de la Asociación De Sordos De La Libertad”, desarrollaron un sistema que convierte las palabras a lenguaje de señas, el cual procesaba un audio de una palabra y lo convertía a gesto. Asi mismo utilizaron herramientas como Matlab para el desarrollo de redes neuronales artificiales (Red Kohonen). Para el analisis de datos se apoyaron de pruebas estadisticas como Wilconxon y Mcnemar. Los resultados experimentales de dicha investigación señalan que mediante el Sistema Interprete de Lenguaje Alternativo, el tiempo promedio de la comunicación en las personas sordas se reduce en un 2.4%, asimismo existe un aumento del 1.6% de la cantidad de medios de comunicación utilizados entre la sociedad y las personas sordas, y con respecto a la inserción laboral de las personas sordas, también aumenta en

un 4.9%. El principal aporte al trabajo de investigación se haya en los resultados obtenidos, ya que nos servirá para poder comparar y contrastar nuestros resultados para la comunicación de persona oyente a sorda.

Marco Teórico

2.1.1. Lenguaje de señas

Es la lengua natural de las personas sordas, la cual tiene una expresión y configuración gesto-espacial y percepción visual, por lo que les permite a las personas sordas poder comunicarse con su entorno social. Se fundamenta en el uso de movimientos y expresiones a través de las manos, los ojos, la boca y el cuerpo. (Plataforma digital unica del Estado Peruano, 2022)

El lenguaje de señas o signos es aquella que tiene sus propias reglas gramaticales por ello se hace más difícil aprenderlo que otros lenguajes (Gálves, 2022)

Asimismo, el lenguaje de señas o signos no es un lenguaje universal y eso se puede saber a través de los datos brindados por la Federación Mundial de Sordos, en donde nos dice que existen más de 300 lenguas de signos en el mundo. Esto ocurre debido a la diversidad de cultura. (Ruiz, 2022)

Si bien existen cientos de diferentes lenguajes de señas, hay 3 tipos principales: el primero es el lenguaje de señas nacionales y regionales; es el lenguaje que las personas totalmente sordas han desarrollado. La estructura gramatical de estos lenguajes es diferente al del lenguaje hablado es por ello que este lenguaje es propio de ellos así que es difícil que una persona que oye pueda aprenderlo (algunos ejemplos: El lenguaje de Señas Americano, Canadá y México). El segundo es el lenguaje de señas basados en lenguaje hablados, este a diferencia del anterior es mucho más fácil de aprender para las personas que oyen ya que tienen una estructura gramatical muy parecida que la

lengua que se habla. Generalmente lo usan personas que antes escuchaban pero que se volvieron sordas. Algunas veces se deletrea la primera letra de una palabra como parte de la seña (algunos ejemplos: Lenguaje de Señas en inglés y el Lenguaje de Señas en español). Por último, está el deletreo con los dedos, en donde cada palabra se deletrea con señas del alfabeto. Generalmente las personas sordas usan este método para expresar palabras nuevas o difíciles. (Werner, 2021)

2.1.2. Imágenes Digitales

Una imagen digital es aquella imagen que se representa de manera bidimensional en una matriz numérica (binaria). Basándonos en el principio de que la resolución de una imagen digital es estática o dinámica, esta puede ser una imagen matricial (mapas de bits) o una imagen vectorial. (Neira, 2022)

2.1.3. Imágenes de mapas de bits

Las imágenes digitales se componen de una cuadrícula de pequeñas celdas (píxeles), organizados en una rejilla. Cada píxel tiene un valor único de color e iluminación, y al observar todo el conjunto de las pequeñas celdas, percibimos una imagen de tono continuo. Es por ello que, si nosotros tratamos de modificar el tamaño de una imagen de mapa de bits va a perder calidad, esto sucede porque al modificarla estamos transformando la distribución y coloración de los píxeles. (Montenegro & Villa, 2019)

2.1.4. Imágenes vectoriales

Una imagen vectorial es aquella imagen digital formada por elementos geométricos independientes. Estas a diferencia de las imágenes de mapas de bits, están conformadas por objetos geométricos, los cuales están definidos por atributos matemáticos que indican su color, posición, etc. Debido a que una imagen vectorial se

construye a partir de cálculos matemáticos que involucran la posición y atributos de cada punto o vértice que la compone, al ajustar sus dimensiones no se produce ninguna distorsión ni pérdida de calidad, ya que el software simplemente recalcula las conexiones entre puntos y produce una nueva imagen igualmente nítida que la original, a diferencia de lo que ocurría con una imagen de mapas de bits. (Gardey & Perez, 2022)

Sin embargo, no todo es ventaja, en las imágenes vectoriales ya que tienen limitaciones para la creación de imágenes real, debido a que es muy complicado reproducir una fotografía a partir de vectores. (Sánchez, 2022)

2.1.5. Inteligencia Artificial

Según John McCarthy en su artículo publicado en 2004, con el título de: “What Is Artificial Intelligence?, define a la inteligencia artificial de la siguiente manera: “Es la ciencia y la ingeniería de la fabricación de máquinas inteligentes, especialmente programas informáticos inteligentes. Está relacionada con la tarea similar de usar computadoras para entender la inteligencia humana, pero la IA no tiene que limitarse a métodos que son biológicamente observables".

Según International Business Machines Corporation (2020) define a la inteligencia artificial como un campo, conformado por la combinación de la ciencia informática y los conjuntos de datos robustos, que permite la resolución de problemas. También señala que en la inteligencia artificial se encuentran los subcampos de Machine Learning y Deep Learning, los cuales están conformados por algoritmos de IA que tienen como objetivo crear sistemas que hagan predicciones o clasificaciones basadas en datos de entrada.

Londoño (2023) define a la inteligencia artificial (IA) como aquella que busca simular la inteligencia humana mediante algoritmos y

sistemas informáticos que les permite ejecutar tareas simples o complejas, tal y como lo realizan las personas.

2.1.6. Machine Learning

También llamado en español “Aprendizaje automático”. Se trata de una disciplina dentro de la inteligencia artificial que emplea algoritmos para que los ordenadores puedan reconocer patrones en los datos y efectuar predicciones. (Iberdrola, 2020)

Según Hurwitz & Kirsch (2018), Machine Learning emplea una amplia gama de algoritmos que progresan de manera iterativa mediante la asimilación de datos con el objetivo de mejorar, describir y anticipar resultados. A medida que estos algoritmos absorben datos de entrenamiento, se tiene la capacidad de generar modelos más precisos basados en dicha información. Un modelo de Aprendizaje Automático es el producto resultante tras entrenar un algoritmo con datos. En la actualidad, el Aprendizaje Automático se ha vuelto esencial para la creación de modelos analíticos. (como se citó con Jimenez Moreano & Qquecho Ccachainca, 2020)

2.1.7. Deep Learning

Según International Business Machines (2020), el aprendizaje profundo constituye un subconjunto del campo de Machine Learning. Esta se caracteriza principalmente por estar conformado por una estructura de red neuronal compuesta por tres o más capas. Estas arquitecturas neuronales buscan imitar ciertos aspectos del funcionamiento del cerebro humano, aunque aún están lejos de alcanzar su capacidad. No obstante, permiten adquirir conocimiento a través del análisis de vastas cantidades de información. Aunque una red neuronal de una única capa ya tiene la capacidad de realizar aproximaciones en sus predicciones, la inclusión de capas adicionales, denominadas capas

ocultas, desempeña un papel fundamental al optimizar y perfeccionar la exactitud de los resultados obtenidos.

2.1.8. Visión Artificial

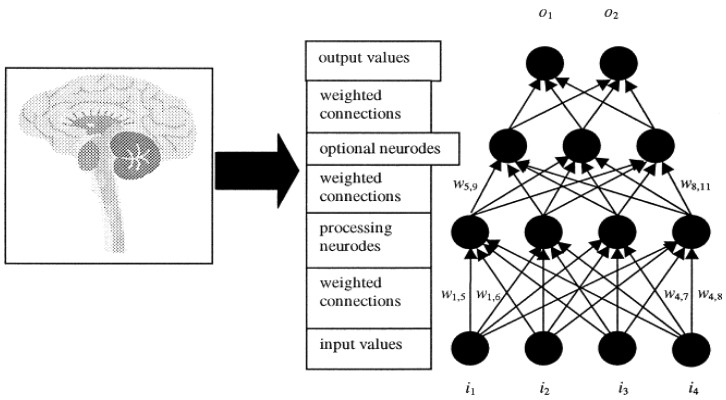
También conocido como Visión por computadora o Visión Técnica. Es aquel campo de la Inteligencia Artificial que se refiere a la capacidad de las computadoras de extraer información valiosa a partir de imágenes digitales y videos, con la finalidad de que puedan realizar acciones específicas basadas en esa información. (International Business Machines Corporation, 2020)

2.1.9. Redes Neuronales Artificiales

Según Sadiq et al. (2019), las redes neuronales artificiales son una técnica de modelado inspirada en el sistema nervioso humano que permite aprender con el ejemplo a partir de datos representativos que describen un fenómeno físico o un proceso de decisión. Estas se componen de capas de entrada, ocultas y de salida con neuronas conectadas (nodos) para simular el cerebro humano, tal y como se muestra en la Figura 1.

Figura 1

Arquitectura de red neuronal artificial



Nota. Adaptado de *Sample artificial neural network architecture* [Fotografía], por Walczak & Cerpa, 2003, Sciencedirect (https://www.sciencedirect.com/science/article/abs/pii/B0122274105008371).

A cada conexión entre nodos se le asigna un peso y tiene a su vez asociado un umbral. Estos son valores que influyen en como la información fluye y se procesa a través de la red. Si a la salida de cualquier nodo, la cual se calculó basado en las entradas y pesos de las conexiones, supera el umbral establecido para ese nodo, se considera que esta activado, lo cual hace que el nodo envíe su salida o resultado a la siguiente capa de la red, donde se procesará aún más. Por otro lado, si la salida calculada de un nodo no supera el umbral, se considera que no está activa y en consecuencia no enviará ninguna información a la siguiente capa de red. (International Business Machines Corporation, 2022)

Las redes neuronales para mejorar su capacidad de hacer predicciones y tomar decisiones precisas, necesitan aprender de ejemplos o datos de entrenamiento. A medida que procesan más ejemplos y ajustan sus conexiones internas, mejoran su capacidad para realizar tareas específicas con mayor precisión. Una vez que estas redes neuronales han sido ajustadas y entrenadas con suficientes datos, se vuelven herramientas poderosas en los campos de la informática y la inteligencia artificial. Esto se debe a que pueden analizar y procesar grandes cantidades de información a velocidades muy altas. (Barba, 2021a)

2.1.10. Redes Neuronales Convolucionales

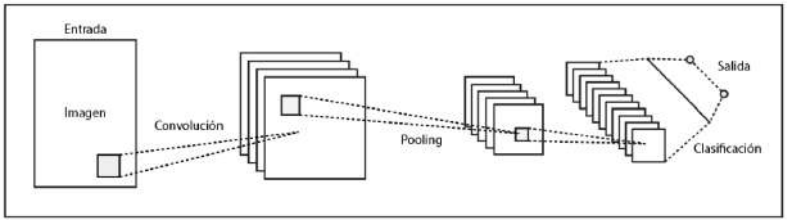
Son redes utilizadas generalmente para el reconocimiento de imágenes, el reconocimiento de patrones y/o la visión por

computadora. Estas redes aprovechan los principios del álgebra lineal, en particular la multiplicación de matrices, para identificar patrones dentro de una imagen. (International Business Machines Corporation, 2022)

Montenegro y Villa (2019) mencionan a través de la Figura 2, que la arquitectura de una red neuronal convolucional está compuesta por un conjunto de capas internas diseñadas de manera progresiva. En un enfoque gradual, las capas iniciales tienen la función de detectar elementos simples, como líneas, curvas, bordes y esquinas, mientras que las capas más profundas adquieren la capacidad de identificar patrones más complejos, como una mano, un rostro o la silueta de un animal. Asimismo, este tipo de red neuronal se caracteriza por incluir múltiples capas de convolución y pooling que se alternan en su disposición. Al final de la estructura, la red cuenta con una capa completamente conectada, también conocida como capa full-connected, que desempeña un papel crucial en el proceso de clasificación. Por lo general, esta red recibe como entrada una imagen con dimensiones [m, m, r], donde “m” corresponde tanto a la altura como al ancho de la imagen, y “c” al número de canales.

Figura 2

Arquitectura de una red neuronal convolucional



Nota. Adaptado de *Estructura básica de una red neuronal convolucional* [Fotografía], por Montenegro & Villa, 2019, Brilliant (https://brilliant.org/wiki/feedforward-neural-networks/).

2.1.11. Python

Cuando se trata de reconocimiento de imágenes, Python es el lenguaje de programación elegido por la mayoría de los científicos de datos e ingenieros de visión por computadora. Admite una gran cantidad de bibliotecas diseñadas específicamente para flujos de trabajo de IA, incluida la detección y el reconocimiento de imágenes. (Bosch, 2023)

2.1.12. Open CV

Se trata de una librería de código abierto que incluye más de 2500 algoritmos y se centra en la visión artificial y Machine Learning. (Rodriguez, 2021)

Con esta herramienta se pueden llevar a cabo diversas tareas, tales como la adquisición y escritura de imágenes, la transformación de imágenes, el reconocimiento de objetos y el seguimiento de estos. (Lopez, 2018)

2.1.13. Mediapipe

MediaPipe es un proyecto de Google que ofrece "soluciones de aprendizaje automático personalizables, multiplataforma y de código abierto para medios en vivo y en streaming". (O'Connor, 2022)

Además, proporciona un conjunto de bibliotecas y herramientas para poder aplicar rápidamente técnicas de inteligencia artificial (IA) y aprendizaje automático (ML) en distintas soluciones, las cuales son compatibles con plataformas como Android, web y Python. Estas son: detección de objetos, clasificación de imágenes, segmentación de imágenes, segmentación interactiva, detección de puntos de referencia de manos, reconocimientos de gestos, incrustación de imágenes, detección de rostro, detección de puntos de referencia faciales, clasificación de texto, incrustación de texto,

detector de idioma y clasificación de audio. (Google for Developers, 2023)

2.1.14. Numpy

Es una biblioteca de Python de código abierto que se utiliza en casi todos los campos de la ciencia y la ingeniería. Es el estándar universal para trabajar con datos numéricos en Python. Esta biblioteca contiene matrices multidimensionales y estructuras de datos matriciales por lo que se puede realizar variedades de operaciones matemáticas en matrices. (Alammar, 2022)

Debido a que una de sus características clave es su capacidad de manejar eficientemente arreglos multidimensionales de datos y que en el caso de imágenes y videos que son esencialmente matrices bidimensionales o tridimensionales de valores de píxeles o cuadros de video, se puede manipular y procesar estos arreglos, permitiendo acceder a valores individuales, realizar operaciones a nivel de pixel y aplicar transformaciones geométricas. (Hualde, 2023)

2.1.15. Math

Es un módulo matemático en Python que nos da acceso a funciones matemáticas definidas por el estándar C, las cuales que mediante este módulo nos permite solo trabajar con números reales. Entre ellas están: las funciones aritméticas, hiperbólicas, trigonométricas, exponenciales y logarítmicas. (Shooken, 2023)

2.1.16. Hand Tracking Module

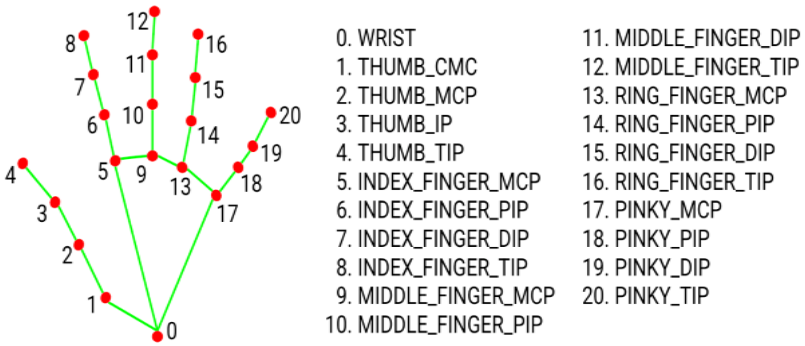
Hand Tracking es el proceso en el que una computadora utiliza la visión artificial para detectar una mano a partir de una imagen de entrada y mantiene el foco en el movimiento y la orientación de la mano. (Dawe, 2022)

Mediante CvZone, se puede utilizar el módulo Hand Tracking, ya que es un paquete de visión por computadora diseñado para facilitar la ejecución de funciones de procesamiento de imágenes mediante las bibliotecas de OpenCV y Mediapipe. (Darshansol, 2023)

Es por ello que a través de CvZone, se puede obtener los 21 puntos de referencia de la mano, tal y como se muestra en la figura 3.

Figura 3

Los 21 puntos de referencia detectados en una mano



Nota. Adaptado de *The 21 hand points that MediaPipe identifies* [Fotografía], por Dawe, 2022, WebScale (https://www.webscale.com/engineering-education/creating-a-hand-tracking-module-using-python-opencv-and-mediapipe/).

2.1.17. Reconocimiento de voz

Es la capacidad de una máquina para escuchar palabras habladas e identificarlas. Posteriormente, este reconocimiento puede ser aprovechado en Python para transformar las palabras habladas en texto, lo que posibilita efectuar búsquedas o generar respuestas. Incluso se puede programar algunos dispositivos para que respondan a estas palabras habladas. (Ravikiran, 2023)

Para realizar el reconocimiento de voz en Python, se debe instalar un paquete de reconocimiento de voz. Hoy día, según Ravikiran (2023) existen varios paquetes en línea tal y como se muestra Tabla 1.

Tabla 1

Paquetes en línea que se utilizan en Python

Paquete	**Funcionalidad**
Apiai	Incluye procesamiento del lenguaje natural para identificar la intención del hablante.
Google-cloud-speech	Es un servicio en la nube ofrecido por Google que utiliza tecnologías avanzadas de procesamiento de lenguaje natural para convertir el habla en texto. Está diseñado para ser utilizado en aplicaciones más complejas y a gran escala.
Speech Recognition	Es una biblioteca de Python que actúa como una interfaz para varios motores de reconocimiento de voz, incluido el reconocimiento de voz de Google. Proporciona una manera más sencilla y directa de realizar tareas básicas de reconocimiento de voz, como convertir audio a texto.
Watson-developer-cloud	Se basa en modelos de deep learning, machine learning y

	procesamiento del lenguaje natural (NLP). Se puede utilizar para realizar tareas básicas de reconocimiento de voz.

***Nota*.** Esta tabla muestra los diferentes paquetes de reconocimiento de voz en Python. Adaptado de *"Picking and installing a speech recognition package"*, por Ravikiran, 2023, Simplilearn (https://www.simplilearn.com/tutorials/python-tutorial/speech-recognition-in-python#:~:text=Speech%20recognition%20is%20a%20machine's,respond%20to%20these%20spoken%20words).

Aranda Garay y Vargas Benites (2020) en el benchmarking de las tecnologías de Speech Recognition que realizaron en su investigación, destacaron diversas tecnologías gratuitas como DialogFlow, Android Speech Recognizer, CMU Sphinx y Google Web Speech Api, las cuales soportan varios idiomas y pueden ser utilizadas como Api de reconocimiento de voz para la biblioteca SpeechRecognition.

2.1.18. Recognizer class

En la biblioteca SpeechRecognition, hay una clase denominada "Recognizer" que posibilita el reconocimiento del habla. Esta clase nos permite crear una instancia mediante "sr.Recognizer()", la cual ofrece una variedad de configuraciones y funcionalidades para detectar la voz proveniente de una fuente de audio. (Ámos, 2019)

Una vez que se ha creado una instancia, tenemos 7 métodos para reconocer la voz desde una fuente de audio mediante varias API como:

- recognize_bing(): Discurso de Microsoft Bing
- recognize_google(): API de voz web de Google
- recognize_google_cloud(): Google Cloud Speech : requiere la instalación del paquete google-cloud-speech
- recognize_houndify(): Houndify de SoundHound
- recognize_ibm(): IBM voz a texto
- recognize_sphinx(): CMU Sphinx
- recognize_wit(): ingenio.ai

De los 7, solo recognize_google() funciona sin necesidad de especificar una clave API, ya que utiliza una predeterminada. Las otras API requieren una autenticación con una clave API o una combinación de nombre de usuario y contraseña. (Ámos, 2019)

2.1.19. Optimizador Adam:

Adam es un algoritmo de optimización que puede emplearse como alternativa al clásico procedimiento de descenso de gradiente estocástico para ajustar iterativamente los pesos de la red basándose en los datos de entrenamiento. El algoritmo de optimización Adam es una versión mejorada del descenso de gradiente estocástico, que ha ganado popularidad recientemente en aplicaciones de aprendizaje profundo, como visión por computadora y procesamiento de lenguaje natural. (Brownlee, Machine learning mastery, 2021)

Adam es un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivo estocásticas, que se basa en estimaciones adaptativas de momentos de orden inferior. Es fácil de implementar, computacionalmente eficiente, y requiere poca memoria. Además, es invariante a los cambios de escala diagonal de

los gradientes y es especialmente adecuado para problemas grandes en términos de datos y/o parámetros. Adam es apropiado para objetivos no estacionarios y problemas con gradientes muy ruidosos y/o escasos. Sus hiperparámetros son intuitivos y generalmente necesitan pocos ajustes. Se discuten algunas conexiones con algoritmos relacionados que inspiraron a Adam, y se analizan sus propiedades teóricas de convergencia, proporcionando un límite de arrepentimiento comparable a los mejores resultados conocidos en la optimización convexa en línea. Los resultados empíricos muestran que Adam funciona bien en la práctica y se compara favorablemente con otros métodos de optimización estocástica. Finalmente, se examina Adamax, una variante de Adam basada en la norma del infinito. (Diederik P. Kingma, 2019).

2.1.20. MobileNetV2

MobileNetV2 es una arquitectura de red neuronal convolucional (CNN) diseñada por Google Research, específicamente para tareas de visión por computadora en dispositivos móviles y embebidos. Esta arquitectura se basa en el éxito de MobileNetV1, mejorando la eficiencia y el rendimiento. Esta función devuelve un modelo de clasificación de imágenes de Keras, opcionalmente cargado con pesos previamente entrenados en ImageNet. (Keras, s.f.)

MobileNetV2 presenta una estructura basada en residuos invertidos, en la cual tanto la entrada como la salida del bloque residual se componen de capas delgadas tipo cuello de botella, a diferencia de los modelos residuales convencionales que emplean representaciones expandidas en la entrada. Esta arquitectura también hace uso de convoluciones de poca profundidad para filtrar características dentro de la capa intermedia de expansión. (Howard, y otros, 2017)

2.1.20.1. Argumentos de entrada de MobileNetV2

Input_shape

El parámetro es opcional y debe especificarse únicamente si include_top es False. De lo contrario, la entrada debe tener la forma (224, 224, 3) en formato "channels_last" o (3, 224, 224) en formato "channels_first". Debe tener exactamente 3 canales y un ancho y alto de al menos 32. (Keras, s.f.a)

Alpha

Regula el ancho de la red y se conoce como multiplicador de ancho en el documento de MobileNet. Si alpha es menor que 1.0, reduce proporcionalmente la cantidad de filtros en cada capa. Si alpha es mayor que 1.0, incrementa proporcionalmente la cantidad de filtros en cada capa. Si alpha es igual a 1, se utiliza la cantidad de filtros predeterminada especificada en el documento. El valor predeterminado es 1.0. (Keras, s.f.b)

Include_top

Un valor booleano que indica si se debe incluir la capa totalmente conectada en la parte superior de la red. El valor predeterminado es True. (Keras, s.f.c)

Weights

Puede ser None (inicialización aleatoria), "imagenet" (preentrenado en ImageNet) o una ruta al archivo de pesos que se desea cargar. El valor predeterminado es "imagenet". (Keras, s.f.d)

Input_tensor

Un tensor de Keras opcional (como el resultado de layers.Input()) que se puede utilizar como entrada de imagen para el modelo. La opción input_tensor es útil para compartir entradas entre diversas redes diferentes. El valor predeterminado es None. (Keras, s.f.e)

Pooling

Modo de agrupación opcional para la extracción de características cuando include_top es False.

- None: Significa que la salida del modelo será el tensor 4D resultante del último bloque convolucional.
- Avg: Se aplicará la agrupación promedio global a la salida del último bloque convolucional, produciendo así un tensor 2D.
- Max: Se aplicará la agrupación máxima global. (Keras, s.f.f)

Classes

Número opcional de clases para clasificar imágenes, que solo debe especificarse si include_top es True y no se ha definido ningún argumento para weights. El valor predeterminado es 1000. (Keras, s.f.g)

Classifier_activation:

Una cadena invocable que especifica la función de activación para usar en la capa "superior". Esta opción se ignora a menos que include_top=True. Se configura classifier_activation=None para devolver los logits de la capa "superior". Si es que se quiere cargar pesos previamente entrenados, classifier_activation será softmax. (Keras, s.f.h)

2.1.21. Image data generator:

Es una clase proporcionada por la librería, consiste en aplicar diversas transformaciones a imágenes originales, generando múltiples copias modificadas de la misma imagen. Cada copia presenta diferencias en ciertos aspectos según las técnicas de aumento utilizadas, como desplazamiento, rotación, volteo, entre otras. Estas variaciones menores en la imagen original no alteran su clase objetivo; simplemente ofrecen una nueva perspectiva para capturar el objeto en situaciones reales. Por esta razón, se emplea frecuentemente en la creación de modelos de aprendizaje profundo. (Bhandari, 2024)

2.1.22. HTML

HTML, que significa Lenguaje de Marcas de Hipertexto en inglés (HyperText Markup Language), constituye el elemento fundamental de la World Wide Web. Este lenguaje define tanto el significado como la estructura del contenido que se encuentra en las páginas web. Además de HTML, en el desarrollo web suelen emplearse otras tecnologías para definir el aspecto visual o presentación de una página (CSS) y para añadir funcionalidades o comportamientos interactivos (JavaScript). (MDN contributors, 2023)

2.1.23. CSS

CSS (Cascading Style Sheets), o Hojas de Estilo en Cascada, no se considera estrictamente un lenguaje de programación ni un lenguaje de marcado, sino más bien un lenguaje de estilo. Su función principal es aplicar estilos de forma selectiva a los elementos presentes en documentos HTML. Por ejemplo, si se desea que todo el texto dentro de los elementos de párrafo en una página HTML tenga un color rojo, se debe escribir una regla CSS para lograrlo. (MDN Contributors, 2023)

2.1.24. JavaScript

JavaScript (JS) es un lenguaje de programación ágil, interpretado y compilado en tiempo justo-a-tiempo (just-in-time), que cuenta con funciones de primera clase. Aunque es conocido principalmente como un lenguaje de secuencias de comandos (scripting) para páginas web, también se utiliza en una variedad de entornos fuera del navegador, como Node.js, Apache CouchDB y Adobe Acrobat.
(MDN Contributors, 2023)

2.1.25. Flask

Flask es un microframework de desarrollo web escrito en Python. Es ligero y fácil de usar, lo que lo hace ideal para desarrollar aplicaciones web simples y escalables. Flask se diseñó para ser extensible, permitiendo a los desarrolladores añadir solo las funcionalidades que necesitan a través de extensiones. (Ionos, 2023)

2.1.26. JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos liviano y fácil de comprender. Es legible y escribible de manera sencilla para los humanos, y también fácil de analizar y generar por las máquinas. Está basado en un subconjunto del estándar de lenguaje de programación JavaScript ECMA-262, tercera edición, publicado en diciembre de 1999. A pesar de su origen en JavaScript, JSON es completamente independiente del lenguaje y sigue convenciones familiares para programadores de lenguajes como C, C++, C#, Java, JavaScript, Perl, Python, entre otros. Esta flexibilidad hace que JSON sea un formato ideal para el intercambio de datos entre diferentes sistemas. (JSON, s.f.)

2.1.27. Jsonify en Flask

En Flask, "jsonify" es una función que convierte objetos de Python en formato JSON. JSON (JavaScript Object Notation), el cual es un formato de intercambio de datos ligero y ampliamente utilizado en aplicaciones web para transmitir datos entre un servidor y un cliente de manera eficiente y legible. (KeepCoding, 2022)

2.2. Marco Conceptual

2.2.1. Sistema de comunicación

Es aquel sistema de elementos en los cuales mediante un emisor, receptor y mensaje se produce un proceso comunicativo entre 2 individuos como mínimo (Llamas, 2021).

2.2.2. Comunicación bidireccional

La comunicación bidireccional es aquella que implica un intercambio constante de retroalimentación, en el cual tanto el receptor como el emisor del mensaje alternan sus roles, permitiendo así la construcción de una conversación que fluye en ambas direcciones. La principal diferencia que existe entre la comunicación bidireccional y la comunicación unidireccional es que en la última no existe posibilidad de retroinformación, es decir el receptor no tiene chance de preguntar o responder al emisor. (Villaverde, 2022)

2.2.3. Pérdida auditiva

Una persona con pérdida auditiva es aquella que escucha por igual o encima de los 20 dB. La pérdida de audición es un problema común causado por el ruido, el envejecimiento, las enfermedades y la herencia. Una persona con pérdida auditiva en muchos casos, se le hace muy difícil o imposible de escuchar el habla u otros sonidos. (National Institute on Aging, 2018)

Hay diferentes tipos de pérdida auditiva, y pueden variar de leves, moderadas, severas o profundas. Algunos tipos de pérdida auditiva son temporales y otros son permanentes. (Healthdirect, 2022)

2.2.3.1. Pérdida auditiva leve

En este tipo de disminución de la capacidad auditiva, la persona puede aún percibir los sonidos del lenguaje hablado, pero tiene dificultades para escuchar sonidos de baja intensidad, como los susurros. (Centros para el Control y la Prevención de Enfermedades, 2020)

2.2.3.2. Pérdida auditiva moderada

En este tipo de pérdida auditiva la persona puede que no escuche nada cuando le hablan a un volumen normal. (Centros para el Control y la Prevención de Enfermedades, 2020)

2.2.3.3. Pérdida auditiva grave

En este tipo de pérdida auditiva la persona no escucha absolutamente nada cuando le hablan a un volumen normal sin embargo aún percibe sonidos fuertes. (Centros para el Control y la Prevención de Enfermedades, 2020)

2.2.3.4. Pérdida auditiva profunda

En este tipo de pérdida auditiva la persona no escucha ni sonidos débiles ni fuertes. (Centros para el Control y la Prevención de Enfermedades, 2020)

2.2.4. Clasificación de imágenes

Entre las tareas de visión artificial, la clasificación de imágenes se destaca con su papel insustituible en la tecnología moderna. Implica asignar una etiqueta a una imagen completa en función de los datos de entrenamiento preexistentes de imágenes ya etiquetadas.

2.2.5. Tipos de clasificación de imágenes

Dependiendo del problema en cuestión, existen diferentes tipos de metodologías de clasificación de imágenes que se pueden emplear. Estos son los de tipo: binario, multiclase, etiqueta múltiple y jerárquico.

2.2.5.1. Clasificación binaria

En el aprendizaje automático, la clasificación binaria es un algoritmo de clasificación plano que se usa para distinguir y/o clasificar los puntos de datos en dos categorías, es decir toma una lógica de 1 o 0. Principalmente se usan cuando se requieren respuestas rápidas y cortas (SI/NO). Algunos ejemplos de aplicación de este clasificador son: Diagnóstico de un médico (Saludable o enfermo), Análisis de correo electrónico (No spam o Spam) y clasificación de imágenes (perro o gato). (Karabiber, 2023)

2.2.5.2. Clasificación multiclase

En el aprendizaje automático, la clasificación multiclase es un algoritmo de clasificación plano que se usa para clasificar más de 2 clases, por ejemplo, clasificar un conjunto de imágenes de frutas que pueden ser naranjas, manzanas o peras. Esta clasificación supone que cada muestra se asigne una etiqueta distinta a las otras, es decir si hablamos del ejemplo mencionado: una fruta puede ser manzana y otra pera, pero no ambas a la vez. (Nabi, 2018)

2.2.5.3. Clasificación con etiqueta múltiple

Es aquella clasificación, que a diferencia de la clasificación multiclase permite que al elemento se le pueda asignar varias etiquetas. Por ejemplo, si tenemos una imagen de ensalada de frutas, es posible asignar como etiquetas varios colores como rojo para la manzana, amarillo para el plátano y así con las demás frutas. Como resultado se tendrá una imagen la cual tendrá varios colores como etiquetas. (Clark, 2023)

2.2.5.4. Clasificación con jerárquica

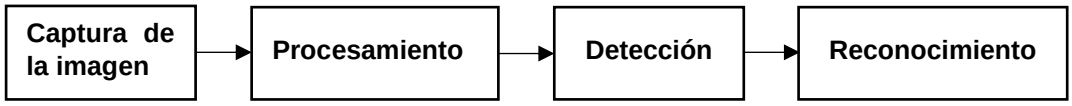
En el aprendizaje automático, la clasificación con jerarquía es un algoritmo de clasificación jerárquica en lo cual las clases que se van a predecir se organizan en una jerarquía de clases.

2.2.6. Estructura básica de un proceso de clasificación de imágenes

La estructura general de un proceso de clasificación y reconocimiento de imágenes por visión artificial, según Zimmemann (2014), se presenta en la Figura 4.

Figura 4

Proceso de clasificación y reconocimiento de imágenes por visión artificial



Nota. Adaptado de *Estructura básica de un proceso de reconocimiento por visión artificial*, por Zimmemann, 2014, Universidad Carlos III de Madrid (https://e-archivo.uc3m.es/entities/publication/266f7034-a1d9-4731-81e2-a7c554bff62d).

2.2.6.1. Captura de la imagen

Es la etapa en la que se obtiene una fotografía en 2D o 3D, o una secuencia de video que se desea analizar para examinar algún objeto en particular (Zimmermann, 2014a).

2.2.6.2. Procesamiento de la imagen

En esta etapa se realizan cambios en la imagen original para hacer más fácil la detección y el reconocimiento de objetos. Algunas de las técnicas más comunes que se utilizan son el filtrado de color, la dilatación, la conversión de la imagen a escala de grises, agudizamiento (sharpening) y las trasformaciones de modelos de color (Zimmermann, 2014b).

Otro método para mejorar la calidad de los datos de una imagen es el aumento de datos, el cual es el proceso de crear nuevas variaciones de las imágenes mediante la transformación de imágenes, con el objetivo de diversificar el conjunto de datos de entrenamiento y aumentar el tamaño, lo que ayudaría a mejorar la precisión del modelo de clasificación de imágenes. (Clark, 2023)

2.2.6.3. Detección

En esta etapa se busca identificar patrones visuales dentro de una imagen que se usarán para distinguir un objeto de otro. En el aprendizaje automático, existen diversas técnicas para llevar a cabo esta tarea, como la detección de vértices mediante filtros, la búsqueda de contornos y la identificación de variaciones entre píxeles en términos de distancia, proximidad, color o agrupamiento. (Zimmermann, 2014c).

Mientras que en Deep Learning se pueden utilizar algoritmos como las Redes Neuronales Convolucionales (CNN), que se utilizan ampliamente para permitir que los algoritmos aprendan directamente

de los datos sin procesar. En este caso la red aprende sobre un gran conjunto de datos de imágenes etiquetadas y distingue los patrones importantes para diferentes clases de imágenes. (Barba Guamán, 2021b)

2.2.6.4. Reconocimiento

Una vez que el modelo ha sido entrenado, se debe probar con imágenes que no forman parte del conjunto de datos de entrenamiento. Esto se utiliza para determinar la usabilidad, el rendimiento y la precisión del modelo. Es por ello que, alrededor del 80-90% del conjunto de datos de imágenes se utiliza para el entrenamiento del modelo, mientras que los datos restantes se reservan para la validación del modelo. (Bosch, 2023)

2.2.7. Tasa de aprendizaje

La tasa de aprendizaje es un hiperparámetro que determina la magnitud del ajuste que se realiza en el modelo en respuesta al error estimado cada vez que se actualizan sus pesos. Seleccionar la tasa de aprendizaje adecuada es un desafío: un valor demasiado pequeño puede hacer que el entrenamiento sea prolongado y propenso a estancarse, mientras que un valor demasiado grande puede llevar a un aprendizaje rápido de pesos subóptimos o a un entrenamiento inestable. (Brownlee, Machine Learning Mastery, 2020)

Una red neuronal aprende o aproxima una función para asignar de manera óptima entradas a salidas. La tasa de aprendizaje, un hiperparámetro, controla la rapidez con la que el modelo aprende. Este hiperparámetro regula la magnitud del error utilizado para ajustar los pesos del modelo en cada actualización, como al final de cada lote de instancias de entrenamiento. Si la tasa de aprendizaje está correctamente ajustada, el modelo optimizará la estimación de la función dada la cantidad de recursos disponibles (el número de capas y nodos

por capa en un número específico de épocas de entrenamiento), procesando de manera efectiva los datos de entrenamiento. (Deepchecks, 2024)

2.2.8. Optimizadores

Los optimizadores son algoritmos que se utilizan para ajustar los parámetros de un modelo durante el proceso de entrenamiento con el fin de minimizar la función de pérdida. La función de pérdida es una medida de qué tan bien el modelo está haciendo predicciones en comparación con los valores reales. El objetivo principal de un optimizador es encontrar los valores óptimos de los pesos y sesgos en la red neuronal para reducir la función de pérdida. Al hacerlo, el modelo se vuelve más preciso en sus predicciones. (Keras, s.f.)

2.2.9. Modelos secuenciales

Son una forma de organizar y construir redes neuronales en la que las capas se apilan una encima de la otra de manera lineal. Cada capa en un modelo secuencial pasa su salida a la siguiente capa en la secuencia. Esto significa que la información fluye en una sola dirección, de la entrada a través de cada capa hasta la salida. Es una forma común y sencilla de crear redes neuronales en muchos frameworks y librerías de aprendizaje profundo, como TensorFlow o Keras. (Fchollet, 2020)

Un modelo secuencial se compone de una serie de capas que pueden incluir capas de neuronas densamente conectadas y capas de convolución que es el componente principal de una CNN (Red Neuronal Convolucional). (Mostafa, Neural Engineering Techniques for Autism Spectrum Disorde, 2021)

2.2.10. Curvas de precisión y pérdida

Las curvas de aprendizaje son una herramienta de diagnóstico comúnmente empleada en el campo del aprendizaje automático para evaluar algoritmos que aprenden de un conjunto de datos de entrenamiento de manera incremental. Durante el entrenamiento, el modelo se evalúa tanto en el conjunto de datos de entrenamiento como en un conjunto de datos de validación después de cada actualización, y se registran las métricas de rendimiento. Estos datos se utilizan para generar gráficos que muestran el rendimiento medido a lo largo del tiempo, lo que permite visualizar las curvas de aprendizaje. Analizar las curvas de aprendizaje durante el entrenamiento puede ayudar a identificar problemas en el proceso de aprendizaje, como un modelo que está infra o sobreajustado, así como determinar si los conjuntos de datos de entrenamiento y validación son representativos y adecuados para el problema en cuestión. (Brownlee, Machine learning mastery, 2021)

2.2.11. Servidor web

Un servidor web puede referirse tanto al hardware como al software, o a la combinación de ambos trabajando en conjunto.

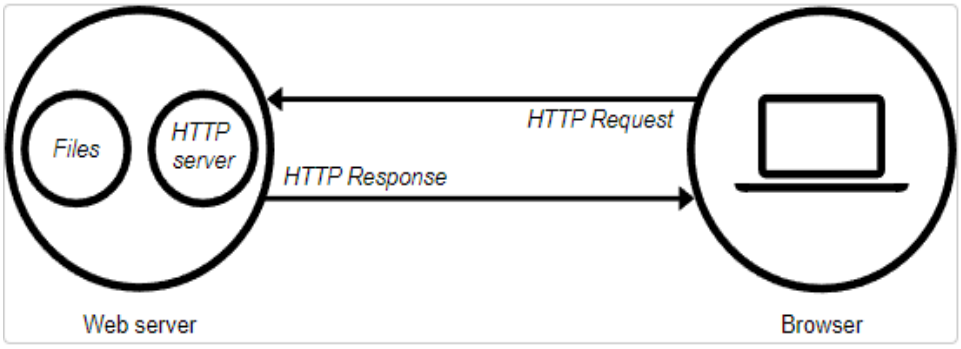
En el contexto del hardware, un servidor web es una computadora que alberga el software del servidor web junto con los archivos que conforman un sitio web, como documentos HTML, imágenes, hojas de estilos CSS y archivos JavaScript. Este hardware de servidor web se conecta a Internet y facilita el intercambio de datos con otros dispositivos conectados a la web. Por otro lado, en términos de software, un servidor web consta de varios componentes que controlan cómo los usuarios de la web acceden a los archivos alojados en el servidor. Básicamente, un servidor web necesita, como mínimo, un servidor HTTP. Este servidor HTTP es un software que puede interpretar las URLs (las direcciones web) y el protocolo HTTP (que es el protocolo que tu navegador utiliza para obtener las páginas

web). Los usuarios pueden acceder a este servidor HTTP a través de los nombres de dominio de los sitios web que aloja, y el servidor entrega el contenido de esos sitios web al dispositivo del usuario final. (Colaboradores Mdn, 2023)

En su nivel más fundamental, cuando un navegador requiere un archivo alojado en un servidor web, solicita ese archivo al servidor utilizando el protocolo HTTP. Una vez que la solicitud llega al hardware del servidor web adecuado, el software del servidor HTTP acepta la solicitud, localiza el documento solicitado y lo devuelve al navegador, también a través del protocolo HTTP. En caso de que el servidor no pueda encontrar el documento solicitado, responderá con un código de respuesta 404 en su lugar. (Colaboradores Mdn, 2023)

Figura 5

Comunicación HTTP Cliente/Servidor



Nota. Adaptado de *Representación básica de una conexión cliente/servidor a través de HTTP* [Fotografía], por Mozilla web Docs, 2024, MDN Web Docs (https://developer.mozilla.org/es/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server).

2.2.12. Comunicación a través de HTTP

HTTP, o Protocolo de Transferencia de Hipertexto, es fundamental para la comunicación en la web, facilitando un intercambio eficiente de datos en Internet. Este protocolo permite la transmisión de documentos hipermedia, como HTML, y establece las reglas y estándares necesarios para que las aplicaciones web estén interconectadas y sean accesibles. (Digital samba, 2024)

HTTP fue desarrollado junto con HTML para crear el primer navegador web interactivo basado en texto, conocido como la World Wide Web original. Actualmente, este protocolo continúa siendo uno de los métodos principales para acceder a Internet. Como un protocolo de solicitud y respuesta, HTTP permite a los usuarios interactuar con recursos web, como archivos HTML, mediante el intercambio de mensajes de hipertexto entre clientes y servidores. Los clientes HTTP usualmente se comunican con los servidores a través de conexiones del Protocolo de Control de Transmisión (TCP). (Extrahop, 2024)

2.2.13. Comunicación GET

GET es un método HTTP utilizado para enviar solicitudes a un servidor, principalmente para obtener datos. Este método se ha diseñado para ser seguro, idempotente, lo que significa que realizar múltiples solicitudes idénticas debería producir el mismo resultado cada vez hasta que otro método cambie el recurso. También es capaz de almacenarse en caché para mejorar la eficiencia. Al utilizar el método GET, se solicita una representación del recurso específico al servidor. Es una forma simple, eficiente y ampliamente utilizada en servicios web para recuperar datos. En esencia, es como la forma estándar de pedir información a los servidores. (Akto, 2023).

2.2.14. Comunicación POST

POST es un método HTTP crucial que se utiliza para enviar datos a un servidor para crear o actualizar un recurso. A diferencia de GET, que se utiliza para recuperar información del servidor, POST se utiliza para enviar datos para su procesamiento a un recurso específico. Generalmente se usa al cargar un archivo o enviar un formulario web completo. A diferencia de GET, POST no es idempotente, lo que significa que realizar varias solicitudes idénticas puede generar resultados diferentes. Cada POST solicitud puede crear un nuevo recurso o cambiar el estado de un recurso existente. Es esencial cuando el cliente necesita enviar datos al servidor para crear o actualizar recursos. Es una forma segura de enviar datos, ya que los datos no se agregan al archivo URL (como en GET), sino que se incluyen en el archivo request body. (Akto, 2023).

2.2.15. Proceso de reconocimiento de voz

El proceso de reconocimiento de voz en Python se basa en algoritmos que realizan dos tipos de modelado: el modelado lingüístico y el modelado acústico. Estas técnicas trabajan conjuntamente para decodificar la información hablada en un formato comprensible. (Ravikiran, 2023a)

El modelado acústico, un componente esencial del reconocimiento de voz, tiene la tarea de identificar los elementos más básicos del habla, como fonemas o patrones fonéticos. Estos elementos forman la base de nuestras expresiones orales. (Ravikiran, 2023b)

Por otro lado, el modelado lingüístico entra en juego para interpretar y asignar significado a estos patrones fonéticos, convirtiéndolos en palabras y oraciones con sentido. El proceso de reconocimiento de voz se inicia capturando la energía sonora generada por el hablante a través de un micrófono. Esta energía sonora se convierte en señales eléctricas y, luego mediante el proceso de

conversión analógica a digital, se transforma en datos digitales y finalmente a texto. (Ravikiran, 2023c)

2.3. Sistema de hipótesis

El uso de un sistema traductor de comunicación bidireccional reduce significativamente los tiempos de reconocimiento de las palabras durante la comunicación entre una persona sorda de la Asociación de Sordos de La Libertad y una persona oyente de La Libertad.

2.4. Variables y Operacionalización

2.4.1. Variables

- ✓ Comunicación de una persona sorda a una persona oyente:
  - **Variable Independiente:** Uso del sistema traductor de comunicación bidireccional.
  - **Variable Dependiente:** Tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad.

- ✓ Comunicación de una persona oyente a una persona sorda:
  - **Variable Independiente:** Uso del sistema traductor de comunicación bidireccional.
  - **Variable Dependiente:** Tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad.

2.4.2. Operacionalización de variables

Tabla 2

Operacionalización de variables

Variable	Definición operacional	Dimensión	Indicador	Unidad de medida	Instrumento
Variable independiente					
Uso de un sistema traductor de comunicación bidireccional.	Presencia o ausencia del uso del sistema traductor de comunicación bidireccional durante la comunicación entre una persona sorda de la Asociación De Sordos De La Libertad y una persona oyente	Uso del sistema	Sin sistema Con sistema	Dicotómica (1=con sistema, 0=sin sistema)	Observación directa
Variables dependientes					
Tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad	Son los tiempos que tardan cada persona oyente de La Libertad en reconocer correctamente cada palabra expresada en el lenguaje de señas peruana por una persona sorda de la Asociación De Sordos De La Libertad, desde el momento en que se expresa la palabra hasta el momento en	Tiempo	Duración del tiempo de reconocimiento	S	Cronómetro

	que indica que ha reconocido la palabra				
Tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad.	Son los tiempos que tardan cada persona sorda de la Asociación De Sordos De La Libertad en reconocer correctamente cada palabra pronunciada por una persona oyente de La Libertad, desde el momento en que se expresa la palabra hasta el momento en que indica que ha reconocido la palabra	Tiempo	Duración del tiempo de reconocimiento	S	Cronómetro

Nota. Elaboración propia.

III. METODOLOGÍA EMPLEADA

3.1. Tipo y nivel de investigación

- **De acuerdo al tipo investigación:** es cuantitativa, porque se busca recolectar datos numéricos como los tiempos de reconocimiento de las palabras, tanto con el uso como sin el uso del sistema traductor de comunicación bidireccional, en personas sordas de la Asociación De Sordos De La Libertad y en personas oyentes de La Libertad.

- **De acuerdo al nivel de investigación**: es explicativa, porque se busca identificar la relación causal entre el uso de un sistema traductor de comunicación bidireccional y los tiempos de reconocimiento de las palabras tanto en las personas sordas de la

Asociación De Sordos De La Libertad como en las personas oyentes de La Libertad.

3.2. Población y muestra de estudio

a) Población

- Personas sordas de la Asociación De Sordos De La Libertad.
- Personas oyentes de La Libertad

b) Muestra

Por conveniencia:

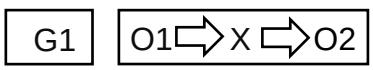
El presente estudio se basa en un muestreo por conveniencia debido a la naturaleza específica de la población objetivo y la disponibilidad de los participantes. Nuestra muestra está conformada por:

- Tres personas sordas de la Asociación De Sordos De La Libertad que utilizan para comunicarse el lenguaje de señas peruana
- Tres personas oyentes de La Libertad que no conocen el lenguaje de señas peruana.

3.3. Diseño de investigación

El diseño de contrastación será de tipo pre – experimental para ambos grupos experimentales:

3.3.1. Para la comunicación de persona sorda a persona oyente



Donde:

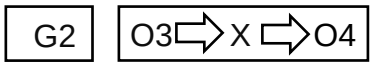
G1: Grupo experimental 1: Personas oyentes del departamento de La Libertad.

O1: Tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad

X: Uso del sistema traductor de comunicación bidireccional.

O2: Tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad, con el uso del sistema traductor de comunicación bidireccional.

3.3.2. Para la comunicación de persona oyente a persona sorda



Donde:

G2: Grupo experimental 2: Personas sordas de la Asociación De Sordos La Libertad.

O3: Tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad.

X: Uso del sistema traductor de comunicación bidireccional.

O4: Tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad, con el uso del sistema traductor de comunicación bidireccional

3.4. Técnicas e instrumentos de recolección de datos

Para la presente investigación, se utilizaron las técnicas e instrumentos de recolección de datos, que se detallan a continuación en la siguiente tabla:

Tabla 3

Técnicas e instrumentos de recolección de datos

Técnicas de recolección de datos	Instrumentos de recolección de datos
Observación	**Equipo de medición**, para medir a través de un cronómetro, los tiempos de reconocimiento de cada

	palabra expresada en el lenguaje de señas peruana por una persona sorda de la Asociación De Sordos De La Libertad para cada persona oyente de La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional. **Equipo de medición**, para medir a través de un cronómetro, los tiempos de reconocimiento de cada palabra pronunciada por una persona oyente de La Libertad para cada persona sorda de la Asociación De Sordos De La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional.

Nota. Elaboración propia.

3.5. Procesamiento y análisis de datos

Para la presente investigación, se utilizaron las técnicas de procesamiento y las técnicas de análisis de datos que se detallan a continuación:

Técnicas de procesamiento:

- Tablas acerca de los tiempos de reconocimiento de cada palabra expresada en el lenguaje de señas peruana por una persona sorda de la Asociación De Sordos De La Libertad para cada persona oyente de La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional.

- Tablas acerca de los tiempos de reconocimiento de cada persona oyente de La Libertad de cada palabra pronunciada por una persona

oyente de La Libertad para cada persona sorda de la Asociación De Sordos De La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional.

Técnicas de análisis:

- Se probará el supuesto de normalidad de la diferencia de los tiempos de reconocimiento de las palabras sin y con el uso del sistema traductor de comunicación bidireccional para cada grupo (personas Sordas de la Asociación De Sordos De La Libertad y personas oyentes de La Libertad). Esto con la finalidad de determinar la prueba estadística que se utilizará para contrastar la hipótesis.

- Si los datos cumplen con el supuesto de normalidad para ambos grupos, se utilizará la prueba t para muestras emparejadas. En caso de que uno de los dos grupos no cumpla con la normalidad de los datos, se aplicará la prueba de Wilcoxon para muestras emparejadas solamente en ese grupo específico que no cumple con la normalidad. Una vez aplicadas las pruebas adecuadas y obtenidos los estadísticos t y los valores p correspondientes para cada grupo, se procederá a interpretar los resultados. Estos resultados permitirán determinar si existe una diferencia significativa en los tiempos de reconocimiento de las palabras antes y después del uso del sistema traductor de comunicación bidireccional, tanto en las personas sordas de la Asociación De Sordos De La Libertad como en las personas oyentes de La Libertad. Si se encuentra una diferencia significativa en ambos grupos, se concluirá que el sistema traductor de comunicación bidireccional reduce de manera significativa los tiempos de reconocimiento de las palabras tanto en las personas sordas de la Asociación De Sordos De La Libertad como en las personas oyentes de La Libertad.

IV. PRESENTACIÓN DE RESULTADOS

4.1. Propuesta de investigación

Para el desarrollo del sistema traductor de comunicación bidireccional, se empleó un equipo informático equipado con un procesador AMD Ryzen 5 5600x y una tarjeta gráfica RTX 3070Ti. Además, se utilizó una cámara digital Full HD para capturar y procesar la señal de video en tiempo real y un micrófono extraíble con cancelación de ruido (auricular inalámbrico “HyperX Cloud II Wireless”) para capturar la voz en tiempo real.

4.1.1. Entorno y Librerías

En el desarrollo del presente proyecto de investigación se utilizó el entorno de desarrollo integrado (IDE) PyCharm, en su versión 2023.3.3, con una distribución de lenguaje Python en su versión 3.9, en donde se importaron las siguientes bibliotecas:

Cvzone: Se utilizó el módulo de detección de manos (HandTrackingModule) y el módulo de clasificación (ClassificationModule) de la biblioteca cvzone para la detección y seguimiento de manos en imágenes y la clasificación de gestos del lenguaje de señas.

TensorFlow: Se utilizaron las bibliotecas layers y models de TensorFlow para construir y entrenar un modelo de aprendizaje automático capaz de reconocer gestos del lenguaje de señas.

MobileNetV2: Se importó la arquitectura de red neuronal preentrenada MobileNetV2 de tensorflow.keras.applications para su uso como base del modelo de clasificación.

ImageDataGenerator: Se utilizó ImageDataGenerator de tensorflow.keras.preprocessing.image para generar lotes de datos de imágenes de forma eficiente durante el entrenamiento del modelo.

Numpy: Se importó numpy como np para realizar operaciones matemáticas eficientes en matrices y vectores.

Matplotlib: Se utilizó matplotlib.pyplot como plt para visualizar gráficos y resultados durante el proceso de desarrollo y entrenamiento del modelo.

Base64: Se importó base64 para la codificación y decodificación de datos binarios, lo que podría ser útil para el procesamiento de imágenes en el sistema.

Time: Se importó time para medir y registrar el tiempo de ejecución de ciertas operaciones y funciones en el sistema.

Cv2: Se importó cv2 como cv2 para acceder a las funciones de OpenCV, lo que permitió el procesamiento de imágenes y la interacción con la cámara digital utilizada en el sistema.

Speech_recognition: Se importó speech_recognition como sr para realizar el reconocimiento de voz a partir de la entrada de audio proveniente de la voz del usuario.

Moviepy: Se importó VideoFileClip y concatenate_videoclips de la biblioteca moviepy.editor para la edición y concatenación de clips de video, lo que podría ser útil para la generación de videos de entrenamiento y evaluación del sistema.

Flask: Se importó Flask para la creación de una aplicación web que permita ejecutar el sistema traductor de comunicación bidireccional en un entorno controlado y accesible a través de un navegador web.

4.1.2. Descripción de la arquitectura del sistema

La arquitectura del sistema se compone de tres códigos, los cuales trabajan en conjunto para ofrecer una solución completa que facilite la comunicación bidireccional entre una persona sorda y una persona oyente. A continuación, se describe cada uno de estos códigos:

4.1.2.1. "Signs_to_text.py"

Es aquel código que contiene funciones, responsables de realizar tareas como capturar imágenes, actualizar conjuntos de datos, entrenar modelos de aprendizaje automático y traducir señas del alfabeto dactilológico del Lenguaje de Señas Peruana a texto mediante una cámara web. Algunas de las funciones clave incluyen:

a) **Captura y actualización del conjunto de datos:** Aquí es donde se desarrolló una función para capturar y agregar nuevas imágenes al conjunto de datos. Además, es la encargada de llevar a cabo el conteo de las capturas realizadas por clase.

b) **Actualización de parámetros:** Aquí es donde se desarrollaron funciones para actualizar los parámetros del: conjunto de datos, entrenamiento y Test.

c) **Entrenamiento de modelo:** Aquí es donde se desarrolló una función para entrenar un modelo clasificador de aprendizaje profundo.

4.1.2.2. "Voice_to_signs.py"

Es aquel código que contiene módulos y funciones relacionadas con el reconocimiento de voz y la generación de videos a partir de las palabras detectadas. Estas funciones son responsables de recibir entradas de voz del usuario, procesarlas para extraer palabras clave

y generar videos personalizados basados en estas palabras. Algunas de las funciones destacadas incluyen:

a) **Reconocimiento de voz:** Aquí se desarrolló una función para transcribir entradas de voz del usuario en texto utilizando servicios de reconocimiento de voz como Google Speech Recognition.

b) **Generación de videos:** Aquí se desarrolló una función para crear videos personalizados basados en palabras clave detectadas en las entradas de voz del usuario. Esto implica la selección y concatenación de clips de video predefinidos para formar un video completo que representa las palabras clave.

4.1.2.3. Servidor web Flask: "Principal.py"

Es el código principal de la aplicación implementado en el framework Flask de Python. En este código se definen las rutas y funciones que manejan las solicitudes del cliente. Esto incluye:

a) **Definición de rutas:** Aquí se definieron rutas para diferentes las funcionalidades respecto a los códigos "Signs_to_tex.py" y "Voice_to_signs.py".

b) **Implementación de funciones auxiliares:** Aquí se desarrollaron funciones auxiliares para cada ruta.

c) **Inicio de la aplicación Flask:** Aquí es donde se inicializa la aplicación Flask y se ejecuta en un servidor local para manejar las solicitudes entrantes del cliente.

4.1.2.4. Interacción entre los códigos

La interacción entre los tres códigos se produce a través de solicitudes y respuestas HTTP entre el cliente y el servidor Flask. En

este contexto, el cliente envía solicitudes al servidor para realizar acciones específicas, el cual, a su vez, invoca funciones en los códigos “Signs_to_text.py” y “Voice_to_signs.py” para llevar a cabo estas acciones solicitadas.

4.1.3. Enfoque del sistema

Para el desarrollo del sistema de comunicación bidireccional se clasificó de la siguiente manera:

- En la comunicación sordo a oyente, nos centramos específicamente en las señas estáticas del alfabeto dactilológico del lenguaje de señas peruana (*ver Anexo 8)*, es decir, aquellas letras que se representan sin ningún movimiento de la mano. Esto debido a su simplicidad y eficacia en la detección y reconocimiento de gestos a través de un modelo clasificador de aprendizaje profundo.

- En la comunicación oyente a sordo nos centramos en cada letra del lenguaje del alfabeto dactilológico del lenguaje de señas peruana, incluida las letras que implican movimiento, ya que no es un factor influyente en la precisión del sistema de reconocimiento de voz.

4.1.4. Interfaz web del sistema

Nuestra interfaz web está compuesta de 2 botones principales: “Sordo a oyente” y “Oyente a sordo”, los cuales han sido desarrollados y diseñados en HTML y CSS, tal y como se observa en la Figura 6.

Figura 6

Interfaz web del sistema

Nota. Elaboración propia.

Con respecto al botón “Sordo a oyente”, inicialmente cuando se le da clic, se carga una nueva página llamada “Menu-code1.html”, el cual contiene un menú de navegación de tres botones (Data, Entrenamiento, Test), tal y como se observa en la Figura 7. Asimismo, al hacer clic en cualquiera de los botones del menú de navegación, se carga una nueva página web con funciones del código "Signs_to_text.py" en donde el botón “Test” es donde se realiza la comunicación de persona sorda a oyente.

Figura 7

Interfaz web traductor de señas a texto

Nota. Elaboración propia.

Con respecto al botón "Oyente a sordo", al hacer clic se carga una nueva página llamada "voice-to-signs.html", tal y como se observa en la figura. En esta página es donde se lleva a cabo la comunicación de persona oyente a sorda y está vinculada a funciones del código "Voice_to_signs.py".

Figura 8

Interfaz web traductor de voz a señas

Nota. Elaboración propia.

4.1.5. Código “Signs_to_text.py”

A continuación, se proporcionará una descripción detallada de las variables y funciones que formar parte de este código:

4.1.5.1. Declaración de variables

a) Variables para la clasificación y/o predicción de señas

Se crea la variable “cap” para captura la señal de video continua mediante el uso de la *clase VideoCapture de OpenCV.* El argumento 0 indica que se usa la cámara web predeterminada del sistema.

cap = cv2.VideoCapture(0)

Se crean dos objetos “HandDetector” para detectar las manos en las imágenes de vídeo. El argumento maxHands=1 indica que detectará como máximo una mano en cada cuadro de video.

detector = HandDetector(maxHands=1),
detector2 = HandDetector(maxHands=1)

Se crea un objeto “Classifier” para poder clasificar en tiempo real cada gesto estático de una mano en cada cuadro de video. Este clasificador está basado en un modelo de red neuronal previamente entrenado, que se carga desde el archivo "Modelo/keras_model.h5". Los índices correspondientes a las etiquetas se cargan desde el archivo "Modelo/labels.txt".

Classifier("Modelo/keras_model.h5", "Modelo/labels.txt")

Se crean dos listas vacías: “previous_predictions” para almacenar predicciones anteriores y “predictions_global” para contener todas las predicciones anteriores concatenadas en una única cadena.

previous_predictions = [], predictions_global = []

Se define un intervalo de tiempo en segundos mediante la variable “interval”. Este intervalo se utilizará para controlar la frecuencia en que se muestran las predicciones en cada cuadro de video.

interval = 2

Se inicializa la variable “last_storage_time” con el valor actual del tiempo en segundos. Nos será útil para rastrear cuándo se realizó la última predicción en cada intervalo.

last_storage_time: float = time.time()

Se representa una lista a través de la variable “labels”, la cual contiene las etiquetas que se asignan a las clases en el conjunto de datos.

labels = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y"]

b) Variables para el procesamiento de cuadros de video

Se define mediante la variable “offset”, el desplazamiento que se utilizará en el procesamiento de cada cuadro de video para poder abarcar completamente la región de interés (mano detectada).

offset = 20

Se define mediante la variable “img_size” el tamaño al que se ajustarán las imágenes. Esto asegura uniformidad en el conjunto de datos y compatibilidad con MobileNetV2.

img_size = 300

c) Variables para la gestión de directorios

Se define a través de la variable “folder_data”, la ruta del directorio que contendrá el conjunto de datos (imágenes) utilizados para el entrenamiento y validación de nuestro modelo.

folder_data = 'Data'

Se define a través de la variable “folder_class”, la ruta del directorio que contendrá el conjunto de datos (imágenes) para una clase específica.

folder_class = f'{folder_data}/A'

Se define a través de la variable “folder_train”, la ruta del directorio que contendrá el conjunto de datos (imágenes) para el entrenamiento del modelo.

folder_train = 'Data/Train'

Se define a través de la variable “folder_validation”, la ruta del directorio que contendrá el conjunto de datos (imágenes) para la validación del modelo.

folder_validation = 'Data/Validation'

Se define a través de la variable “folder_modelo”, la ruta del directorio en donde se guardará nuestro modelo de clasificación gestos estáticos (.h5) así como los índices (.txt) asociados a cada una de las salidas del modelo.

folder_modelo = 'Modelo'

Se define a través de la variable “Folder_curves_acurracy”, la ruta donde se guardará la curva de precisión durante el entrenamiento y validación del modelo.

Folder_curves_accuracy = 'Curves (train)/Precision'

Similar a la variable anterior, se define a través de la variable “Folder_curves_loss”, la ruta donde se guardará la curva de pérdida durante el entrenamiento y validación del modelo.

Folder_curves_loss = 'Curves (train)/Loss'

d) Variables para el control y gestión del proceso de captura de cuadros de video

Se define un contador de imágenes capturadas mediante la variable “counter_img”. Se inicializa en 0 y se incrementará cada vez que se captura una nueva imagen.

counter_img = 0

Se define el tamaño total del conjunto de datos para cada clase a través de la variable “dataset_size”. En este caso, se establece en 3000, lo que significa que se utilizarán 3000 muestras en total para cada clase.

dataset_size = 3000

Se define la variable “capture_requested” para indicar que se ha solicitado iniciar el proceso de capturar imágenes. Su estado inicial es 'False'.

capture_requested = False

Se define la variable "create_dataset" para indicar que se ha solicitado iniciar el proceso de creación de cada subdirectorio (clase). Su estado inicial es 'False'.

create_dataset = False

Se define la variable “dataset_class_request” para indicar que se ha solicitado notificar al usuario por medio de un mensaje en cada cuadro de video, indicando que ya puede capturar imágenes para una clase específica. Su estado inicial es 'False'.

dataset_class_request = False

e) Variables para los parámetros de entrenamiento del modelo de aprendizaje profundo

Se define la variable “input_shape” para especificar la forma de entrada que espera el modelo. En este caso, como se utiliza la red neuronal convolucional “MobileNetV2”, se definió una entrada con un tamaño de 224x224 píxeles y tres canales de color (RGB).

input_shape = (224, 224, 3)

Se define la variable “history” y su estado para almacenar el historial del entrenamiento del modelo.

history = None

Se define la variable “num_class” para indicar el número total de clases en el conjunto de datos. Se fijo en 24, debido a que en el alfabeto del lenguaje de señas peruano existen 24 letras con gesto estático.

num_class = 24

Se define la variable “batch_size” para especificar el tamaño del lote utilizado durante el entrenamiento y validación del modelo.

$$batch\_size = 64$$

Se define la variable “epoch” para indicar la cantidad de veces que el modelo pasará por todo el conjunto de datos durante el entrenamiento y la validación.

$$epochs = 20$$

Se define la variable “learning_rate” para especificar la tasa de aprendizaje del modelo.

$$learning\_rate = 0.001$$

Nota: Es necesario mencionar que, para los parámetros de entrenamiento como el tamaño de lote, las épocas y la tasa de aprendizaje, se fijaron los valores que se mencionaron anteriormente, debido a que nuestro modelo de aprendizaje profundo reconocía en tiempo real con más exactitud las letras con gesto estático del alfabeto dactilológico del lenguaje de señas peruana, con las personas sordas de la Asociación De Sordos De La Libertad.

4.1.5.2. Funciones

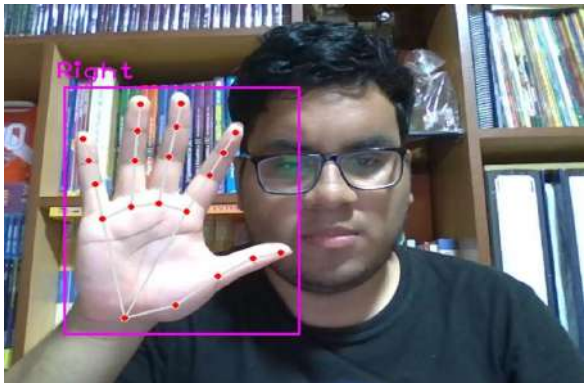
4.1.5.2.1. image_principal ()

a) Descripción

Es aquella función que se llama a través del servidor Flask cuando se carga la página web “Data.html”, es decir cuando previamente se ha dado clic al botón con la etiqueta “Data” de la página web “Menu-code1.html”. Su objetivo es llevar a cabo dos procesos: la transmisión de video con la detección de una de las manos y la creación de carpetas para el conjunto de datos. Asimismo, dependiendo del estado de la variable “dataset_class_request” se verá o no superpuesto un mensaje en la parte superior del mismo. No obstante, por defecto, al iniciar esta función, no se verá superpuesto ningún mensaje, tal y como se muestra en la Figura 9.

Figura 9

Ilustración del uso de la función image_principal() en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Mediante un bucle “While”, se lee constantemente un cuadro de la transmisión de video utilizando el objeto cap, que representa la cámara de entrada. Esta operación devuelve dos valores: “success”, que indica si la lectura fue exitosa, y “frame”, que contiene el cuadro de video leído.

```
while True:
    success, frame = cap.read()
```

Después de leer un cuadro de video, la función verifica si la operación de lectura fue exitosa. En caso contrario, el bucle se interrumpe mediante la instrucción “break”, lo que detiene la transmisión de video.

```
if not success:
    break
```

Si no se interrumpe el bucle, se procede a detectar solo una de las dos manos (izquierda o derecha) en el cuadro de video “frame” mediante “detector.findHands(frame)”. El resultado de esta detección, nos permite obtener dos valores: “hand” y “frame”, el primero contiene la información sobre la mano detectada y el segundo es el cuadro de video modificado con la detección dibujada.

```
else:
    hand, frame = detector.findHands(frame)
```

Se verifica si dataset_class_request es verdadero.

```
if dataset_class_request:
```

Si es verdadero, se llama a la función draw_image_principal(frame) para dibujar un mensaje en la parte superior del cuadro de video.

```
draw_image_principal(frame)
```

Independientemente de si se cumple o no la condición anterior, se codifica el cuadro de video (“frame”) en formato JPEG utilizando OpenCV (cv2). Esta función de codificación devuelve dos valores: “suc”, que indica si la codificación fue exitosa y “encode” que contiene los bytes del cuadro de video codificado.

```
suc, encode = cv2.imencode('.jpg', frame)
```

Se verifica si la codificación fue exitosa. Si no lo fue, el bucle continúa con la próxima iteración utilizando “continue”.

```
if not suc:
    continue
```

Caso contrario, se convierte los bytes del cuadro de video codificado en una secuencia de bytes para poder así enviarlos como parte de un generador de cuadros codificados en JPEG, utilizando “yield”.

```
frame3 = encode.tobytes()
yield (b'--frame\r\n'
       b'Content-Type: image/jpeg\r\n\r\n' + frame3 + b'\r\n')
```

Luego se verifica si el estado de la variable create_dataset es verdadero (True).

```
if create_dataset:
```

Si es verdadero, se verifica entonces si el número de etiquetas (“labels”) es igual al número de clases (“num_class”). Esto se realiza con la finalidad de asegurar la uniformidad en ambos.

```
if len(labels) == num_class:
```

Si las dos variables son iguales, se crea una carpeta para cada etiqueta en la ruta especificada en el directorio “folder_data”.

```
for label in labels:
    if not os.path.exists(os.path.join(folder_data, label)):
        os.makedirs(os.path.join(folder_data, label))
```

Finalmente, la variable “create_dataset” vuelve a su estado inicial (False), lo que evita que se vuelvan a crear las carpetas hasta que se active nuevamente.

```
create_dataset = False
```

4.1.5.2.2. draw_image_principal(frame)

a) Descripción

Es aquella función que recibe como argumento un cuadro de video (“frame”) de la función “image_principal()”. Su objetivo es agregar un mensaje en la parte superior del mismo con la finalidad de indicar al usuario que el sistema se encuentra listo para comenzar el proceso de capturar y guardar cuadros de video como imágenes mediante la letra S, tal y como se observa en la Figura 10.

Figura 10

Ilustración del uso de la función draw_image_principal(frame) en Python

Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado esta función:

Se define una variable llamada text, la cual almacena el mensaje: "¿Estas listo? Presiona la letra S".

```
text = 'Estas listo? Presiona la letra S'
```

Además, se establece la variable "font" con el valor "Simplex" para el tipo de fuente y la variable "color" con la tupla (0, 255, 0), que representa el color verde en el formato RGB.

```
font = cv2.FONT_HERSHEY_SIMPLEX
color = (0, 255, 0)
```

Para posicionar el texto en el cuadro de imagen, se realiza un cálculo para determinar el tamaño del texto utilizando la función cv2.getTextSize(), la cual devuelve las dimensiones del texto en píxeles.

(text_width, text_height), baseline = cv2.getTextSize(text, font, 1.2, 3)

Luego, se calcula la posición horizontal del texto (text_x) para centralizarlo en "frame", considerando su ancho. Para la posición vertical del texto (text_y), se define un valor que establece la distancia desde la parte superior del "frame".

text_x = (frame.shape[1] - text_width) // 2
text_y = frame.shape[0] - 400

Para resaltar el texto, se dibuja un rectángulo blanco debajo del área donde se lo colocará utilizando la función cv2.rectangle(), en donde se especifica su posición y el tamaño a través de un cálculo en el que se tiene en cuenta la posición y el tamaño del texto calculado.

cv2.rectangle(frame, (text_x, text_y - text_height), (text_x + text_width, text_y + baseline), (255, 255, 255), cv2.FILLED)

Finalmente, el mensaje de texto se agrega a "frame" utilizando la función cv2.putText(), en donde se especifica su posición utilizando las coordenadas (text_x, text_y) y las propiedades de texto como: el color, tamaño, grosor de línea y el tipo de línea.

cv2.putText(frame, text, (text_x, text_y), font, 1.2, color, 3, cv2.LINE_AA)

4.1.5.2.3. image_white()

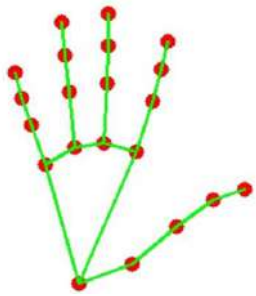
a) Descripción

Es aquella función que al igual que la función “image_principal()”, se llama a través del servidor Flask cuando

se carga la página web “Data.html”, es decir cuando previamente se ha dado clic en el botón con la etiqueta “Data” de la página web “Menu-code1.html”. Su principal objetivo es obtener un cuadro de video procesado (“img_white”), en el cual se represente a la mano detectada como los 21 puntos de referencia, tal y como se muestra en la Figura 11. Asimismo, dependiendo del estado de la variable “capture_requested, la función puede comenzar o detener el proceso de capturar y guardar cada cuadro de video procesado como imagen en un directorio específico.

Figura 11

Ilustración del uso de la función image_white() en Python



Nota. Elaboración propia.

Es importante destacar que el procesamiento que se obtiene en esta función sobre la mano detectada, se basa en los excelentes resultados logrados por los investigadores Montenegro y Villa (2019) en la proporción de predicciones correctas realizadas por su modelo de aprendizaje profundo sobre el total de instancias de su conjunto de datos (“accuracy”), alcanzando un total del 91%. Este éxito se atribuye al hecho de que en su investigación lograron aislar la mano en cada cuadro de video utilizando un guante de color verde, lo cual permite que su modelo se enfoque en las características relevantes para el reconocimiento de gestos.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Mediante un bucle principal (“while”), esta función se encarga de capturar y procesar cuadros de video constantemente.

while True:

Dentro de este bucle principal, existen otros dos bucles. El primero se ejecuta cuando capture_requested es “False”.

while not capture_requested:

Asimismo dentro de este bucle, se lee un cuadro con cap.read().

success, frame = cap.read()

Si la lectura del fotograma es exitosa (success es True), se llama a la función “process_frame_white(frame)” para obtener un cuadro de video procesado (“img_white”), en el cual se represente a la mano detectada como los 21 puntos de referencia junto con sus conexiones.

img_white = process_frame_white(frame)

Si “img_white” no es None, se codifica el cuadro en formato JPEG usando cv2.imencode('.jpg', img_white).

```
if img_white is not None:
    suc, encode = cv2.imencode('.jpg', img_white)
```

Si la codificación (“suc”) fue exitosa, se convierte los bytes del cuadro codificado (“encode”) en una secuencia de bytes para

poder así enviarlos como parte de un generador de cuadros codificados en JPEG, utilizando “yield”.

```
frame2_1 = encode.tobytes()
yield (b'--frame\r\n'
       b'Content-Type: image/jpeg\r\n\r\n' +
       frame2_1 + b'\r\n')
```

Ahora, si el estado de la variable “capture_requested” cambia a True, se rompe el primer bucle y se restablece el contador de imágenes (counter_img) a 0 para iniciar la captura del conjunto de datos mediante el segundo bucle.

```
counter_img = 0
```

El segundo bucle se ejecuta siempre y cuando “counter_img” sea menor que “dataset_size” (volumen de datos por clase), ya que lo que se requiere, es que “counter_img” al ser un contador de imágenes, deberá ir aumentando durante este bucle hasta alcanzar al tamaño o volumen prefijado (“dataset_size”). De esta manera también podremos romper este bucle cuando no se necesite capturar imágenes.

```
while counter_img < dataset_size:
```

Dentro de este bucle, se realiza el mismo proceso que se realizó en el primer bucle, solo que a diferencia de este, aquí cada cuadro procesado se guarda en el directorio “folder_class” con un nombre único basado en el tiempo (time.time()).

```
cv2.imwrite(f'{folder_class}/{time.time()}.jpg', img_white)
```

Finalmente, cuando se han capturado todas las imágenes necesarias de una clase (“counter_img” alcanza “dataset_size”),

se restablece la variable “capture_requested” a False, indicando que la captura ha finalizado. Dado que la función se encuentra en un bucle continuo y la variable “capture_requested” es False, el primer bucle interno se ejecuta nuevamente.

capture_requested = False

4.1.5.2.4. process_frame_white(frame)

a) Descripción

Es aquella función que tiene como objetivo obtener un cuadro de video ("frame") de la función "img_white()" y realizar las siguientes acciones: primero, se encarga de realizar una copia del cuadro de video “frame”. Luego, intenta detectar una mano, si logra detectar con éxito una mano procede a extraer las coordenadas y el tamaño del cuadro delimitador que rodea a la mano. Posteriormente estos datos, así como la copia de “frame” se pasan como argumentos a la función "hand_points", el cual retorna un cuadro de video procesado (“img_white”) que representa a la mano detectada como los 21 puntos de referencia. Finalmente, este cuadro de video procesado se retorna también desde esta función.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se realiza una copia del cuadro de video original (“frame”) a través de “np.copy()” de NumPy.

frame_copy = np.copy(frame)

Se utiliza “detector.findHands(frame)” para detectar una de las dos manos en el cuadro de video “frame”. El resultado de esta detección se almacena en la variable “hands”.

hands, _ = detector.findHands(frame)

Luego se verifica si hay efectivamente una mano detectada en el cuadro de video “frame”.

if hands:

Si es así, la función toma la primera mano detectada y la asigna a la variable “hand”. Luego, extrae las coordenadas y el tamaño del cuadro delimitador de la mano detectada.

hand = hands[0]:
x, y, w, h = hand['bbox']

Después se llama a la función “hand_points*(frame_copy, x, y, w, h)* con los parámetros del cuadro de video (“frame_copy”) y las coordenadas y tamaño del cuadro delimitador de la mano (x, y, w, h), con la finalidad de obtener un cuadro de video procesado (“img_white”), que represente a la mano detectada como los 21 puntos de referencia junto con sus conexiones.

img_white = hand_points(frame_copy, x, y, w, h)

Finalmente se retorna la variable “img_white”.

return img_white

4.1.5.2.5. def hand_points(frame_copy, x, y, w, h)

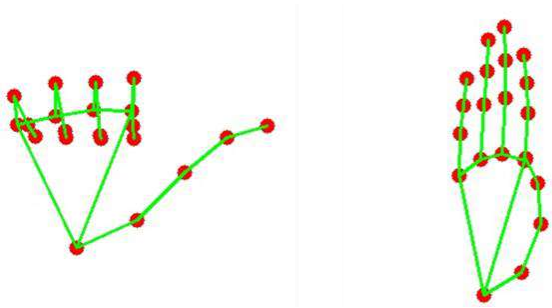
a) Descripción

Es aquella función que recibe como argumentos desde la función “process_frame_white(frame)” una copia del cuadro de video original (“frame_copy”) junto con las coordenadas (x, y) y el tamaño (w, h) del cuadro delimitador de la mano detectada. Su objetivo es procesar dicha región para generar una representación visual de la mano, utilizando los 21 puntos de referencia y sus conexiones.

Para asegurar la efectividad de nuestro modelo de aprendizaje profundo “MobileNetV2”, necesitamos que todas las imágenes que forman parte del conjunto de datos de entrenamiento y validación tengan un tamaño fijo de 300x300 pixeles, sin embargo, cuando se detecta la mano y se extrae la región que lo rodea (cuadro delimitador), esta puede tener diferentes proporciones entre anchura y altura, dependiendo del gesto que se realice, por ejemplo: cuando se realiza la letra A en lenguaje de señas, el ancho de la región es más grande que su alto y lo contrario pasa con la letra B en la que la que tiene un alto más grande que su ancho. Es por ello que mediante esta función podemos ajustar automáticamente el tamaño de la región de interés a un tamaño fijo de una imagen en blanco (300x300 pixeles) manteniendo esa misma proporción entre el ancho y el alto, tal y como se observa en la Figura 12. Al mantener esta proporción, nos aseguramos que las imágenes resultantes conserven toda la información relevante de la mano, lo cual es crucial para un correcto entrenamiento del modelo de aprendizaje profundo.

Figura 12

Ilustración del uso de la función hand_points(frame_copy, x, y, w, h) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se crea un cuadro en blanco de tamaño “img_size x img_size” con tres canales (RGB) y todos los píxeles inicializados en 255, lo que representa el color blanco.

*img_white = np.ones((img_size, img_size, 3), np.uint8) * 255*

Luego, se recorta una región del cuadro de video “frame_copy” alrededor del rectángulo que delimita la mano. Esto se lleva a cabo utilizando las coordenadas y el tamaño del cuadro delimitador, así como el uso de la variable 'offset' para ampliar ligeramente el área recortada.

img_crop = frame_copy[y - offset:y + h + offset, x - offset:x + w + offset]

Se verifica si el tamaño de la región recortada “img_crop” es mayor a cero, esto se hace con la finalidad de asegurarse que el

recorte se haya realizado correctamente y que la región de interés (mano) esté presente en la imagen original.

if img_crop.size > 0:

Si está vacío o no tiene píxeles, la función devuelve None.

else:
return None

Caso contrario, se calcula el “aspect_ratio” de la región de interés, el cual es la relación entre el alto y ancho del cuadro delimitador de la mano detectada.

aspect_ratio = h / w

Se verifica si el “aspect_ratio” es mayor que 1.

if aspect_ratio > 1:

Si “aspecto_ratio” es mayor que 1, lo cual significa que el cuadro delimitador de la mano detectada es más alto que ancho, se calcula un factor de escalamiento (“scaling_factor”) basado en el tamaño deseado (“img_size”) y la altura de la región.

scaling_factor = img_size / h

A partir de este factor de escalamiento se calcula el nuevo ancho de la región recortada manteniendo la proporción de aspecto.

*w_calculated = math.ceil(scaling_factor * w)*

Luego se redimensiona la región recortada (“img_crop”) al tamaño deseado (*w_calculated*, “img_resized”) y se calcula un

desplazamiento horizontal (“w_gap”) con la finalidad de poder centrar el cuadro final redimensionado en “img_white”.

```
img_resized = cv2.resize(img_crop, (w_calculated, img_size))
w_gap = math.ceil((img_size - w_calculated) / 2)
```

Posteriormente se procede a detectar una de las dos manos (izquierda o derecha) en la región redimensionada (“img_resized”) con “detector2.findHands(img_resized)”. El resultado de esta detección se almacena en la variable “hand”.

```
hand, _ = detector2.findHands(img_resized)
```

A partir de “hand” podemos verificar si se ha detectado correctamente una mano.

```
if hand:
```

Si se ha detectado correctamente una mano, se procede a almacenarla en la variable “hand_selected”. Para luego extraer a partir de ella, una lista (“lmList”) con las coordenadas de los 21 puntos de referencia (“hand_landmarks”).

```
    hand_selected = hand[0]
    hand_landmarks = hand_selected['lmList']
```

Asimismo, se crea un cuadro en blanco, del mismo tamaño que la región redimensionada.

```
img_resized_point = np.ones_like(img_resized) * 255
```

Posteriormente a través de la función “draw_landmarks_and_connections (img_resized_point,

hand_landmarks)” se dibuja los puntos de referencia juntos con sus conexiones en el cuadro de video “img_resized_point”.

```
draw_landmarks_and_connections(img_resized_point,
                               hand_landmarks)
```

Finalmente, se superpone y se centra esta imagen redimensionada “img_resized_point” en “img_white”, teniendo cuenta el ancho y el desplazamiento calculado anteriormente.

img_white[:, w_gap:w_calculated + w_gap] = img_resized_point

Ahora si “aspect_ratio” es menor a 1, lo que indica que el cuadro delimitador de la mano detectada es más ancho que alto, se realiza el mismo proceso que se explicó cuando era mayor a 1, no obstante, a diferencia de este, se calcula la nueva altura ('h_calculated') para poder redimensionar la imagen recortada ('img_resized') y determinar el desplazamiento vertical (“h_gap”).

```
scaling_factor = img_size / w
h_calculated = math.ceil(scaling_factor * h)
img_resized = cv2.resize(img_crop, (img_size,
                                    h_calculated))
h_gap = math.ceil((img_size - h_calculated) / 2)
```

4.1.5.2.6. draw_landmarks_and_connections(img_resized_point, hand_landmarks)

a) Descripción

Es aquella función que recibe como argumentos desde la función “hand_points(frame_copy, x, y, w, h)” un cuadro de video redimensionada en blanco (“img_resized_point”) y una lista de coordenadas de los puntos de referencia de la mano detectada

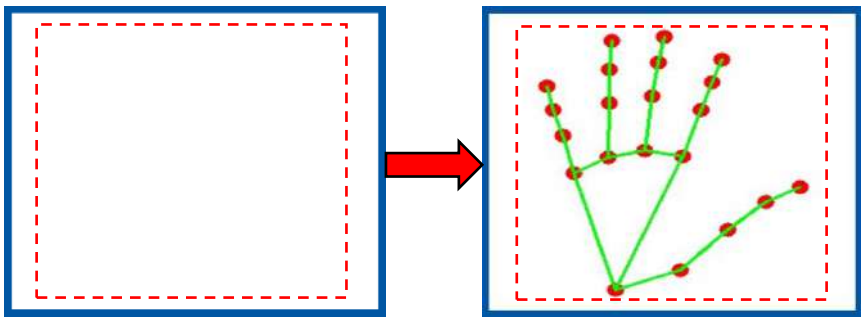
(hand_landmarks). Su objetivo es dibujar los puntos y conexiones de una mano, tal y como se muestra en la Figura 13.

Figura 13

Ilustración del uso de la función

draw_landmarks_and_connections(img_resized_point,

hand_landmarks) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Primero, se itera sobre cada punto de referencia en la lista “hand_landmarks”, en donde se extraen las coordenadas x e y. Asimismo utilizando estas coordenadas se dibuja un círculo relleno de color rojo para cada punto de referencia en el cuadro “img_resized_point”.

```
for point in hand_landmarks:
    x_lm, y_lm = point[:2]
    cv2.circle(img_resized_point, (int(x_lm), int(y_lm)),
    7, (0, 0, 255), -1)
```

Luego, se define la lista de conexiones mediante la variable “connections”, la cual contiene los pares de puntos que deber ir conectados para representar una mano mediante el trazado de líneas.

```
connections = [[0, 1], [1, 2], [2, 3], [3, 4], [0, 5], [5, 9], [9, 13],
               [13, 17], [0, 17], [17, 18], [18, 19], [19, 20], [13,
               14], [14, 15], [15, 16], [9, 10], [10, 11], [11, 12], [5,
               6], [6, 7], [7, 8]]
```

Después, se itera sobre cada conexión en la lista “connections”. Para cada conexión, se obtienen los índices de los puntos de inicio y final. Estos índices se utilizan para extraer las coordenadas x e y de los puntos conectados.

```
for connection in connections:
    start, end = connection
    x_c1, y_c1 = hand_landmarks[start][:2]
    x_c2, y_c2 = hand_landmarks[end][:2]
```

Finalmente, se dibuja en el cuadro de video “img_resized_point”, una línea entre cada par de puntos conectados utilizando la función cv2.line(). La línea se dibuja en color verde y con un grosor de 2 píxeles.

```
cv2.line(img_resized_point, (int(x_c1), int(y_c1)), (int(x_c2),
                int(y_c2)), (0, 255, 0), 2)
```

4.1.5.2.7. cap_photo()

a) Descripción

Es aquella función que se llama a través del servidor Flask (“principal.py”) después de que se ha llamado previamente a la función “reset_counter()” y se haya presionado la tecla 's' en el

teclado mientras se navega en la página web “Data.html”. Su objetivo es iniciar el proceso de captura de imágenes y retornar su totalidad cuando se alcance el tamaño deseado, tal y como se muestra en la Figura 14.

Figura 14

Ilustración del uso de la función cap_photo() en Python

Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se inicia estableciendo la variable “capture_requested” en True. Esto hace que se inicie el proceso de captura y almacenamiento de cada cuadro de video como imágenes de la función "image_white ()”.

capture_requested = True

Mediante un bucle "while", la función seguirá ejecutándose hasta que se hayan capturado todas las imágenes requeridas de una clase específica, es decir cuando el valor de “counter_img” sea igual que “*dataset_size”*. Dentro del bucle, se establece un tiempo de espera de 25 ms en cada iteración. Este tiempo de

espera se debe a la necesidad de sincronizarse con el proceso de captura y almacenamiento de cada cuadro de video como imágenes de la función "image_white ()", el cual se fijó también en 25 ms.

```
while counter_img < dataset_size:
    cv2.waitKey(25)
```

Finalmente, cuando "counter_img" sea igual en valor que "dataset_size", la función saldrá del bucle y devolverá la variable global "counter_img" con su valor actualizado.

```
return counter_img
```

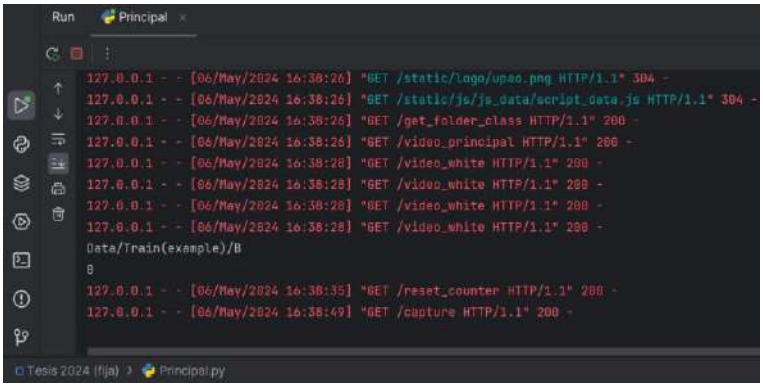
4.1.5.2.8. reset_counter()

a) Descripción

Es aquella función que se llama a través del servidor Flask (“principal.py”), cuando se presiona la tecla 's' en el teclado mientras se navega en la página web “Data.html”. Su objetivo es restablecer el contador de imágenes a cero cuando queremos volver a ejecutar el proceso de captura y almacenamiento de imágenes para otra clase específica, tal y como se muestra en la Figura 15.

Figura 15

Ilustración del uso de la función reset_counter() en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se asigna el valor cero a la variable global “counter_img”, lo que implica que se reinicia el contador de imágenes.

counter_img = 0

Finalmente, la función devuelve esta variable global, pero con su valor actualizado

return counter_img

4.1.5.2.9. update_dataset_file(new_labels, new_num_class, new_folder_data)

a) Descripción

Es aquella función que se llama a través del servidor Flask (“principal.py”), cuando se hace uso del formulario “Carpeta y Subcarpetas” en la página web “Data.html”. Su objetivo es actualizar en el sistema los parámetros del conjunto de datos

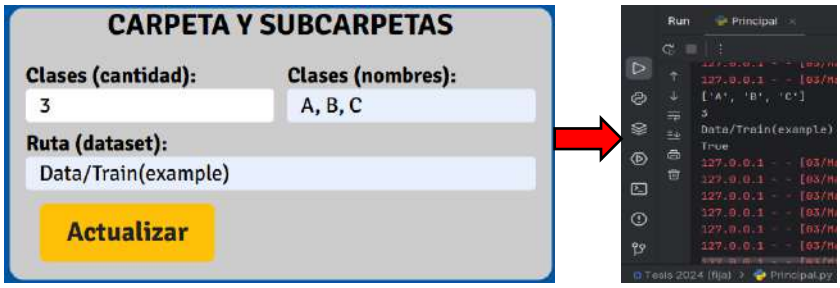
como: el número de clases, las etiquetas y su ruta. Todo ello, a partir de los valores (argumentos de entrada) obtenidos del formulario, tal y como se muestra en la Figura 16. Además, actúa como una señal para iniciar el proceso de generar una subcarpeta por cada clase de la función “image_principal()”.

Figura 16

Ilustración del uso de la función

update_dataset_file(new_labels, new_num_class,

new_folder_data) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se verifica si todos los valores de los argumentos de entrada son proporcionados.

if new_labels and new_num_class and “new_folder_data”:

Si es que al menos uno de ellos no ha sido proporcionado, la función devuelve “None”.

else:

return None

Caso contrario, entonces se actualizan las variables globales: “labels”, “num_class” y “folder_data”, con los valores proporcionados por los argumentos de entrada. Es importante señalar que ya se han definido valores para cada una de estas variables globales, tal y como se mencionaron en el apartado **“4.1.5.1. Declaración de variables”**. Por lo que, si se quiere cambiar alguno de estos valores, se tendrá que hacer uso de este formulario.

labels = new_labels
num_class = new_num_class
folder_data = new_folder_data

Luego, se establece la variable global “create_dataset” en True, lo cual repercute en la función “image_principal()”, debido a que se inicia el proceso de creación de cada subcarpeta del conjunto de datos.

create_dataset = True

Finalmente, la función devuelve las variables globales actualizadas como resultado.

return labels, create_dataset, num_class, folder_data

4.1.5.2.10. update_dataset_size(new_dataset_size)

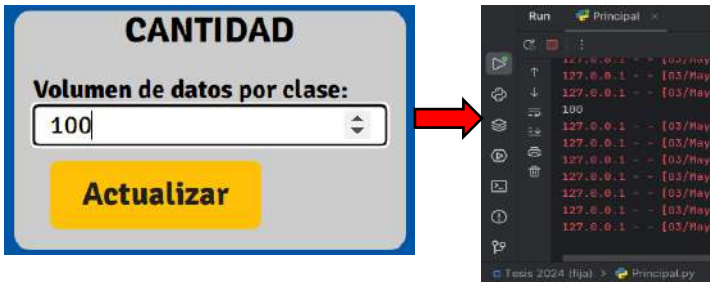
a) Descripción

Es aquella función que se llama a través del servidor Flask, cuando se hace uso del formulario “Cantidad” en la página web “Data.html”. Su objetivo es actualizar en el sistema, el volumen de datos (imágenes) por clase, a partir del valor que se obtiene en el formulario (argumento de entrada), tal y como se muestra en la Figura 17.

Figura 17

Ilustración del uso de la función

update_dataset_size(new_dataset_size) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Esta línea verifica si “new_dataset_size” tiene algún valor.

```
if new_dataset_size:
```

Si está vacía, la función devuelve “None”.

```
else:
    return None
```

Caso contrario, se actualiza la variable global “dataset_size” con el valor proporcionado por el argumento “new_dataset_size”.

```
dataset_size = new_dataset_size
```

Finalmente, la función devuelve esta variable global actualizada como resultado.

```
return dataset_size
```

Es importante señalar que se ha definido por defecto que el valor de la variable “dataset_size” sea 3000, tal y como se mencionó en el apartado **“4.1.5.1. Declaración de variables”**. Es por ello que, si se requiere un volumen de datos por clase mayor o menor a 3000, se tendrá que hacer uso de este formulario.

4.1.5.2.11. update_folder_class(new_folder_class)

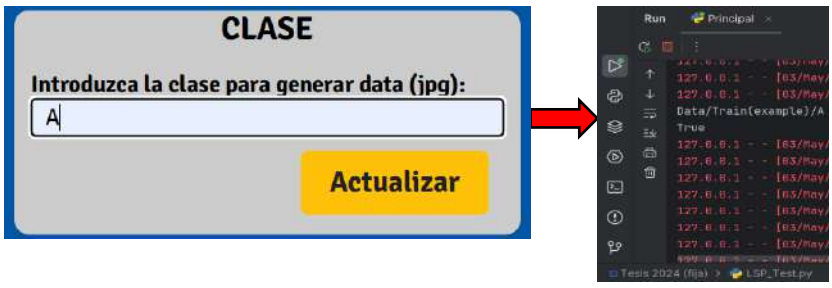
a) Descripción

Es aquella función que se llama a través del servidor Flask, cuando se hace uso del formulario “Cantidad” en la página web “Data.html”. Su objetivo es actualizar en el sistema, la ruta de la carpeta de una clase específica a partir del valor (argumento de entrada) que se obtiene del formulario, tal y como se muestra en la Figura 18. Además, actúa como una señal para invocar a la función “draw_image_principal(frame)” dentro de la función “image_principal()”.

Figura 18

Ilustración del uso de la función

update_folder_class(new_folder_class) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se verifica si el argumento de entrada (“new_folder_class”) tiene algún valor.

if new_folder_class:

Si está vacía, la función devuelve None.

else:
    return None

Caso contrario, se actualiza la variable global (“folder_class”), concatenándolo el nombre de la carpeta de la clase específica (“new_folder_class”) a la ruta donde se almacena todo el conjunto de datos (“folder_data”).

folder_class = f'{folder_data} /{new_folder_class}'

También se establece la variable global dataset_class_request en True, lo cual repercute en la función “image_principal()”, ya que permite que se muestra el mensaje: “¿Estas listo? Presiona la letra S”.

dataset_class_request = True

Finalmente, la función devuelve estas variables globales actualizadas como resultado.

return folder_class, dataset_class_request

Es importante señalar que se ha establecido por defecto que la variable global “folder_class” sea la ruta de la carpeta de la clase correspondiente a la letra "A", tal y como se mencionó en el apartado **“4.1.5.1. Declaración de variables”**. Es por ello que, si se desea utilizar otra clase para el proceso de captura y

almacenamiento de imágenes será necesario actualizarla mediante este formulario.

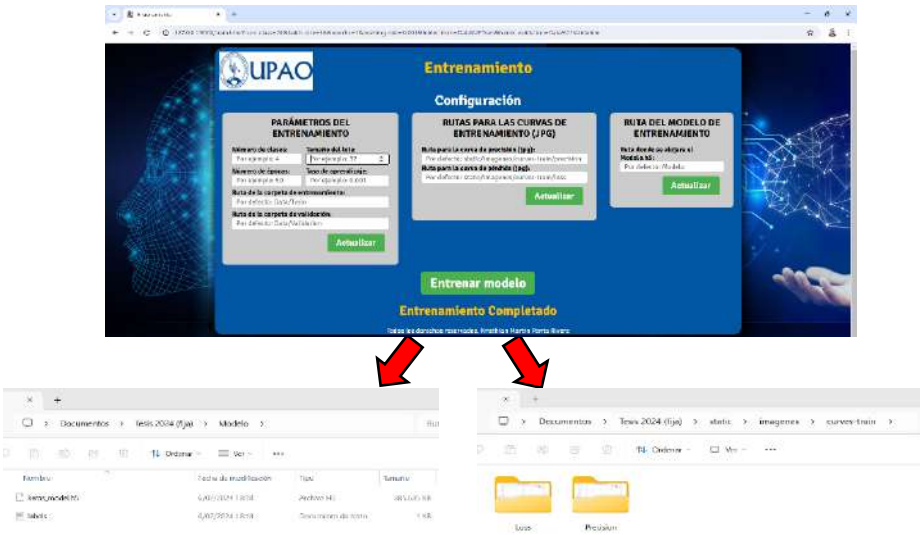
4.1.5.2.12. Train()

a) Descripción

Es aquella función que se llama a través del servidor Flask, cuando se hace clic en el botón con la etiqueta “Entrenar modelo” de la página web “Train.html”. Su objetivo es entrenar un modelo de aprendizaje profundo para que pueda reconocer y clasificar correctamente las señas o gestos estáticos del alfabeto dactilológico del lenguaje de señas peruano. Una vez finalizado el entrenamiento, la función guardará el modelo (.h5) en un directorio local específico, así como las imágenes de las curvas de precisión y pérdida generadas durante el entrenamiento y la validación del modelo, tal y como se muestra en la Figura 19.

Figura 19

Ilustración del uso de la función Train()



Nota. Elaboración propia.

Con respecto al entrenamiento, se emplea la técnica de transferencia de aprendizaje utilizando la arquitectura

preentrenada de MobileNetV2. Este enfoque permite aprovechar el conocimiento adquirido por MobileNetV2 en un conjunto de grandes datos como “ImageNet”, y aplicarlo a nuestro problema de clasificación.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se descarga la arquitectura de MobileNetV2 con los pesos preentrenados en ImageNet y se congela para que no se modifiquen durante el entrenamiento.

```
base_model = MobileNetV2(input_shape=input_shape,
                         include_top=False,
                         weights='imagenet')
for layer in base_model.layers:
    layer.trainable = False
```

Se crea un modelo secuencial que consiste en la base de MobileNetV2 seguida de capas personalizadas.

```
model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(num_class, activation='softmax')
])
```

En este modelo:

- La base de MobileNetV2 nos sirve como extractor de características.

- La capa de aplanamiento (Flatten()) nos sirve para convertir las características en un vector unidimensional.

- Las dos últimas capas densas (Dense()) nos sirve para la clasificación. La última capa tiene neuronas igual al número de clases con una función de activación softmax.

Luego se compila el modelo personalizado especificando el optimizador, la función de pérdida y las métricas a monitorear durante el entrenamiento. En este caso se ha definido el optimizador Adam cuya tasa de aprendizaje puede ser modificable a través de un formulario desde la página web “train.html”.

```
model.compile(optimizer=tf.keras.optimizers.Adam(
                learning_rate=learning_rate),
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Se crean generadores de datos utilizando “ImageDataGenerator” para preprocesar las imágenes de la carpeta del conjunto de datos de entrenamiento y de validación.

```
train_datagen=ImageDataGenerator(rescale=1. / 255,
                                 rotation_range=20,
                                 horizontal_flip=True,
                                 vertical_flip=False)

validation_datagen=ImageDataGenerator(rescale=1. / 255,
                                      rotation_range=20,
                                      horizontal_flip=True,
                                      vertical_flip=False)
```

En ella se aplican transformaciones como el reescalado, rotación y volteo horizontal a las imágenes; los cuales se detallan a continuación:

- Con lo que respecta al reescalado, se ha definido con un valor de 1. /255. Esto significa que cada pixel de una imagen se dividide por 255, lo que resulta que cada uno de ellos estén en el rango de [0,1]. Con esto aseguramos que los datos de entrada estén en una escala apropiada para el entrenamiento, lo que ayudará a mejorar la velocidad de convergencia y la estabilidad del modelo.

- Con lo que respecta a la rotación, se ha definido un valor de 20, esto significa que cada imagen se rotará 20 grados en cualquier dirección. Con esto aseguramos que nuestro modelo pueda generalizar mejor las señas estáticas del alfabeto dactilológico del lenguaje de señas peruana a diferentes orientaciones.

- Con lo que respecta al volteo horizontal, para el propósito de la investigación se le aplicó para que cualquier persona sorda pueda hacer señas con cualquiera de las dos manos (derecha o izquierda) y pueda ser detectada con la misma precisión.

Luego, a través de los métodos “flow_from_directory” se cargan y se preprocesan las imágenes de los directorios “folder_train” y “folder_validation” con las transformaciones de los generadores de datos mencionados anteriormente, en donde se especifican parámetros como el tamaño de las imágenes de entrada (“target_size”), el tamaño de lote (“batch_size”) y el modo de clasificación (“class_mode”).

train_generator=train_datagen.flow_from_directory(folder_train, target_size=input_shape[:2], batch_size=batch_size, class_mode='categorical')

validation_generator=validation_datagen.flow_from_directory(folder_validation, target_size=input_shape[:2], batch_size=batch_size, class_mode='categorical')

Una vez que el modelo y los generadores de datos están configurados, se procede a entrenar el modelo. El modelo se entrena utilizando el método fit de Keras, donde se especifica el generador de datos de entrenamiento y validación, así como el número de épocas.

history = model.fit(train_generator, epochs=epochs, validation_data=validation_generator)

Una vez finalizado el entrenamiento, se obtienen los índices de las clases del generador de datos de entrenamiento para después guardarlos en un archivo de texto (.txt) en el mismo directorio donde se guardará el modelo.

```
Class_names = list(train_generator.class_indices.keys())
with open(f'{folder_model}/labels.txt', 'w') as file:
    for class_name in class_names:
        file.write(f"{class_name}\n")
```

Luego de que se han guardado los índices de las clases, se procede a guardar el modelo en la ruta especificada en la variable “folder_modelo”, con el nombre “keras_model.h5”.

Model.save(f’{folder_modelo}/keras_model.h5’)

Después de haber guardado el modelo, se generan las rutas de los directorios donde se almacenarán las imágenes de curvas de precisión y pérdida, las cuales serán generadas a partir del entrenamiento y la validación del modelo.

```
accuracy_image_path=f'{Folder_curves_accuracy}/
                        accuracy_{time.time()}.png'

loss_image_path=f'{Folder_curves_loss}/
                        loss_{time.time()}.png'
```

Primero, se genera la curva de precisión del modelo y se guarda como una imagen en su respectivo directorio local (“*accuracy_image_path”)*.

```
plt.figure(1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.4, ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.title('Curva de precisión')
plt.savefig(accuracy_image_path)
```

Luego, se genera la curva de pérdida del modelo durante el entrenamiento y validación, y se guarda como una imagen en su respectivo directorio local (“*loss_image_path”)*.

```
plt.figure(2)
plt.plot(history.history['loss'])
```

```
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.title('Curva de pérdida')
plt.savefig(loss_image_path)
```

Después de generar las imágenes de las curvas, estas se convierten al formato “base64”. Esta codificación convierte los datos binarios de las imágenes en cadenas de texto. Esta técnica resulta útil, ya que podremos enviar estas imágenes a la página web “train.html” como parte de una solicitud HTTP.

```
with open(accuracy_image_path, 'rb') as f:
    accuracy_image = base64.b64encode(f.read()).
                                        decode('utf-8')
plt.close()
with open(loss_image_path, 'rb') as f:
    loss_image = base64.b64encode(f.read()).decode('utf-8')
plt.close()
```

Finalmente, la función retorna estas imágenes a través de las variables “accuracy_image” y “loss_image”.

```
return accuracy_image, loss_image
```

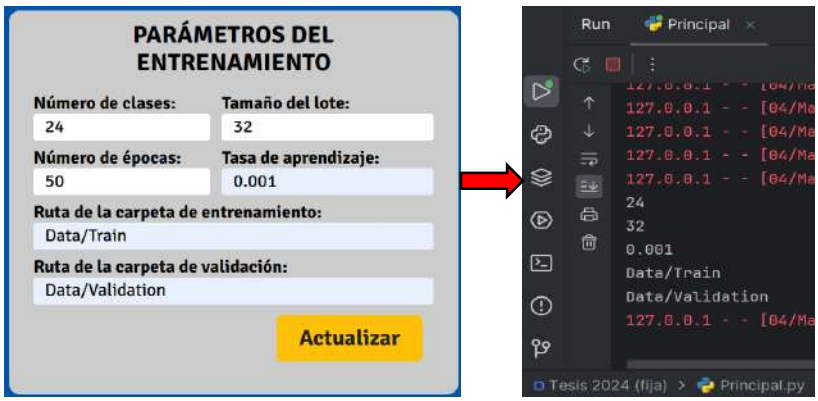
4.1.5.2.13. parameters_train(new_num_class, new_batch_size, new_epochs, new_learning_rate, new_folder_train, new_folder_validation)

a) Descripción

Es aquella función que se llama a través del servidor Flask, cuando se hace uso del formulario “Parámetros del Entrenamiento” en la página web “Train.html”. Su objetivo es actualizar en el sistema, los parámetros clave del proceso de entrenamiento del modelo como: el número de clases (“num_class”), el tamaño del lote (“batch_size”), el número de épocas (“epochs”), la tasa de aprendizaje (“new_learning_rate”), la ruta de la carpeta donde se aloja la data del entrenamiento (“folder_train”) y la ruta de la carpeta donde se aloja la data de validación (“folder_validation”). Todo ello, a partir de los valores obtenidos del formulario (argumentos de entrada), tal y como se muestra en la Figura 20.

Figura 20

Ilustración del uso de la función parameters_train(new_num_class, new_batch_size, new_epochs, new_learning_rate, new_folder_train, new_folder_validation) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se verifica si todos los valores de los argumentos de entrada son proporcionados.

if (new_num_class and new_batch_size and new_epochs new_learning_rate and new_folder_train and new_folder_validation):

Si es que al menos uno de estos está vacío, la función devuelve “None”

```
else:
    return None
```

Caso contrario, las variables globales del proceso de entrenamiento del modelo se actualizan con estos valores proporcionados.

```
num_class = new_num_class
batch_size = new_batch_size
epochs = new_epochs
learning_rate = new_learning_rate
folder_train = new_folder_train
folder_validation = new_folder_validation
```

Finalmente, la función devuelve estas variables globales actualizadas como resultado

return num_class, batch_size, epochs, folder_train, folder_validation

Es importante resaltar que los valores de estas variables globales ya están definidos por defecto en el código, tal y como se mencionó en el apartado **“4.1.5.1. Declaración de variables”**. Por lo tanto, solo cuando se desea cambiar estos valores prefijados, se hará uso de este formulario.

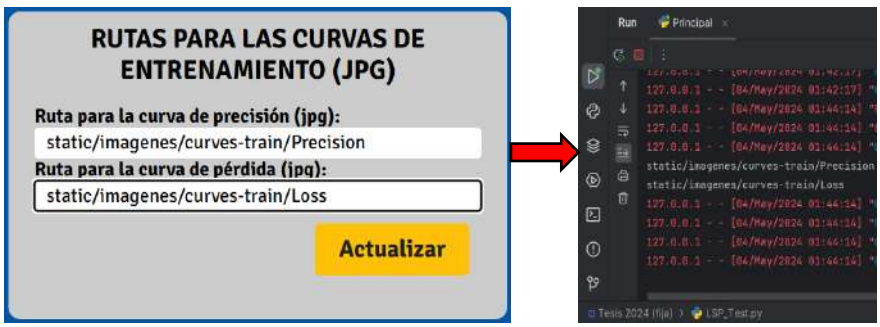
4.1.5.2.14. update_folder_curves(new_folder_acurracy, new_folder_loss)

a) Descripción

Es aquella función que se llama a través del servidor Flask (“principal.py”) cuando se hace uso del formulario “Rutas para las curvas de entrenamiento” en la página web “Train.html”. Su objetivo es actualizar en el sistema, las rutas de las carpetas donde se guardarán las imágenes de las curvas de precisión y pérdida durante el entrenamiento y validación del modelo, a partir de los valores (argumentos de entrada) obtenidos del formulario, tal y como se muestra en la Figura 21.

Figura 21

Ilustración del uso de la función update_foler_curves(new_folder_acurracy, new_folder_loss) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se verifica si todos los valores de los argumentos de entrada son proporcionados (“new_folder_accuracy”, “new_folder_loss”).

if new_folder_accuracy and new_folder_loss:

Si es que al menos uno de estos está vacío, la función devuelve “None”.

else:
    return None

Caso contrario, se actualizan las variables globales que almacenan las rutas de las curvas con los valores proporcionados.

Folder_curves_accuracy = new_folder_accuracy
Folder_curves_loss = new_folder_loss

Finalmente, la función devuelve estas variables globales actualizadas como resultado.

return Folder_curves_accuracy, Folder_curves_loss

Es importante resaltar que estas variables ya tienen definidas por defecto las rutas hacia unos directorios designados para alojar estas curvas, tal y como se mencionó en el apartado **“4.1.5.1. Declaración de variables”**. Por lo tanto, solo cuando se desea cambiar estas rutas, se hará uso de este formulario.

4.1.5.2.15. update_folder_modelo(new_folder_modelo)

a) Descripción

Es aquella función que se llama a través del servidor Flask (“Principal.py”), cuando se hace uso del formulario “Ruta del modelo de entrenamiento” en la página web “Train.html”. Su objetivo es actualizar en el sistema, la ruta de la carpeta donde se guardará el modelo entrenado, a partir del valor (argumento de entrada) obtenido del formulario, tal y como se observa en la Figura 22.

Figura 22

Ilustración del uso de la función

update_folder_modelo(new_folder_modelo) en Python

Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se verifica si se ha proporcionado algún valor (ruta) para el argumento de entrada “new_folder_modelo”.

if new_folder_modelo:

Si es que no se ha proporcionado ningún valor, la función devuelve “None”.

```
else:
    return None
```

Caso contrario, se actualiza la variable global “folder_modelo” con este nuevo valor.

folder_modelo = new_folder_modelo

Finalmente, la función retorna esta variable global, pero con su valor actualizado.

return folder_modelo

Es importante resaltar que esta variable ya tiene por defecto una ruta hacia un directorio designado para almacenar el modelo, tal y como se mencionó anteriormente en la sección **“4.1.5.1. Declaración de variables”**. Por lo tanto, solo cuando se desea alojar el modelo en otro directorio que no sea el predeterminado, se hará uso de este formulario.

4.1.5.2.16. Image_output()

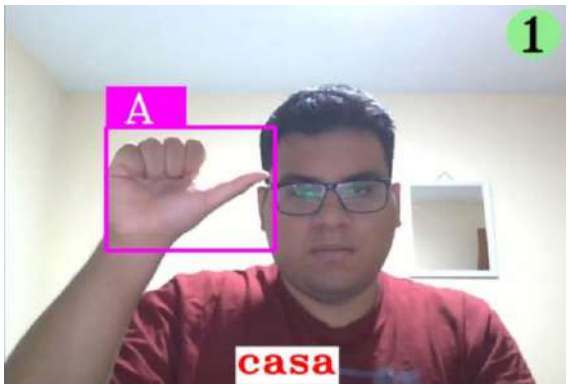
a) Descripción

Es aquella función que se llama a través del servidor Flask cuando se carga la página web “Signs-to-text.html”, es decir cuando previamente se ha dado clic al botón con la etiqueta “Test” de la página web “Menu-code1.html”. Su objetivo es procesar continuamente los fotogramas de un flujo de video proveniente de una cámara, detectar manos en cada fotograma y realizar predicciones basadas en la mano detectada, en donde la última predicción de cada intervalo regular de dos segundos se almacenará y se dibujará en la parte inferior de cada cuadro de video, con la finalidad de poder formar una palabra, tal y como se muestra en la Figura 23. Es importante resaltar que el tiempo de este intervalo se definió para que la persona sorda se pueda

tomar su tiempo en realizar las señas de maneras de manera clara y precisa.

Figura 23

Ilustración del uso de la función image_output() en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Mediante un bucle “While”, se lee constantemente un cuadro de video utilizando el objeto cap, que representa la cámara. Esta operación devuelve dos valores: success, que indica si la lectura fue exitosa, y frame, que contiene el cuadro de video leído.

```
while True:
    success, frame = cap.read()
```

Para verificar que no exista problemas con la cámara o algún otro problema en la lectura del flujo de vídeo, se verifica si la operación de lectura ha sido exitosa. Si no ha sido exitosa, el bucle se interrumpe mediante la instrucción break, lo que detiene la transmisión de video.

```
if not success:
    break
```

Caso contrario, se realiza una copia del cuadro original (“frame”), se realiza la detección de solo una de las dos manos y se obtiene el tiempo actual.

```
frame_copy = np.copy(frame)
hand_detected, _ = detector.findHands(frame)
current_time = time.time()
```

Luego, se implementa la lógica para el almacenamiento de predicciones. Mediante una condicional “if”, verificamos si es que no ha pasado dos segundos (“interval”) desde la última predicción.

```
if current_time - last_storage_time < interval:
```

Si es así, se agrega una entrada vacía a la lista “previous_predictions”.

```
previous_predictions.append("")
```

Dentro de esta condicional, también se verifica además si se ha detectado correctamente una mano.

```
if hand_detected:
```

Si es así, se procede a aplicar el mismo procesamiento utilizado en la función image_white() a partir de la mano detectada, por lo que se obtiene mediante la variable “img_white” una representación visual de la mano detectada, utilizando los 21 puntos de referencia y sus conexiones.

```
x, y, w, h = hand['bbox']
img_white = hand_points(frame_copy, x, y, w, h)
```

Antes de realizar una predicción sobre esta variable, primero se verifica si ha recibido algún valor por parte de esta variable.

if img_white is not None:

Si es así, se realiza una predicción a partir de ella mediante la función “classifier.getPrediction()”.

```
prediction, index = classifier.getPrediction(img_white,
                                              draw=False)
```

A partir del índice de la predicción y mediante la función “draw_image_output”, se dibuja la etiqueta resultante de la predicción de nuestro modelo, y se la sitúa en la parte inferior y centralizada del cuadro de video (“frame_copy”).

```
draw_image_output(frame_copy, x, y, w, h, index)
```

Ahora si ha pasado dos segundos desde la última predicción, se realiza el mismo procesamiento acerca de la mano detectada solo que, a diferencia del anterior, se agrega la última predicción en la lista “previous_predictions”.

previous_predictions[-1] = labels[index]

Asimismo, se iguala la variable “last_storage_time” con “current_time”, para restablecer la lógica del almacenamiento de predicciones, y se concatenan todas las etiquetas en minúsculas de cada predicción acumulada (“previous_predictions”) en la variable “predictions_global”.

last_storage_time = current_time
predictions_global = ''.join(prediction.lower() for prediction in previous_predictions)

Luego, se dibujan las predicciones acumuladas y el intervalo de tiempo en el cuadro de vídeo "frame_copy", utilizando las funciones "draw_previous_prediction(frame_copy)" y "draw_counter(frame_copy, current_time)".

draw_previous_prediction(frame_copy)
draw_counter(frame_copy, current_time)

Posteriormente, se codifica el cuadro de video (“frame_copy”) en formato JPEG utilizando OpenCV (cv2). Esta función de codificación devuelve dos valores: “suc”, que indica si la codificación fue exitosa y “encode” que contiene los bytes del cuadro de video codificado en formato JPEG.

suc, encode = cv2.imencode('.jpg', frame_copy)

Se verifica si la codificación no ha sido exitosa. Si es así, el bucle continúa con la próxima iteración utilizando “continue”.

```
if not suc:
    continue
```

Caso contrario, se convierte los bytes del cuadro de video codificado en una secuencia de bytes para poder así enviarlos como parte de un generador de cuadros codificados en JPEG, utilizando “yield”.

```
frame1 = encode.tobytes()
yield (b'--frame\r\n'
b'Content-Type: image/jpeg\r\n\r\n' + frame1 + b'\r\n')
```

4.1.5.2.17. draw_image_output(frame_copy, x, y, w, h, index)

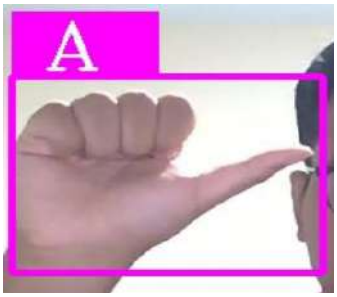
a) Descripción

Es aquella función que se ejecuta cuando previamente se ha detectado una mano. Su objetivo es dibujar una caja delimitadora alrededor de la mano detectada en el cuadro de video “frame_copy” y asimismo mostrar la etiqueta correspondiente a la predicción de una seña estática del alfabeto dactilológico del lenguaje de señas peruana, tal y como se muestra en la Figura 24.

Figura 24

Ilustración del uso de la función

draw_image_output(frame_copy, x, y, w, h, index) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se dibuja un rectángulo relleno de color encima del cuadro delimitador de la mano detectada mediante la función “cv2.rectangle”, en donde se definen parámetros como: las coordenadas donde se posicionará, sus dimensiones, el color y el tipo de grosor de línea. Con respecto a los dos primeros, se realiza un cálculo en el que se utiliza la variable “offset” y los

argumentos acerca de las coordenadas del cuadro delimitador (x, y).

cv2.rectangle(frame_copy, (x - offset, y - offset - 50), (x – offset + 90, y - offset), (255, 0, 255), cv2.FILLED)

Después, se coloca la etiqueta dentro de ese rectángulo a través de la función “cv2.putText” de la biblioteca OpenCV. Para poder utilizar esta función se necesita definir previamente algunos parámetros como: las coordenadas donde se posicionará, sus dimensiones, el color y el tipo de grosor de línea. En cuanto a las coordenadas del texto, se utiliza los argumentos acerca de las coordenadas del cuadro delimitador (x, y) en la que se le resta un valor de compensación a la coordenada “y” para evitar que se superponga con el cuadro delimitador.

```
cv2.putText(frame_copy, labels[index], (x, y - 26),
            cv2.FONT_HERSHEY_COMPLEX, 1.7,
            (255, 255, 255), 2)
```

Finalmente se dibuja otro rectángulo alrededor de la mano detectada, en el que se realiza un cálculo para determinar sus coordenadas y sus dimensiones, utilizando los argumentos acerca de las coordenadas y dimensiones del cuadro del cuadro delimitador (x, y, w, h).

cv2.rectangle(frame_copy, (x - offset, y - offset), (x + w + offset, y + h + offset), (255, 0, 255), 4)

4.1.5.2.18. draw_previous_prediction(frame_copy)

a) Descripción

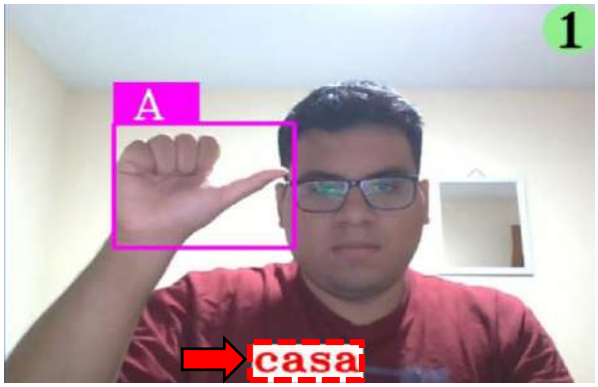
Es aquella función que recibe como argumento desde la función “image_output()”, a la copia del cuadro de video original

(frame_copy). Su objetivo es dibujar las etiquetas correspondientes a la última predicción durante cada intervalo de dos segundos, situándolas en la parte inferior y en el centro del cuadro de video "frame_copy", tal y como se muestra en la Figura 25.

Figura 25

Ilustración del uso de la función

draw_previous_prediction(frame_copy) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se define mediante la variable “font”, la fuente de texto con el que se van a dibujar las etiquetas de las predicciones, y mediante la variable “color”, el color de estos mismos.

```
font = cv2.FONT_HERSHEY_TRIPLEX
color = (0, 0, 255)
```

Luego, a través de la función “cv2.getTextSize”, se obtiene el ancho, el alto y la línea base de la palabra generada a partir de

la concatenación en minúsculas de las predicciones almacenadas en la variable “previous_predictions”. Estos valores son fundamentales ya que nos permitirá calcular la posición exacta donde se desea dibujar la palabra dentro del cuadro de video “frame_copy”.

```
(text_width, text_height), baseline = cv2.getTextSize
                                      (''.join(prediction.lower() for
                                      prediction in
                                      previous_predictions), font,
                                      1.5, 2)
```

Después, se calcula la posición para centrar la palabra en la parte inferior del cuadro de video “frame_copy” en donde “text_x” representa la posición horizontal de la palabra generada y "text_y" la posición vertical.

```
text_x = (frame_copy.shape[1] - text_width) // 2
text_y = frame_copy.shape[0] – 20
```

Posteriormente, a través de la función “cv2.rectangle” se dibuja un rectángulo relleno de color blanco detrás de la palabra generada y dibujada en el cuadro de video “frame_copy”. Este rectángulo tiene como finalidad mejorar la legibilidad de la palabra mostrada. Con respecto al posicionamiento del rectángulo, se realiza un cálculo en el que se tiene en cuenta la posición de la palabra (text_x, text_y), sus dimensiones (text_width, text_height) y su línea base.

```
cv2.rectangle(frame_copy, (text_x, text_y - text_height), (text_x
                + text_width, text_y + baseline), (255, 255, 255),
                 cv2.FILLED)
```

Finalmente se dibuja la palabra generada a partir de la concatenación en minúsculas de las predicciones almacenadas en la variable “previous_predictions”, situándolas en la parte inferior y en el centro del cuadro de video "frame_copy".

cv2.putText(frame_copy, ''.join(prediction.lower() for prediction in previous_predictions), (text_x, text_y), font, 1.5, color, 2, cv2.LINE_AA)

4.1.5.2.19. draw_counter(frame_copy, current_time)

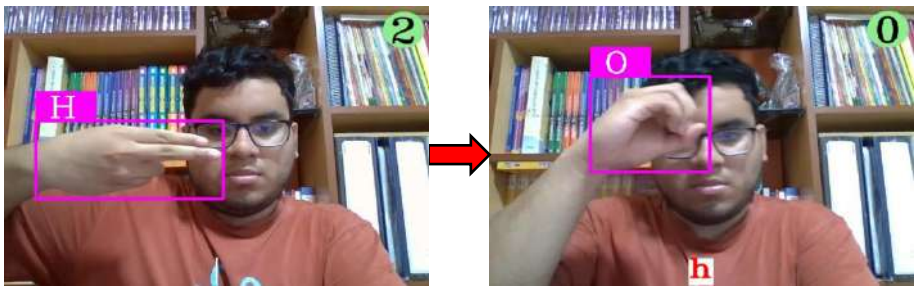
a) Descripción

Es aquella función que recibe como argumentos desde la función “image_output()” a la copia del cuadro de video original (frame_copy) y al tiempo actual (current_time). Su objetivo es dibujar un intervalo de tiempo en forma de círculo en la esquina superior derecha del cuadro de video “frame_copy”, de modo que se reinicie regularmente cada vez que transcurran dos segundos, tal y como se muestra en la Figura 26.

Figura 26

Ilustración del uso de la función

draw_counter(frame_copy_current_time) en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se obtiene el tiempo transcurrido (“counter”) a partir de la resta del tiempo actual (“current_time”) con el tiempo que se almacenó por última vez (“last_storage_time”).

```
counter = int(current_time - last_storage_time)
```

Asimismo, se define la fuente de letra y el color del intervalo de tiempo a través de las variables “font_counter” y “color_counter”.

```
font_counter = cv2.FONT_HERSHEY_TRIPLEX
color_counter = (144, 238, 144)
```

Luego a través de la función “cv2.getTexSize” se obtiene el tamaño del del intervalo de tiempo, el cual nos devuelve su ancho y su alto en píxeles. Estos valores son fundamentales ya que nos permite calcular la posición exacta donde se desea dibujar el intervalo dentro del cuadro de video “frame_copy”.

```
(counter_width, counter_height), _ = cv2.getTextSize(str(
                                     counter), font_counter,
                                     2,2)
```

Teniendo en cuenta estas dimensiones y con las dimensiones del cuadro de video “frame_copy”, se calcula la posición para centrar el intervalo en la esquina superior derecha del fotograma.

```
text_x_counter = frame_copy.shape[1] - counter_width - 20
text_y_counter = 20
```

Posteriormente, se dibuja un círculo relleno de color verde detrás de intervalo. Este círculo tiene como finalidad mejorar la legibilidad del intervalo mostrado. Con respecto al posicionamiento del círculo dentro del cuadro de video “frame_copy”, se ajusta su radio para ser un poco más grande que el intervalo.

```
circle_radius = max(counter_width, counter_height) // 2 + 10
cv2.circle(frame_copy, (text_x_counter + counter_width // 2,
        text_y_counter + counter_height // 2),
    circle_radius, color_counter, cv2.FILLED)
```

Finalmente se dibuja el intervalo de tiempo en el cuadro de video “frame_copy”, teniendo en cuenta todo lo calculado y definido anteriormente.

```
cv2.putText(frame_copy, str(counter), (text_x_counter,
        text_y_counter + counter_height),
    font_counter, 2, (0, 0, 0), 2, cv2.LINE_AA)
```

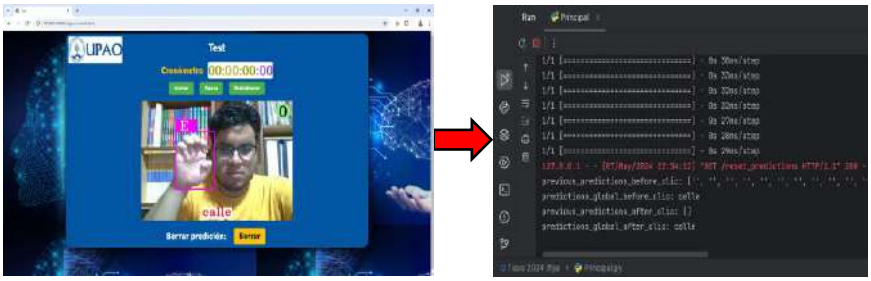
4.1.5.2.20. reset_predictions()

a) Descripción

Es aquella función que se llama a través del servidor Flask cuando se hace clic al botón con la etiqueta “borrar” de la página web “signs-to-text.html”. Su objetivo es eliminar las etiquetas correspondientes a las predicciones realizadas por nuestro modelo en el cuadro de video “frame_copy”, tal y como se muestra en la Figura 27.

Figura 27

Ilustración del uso de la función reset_predictions() en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se modifica el valor de la variable global “previous_predictions” asignándole una lista vacía, lo cual elimina las etiquetas correspondientes a las predicciones realizadas por el modelo.

previous_predictions = []

Finalmente, la función retorna esta variable global, pero con su valor actualizado.

return previous_predictions

4.1.6. Código “Voice_to_signs.py”

A continuación, se proporcionará una descripción detallada de las variables y funciones que formar parte de este código:

4.1.6.1. Declaración de variables

a) Video_by_letter

Es aquella variable que almacena un diccionario que relaciona cada letra del alfabeto con la ruta de un video en lenguaje de señas peruana.

```
video_by_letter = {"a": "static/Videos_letras/A.mp4",
                   "b": "static/Videos_letras/B.mp4",
                   "c": "static/Videos_letras/C.mp4",
                   # Resto de las letras y sus respectivos videos”}
```

b) Listener_words

Es aquella variable que almacena una lista de las palabras gramaticalmente correctas, los cuales se usarán en la comunicación de persona oyente a sordo.

```
listener_words = {"internet", "salud", "noticias", "ayer", "hoy", "mañana", "beber", "comer", "trabajar", "familia"}
```

c) Audio_request

Es aquella variable que se utilizará para indicar si se está solicitando capturar audio o no. Su valor inicial es: “False”.

```
audio_request = False
```

d) Text_recognized

Es aquella variable que nos indicará que se ha reconocido un texto.

```
text_recognized = "texto reconocido"
```

4.1.6.2. Funciones

4.1.6.2.1. confirmation_to_speak()

a) Descripción

Es aquella función que se llama a través del servidor Flask (“Principal.py) cuando se hace clic al ícono de un micrófono de la página web “voice-to-signs.hmtl”. Su objetivo es retornar un mensaje con la finalidad de indicar que ya se puede comenzar a hablar, tal y como se muestra en la Figura 28.

Figura 28

Ilustración del uso de la función confirmation_to_speak() en Python

Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se inicia modificando el valor de la variable global “audio_request” a “True”.

audio_request = True

Luego se crea un mensaje (“message”) que le indica al usuario oyente que puede comenzar a hablar.

message = "Ya puede comenzar a hablar"

Finalmente, la función retorna la variable “message” con su valor actualizado.

return message

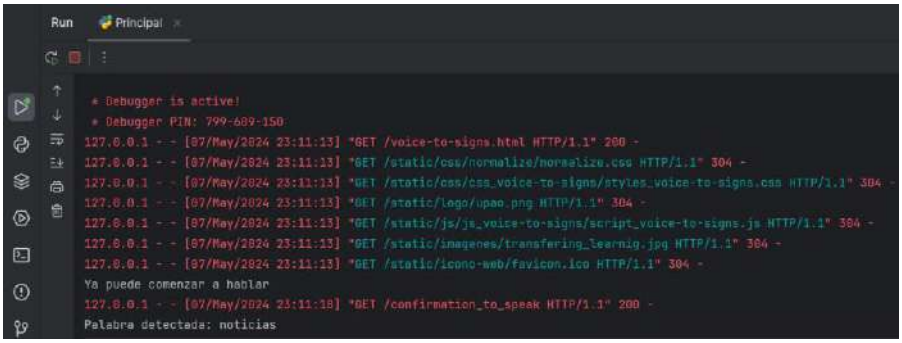
4.1.6.2.2. recognize_speech()

a) Descripción

Es aquella función que se llama a través del servidor Flask (“Principal.py) cuando se carga la página web “voice-to-signs.html”. Al iniciarse, esta función establece un bucle que se interrumpe al hacer clic en el ícono de un micrófono de la página mencionada. Su objetivo es reconocer la voz utilizando la API de reconocimiento de voz de Google, con el fin de devolver la palabra detectada, tal y como se muestra en la Figura 29.

Figura 29

Ilustración del uso de la función recognize_speech() en Python



Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se crea un objeto Recognizer de la librería speech_recognition que se utilizará para reconocer la voz.

recognizer = sr.Recognizer()

Se crea un contexto (“with”) utilizando un micrófono como fuente de entrada de audio

with sr.Microphone() as source:

Se ajusta el reconocedor para adaptarse al ruido ambiental, lo que ayuda a mejorar la precisión del reconocimiento de voz al eliminar el ruido de fondo.

recognizer.adjust_for_ambient_noise(source)

Luego, se inicia un bucle “while”, que espera hasta que el valor de la variable global “audio_request” sea True.

```
while not audio_request:
    pass
```

Una vez que el valor de la variable global “audio_request” sea True, se captura el audio del micrófono durante un máximo de 5 segundos utilizando el método “recognizer.listen(source, timeout=5)” de la clase “recognizer”. El audio capturado se guarda en la variable “audio”.

audio = recognizer.listen(source, timeout=5)

Posteriormente, se intenta reconocer la voz utilizando el servicio de reconocimiento de voz de Google (recognize_google()). Si el reconocimiento es exitoso, el texto reconocido se guarda en la variable texto en minúsculas.

```
try:
    texto = recognizer.recognize_google(audio,
                                        language="es-ES")
```

Si es que existe un error al intentar reconocer la voz, se establece la variable “texto” como una cadena vacía.

```
except sr.UnknownValueError:
    texto = ""
```

Finalmente, se establece la variable global “audio_request” como False para indicar que el proceso de reconocimiento de voz ha terminado. El texto reconocido en minúsculas se guarda en la variable global “text_recognized” y se devuelve como resultado de la función.

```
audio_request = False
text_recognized = texto.lower()
return text_recognized
```

4.1.6.2.3. remove_accents(input_str)

a) Descripción

Es aquella función que se llama a través de la función concat_videos_letters(server_word). Su objetivo es eliminar los acentos de los caracteres de una cadena de texto, que en este caso vendría a ser el argumento de entrada proporcionado.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Para descomponer los caracteres acentuados en caracteres base y marcas diacrítica, se utiliza el argumento 'NFD' del módulo “unicodedata”.

```
unicodedata.normalize('NFD', input_str)
```

Finalmente se itera sobre cada carácter, en donde se unen y se retorna los caracteres que no son marcas diacríticas. Esto se logra a través del filtrado de la categoría 'Mn' (Mark, Nonspacing)

if unicodedata.category(char) != 'Mn'

4.1.6.2.4. concat_videos_letters(server_word)

a) Descripción

Es aquella función que se llama a través del servidor Flask (Principal.py") una vez que se haya detectado la palabra a través del sistema de reconocimiento de voz. Su objetivo generar un único video a través de la concatenación de los videos correspondientes a cada letra del alfabeto en lenguaje de señas peruana, tal y como se muestra en la Figura 30.

Figura 30

Ilustración del uso de la función

concat_videos_letters(server_word)

Nota. Elaboración propia.

b) Funcionamiento

A continuación, se procederá a explicar el código relacionado a esta función:

Se crea una lista vacía video_clips que se utilizará para almacenar los clips de video que se van a concatenar.

video_clips = []

Asimismo, se llama a la función “remove_accents(server_word)” con el objetivo de obtener la palabra detectada en el reconocimiento de voz (“server_word”), sin ninguna tilde o marca diacrítica.

normalized_word = remove_accents(server_word)

Luego, comienza un bucle “for”, el cual recorre cada letra de la palabra filtrada (“normalized_word”).

for letter in normalized_word:

Se verifica si la letra actual está en el diccionario de la variable “video_by_letter”.

if letter in video_by_letter:

Si la letra está en el diccionario, se crea un objeto VideoFileClip para el video correspondiente a esa letra. Asimismo, se agrega este video a la lista de la variable “video_clips”

video_clip = VideoFileClip(video_by_letter[letter])
video_clips.append(video_clip)

Después de que se hayan almacenado en "video_clips" todos los videos correspondientes a cada letra de la palabra detectada por el reconocimiento de voz (“server_word”), se verifica si la lista de la variable “video_clips” no está vacía.

if video_clips:

Si no está vacía, se define la ruta para poder guardar el video final concatenado. Esta ruta se basa en el nombre de la palabra

detectada (“server_word”) junto con la ruta de la carpeta local “video_final”.

```
final_video_path = f'static/Video_final/video_final_
                   {server_word}.mp4'
```

Asimismo se utiliza la función “concatenate_videoclips()” de “moviepy” para concatenar todos videos almacenados en “video_clips” en un único video llamado “final_video”.

```
final_video = concatenate_videoclips(video_clips)
```

Después de haber concatenado los videos almacenados en “video_clips” en un único video, se utiliza la función “write_videofile()”, para poder escribir este video en el archivo de video de la ruta “final_video_path”.

```
final_video.write_videofile(final_video_path, codec="libx264")
```

Una vez que se ha terminado de escribir el video, se cierra el objeto “final_video” para liberar los recursos.

```
final_video.close()
```

Finalmente, la función devuelve la ruta del video final concatenado que se acaba de crear.

```
return final_video_path
```

4.1.7. Servidor web Flask (“Principal.py”)

4.1.7.1. Importaciones de módulos y configuración de la aplicación Flask

Aquí es donde se importa el popular framework de Python llamado “Flask” y algunos de sus clases, funciones y objetos relacionados a este. Esto con la finalidad de poder crear un servidor web con Flask que pueda manejar solicitudes y respuestas (HTTP), renderizar plantillas HTML y trabajar con datos JSON. Asimismo, se importan las funciones definidas en los códigos “Signs_to_text.py” y “Voice_to_signs.py” y se crea una instancia del servidor Flask con el nombre app.

```
from flask import Flask, render_template, Response, jsonify, request
from Signs_to_text import (image_output, image_white,
                           image_principal, reset_counter,
                           update_dataset_file, cap_photo,
                           update_folder_class, get_folder_class,
                           update_dataset_size, parameters_train,
                           update_folder_curves, reset_predictions
                           update_folder_model, train_model)
from Voice_to_signs import (confirmation_to_speak,
                            recognize_speech,
                            concat_videos_letters)
app = Flask(__name__)
```

4.1.7.2. Definición de rutas para la interacción con la aplicación

4.1.7.2.1. “/video_output”

Es aquella ruta que responde a solicitudes HTTP que se generan cuando un usuario accede a la página web "signs-to-text.html". En este contexto, cuando se accede a esta página, se ejecuta la función llamada “video_output_route()”, la cual a su vez llama a la función “image_output()” del código “Signs_to_text.py”, para devolver una secuencia de imágenes procesadas en formato “multipart/x-mixed-replace”, lo que permite que se visualice una

transmisión de video procesado en tiempo real desde la página web mencionada.

```
@app.route('/video_output')
def video_output_route():
    return Response(image_output(), mimetype='multipart/x-
                    mixed-replace; boundary=frame')
```

4.1.7.2.2. “/video_white”

Es aquella ruta que responde a las solicitudes HTTP que se generan cuando un usuario accede a la página web "Data.html". En este contexto, cuando se accede a esta página, se ejecuta la función llamada “video_white_route()”, la cual a su vez llama a la función “image_white()” del código “Signs_to_text.py”, para poder devolver una secuencia de imágenes procesadas en formato “multipart/x-mixed-replace”, lo que permite que se visualice una transmisión de video procesado en tiempo real desde la página web mencionada.

```
@app.route('/video_white')
def video_white_route():
    return Response(image_white(), mimetype='multipart/x-
                    mixed-replace; boundary=frame')
```

4.1.7.2.3. “/video_principal”

Es aquella ruta que responde a las solicitudes HTTP que se generan cuando un usuario accede a la página web "Data.html". En este contexto, cuando se accede a esta página, se ejecuta la función llamada “video_principal_route()”, la cual a través de la función “image_principal()” del código “Signs_to_text.py”, devuelve una secuencia de imágenes procesadas en formato multipart/x-mixed-replace, lo que permite que se visualice una

transmisión de video procesado en tiempo real desde la página web mencionada.

```
@app.route('/video_principal')
def video_principal_route():
    return Response(image_principal(), mimetype='multipart/x-
                                    mixed-replace; boundary=frame')
```

4.1.7.2.4. "/capture"

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET, que se generan desde la página web "Data.html" cuando el usuario presiona la letra "S" (teclado). En este contexto, cuando esta ruta recibe una petición GET, se ejecuta la función llamada "capture_route()", la cual a su vez llama a la función "cap_photo()" del código "Signs_to_text.py", para obtener el número de imágenes capturadas. Una vez que se obtiene el número de imágenes capturadas ("counter"), se lo devuelve como un objeto JSON con la clave 'counter_img', lo que permite que se visualice desde la página web mencionada.

```
@app.route('/capture', methods=['GET'])
def capture():
    counter = cap_photo()
    return jsonify({'counter_img': counter})
```

4.1.7.2.5. "/reset_counter"

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET, que se generan desde la página web "Data.html" cuando el usuario presiona la letra "S" (teclado). En este contexto, cuando esta ruta recibe una petición GET, se ejecuta la función "reset_counter_route()", la cual a su vez llama a la función "reset_counter()" del código "Signs_to_text.py" para reiniciar el contador de imágenes. Una vez completada la

ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/reset_counter', methods=['GET'])
def reset_counter_route():
    reset_counter()
    return jsonify({'message': 'Counter se reinició exitosamente'})
```

4.1.7.2.6. "/new_dataset_file"

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web "Data.html" a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función llamada "new_dataset_file_route()", la cual a su vez llama a la función "update_dataset_file(new_labels, new_num_class, new_folder_data)" del código "Signs_to_text.py" junto con los valores obtenidos del formulario como argumentos, esto con la finalidad de poder actualizar en el sistema la configuración de la carpeta del conjunto de datos. Una vez completada la ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/new_dataset_file', methods=['POST'])
def new_dataset_file_route():
    new_labels_str = request.form.get('label_class_data')
    new_labels = [label.strip() for label in new_labels_str.split(',')
                  if label]
    new_num_class = int(request.form.get('new_num_class'))
    new_folder_data = request.form.get('new_folder_data')
    update_dataset_file(new_labels, new_num_class,
                        new_folder_data)
    return jsonify({'message': 'Etiquetas, clases y ruta
                    actualizadas exitosamente'})
```

4.1.7.2.7. “/new_dataset_size”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web “Data.html” a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función llamada “new_dataset_size_route()”, la cual a su vez llama a la función “update_dataset_size(new_dataset_size)” del código “Signs_to_text.py” junto con el valor obtenido del formulario como argumento, esto con la finalidad de poder actualizar en el sistema, el tamaño del conjunto de datos (imágenes) para una clase específica. Una vez completada la ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/new_dataset_size', methods=['POST'])
def new_dataset_size_route():
    new_dataset_size = int(request.form.get('Dataset_size'))
    update_dataset_size(new_dataset_size)
    return jsonify({'message': 'Volumen de data actualizada
                    exitosamente'})
```

4.1.7.2.8. “/update_folder_class”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web “Data.html” a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función llamada “update_fold_class_route()”, la cual a su vez llama a la función “update_folder_class(new_name_class)” del código “Signs_to_text.py” junto con el valor obtenido del formulario como argumento, esto con la finalidad de poder actualizar en el sistema, la ruta de la carpeta donde se van a guardar los datos (imágenes) para una clase específica. Una vez completada la ejecución de

esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/update_folder_class', methods=['POST'])
def update_folder_class_route():
    new_name_class = request.form.get('new_name_class')
    update_folder_class(new_name_class)
    return jsonify({'message': 'Ruta (clase) actualizada
                exitosamente'})
```

4.1.7.2.9. "/get_folder_class"

Es aquella ruta que está configurada para responder a las solicitudes HTTP de tipo GET, que se generan cuando un usuario accede a la página web "Data.html". En este contexto, cuando se accede a esta página, se ejecuta la función llamada "get_folder_class_route()", la cual a su vez llama a la función "get_folder_class()" para obtener la ruta de la carpeta donde se van a guardar el conjunto de datos (imágenes) para una clase específica. Una vez que se obtiene esta ruta (folder_class), se lo devuelve como un objeto JSON a través de la clave 'folder_class', lo que permite que se visualice desde la página web mencionada.

```
@app.route('/get_folder_class', methods=['GET'])
def get_folder_class_route():
    folder_class = get_folder_class()
    return jsonify({'folder_class': folder_class})
```

4.1.7.2.10. "/parameters"

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web "Train.html" a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función

llamada “parameters_route()”, la cual a su vez llama a la función “parameters_train(new_num_class, new_batch_size, new_epochs, new_learning_rate, new_folder_train, new_folder_validation)” del código “Signs_to_text.py” junto con los valores obtenidos del formulario como argumentos, esto con la finalidad de poder actualizar en el sistema los parámetros del entrenamiento del modelo. Una vez completada la ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/parameters', methods=['POST'])
def parameters_route():
    new_num_class = int(request.form.get('num_class'))
    new_batch_size = int(request.form.get('batch_size'))
    new_epochs = int(request.form.get('epochs'))
    new_learning_rate = float(request.form.get('learning_rate'))
    new_folder_train = request.form.get('folder_train')
    new_folder_validation = request.form.get('folder_validation')
    parameters_train(new_num_class, new_batch_size,
                     new_epochs, new_learning_rate,
                     new_folder_train, new_folder_validation)
    return jsonify({'message': 'Parámetros actualizados
                    exitosamente'})
```

4.1.7.2.11. “/update_folder_curves”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web “Train.html” a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función llamada “update_folder_curves_route()”, la cual a su vez llama a la función “update_folder_curves(new_folder_accuracy, new_folder_loss)” del código “Signs_to_text.py” junto con los valores obtenidos del formulario como argumentos, esto con la

finalidad de poder actualizar en el sistema las rutas de las carpetas de curvas de precisión y pérdida. Una vez completada la ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/update_folder_curves', methods=['POST'])
def update_folder_curves_route():
    new_folder_acc = request.form.get('accuracy_folder')
    new_folder_loss = request.form.get('loss_folder')
    update_folder_curves(new_folder_acc, new_folder_loss)
    return jsonify({'message': 'Carpeta actualizada exitosamente'})
```

4.1.7.2.12. "/update_folder_model"

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo POST, que se generan desde la página web "Train.html" a través de un formulario. En este contexto, cuando esta ruta recibe una petición POST, se ejecuta la función llamada "update_folder_model()", la cual a su vez llama a la función "update_folder_model(new_folder_model)" del código "Signs_to_text.py" junto con los valores obtenidos del formulario como argumentos, esto con la finalidad de poder actualizar en el sistema la ruta de la carpeta del modelo. Una vez completada la ejecución de esta función, se devuelve a través de la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/update_folder_model', methods=['POST'])
def update_folder_model_route():
    new_folder_model = request.form.get('new_folder_model')
    update_folder_model(new_folder_model)
    return jsonify({'message': 'Carpeta actualizada exitosamente'})
```

4.1.7.2.13. “/train”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET, que se generan desde la página web “Train.html cuando el usuario hace clic en el botón con la etiqueta “Entrenar”. En este contexto, cuando se recibe una solicitud GET en esta ruta, se ejecuta la función “train_route()”, la cual a su vez llama a la función “train_model()” del código “Signs_to_text.py” para poder obtener las rutas donde se generaron las imágenes de precisión y pérdida generadas durante el entrenamiento y validación. Una vez que se obtiene las rutas de las imágenes de estas curvas (accuracy_image, loss_image), se los devuelve como un objeto JSON a través de las claves: 'accuracy_image' y 'loss_image', lo que permite que se visualicen estas imágenes desde la página web mencionada.

```
@app.route('/train', methods=['GET'])
def train_route():
    accuracy_image, loss_image = train_model()
    return jsonify({'accuracy_image': accuracy_image,
                    'loss_image': loss_image})
```

4.1.7.2.14. “/reset_prediction”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET, que se generan desde la página web “signs-to-text.html”, cuando el usuario hace clic en el botón con la etiqueta “Borrar”. En este contexto, cuando se recibe una solicitud GET en esta ruta, se ejecuta la función “reset_predictions_route()”, la cual a su vez llama a la función “reset_predictions()” del código “Signs_to_text.py” para borrar las etiquetas correspondientes a las predicciones realizadas por nuestro modelo de clasificación de señas estáticas. Una vez completada la ejecución de esta función, se devuelve a través de

la clave 'message', un mensaje de confirmación como un objeto JSON.

```
@app.route('/reset_predictions', methods=['GET'])
def reset_predictions_route():
    reset_predictions()
    return jsonify({'message': Las predicciones se borraron
                exitosamente'})
```

4.1.7.2.15. "/confirmation_to_speak"

Es aquella ruta que está configurada para responder a las solicitudes HTTP de tipo GET, que se generan desde la página web "voice-to-signs.html", cuando el usuario hace clic al ícono de un micrófono. En este contexto, cuando se recibe una solicitud GET en esta ruta, se ejecuta la función llamada "confirmation_to_speak_route()", la cual a su vez llama a la función "confirmation_to_speak()" del código "Voice_to_signs.py" para obtener el mensaje que le confirme al usuario que puede comenzar a hablar. Una vez que se obtiene este mensaje (confirmation_speak), se lo devuelve como objeto JSON a través de la clave 'confirmation_speak', lo que permite que se visualice desde la página web mencionada.

```
@app.route('/confirmation_to_speak', methods=['GET'])
def confirmation_to_speak_route():
    confirmation_speak = confirmation_to_speak()
    return jsonify({'confirmation_speak': confirmation_speak})
```

4.1.7.2.16. "/recognize_speech"

Es aquella ruta que está configurada para responder a las solicitudes HTTP de tipo GET, que se generan cuando un usuario accede a la página web "voice-to-signs.html". En este contexto, cuando se accede a esta página, se ejecuta la función llamada

“recognize_speech_route()”, la cual a su vez llama a la función “recognize_speech()” del código “Voice_to_signs.py” para obtener la palabra detectada del reconocimiento de voz. Una vez que se obtiene esta palabra (converted_text), se lo devuelve como objeto JSON a través de la clave 'converted_text', lo que permite que se visualice desde la página web mencionada.

```
@app.route('/recognize_speech', methods=['GET'])
def recognize_speech_route():
    converted_text = recognize_speech()
    return jsonify({'converted_text': converted_text})
```

4.1.7.2.17. “/video/<server_word>”

Es aquella ruta que está configurada para responder a las solicitudes HTTP de tipo GET, que se generan a través de la página web “voice-to-signs.html”, una vez detectada la palabra del reconocimiento de voz. En este contexto, cuando se recibe una solicitud GET en esta ruta, se ejecuta la función asociada a esta, llamada “video_route(server_word)”, la cual a su vez llama a la función “concat_videos_letters(server_word)” del código “Voice_to_signs.py” para obtener la ruta del video concatenado correspondiente a la palabra detectada del reconocimiento de voz (server_word). Una vez que se obtiene esta ruta (video_path), se lo devuelve como objeto JSON a través de la clave 'video_path', lo que permite que se visualice desde la página web mencionada.

```
@app.route('/video/<server_word>')
def video_route(server_word):
    video_path = concat_videos_letters(server_word)
    return jsonify({'video_path': video_path})
```

4.1.7.2.18. “/”:

Es aquella ruta raíz que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “index_html()” renderiza la plantilla “index.html”, la cual está ubicada dentro del directorio “templates”.

```
@app.route('/')
def index_html():
    return render_template('index.html')
```

4.1.7.2.19. “/menu-code1.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “menu_code1_html()” renderiza la plantilla “Menu-code1.html”, la cual está ubicada en el directorio “Code1” con ruta “templates/nav/Code1”.

```
@app.route('/menu-code1.html')
def menu_code1_html():
    return render_template('nav/Code1/Menu-code1.html')
```

4.1.7.2.20. “/data.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “data_html()” renderiza la plantilla “Data.html”, la cual está ubicada en el directorio “Code1” con ruta “templates/nav/Code1”.

```
@app.route('/data.html')
def data_html():
    return render_template('nav/Code1/Data.html')
```

4.1.7.2.21. “/train.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “train_html()” renderiza la plantilla 'Train.html', la cual está ubicada en el directorio “Code1” con ruta “templates/nav/Code1”.

```
@app.route('/train.html')
def train_html():
    return render_template('nav/Code1/Train.html')
```

4.1.7.2.22. “/clasification-report.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “clasification_report_html()” renderiza la plantilla “Report.html”, la cual está ubicada en el directorio “Code1” con ruta “templates/nav/Code1'.

```
@app.route('/clasification-report.html')
def clasification_report_html():
    return render_template('nav/Code1/Report.html')
```

4.1.7.2.23. “/signs-to-text.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “signs_to_text_html()” renderiza la plantilla “signs-to-text.html”, la cual está ubicada en el directorio “Code1” con ruta “templates/nav/Code1'.

```
@app.route('/signs-to-text.html')
def signs_to_text_html():
    return render_template('nav/Code1/signs-to-text.html')
```

4.1.7.2.24. “/voice-to-signs.html”

Es aquella ruta que está configurada para responder las solicitudes HTTP de tipo GET. En este contexto, cuando se recibe una solicitud GET en esta ruta, la función “voice_to_signs_html()” renderiza la plantilla “voice-to-sign.html’, la cual está ubicada en el directorio “Code2” con ruta “templates/nav/Code2'.

```
@app.route('/voice-to-signs.html')
def voice_to_signs_html():
    return render_template('nav/Code2/voice-to-signs.html')
```

4.1.7.3. Configuración de ejecución

Este bloque de código define la configuración para ejecutar el servidor web, especificando la activación del modo de depuración (debug=True) y el puerto (port=5000) en caso de que se ejecute directamente desde este archivo (if _name_== '_main_').

```
if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

4.1.8. Cronómetro digital

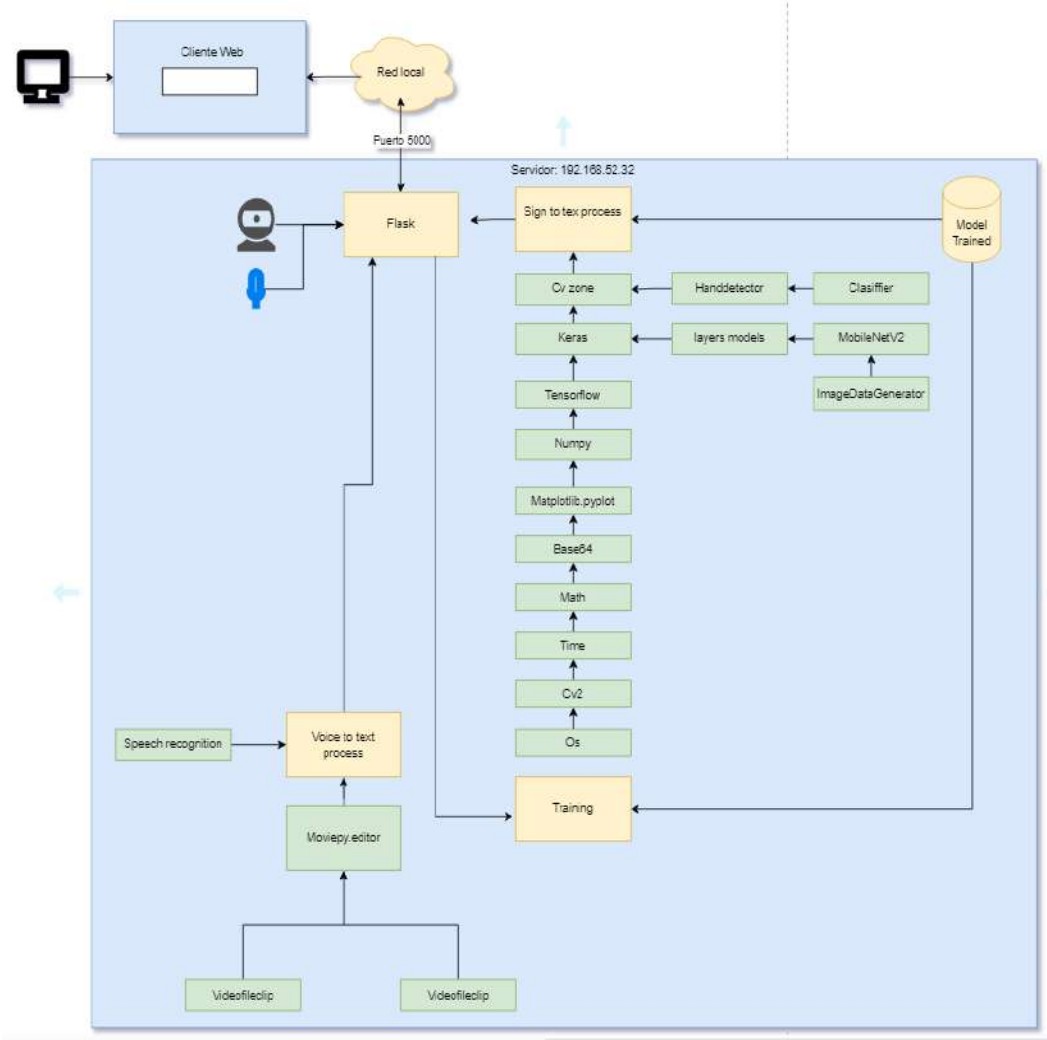
Se implementó un cronómetro digital en Java Script (JS), con la finalidad de medir los tiempos de reconocimiento de las palabras tanto en la comunicación de persona sorda a oyente como en la comunicación de persona oyente a sorda, esto solo cuando se utiliza el sistema traductor de comunicación bidireccional. Es por ello que, en las páginas web “sign-to-text.html” y “voice-to-signs.html” se puede visualizar un cronómetro con las funciones de inicio, detección y reinicio; el cual es controlable tanto por los botones en la interfaz como por las teclas del teclado: “s” (start), “d” (stop”), “a” (reset).

4.1.9. Diagrama de bloques

A continuación, se muestra a través de la Figura 31, la arquitectura del sistema traductor de comunicación bidireccional, representado en un diagrama de bloques.

Figura 31

Diagrama de bloques acerca de la arquitectura del sistema traductor de comunicación bidireccional



Nota. Elaboración propia.

4.1.10. Preparación del sistema traductor de comunicación bidireccional

Antes de analizar e interpretar los resultados que se obtuvieron, es importante mencionar el proceso de preparación del sistema traductor de comunicación bidireccional que se realizó tanto para la comunicación de persona sorda a oyente como en la comunicación de persona oyente a sorda.

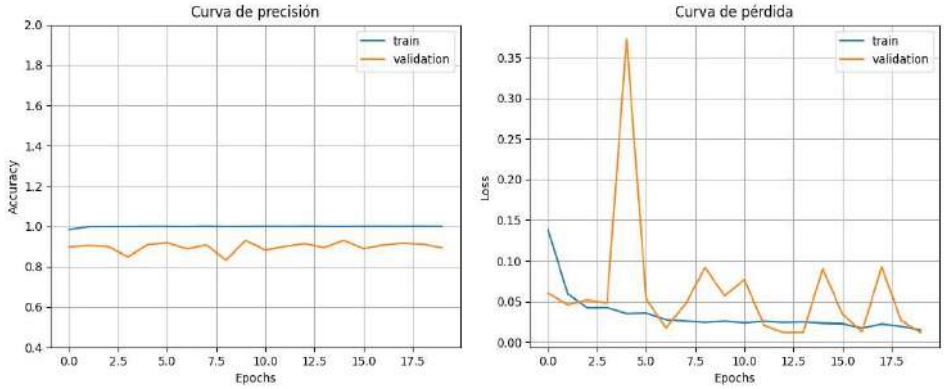
a) Comunicación de persona sorda a oyente

Para que nuestro modelo clasificador de aprendizaje profundo pueda clasificar las señas estáticas del alfabeto dactilológico del lenguaje de señas peruana realizado por una persona sorda en tiempo real, se tomó un conjunto de 3000 fotos de manos de tres personas sordas, por cada gesto estático (24 clases). Estas imágenes se dividieron en 64800 para el conjunto de datos de entrenamiento y 7200 para el conjunto de datos de validación del modelo. Es importante destacar que no existe un número mínimo establecido de datos de entrenamiento para construir un modelo de aprendizaje profundo. Sin embargo, según la investigación de Montenegro y Villa (2019), existe un consenso en la comunidad científica de que un conjunto de datos de al menos 1000 imágenes etiquetadas por clase es considerado adecuado cuando se utiliza un modelo de aprendizaje profundo para clasificar imágenes. Es por ello que se tomó 1000 imágenes para cada clase por persona sorda, en donde el 90% se utiliza para el entrenamiento y el 10% restante para la validación del modelo.

Luego una vez que se completó el tamaño del conjunto de datos para el entrenamiento y validación del modelo de aprendizaje profundo, se procedió a entrenarlo utilizando los parámetros que se mencionaron en el apartado “4.1.5.1. Declaración de variables”. se obtuvieron: el modelo de red neuronal de aprendizaje profundo (.h5), los índices de las clases (.txt) y las curvas de precisión y de pérdida, las cuales estas últimas se visualizan en la Figura 32.

Figura 32

Curvas de precisión y pérdida generadas durante el entrenamiento y validación del modelo



Nota. Elaboración propia.

Al analizar las imágenes de las curvas de precisión generadas durante el entrenamiento y validación del modelo, se observó que a medida que van avanzando las épocas, la precisión del modelo en lo que respecta al entrenamiento, se va acentuando en 1, es decir 100%. Por otro lado, la curva de precisión en la validación del modelo es menos estable que con la que se observó en el entrenamiento, aun así, se podría considerar que la precisión se mantiene estable ya que los picos de variación son pequeños, siendo la precisión más pequeña 0.82%.

Ahora, con respecto a las imágenes de las curvas de pérdida generadas durante el entrenamiento y validación del modelo, se observó que a medida que van avanzando las épocas, la precisión del modelo en lo que respecta al entrenamiento, muestra una disminución constante y gradual a lo largo de las épocas, acercándose a 0. Esto indica que el modelo está aprendiendo y ajustando sus parámetros para minimizar la pérdida durante el entrenamiento. Por otro lado, la curva de pérdida de validación, muestra un comportamiento más errático, siendo el pico más alto 0.38, el cual se obtuvo alrededor entre

las épocas 4 y 5, seguido de fluctuaciones a lo largo de las demás épocas. Esto probablemente se deba a la variabilidad en los tamaños de las manos y en las orientaciones con respecto a la cámara. No obstante, la estabilidad relativa hacia el final sugiere que el modelo está logrando un equilibrio en su capacidad para generalizar nuevos datos.

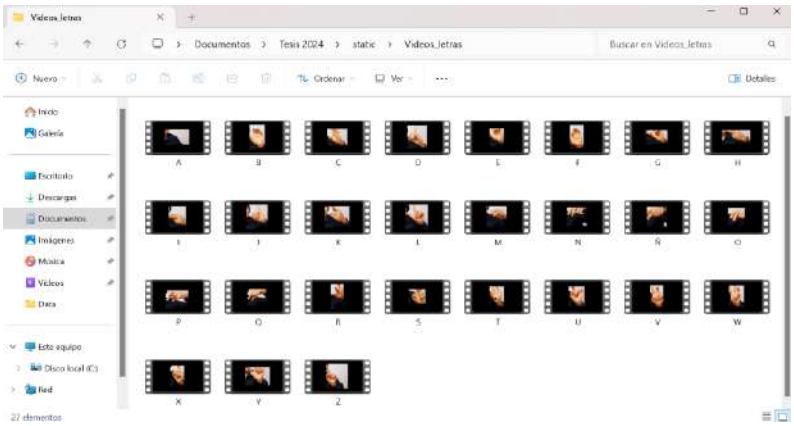
Con el archivo generado que contiene el modelo de red neuronal de aprendizaje profundo (keras_model.h5) y con el archivo de texto que contiene los índices de las clases (labels.txt), el sistema podrá clasificar en tiempo real las señas estáticas del alfabeto dactilológico del lenguaje de señas peruana mediante una cámara de video desde la página web “signs-to-text.html” (ver *Anexos 09, 10 y 11)*.

b) Comunicación de persona oyente a sorda

Se utilizó el video “ABECEDARIO LENGUA DE SEÑAS PERUANA LSP” de IncluSeñas_Perú (2021), el cual se recortó en 27 partes debido a que en este video se representa cada letra del abecedario del lenguaje de señas peruana, tal y como se muestra en la Figura 33.

Figura 33

Representación de las 27 letras del abecedario dactilológico del Lenguaje de Señas Peruana en videos



Nota. Elaboración propia.

De esta forma cuando una persona oyente acceda a la página web “voice-to-signs.html”, de clic al ícono de un micrófono y pronuncie una palabra, el sistema reconocerá la palabra y se concatenarán cada parte recortada de este video con el fin de mostrar un único video, el cual represente la palabra pronunciada de manera deletreada (*ver Anexos 12, 13 y 14)*.

4.2. Análisis e interpretación de resultados

4.2.1. Comunicación de persona sorda a oyente

Para la comunicación de persona sorda a oyente, se emparejó a cada una de las tres personas oyentes de La Libertad con una persona sorda de la Asociación De Sordos De La Libertad. A partir de ello, se midió el tiempo en que se demora cada una de las tres personas oyentes de La Libertad (*ver Anexo 4, 5 y 6)* en reconocer correctamente cada palabra expresada mediante el lenguaje de señas peruana, sin y con el uso del sistema traductor de comunicación bidireccional. Para ello, previamente se listó 30 palabras, las cuales fueron obtenidas mediante la encuesta (*ver Anexo 15*) realizada a estas personas.

Es importante mencionar que se estableció un límite de 5 minutos como tiempo máximo para que una persona oyente de La Libertad pueda reconocer correctamente una palabra.

Tabla 4

Tiempos de reconocimiento de cada palabra expresada en el Lenguaje de Señas Peruana por una persona sorda de la Asociación De Sordos De La Libertad para cada persona oyente de La Libertad, sin el uso del sistema traductor de comunicación bidireccional

Palabras	**Persona Oyente 1**	**Persona Oyente 2**	**Persona Oyente 3**
Aceptar	No reconoció	No reconoció	No reconoció
Aguantar	29.03 s	35.19 s	32.84 s
Árbol	No reconoció	No reconoció	100.85 s
Buscar	No reconoció	No reconoció	116.49 s
Canción	33.87 s	40.93 s	27.78 s
Comprender	No reconoció	No reconoció	No reconoció
Contador	40.11 s	31.38 s	25.60 s
Día	15.37 s	21.05 s	18.16 s
Diccionario	No reconoció	No reconoció	90.24 s
Ensayo	No reconoció	No reconoció	No reconoció
Equivocar	No reconoció	No reconoció	No reconoció
Examen	No reconoció	No reconoció	81.31 s
Explicar	No reconoció	No reconoció	No reconoció
Iglesia	No reconoció	No reconoció	No reconoció
Manta	28.83 s	32.17 s	15.35 s
Navidad	No reconoció	No reconoció	No reconoció
País	No reconoció	No reconoció	102.63 s

Panadería	No reconoció	No reconoció	No reconoció
Piedra	156.10 s	No reconoció	No reconoció
Quien	19.78 s	28.26 s	22.14 s
Refrigerador	163.87 s	No reconoció	No reconoció
Siempre	No reconoció	No reconoció	70.18 s
Sufrir	21.88 s	17.90 s	33.81 s
Tentación	No reconoció	No reconoció	No reconoció
Terremoto	19.44 s	29.45 s	35.67 s
Trapear	26.70 s	34.20 s	29.22 s
Universidad	35.62 s	43.44 s	27.69 s
Vacaciones	No reconoció	No reconoció	No reconoció
Verdad	No reconoció	No reconoció	122.11 s
Visitar	No reconoció	No reconoció	No reconoció

Nota. Elaboración propia.

Interpretación: Al analizar los tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad sin el uso del sistema traductor de comunicación bidireccional, se observó que 11 de las 30 palabras medidas no fueron reconocidas por ninguna de las tres personas oyentes. No obstante, hubo 19 palabras en las cuales al menos una persona oyente logró reconocerlas. Asimismo, de estas 19 palabras, solo 10 fueron reconocidas por las tres personas oyentes. En contraste, el resto de las palabras presentaron una mayor dificultad de reconocimiento y por ende no fueron reconocidas por el total de la muestra de personas oyentes de La Libertad.

Tabla 5

Tiempos de reconocimiento de cada palabra expresada en el Lenguaje de Señas Peruana por una persona sorda de la Asociación De Sordos De La Libertad para cada persona oyente de La Libertad, con el uso del sistema traductor de comunicación bidireccional

Palabras	**Persona Oyente 1**	**Persona Oyente 2**	**Persona Oyente 3**
Aceptar	14.14 s	14.50 s	14.27 s
Aguantar	16.09 s	16.20 s	16.16 s
Árbol	10.12 s	10.21 s	10.25 s
Buscar	12.59 s	12.72 s	12.44 s
Canción	14.62 s	14.74 s	14.55 s
Comprender	18.55 s	18.49 s	18.61 s
Contador	16.58 s	16.30 s	16.47 s
Día	6.06 s	6.22 s	6.11 s
Diccionario	22.26 s	22.09 s	22.13 s
Ensayo	12.92 s	12.98 s	12.86 s
Equivocar	18.34 s	18.27 s	18.41 s
Examen	12.63 s	12.49 s	12.37 s
Explicar	16.61 s	16.68 s	16.54 s
Iglesia	15.03 s	15.09 s	14.97 s
Manta	10.56 s	10.33 s	10.49 s
Navidad	14.23 s	14.17 s	14.39 s
País	8.73 s	8.68 s	8.53 s

Panadería	18.32 s	18.15 s	18.41 s
Piedra	12.30 s	12.42 s	12.25 s
Quien	10.25 s	10.11 s	10.19 s
Refrigerador	24.81 s	24.64 s	24.78 s
Siempre	14.72 s	14.56 s	14.52 s
Sufrir	12.24 s	12.38 s	12.32 s
Tentación	19.04 s	18.97 s	19.11 s
Terremoto	18.38 s	18.51 s	18.65 s
Trapear	14.08 s	14.15 s	14.03 s
Universidad	22.83 s	22.76 s	22.90 s
Vacaciones	21.08 s	21.02 s	21.14 s
Verdad	13.06 s	13.12 s	13.00 s
Visitar	14.29 s	14.36 s	14.22 s

Nota. Elaboración propia.

Interpretación: Al analizar los tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad con el uso del sistema traductor de comunicación bidireccional, se observó que, a diferencia de los resultados sin el uso del sistema, todas las palabras fueron reconocidas correctamente. Asimismo, los tiempos de reconocimiento de las palabras son más bajos en comparación que cuando no se usó el sistema. No obstante, para cuantificar y validar estadísticamente esta diferencia, se tendrá que aplicar una prueba estadística, como la prueba t de Student (prueba paramétrica) o la prueba de Wilcoxon (prueba no paramétrica). La elección de una de estas pruebas se evaluará con la prueba de normalidad de la diferencia de los tiempos de reconocimiento de las palabras sin y con el uso del sistema. Es por ello que, para poder facilitar la aplicación de las pruebas estadísticas mencionadas, se

organizó los tiempos de reconocimiento de cada palabra para cada persona oyente, sin y con el uso del sistema, tal y como se observa en la Tabla N°6.

Tabla 6

Tiempos de reconocimiento de las palabras para cada persona oyente de La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional

Persona Oyente	Tiempo sin sistema (s)	Tiempo con sistema (s)
1	29.03 s	16.09 s
1	33.87 s	14.62 s
1	40.11 s	16.58 s
1	15.37 s	6.06 s
1	28.83 s	10.56 s
1	19.78 s	10.25 s
1	21.88 s	12.24 s
1	19.44 s	18.38 s
1	26.70 s	14.08 s
1	35.62 s	22.83 s
2	35.19 s	16.20 s
2	40.93 s	14.74 s
2	31.38 s	16.30 s
2	21.05 s	6.22 s
2	32.17 s	10.33 s

2	28.26 s	10.11 s
2	17.90 s	12.38 s
2	29.45 s	18.51 s
2	34.20 s	14.15 s
2	43.44 s	22.76 s
3	32.84 s	16.16 s
3	27.78 s	14.55 s
3	25.60 s	16.47 s
3	18.16 s	6.11 s
3	15.35 s	10.49 s
3	22.14 s	10.19 s
3	33.81 s	12.32 s
3	35.67 s	18.65 s
3	29.22 s	14.03 s
3	27.69 s	22.90 s

Nota. Elaboración propia.

Importante: Para no introducir sesgos debido a muestras desiguales, se colocó los tiempos de reconocimiento de aquellas palabras que fueron reconocidas por el total de la muestra de las personas oyentes de La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional.

4.2.2. Comunicación de persona oyente a sorda

Para la comunicación de persona oyente a sorda, se midió el tiempo que se demora cada una de las tres sordas de la Asociación De Sordos De La Libertad (*ver Anexo 1, 2 y 3)* en reconocer correctamente cada palabra pronunciada por una persona oyente de La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional. Para ello, previamente se listó 30 palabras, las cuales fueron obtenidas mediante la encuesta (ver Anexo 15) realizada a estas personas.

Es importante mencionar que se estableció 5 minutos, como tiempo máximo para que una persona sorda pueda reconocer correctamente una palabra.

Tabla 7

Tiempos de reconocimiento de cada palabra pronunciada por una persona oyente de La Libertad para cada persona sorda de la Asociación De Sordos De La Libertad, sin el uso del sistema traductor de comunicación bidireccional

Palabras	Persona Sorda 1	Persona Sorda 2	Persona Sorda 3
Aeropuerto	No reconoció	No reconoció	No reconoció
Almohada	188.59 s	192.14 s	No reconoció
Anfitrión	No reconoció	No reconoció	No reconoció
Ardilla	155.18 s	No reconoció	No reconoció
Armario	175.46 s	No reconoció	179.92 s
Biblioteca	121.92 s	141.09 s	No reconoció
Crucigrama	No reconoció	No reconoció	No reconoció

Disimular	167.93 s	46.02 s	No reconoció
Encuesta	79.93 s	No reconoció	94.98 s
Escalera	83.80 s	118.27 s	111.44 s
Espejo	87.11 s	96.12 s	102.31 s
Fábrica	45.07 s	No reconoció	No reconoció
Galleta	45.80 s	No reconoció	No reconoció
Hielo	88.67 s	106.11 s	102.73 s
Hierbabuena	No reconoció	No reconoció	No reconoció
Hospital	118.51 s	No reconoció	No reconoció
Impresora	133.04 s	No reconoció	No reconoció
Intérprete	No reconoció	No reconoció	No reconoció
Jardín	130.17 s	No reconoció	151.71 s
Lámpara	95.92 s	117.37 s	104.85 s
Lista	138.72 s	No reconoció	No reconoció
Mariposa	82.29 s	104.56 s	120.22 s
Montaña	No reconoció	No reconoció	No reconoció
Película	81.94 s	108.80 s	83.14 s
Relámpago	No reconoció	No reconoció	No reconoció
Semáforo	222.57 s	No reconoció	No reconoció
Simulacro	No reconoció	No reconoció	No reconoció
Sistema	No reconoció	No reconoció	No reconoció
USB	74.31 s	101.43 s	95.79 s
Zapato	90.61 s	103.98 s	84.65 s

Nota. Elaboración propia.

Interpretación: Al analizar los tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad sin el uso del sistema traductor de comunicación bidireccional, se observó que 6 de las 30 palabras medidas no fueron reconocidas por ninguna de las tres personas sordas. No obstante, hubo 24 palabras en las cuales al menos 1 persona sorda logró reconocerlas. Asimismo, de estas 24 palabras, solo 8 fueron reconocidas por las tres personas oyentes. En contraste, el resto de las palabras presentaron una mayor dificultad de reconocimiento y por ende no fueron reconocidas por el total de la muestra de personas sordas de la Asociación De Sordos De La Libertad.

Tabla 8

Tiempos de reconocimiento de cada palabra pronunciada por una persona oyente de La Libertad para cada persona sorda de la Asociación De Sordos De La Libertad, con el uso del sistema traductor de comunicación bidireccional

Palabras	Persona Sorda 1	Persona Sorda 2	Persona Sorda 3
Aeropuerto	29.92 s	30.10 s	29.80 s
Almohada	24.16 s	24.30 s	24.00 s
Anfitrión	26.01 s	26.20 s	25.90 s
Ardilla	20.73 s	20.80 s	20.60 s
Armario	22.59 s	22.70 s	22.50 s
Biblioteca	28.90 s	29.00 s	28.80 s
Crucigrama	30.17 s	30.30 s	30.00 s
Disimular	26.30 s	26.40 s	26.20 s

Encuesta	23.45 s	23.60 s	23.30 s
Escalera	24.30 s	24.40 s	24.20 s
Espejo	18.08 s	18.20 s	18.07 s
Fábrica	25.90 s	26.00 s	25.80 s
Galleta	20.80 s	20.90 s	20.70 s
Hielo	14.69 s	14.80 s	14.60 s
Hierbabuena	31.71 s	31.80 s	31.60 s
Hospital	23.23 s	23.30 s	23.10 s
Impresora	26.38 s	26.50 s	26.20 s
Intérprete	29.29 s	29.40 s	29.20 s
Jardín	17.52 s	17.60 s	17.40 s
Lámpara	20.84 s	20.90 s	20.70 s
Lista	17.42 s	17.50 s	17.30 s
Mariposa	22.91 s	23.10 s	22.80 s
Montaña	20.41 s	20.50 s	20.30 s
Película	23.68 s	23.80 s	23.60 s
Relámpago	25.79 s	25.90 s	25.70 s
Semáforo	23.30 s	23.40 s	23.20 s
Simulacro	26.88 s	27.00 s	26.70 s
Sistema	20.76 s	20.90 s	20.60 s
USB	9.42 s	9.50 s	9.30 s
Zapato	17.74 s	17.80 s	17.60 s

Nota. Elaboración propia.

Interpretación: Al analizar los tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad con el uso del sistema traductor de comunicación bidireccional, se observó que a diferencia que cuando no se usó el sistema, acá todas las palabras fueron reconocidas correctamente. Asimismo, los tiempos de reconocimiento de las palabras son más bajos en comparación que cuando no se usó el sistema. No obstante, para cuantificar y validar estadísticamente esta diferencia, se tendrá que aplicar una prueba estadística, como la prueba t de Student (prueba paramétrica) o la prueba de Wilcoxon (prueba no paramétrica). La elección de una de estas pruebas se evaluará con la prueba de normalidad de la diferencia de los tiempos de reconocimiento de las palabras sin y con el uso del sistema. Es por ello que, para poder facilitar la aplicación de las pruebas estadísticas mencionadas, se organizó los tiempos de reconocimiento de las palabras sin y con el uso del sistema para cada persona sorda, tal y como se observa en Tabla N°9.

Tabla 9

Tiempos de reconocimiento de las palabras para cada persona sorda de la Asociación De Sordos De La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional

Persona Sorda	Tiempo sin sistema (s)	Tiempo con sistema (s)
1	83.80 s	24.30 s
1	87.11 s	18.08 s
1	88.67 s	14.69 s
1	95.92 s	20.84 s
1	82.29 s	22.91 s
1	81.94 s	23.68 s

1	74.31 s	9.42 s
1	90.61 s	17.74 s
2	118.27 s	24.40 s
2	96.12 s	18.20 s
2	106.11 s	14.80 s
2	117.37 s	20.90 s
2	104.56 s	23.10 s
2	108.80 s	23.80 s
2	101.43 s	9.50 s
2	103.98 s	17.80 s
3	111.44 s	24.20 s
3	102.31 s	18.07 s
3	102.73 s	14.60 s
3	104.85 s	20.70 s
3	120.22 s	22.80 s
3	83.14 s	23.60 s
3	95.79 s	9.30 s
3	84.65 s	17.60 s

Nota. Elaboración propia.

Importante: Para no introducir sesgos debido a muestras desiguales, solo se colocaron los tiempos de reconocimiento de aquellas palabras que fueron reconocidas por el total de la muestra de las personas sordas de la Asociación De Sordos De La Libertad, sin y con el uso del sistema traductor de comunicación bidireccional.

4.3. Docimasia de hipótesis

4.3.1. Comunicación de persona sorda a oyente

a) Supuesto de normalidad

Para determinar el método estadístico adecuado para contrastar la hipótesis planteada en la comunicación de persona sorda a oyente, es necesario verificar si nuestros datos siguen o no una distribución normal. Si los datos siguen una distribución normal se utilizará la Prueba t de Student (para muestras emparejadas) ya que es una prueba paramétrica que asume que los datos se distribuyen normalmente. En caso de que los datos no cumplan con esta suposición, se utilizará la Prueba de Wilcoxon. Dicho esto, la hipótesis de normalidad es la siguiente:

Hipótesis nula (H_0): Los datos de la diferencia de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad sin y con el uso del sistema traductor de comunicación bidireccional siguen una distribución normal.

Hipótesis alterna (H_1): Los datos de la diferencia de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad sin y con el uso del sistema traductor de comunicación bidireccional no siguen una distribución normal.

El nivel de significancia (α), que es el umbral que se establece para determinar si un resultado es estadísticamente significativo, se fijó en 0.05 (5%). Esto quiere decir que si el valor de significancia (valor p), es menor que el nivel de significancia se rechaza la hipótesis nula (H_0) y se acepta la hipótesis alterna (H_1), caso contrario, no se rechaza la hipótesis nula (H_0) y no se acepta la hipótesis alterna (H_1).

$$\alpha=0.05$$

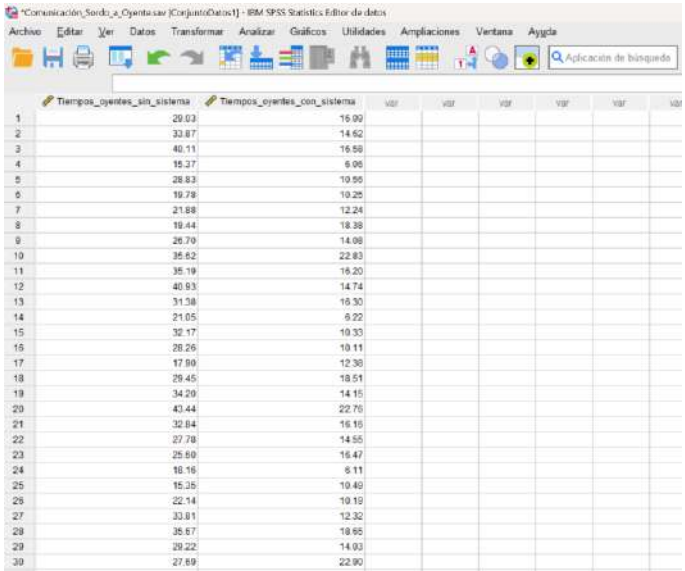
Por otro lado, el nivel de confianza (1−α) es del 95%, lo que representa la probabilidad de obtener resultados consistentes con la muestra si la hipótesis nula es verdadera.

$$1-\alpha = 0.95$$

Con todo lo mencionado anteriormente, se procedió a procesar los datos de la Tabla N°6 mediante el software IBM SPSS Statistics V29.0.2.0 (SPSS), con el fin de determinar su normalidad. Para ello, primero se ingresaron en SPSS, los tiempos de reconocimiento de las palabras de cada persona oyente de La Libertad, tanto sin como con el uso del sistema traductor de comunicación bidireccional, en dos variables separadas en 2 columnas, tal y como muestra en la Figura 34.

Figura 34

Tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics

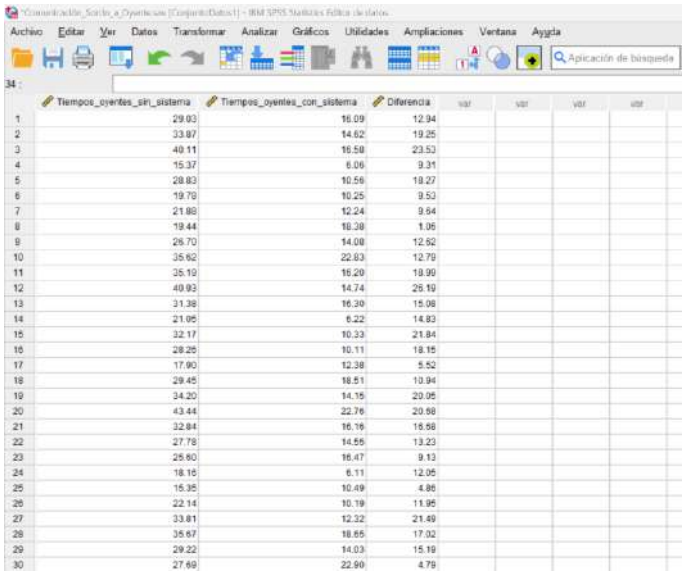


Nota. Elaboración propia.

Dado que se espera que los tiempos de reconocimiento de las palabras en cada persona oyente de La Libertad disminuyan después de usar el sistema traductor de comunicación bidireccional, se calculó la diferencia entre los tiempos de reconocimiento de las palabras sin y con el uso del sistema. Los resultados obtenidos de esta diferencia se muestran en la Figura 35.

Figura 35

Diferencia de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics



Nota. Elaboración propia.

Teniendo en cuenta los valores del nivel de significancia (α) y nivel de confianza (1−α) mencionados anteriormente y la diferencia calculada en los tiempos de reconocimiento de las palabras para cada persona oyente de La Libertad, sin y con el uso del sistema, se procedió a calcular la normalidad en SPSS. El resultado de este análisis se visualiza en la Figura 36.

Figura 36

Resultados de la prueba de normalidad de la diferencia de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Diferencia	.074	30	.200*	.987	30	.964

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors

Nota. Elaboración propia.

Interpretación: En las pruebas de normalidad realizadas en SPSS, se llevaron a cabo dos análisis: uno mediante Kolmogorov-Smirnov y otro mediante Shapiro-Wilk. Dado que el número de observaciones acerca de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad es menor a 50, se enfocó únicamente en el análisis de Shapiro-Wilk. El resultado de este análisis arrojó un valor p de 0.964. Como este valor es mayor que el nivel de significancia propuesto (α=0.05), se concluye que hay evidencia suficiente para no rechazar la hipótesis nula (H0) y no aceptar la hipótesis alterna (H1), lo que indica que los datos siguen una distribución normal.

b) Hipótesis

Dado que, en el análisis de normalidad se concluyó que la diferencia de los tiempos de reconocimiento de las palabras por parte de las personas oyentes de La Libertad sin y con el uso del sistema siguen una distribución normal, se determinó que la Prueba t de Student (para muestras emparejadas), sería la adecuada para contrastar la siguiente hipótesis:

Hipótesis nula (H_0): No hay una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas oyentes de La Libertad durante la comunicación de persona oyente a persona sorda, independientemente del uso del sistema traductor de comunicación bidireccional.

Hipótesis alterna (H_1): Hay una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas oyentes de La Libertad durante la comunicación de persona oyente a persona sorda, cuando se utiliza el sistema traductor bidireccional de comunicación bidireccional.

Al igual que en la prueba de normalidad, los valores de los niveles de significancia (α) y confianza (1−α) serán: 0.05 y 0.95 respectivamente. Esto quiere decir que, si el valor p es menor que el nivel de significancia, se rechaza la hipótesis nula (H_0) y se acepta la hipótesis alterna (H_1), caso contrario, no se rechaza la hipótesis nula (H_0) y no se acepta la hipótesis alterna (H_1)**.**

Con todo lo mencionado anteriormente y a partir de las dos variables creadas en SPSS durante la prueba de normalidad, se realizó la Prueba t de Student (para muestras emparejadas). Los resultados de este análisis se muestran en la Figura 37.

Figura 37

Resultados de la aplicación de la Prueba t (muestras emparejadas) a los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad antes y después del uso del sistema

traductor de comunicación bidireccional, a través del software IBM SPSS Statistics

Prueba T

Estadísticas de muestras emparejadas

		Media	N	Desv. estándar	Media de error estándar
Par 1	Tiempos (antes)	28.4287	30	7.58064	1.38403
	Tiempos (después)	14.1753	30	4.56005	.83255

Prueba de muestras emparejadas

		Diferencias emparejadas					t	gl	Significación	
		Media	Desv. estándar	Media de error estándar	95% de intervalo de confianza de la diferencia Inferior	Superior			P de un factor	P de dos factores
Par 1	Tiempos (antes) - Tiempos (después)	14.25333	6.04195	1.10310	11.99723	16.50943	12.921	29	<.001	<.001

Nota. Elaboración propia.

Además, debido a que SPSS no proporciona el valor crítico t como resultado del análisis de la Prueba t de Student, este se lo calculó utilizando la función "INV.T()" del software de hoja de cálculo Microsoft Excel, la cual requiere como entrada, los valores del nivel de significancia (α) y los grados de libertad (gl). El resultado de la aplicación de esta función se muestra en la Figura 38.

Figura 38

Cálculo del valor crítico t para la prueba t (muestras emparejadas) de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software de hoja de cálculo Microsoft Excel

Valor crítico:	gl = (n-1) =	29
	α =	0.05
	$t_{(1-\alpha),(n-1)}$ =	1.699127

Nota. Elaboración propia.

Interpretación: Al analizar los resultados de la tabla "Estadísticas de muestras emparejadas", se observó una disminución del 50.14% en la media de los tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad después de utilizar el sistema traductor de comunicación bidireccional. Sin embargo, para validar estadísticamente que esta reducción en los tiempos de reconocimiento es significativa, es necesario interpretar el valor t de la prueba y el valor p, los cuales se obtuvieron en la tabla “Prueba de muestras emparejadas”. Dado que el valor p (< 0.001) es inferior al nivel de significancia propuesto (α = 0.05) y el valor t (12.921) de la prueba es mayor que el valor crítico t (1.690), se concluye que hay evidencia suficiente para rechazar la hipótesis nula (H0) y aceptar la hipótesis alternativa (H1). Esto indica que existe una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas oyentes de La Libertad durante la comunicación de persona oyente a persona sorda, cuando se utiliza el sistema traductor bidireccional de comunicación bidireccional. Por lo tanto, se puede afirmar que el uso de un sistema traductor de comunicación bidireccional reduce significativamente los tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad, durante la comunicación de persona sorda a oyente.

4.3.2. Comunicación de persona oyente a sorda

a) Supuesto de normalidad

Para determinar el método estadístico adecuado para contrastar la hipótesis planteada en la comunicación de persona oyente a sorda, es necesario verificar si nuestros datos siguen o no una distribución normal. Si los datos siguen una distribución normal se utilizará la Prueba t de Student (para muestras relacionadas) ya que es una prueba paramétrica que asume que los datos se distribuyen normalmente. En caso de que los datos no cumplan con esta suposición, se utilizará la Prueba de Wilcoxon. Dicho esto, la hipótesis de normalidad es la siguiente:

Hipótesis nula (H0): Los datos de la diferencia de los tiempos de reconocimiento de las palabras sin y con el uso del sistema traductor de comunicación bidireccional siguen una distribución normal.

Hipótesis alterna (H1): Los datos de la diferencia de los tiempos de reconocimiento de las palabras sin y con el uso del sistema traductor de comunicación bidireccional no siguen una distribución normal.

El nivel de significancia (α), se fijó en un 5%. Esto quiere decir que si el valor p (grado de significancia) es menor que el nivel de significancia (0.05) se rechaza la hipótesis nula (H0) y se acepta la hipótesis alterna (H1), caso contrario, se acepta la hipótesis nula (H0) y se rechaza la hipótesis alterna (H1).

$$\alpha=0.05$$

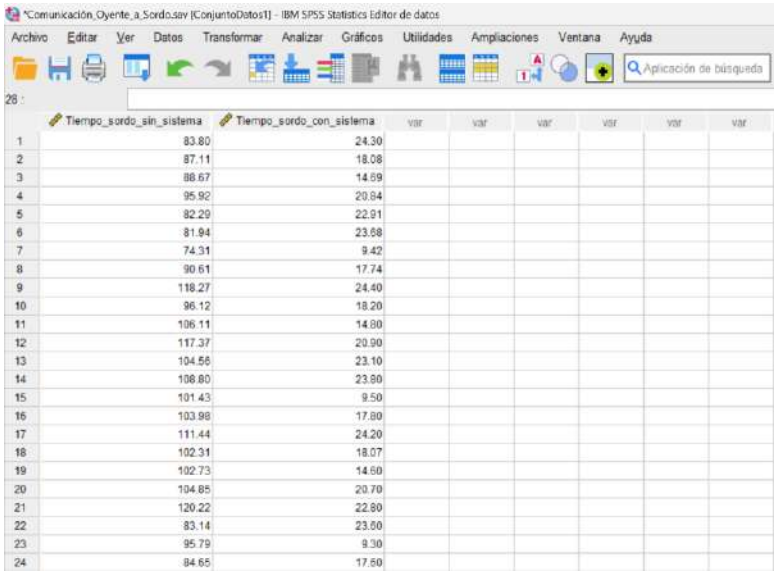
Por otro lado, el nivel de confianza (1−α) es del 95%, lo que representa la probabilidad de obtener resultados consistentes con la muestra si la hipótesis nula es verdadera.

$$1-\alpha = 0.95$$

Con todo lo mencionado anteriormente, se procedió a procesar los datos de la Tabla N°9 mediante el software IBM SPSS Statistics (SPSS), con el fin de determinar su normalidad. Para ello, primero se ingresaron en SPSS, los tiempos de reconocimiento de las palabras para cada persona sorda de la Asociación De Sordos De La Libertad, tanto sin como con el uso del sistema traductor de comunicación bidireccional, en dos variables separadas en 2 columnas, tal y como muestra en la Figura 39.

Figura 39

Tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics



	Tiempo_sordo_sin_sistema	Tiempo_sordo_con_sistema	var	var	var	var	var	var
1	83.80	24.30						
2	87.11	18.08						
3	88.67	14.69						
4	95.92	20.84						
5	82.29	22.91						
6	81.94	23.68						
7	74.31	9.42						
8	90.61	17.74						
9	118.27	24.40						
10	96.12	18.20						
11	106.11	14.80						
12	117.37	20.90						
13	104.56	23.10						
14	108.80	23.80						
15	101.43	9.50						
16	103.98	17.80						
17	111.44	24.20						
18	102.31	18.07						
19	102.73	14.60						
20	104.85	20.70						
21	120.22	22.80						
22	83.14	23.60						
23	95.79	9.30						
24	84.65	17.60						

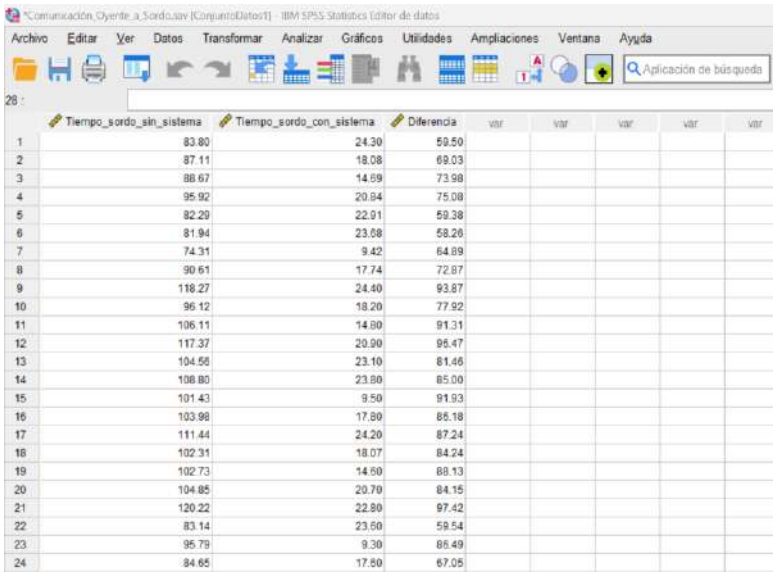
Nota. Elaboración propia.

Dado que se espera que los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad disminuyan después de usar el sistema traductor de comunicación bidireccional, se calculó la diferencia entre los tiempos de reconocimiento de las palabras, sin y con el uso del sistema. Los resultados obtenidos de esta diferencia se muestran en la Figura 40.

Figura 40

Diferencia de los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad antes y

después del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics



	Tiempo_sordo_sin_sistema	Tiempo_sordo_con_sistema	Diferencia	var	var	var	var	var
1	83.80	24.30	59.50					
2	87.11	18.08	69.03					
3	88.67	14.69	73.98					
4	95.92	20.84	75.08					
5	82.29	22.91	59.38					
6	81.94	23.68	58.26					
7	74.31	9.42	64.89					
8	90.61	17.74	72.87					
9	118.27	24.40	93.87					
10	96.12	18.20	77.92					
11	106.11	14.80	91.31					
12	117.37	20.90	96.47					
13	104.56	23.10	81.46					
14	108.80	23.80	85.00					
15	101.43	9.50	91.93					
16	103.98	17.80	86.18					
17	111.44	24.20	87.24					
18	102.31	18.07	84.24					
19	102.73	14.60	88.13					
20	104.85	20.70	84.15					
21	120.22	22.80	97.42					
22	83.14	23.60	59.54					
23	95.79	9.30	86.49					
24	84.65	17.60	67.05					

Nota. Elaboración propia.

Teniendo en cuenta los valores del nivel de significancia (α), el nivel de confianza (1−α) y la diferencia calculada en los tiempos de reconocimiento de las palabras para cada persona sorda de la Asociación De Sordos De La Libertad sin y con el uso del sistema, se procedió a calcular la normalidad en SPSS. El resultado de este análisis se visualiza en la Figura 41.

Figura 41

Resultados de la prueba de normalidad de la diferencia de los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad antes y después del uso

del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Diferencia	.165	24	.092	.930	24	.097

a. Corrección de significación de Lilliefors

Nota. Elaboración propia.

Interpretación: En las pruebas de normalidad realizadas en SPSS, se llevaron a cabo dos análisis: uno mediante Kolmogorov-Smirnov y otro mediante Shapiro-Wilk. Dado que el número de observaciones acerca de los tiempos de reconocimiento de las palabras para cada persona sorda de la Asociación De Sordos De La Libertad es menor a 50, se enfocó únicamente en el análisis de Shapiro-Wilk. El resultado de este análisis arrojó un valor p de 0.097. Como este valor es mayor que el nivel de significancia propuesto (α=0.05), se concluye que hay evidencia suficiente para no rechazar la hipótesis nula (H0) y no aceptar la hipótesis alterna (H1), lo que indica que los datos siguen una distribución normal.

b) Hipótesis

Dado que, en el análisis de normalidad se concluyó que la diferencia de los tiempos de reconocimiento de las palabras en cada persona sorda de la Asociación De La Libertad sin y con el uso del sistema traductor de comunicación bidireccional siguen una distribución normal, se determinó que la Prueba t de Student (para muestras relacionadas), sería la adecuada para contrastar la siguiente hipótesis:

Hipótesis nula (H_0): No hay una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas sordas de la Asociación De Sordos De La Libertad durante la

comunicación de persona sorda a persona oyente, independientemente del uso del sistema traductor bidireccional.

Hipótesis alterna (H_1): Hay una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas sordas de la Asociación De Sordos De La Libertad durante la comunicación de persona sordo a persona oyente, cuando se utiliza el sistema traductor bidireccional de comunicación bidireccional.

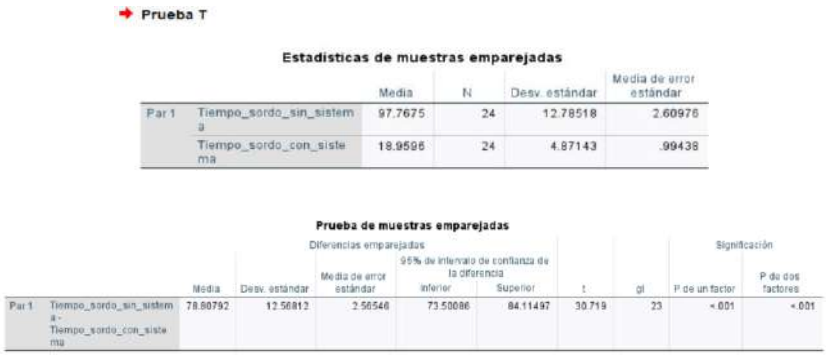
Al igual que en la prueba de normalidad, los valores de los niveles de significancia (α) y confianza (1−α) serán: 0.05 y 0.95 respectivamente. Esto quiere decir que, si el valor p es menor que el nivel de significancia, se rechaza la hipótesis nula (H0) y se acepta la hipótesis alterna (H1), caso contrario, no se rechaza la hipótesis nula (H0) y no se acepta la hipótesis alterna (H1).

Con todo lo mencionado anteriormente y a partir de las dos variables creadas en SPSS durante la prueba de normalidad, se realizó un análisis en SPSS, utilizando la Prueba t de Student (para muestras relacionadas). Los resultados de este análisis se presentan en la Figura 42.

Figura 42

Resultados de la aplicación de la Prueba t (muestras emparejadas) a los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad antes y después

del uso del sistema traductor de comunicación bidireccional, a través del software IBM SPSS Statistics



Prueba T

Estadísticas de muestras emparejadas

		Media	N	Desv. estándar	Media de error estándar
Par 1	Tiempo_sordo_sin_sistema	97.7675	24	12.78518	2.60976
	Tiempo_sordo_con_sistema	18.9596	24	4.87143	.99438

Prueba de muestras emparejadas

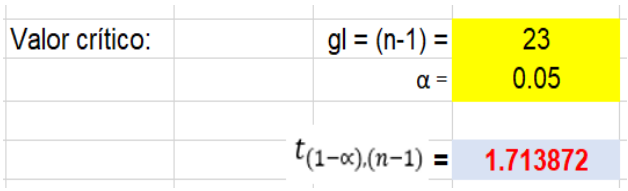
		Diferencias emparejadas					t	gl	Significación	
		Media	Desv. estándar	Media de error estándar	95% de intervalo de confianza de la diferencia Inferior	Superior			P de un factor	P de dos factores
Par 1	Tiempo_sordo_sin_sistema - Tiempo_sordo_con_sistema	78.80792	12.56812	2.56548	73.50086	84.11497	30.719	23	< .001	< .001

Nota. Elaboración propia.

Además, debido a que SPSS no proporciona el valor crítico t como resultado del análisis de la Prueba t de Student, este se lo calculó utilizando la función "INV.T()" del software de hoja de cálculo Microsoft Excel, la cual requiere como entrada, los valores del nivel de significancia (α) y los grados de libertad (gl). El resultado de la aplicación de esta función se muestra en la Figura 43.

Figura 43

Cálculo del valor crítico t para la prueba t (muestras emparejadas) de los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad antes y después del uso del sistema traductor de comunicación bidireccional, a través del software de hoja de cálculo Microsoft Excel



Valor crítico:		gl = (n-1) =	23
		α =	0.05
		$t_{(1-\alpha),(n-1)}$ =	1.713872

Nota. Elaboración propia.

Interpretación: Al analizar los resultados de la tabla "Estadísticas de muestras emparejadas", se observó una disminución del 80.61% en la media de los tiempos de reconocimiento de las palabras en las personas oyentes de La Libertad después de utilizar el sistema traductor de comunicación bidireccional. Sin embargo, para validar estadísticamente que esta reducción es significativa, es necesario interpretar el valor t de la prueba y el valor p, los cuales se obtuvieron en la tabla “Prueba de muestras emparejadas”. Dado que el valor p (< 0.001) es inferior al nivel de significancia propuesto (α = 0.05) y el valor t (30.719) de la prueba es mayor que el valor crítico t (1.690), se concluye que hay evidencia suficiente para rechazar la hipótesis nula (H0) y aceptar la hipótesis alternativa (H1). Esto indica que existe una diferencia significativa en los tiempos de reconocimiento de las palabras por parte de las personas sordas de la Asociación De Sordos De La Libertad durante la comunicación de persona oyente a persona sorda, cuando se utiliza el sistema traductor bidireccional de comunicación bidireccional. Por lo tanto, se puede afirmar que el uso de un sistema traductor de comunicación bidireccional reduce significativamente los tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad, durante la comunicación de persona sorda a oyente.

V. DISCUSIÓN DE LOS RESULTADOS

- Con respecto al objetivo 1, los resultados arrojaron una disminución del 50.14% en la media de los tiempos de reconocimiento de las palabras de las personas oyentes de La Libertad, debido al uso del sistema traductor de comunicación bidireccional. Asimismo, a través de la prueba t de Student (para muestras relacionadas), se obtuvo un valor p menor que 0.05 (nivel de significancia) y un valor t mayor que el nivel crítico t, los cuales nos permiten rechazar la hipótesis nula (H0) y aceptar la hipótesis alternativa (H1), confirmando que el uso de un sistema traductor de comunicación bidireccional reduce significativamente los tiempos de reconocimiento de las palabras en las personas oyentes de La libertad. Este hallazgo se alinea con la investigación Montenegro y Villa (2019), quienes demostraron, a través de los resultados obtenidos, una diferencia de 1.28 min en el tiempo promedio de comunicación de un estudiante con discapacidad auditiva en el Colegio Bautista para Sordos Harvest, al comparar el uso y la ausencia de un sistema inteligente de reconocimiento del lenguaje de señas peruano.

- Con respecto al objetivo 2, los resultados arrojaron una disminución del 80.61% en la media de los tiempos de reconocimiento de las palabras de las personas sordas de la Asociación De Sordos De La Libertad, debido al uso del sistema traductor de comunicación bidireccional. Asimismo, a través de la prueba t de Student (para muestras relacionadas), se obtuvo un valor p menor que 0.05 (nivel de significancia) y un valor t mayor que el nivel crítico t, los cuales nos permiten rechazar la hipótesis nula (H_0) y aceptar la hipótesis alternativa (H_1), confirmando que el uso de un sistema traductor de comunicación bidireccional reduce significativamente los tiempos de reconocimiento de las palabras en las personas sordas de la Asociación De Sordos De La Libertad con el uso del sistema traductor de comunicación bidireccional. Este hallazgo se alinea con la investigación de Vilchez y Rommel (2015), quienes demostraron también a través del análisis estadistico Wilconxon, un valor P menor que 0.05, lo cual les permite rechazar la hipotesis nula (H_0) y aceptar la hipótesis alternativa (H_1), demostrando

que el tiempo promedio de comunicación de las personas sordas con el sistema actual es mayor que con el sistema propuesto, es decir, el tiempo que una persona sorda tarda en entender a una persona oyente es significativamente menor cuando se utiliza el sistema propuesto.

CONCLUSIONES

- Se evaluó el efecto del sistema traductor de comunicación bidireccional en los tiempos de reconocimiento de las personas oyentes de La Libertad para las palabras expresadas a través del lenguaje de señas peruano por las personas sordas de la Asociación De Sordos De La Libertad, en el que se determinó a través del análisis estadístico de la prueba t de Student para muestras emparejadas, que hubo una reducción significativa en los tiempos de reconocimiento de las palabras debido al uso del sistema traductor de comunicación bidireccional.

- Se evaluó el efecto del sistema traductor de comunicación bidireccional en los tiempos de reconocimiento de las personas sordas de la Asociación De Sordos De La Libertad para las palabras pronunciadas por las personas oyentes de La Libertad, en el que se determinó a través del análisis estadístico de la prueba t de Student para muestras emparejadas, que hubo una reducción significativa en los tiempos de reconocimientos de las palabras debido al uso del sistema traductor de comunicación bidireccional.

RECOMENDACIONES

- A pesar de que se reduce en gran parte el ruido mediante la extracción de la mano a través de los 21 puntos de referencia con un fondo blanco, se recomienda ubicarse en un ambiente con una iluminación moderada, ya que al fin y al cabo nuestro sistema antes de realizar la extracción de la mano mediante los 21 puntos de referencia, recoge una copia del frame original, y si ese frame original se ve afectado por el ruido del ambiente, la detección de la mano también se verá afectada.

- Para obtener una predicción acertada acerca de una de las señas del alfabeto dactilológico del Lenguaje de Señas Peruana, se recomienda estar situado delante y al centro de una cámara web con una distancia prudente, donde se pueda visualizar toda la dimensión de la mano con la que se va a realizar la seña.

- Para futuras investigaciones, se recomienda entrenar un modelo de aprendizaje profundo (Deep Learning) utilizando un conjunto de datos extenso que incluya una diversidad representativa de personas sordas. Esto permitirá al modelo generalizar de manera más efectiva y reconocer gestos realizados por diferentes tipos de individuos dentro de la comunidad sorda.

- Para futuras investigaciones, se recomienda explorar técnicas avanzadas de seguimiento y reconocimiento de gestos dinámicos, específicamente en el contexto de letras o palabras en movimiento dentro del Lenguaje de Señas Peruana. Esto podría incluir el desarrollo de modelos de aprendizaje profundo que integren el seguimiento temporal y espacial de gestos para capturar y reconocer con precisión secuencias de gestos dinámicos.

- Para futuras investigaciones, se recomienda mejorar el sistema traductor de comunicación bidireccional para que sea capaz de mostrar en un único video gestos que representen una palabra completa en lugar de limitarse al deletreo en señas. Esto con la finalidad de poder reducir aún más los tiempos de reconocimiento de las palabras en una persona sorda.

- Para futuras investigaciones, se recomienda mejorar el sistema traductor de comunicación bidireccional para que pueda integrarse como una aplicación móvil en teléfonos celulares, en lugar de estar limitado a una página web accesible desde una computadora. Esto permitiría que personas sordas y oyentes utilicen diariamente una aplicación que facilite la comunicación entre ambos grupos.

REFERENCIAS BIBLIOGRÁFICAS

Akto. (2023). *POST Method*. Obtenido de https://www.akto.io/academy/post

Akto. (2023). *What is GET Method?* Obtenido de https://www.akto.io/academy/get

Alammar, J. (12 de mayo de 2022). *NumPy: the absolute basics for beginners*. Obtenido de https://numpy.org/doc/stable/user/absolute_beginners.html

Ámos, D. (2019). *La guía definitiva para el reconocimiento de voz con Python*. Obtenido de Real Python: https://realpython.com/python-speech-recognition/

Aranda Garay, A., & Vargas Benites, V. (2020). *Propuesta de solución para la gestión remota de terapias del habla de pacientes postoperatorios de labio leporino utilizando Speech Recognition y Gamificacion.* Universidad Peruana De Ciencias Aplicadas, Lima. Obtenido de https://repositorioacademico.upc.edu.pe/bitstream/handle/10757/655646/Aranda_GA.pdf?sequence=3&isAllowed=y

Arevalo Miró Quesada, J. A. (29 de setiembre de 2022). *Día Mundial de las Personas Sordas: ¿Dónde aprender la lengua de señas peruana?* Obtenido de Comercio: https://elcomercio.pe/casa-y-mas/dia-mundial-de-las-personas-sordas-donde-aprender-la-lengua-de-senas-peruana-rmmn-emcc-noticia/

Barba Guamán, L. (2021). *Uso de técnica deep learning para reconocimiento de objetos en áreas rurales.* Universidad Politécnica de Madrid. Recuperado el 20 de agosto de 2023, de https://oa.upm.es/67969/1/LUIS_RODRIGO_BARBA_GUAMAN_01.pdf

Bhandari, A. (19 de Febrero de 2024). *Analytics Vidhya*. Obtenido de https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/#:~:text=ImageDataGenerator%20class%20ensures%20that%20the,the%20original%20corpus%20of%20images.

Bosch, G. (2023). *Image Recognition: The Basics and Use Cases (2023 Guide)*. Obtenido de Viso.ai: https://viso.ai/computer-vision/image-recognition/

Brownlee, J. (12 de Septiembre de 2020). *Machine Learning Mastery*. Obtenido de https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Brownlee, J. (13 de enero de 2021). *Machine learning mastery*. Obtenido de https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Centros para el Control y la Prevención de Enfermedades. (1 de julio de 2020). *Tipos de pérdida auditiva*. Obtenido de https://www.cdc.gov/ncbddd/spanish/hearingloss/types.html

Clark, J. (23 de mayo de 2023). *¿Qué es la clasificación de imágenes?* Obtenido de SuperAnnotate: https://www.superannotate.com/blog/image-classification-basics

Colaboradores Mdn. (2023). *MDN Web Docs*. Obtenido de https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server

Darshansol. (03 de enero de 2023). *Creating a Finger Counter Using Computer Vision and OpenCv in Python*. Obtenido de GeeksforGeeks: https://www.geeksforgeeks.org/creating-a-finger-counter-using-computer-vision-and-opencv-in-python/#article-meta-div

Dawe, D. (11 de febrero de 2022). *Creating a Hand Tracking Module using Python, OpenCv, and MediaPipe*. Obtenido de Section: https://www.section.io/engineering-education/creating-a-hand-tracking-module/

Deep, A., Litoriya, A., Ingole, A., Asare, V., M Bhle, S., & Pathak, S. (11 de octubre de 2022). Realtime Sign Language Detection and Recognition. *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, 1-4. doi:10.1109/ASIANCON55314.2022.9908995

Deepchecks. (2024). *Deepcheks*. Obtenido de https://deepchecks.com/glossary/learning-rate-in-machine-learning/

Defensoría del Pueblo. (24 de setiembre de 2020). *Defensoría del Pueblo: debe facilitarse el aprendizaje de la lengua de señas peruana y promover la identidad lingüística y cultural de las personas sordas*. Obtenido de 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/

Diederik P. Kingma, J. B. (2019). *Arxiv*. Obtenido de https://arxiv.org/abs/1412.6980

Digital samba. (27 de marzo de 2024). *Digitalsamba*. Obtenido de https://www.digitalsamba.com/blog/websocket-vs-http

Dirección General de Servicios Educativos Especializados. (2019). *Lengua de señas peruana guía para el aprendizaje de la lengua de Señas Peruana*. Obtenido de Ministerio de Educación: https://repositorio.minedu.gob.pe/handle/20.500.12799/5545

Extrahop. (2024). *Extrahop*. Obtenido de https://hop.extrahop.com/resources/protocols/http/

FEDERACIÓN DE PERSONAS SORDAS DE LA COMUNIDAD DE MADRID. (2019). *Definición de LSE*. Obtenido de Fesocarm: https://www.fesorcam.org/definicion-de-lse/

Figueroa Carlos, B. (2021). *Desarrollo de una Solución mediante el Uso de Speech Recognition y Generative Adversarial Networks para la Generación Automática del Retrato Hablado.* Universidad Nacional Hermilio Valdizán, Huánuco, Perú. Obtenido de https://repositorio.unheval.edu.pe/handle/20.500.13080/6738

Gálves, E. (15 de noviembre de 2022). *6 Reglas que desconoces sobre la lengua de signos*. Obtenido de audiolis: https://www.audiolis.com/cursos-de-formacion/blog/6-reglas-que-desconoces-sobre-el-lenguaje-de-signos/

Gardey, A., & Perez Porto, J. (12 de mayo de 2022). *Definición.DE*. Obtenido de Imagen vectorial - Qué es, usos, definición y concepto: https://definicion.de/imagen-vectorial/

Google for Developers. (9 de mayo de 2023). *MediaPipe Solutions guide*. Obtenido de https://developers.google.com/mediapipe/solutions/guide

Healthdirect. (22 de julio de 2022). *Pérdida de la audición*. Obtenido de https://www.healthdirect.gov.au/hearing-loss

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision. *Cornell University*.

Hualde. (15 de enero de 2023). *05:Tutorial visión artificial con OpenCV y Python: Deteccion de objetos y seguimiento de movimiento*. Obtenido de ROS Tutorial: https://rostutorial.com/vision-artificial-deteccion-seguimiento/

Iberdrola. (15 de octubre de 2020). *¿Que es el machine Learning?* Obtenido de Iberdrola: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico

International Business Machines Corporation. (2020). *¿Qué es la inteligencia artificial?* Obtenido de IBM: https://www.netapp.com/es/artificial-intelligence/what-is-artificial-intelligence/

International Business Machines Corporation. (2020). *¿Que es la visión artificial?* Obtenido de https://www.ibm.com/cl-es/topics/computer-vision

International Business Machines Corporation. (2020). *What is deep learning?* Recuperado el 21 de Agosto de 2023, de IBM: https://www.ibm.com/topics/deep-learning

International Business Machines Corporation. (2022). *¿Qué son las redes neuronales?* Obtenido de IBM: https://www.ibm.com/topics/neural-networks

Ionos. (2023). *Digital Guide Ionos*. Obtenido de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/flask/

Jimenez Moreano, A., & Qquecho Ccachainca, B. A. (2020). *PROTOTIPO DE TRADUCTOR DE LENGUAJE DE SEÑAS.* Universidad Andina del Cuzco, Cuzco, Perú. Recuperado el 22 de noviembre de 2022

JSON. (s.f.). *json.org*. Obtenido de https://www.json.org/json-en.html

Karabiber, F. (2023). *Clasificación binaria*. Obtenido de LearnDataSci: https://www.learndatasci.com/glossary/binary-classification/#:~:text=each%20binary%20classifier-,What%20is%20Binary%20Classification%3F,Application

KeepCoding, R. (15 de Diciembre de 2022). *keepcoding tech school*. Obtenido de https://keepcoding.io/blog/que-es-jsonify-en-flask/#Que_es_jsonify_en_Flask

Keras. (s.f.). *MobileNet, MobileNetV2, and MobileNetV3*. Obtenido de Keras: https://keras.io/api/applications/mobilenet/

Keras. (s.f.). *Optimizers*. Obtenido de K Keras: https://keras.io/api/optimizers/

Khalkar, R., Singh Dikhit, A., & Goel, A. (2021). *Handwritten Text Recognition using Deep Learning (CNN & RNN).* Obtenido de https://www.researchgate.net/figure/Venn-Diagram-for-AI-ML-NLP-DL_fig1_353939315

Kiversal. (12 de abril de 2019). *Sordos o sordomudos: uso correcto del lenguaje inclusivo.* Obtenido de Blog de Kiversal: https://blog.kiversal.com/sordos-lenguaje-inclusivo/

Llamas, J. (14 de octubre de 2021). *Sistema de Comunicacion.* Obtenido de Economipedia: https://economipedia.com/definiciones/sistemas-de-comunicacion.html

Londoño, P. (2023). *Inteligencia artificial: qué es, cómo funciona e importancia en 2023*. Obtenido de HubSpot: https://blog.hubspot.es/marketing/inteligencia-articial-esta-aqui

Lopez Roca, K. A. (2018). *Aplicación móvil de interpretación del lenguaje de señas peruanas para discapacitados auditivos en la Asociacion de Sordos de la región Lima.* Universidad César Vallejo, Lima, Perú. Recuperado el 15 de agosto de 2023, de https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/38179/Lopez_RK.pdf?sequence=1&isAllowed=y

MathWorks. (2023). *¿Qué es el reconocimiento de imágenes?* Obtenido de https://la.mathworks.com/discovery/image-recognition-matlab.html

MDN contributors. (24 de Julio de 2023). *HTML: Lenguaje de etiquetas de hipertexto*. Obtenido de Resources for Developers: https://developer.mozilla.org/es/docs/Web/HTML

MDN Contributors. (18 de Julio de 2023). *Resources for Developers*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/CSS_basics

Mohseni Dargah, M., Falahati, Z., Dabirmanesh, B., Nasrollahi, P., & Khajeh, K. (2022). Artificial Intelligence and Data Science in Environmental Sensing. 269-298.

Montenegro Cachay, C. A., & Villa Rodriguez, D. R. (2019). *Sistema inteligente de reconocimiento de lenguaje de señas peruano para mejorar la comunicación entre las personas sordomudas de la Institución Educativa Bautista para sordos Harvest en Chiclayo.* Universidad Nacional Pedro Ruiz Gallo, Chiclayo, Perú. Recuperado el 20 de agosto de 2023, de https://repositorio.unprg.edu.pe/bitstream/handle/20.500.12893/8207/BC-4599%20MONTENEGRO%20CACHAY-VILLA%20RODRIGUEZ.pdf?sequence=1&isAllowed=y

Nabi, J. (18 de diciembre de 2018). *Aprendizaje automático: clasificación multiclase con conjunto de datos desequilibrado*. Obtenido de Medium: https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a

National Institute on Aging. (20 de noviembre de 2018). *Hearing Loss: A Common Problem for Older Adults*. Obtenido de National Institute on Aging: https://www.nia.nih.gov/health/hearing-loss-common-problem-older-adults

Neira, J. (2022). *Imágenes digitales, tipos y características*. Obtenido de Creativos Online: https://www.creativosonline.org/imagenes-digitales-tipos-caracteristicas.html

Nurfirdausi, A. F., Soekirno, S., & Aminah, S. (2021). Implementation of Single Shot Detector (SSD) MobileNet V2 on Disabled Patient's Hand Gesture Recognition as a Notification System. *2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 1-6. doi:10.1109/ICACSIS53237.2021.9631333

O'Connor, R. (22 de abril de 2022). *MediaPipe for Dummies*. Obtenido de Assembly AI: https://www.assemblyai.com/blog/mediapipe-for-dummies/

Organización Mundial de la Salud. (2 de marzo de 2021). *Sordera y pérdida de la audición*. Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss#:~:text=Se%20dice%20que%20alguien%20sufre,%2C%20moderada%2C%20grave%20o%20profunda.

Otiniano, A. (25 de setiembre de 2022). *La Población Sorda Es Educada En El Lenguaje De Señas Siempre Y Cuando Tengan Los Medios Para Hacerlo*. Obtenido de SienteTrujillo: https://sientetrujillo.com/poblacion-sorda-es-educada-en-lenguaje-de-senas-si-tiene-los-medios-para-hacerlo/

Plataforma digital unica del Estado Peruano. (25 de marzo de 2022). *¿Por qué es importante la lengua de señas o el idioma de las personas sordas?* Obtenido de Plataforma digital unica del Estado Peruano: https://www.gob.pe/institucion/munisullana/noticias/595026-por-que-es-importante-la-lengua-de-senas-o-el-idioma-de-las-personas-sordas

Quijada Lovaton, K. J. (23 de diciembre de 2021). La importancia del emprendimiento en la comunidad sorda peruana: Un reto para la inclusión. *Gestión en el Tercer Milenio*, 113-120. doi:https://orcid.org/0000-0002-6752-4701

Ravikiran. (13 de febrero de 2023). *A Guide to Speech Recognition in Python: Everything You Should Know*. Obtenido de Simplearn: https://www.simplilearn.com/tutorials/python-tutorial/speech-recognition-in-python#:~:text=Speech%20recognition%20is%20a%20machine's,respond%20to%20these%20spoken%20words.

Rodriguez, H. (27 de abril de 2021). *¿Que es OpenCV?Descubre todo acerca de la vision artificial.* Obtenido de crehana: https://www.crehana.com/blog/desarrollo-web/que-es-opencv/

Saavedra, N. (1 de diciembre de 2022). ¿Por qué las personas sordas tienen dificultades para leer y escribir en español? *Diario La República*. Recuperado el 15 de agosto de 2023, de https://larepublica.pe/sociedad/2022/11/30/por-que-las-personas-sordas-tienen-dificultades-para-leer-y-escribir-el-espanol

Sadiq, R., Rodriguez, M., & R.Mian, H. (2019). Encyclopedia of Environmental Health (Second Edition). 282-295.

Sánchez Perdomo, A. &. (2022). *Sistema de reconocimiento del alfabeto dactilológico colombiano completo por medio de visión artificial.* Colombia.

Shooken, M. (23 de febrero de 2023). *Módulos Matemáticos en Python: Math y Cmath*. Obtenido de envatotuts+: https://code.tutsplus.com/es/mathematical-modules-in-python-math-and-cmath--cms-26913t

Vilchez Sandoval, R. K. (2015). Sistema intérprete de Lenguaje Alternativo para mejora la comunicación de la personas sordas en la Asociación De Sordos De La Libertad. *Innovación en ingeniería*, 3-16.

Villaverde, C. (25 de marzo de 2022). *Comunicación bidireccional: qué es, ventajas, ejemplos y cómo conseguirla en redes sociales*. Obtenido de InBoundcycle: https://www.inboundcycle.com/blog-de-inbound-marketing/comunicacion-bidireccional-que-es-y-como-conseguirla-en-redes-sociales

wang, p., Yin, N., & Sujatha, K. (11 de diciembre de 2022). A Web-Based Audio-to-Sign Language Converter for Chinese National Sign Language. *2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*. Recuperado el 13 de agosto de 2023

Werner, D. (1 de febrero de 2021). *El Niño campesino deshabilitado.* Berkeley, California, Estados Unidos: Hesperian – Guías de salud. doi:978-0-942364-07-1

Zimmermann Casado, M. (2014). *Sistema de recolección de objetos mediante visión artificial y planificación automática.* Universidad Carlos III de Madrid, Madrid.

ANEXOS

Anexo 1: Carne de registro del CONADIS de Renan Miguel Begazo Torres (Persona sorda 1)

Anexo 2: Carne de registro del CONADIS de Sofia Raquel León Saldaña (Persona sorda 2)

Anexo 3: Carne de registro del CONADIS de Mirla Consuelo Castillo Abad (Persona sorda 3)

Anexo 4: Documento Nacional de Identidad (DNI) de Anthony Paul Panta Rivera (Persona oyente 1)

Anexo 5: Documento Nacional de Identidad (DNI) de Milagros Stefany Sánchez Aguilar (Persona oyente 2)

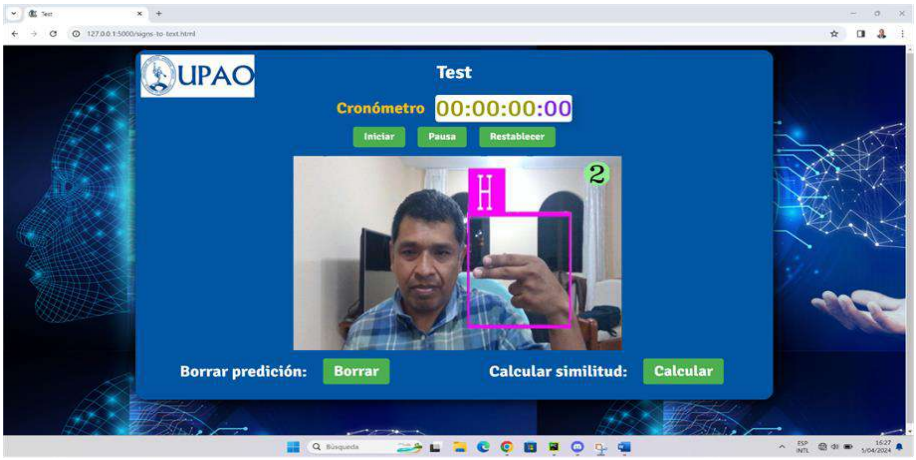
Anexo 6: Documento Nacional de Identidad (DNI) de Marcos Miguel Paredes Carrasco (Persona oyente 3)

Anexo 7: Muestra de Encuestados en Fotografía

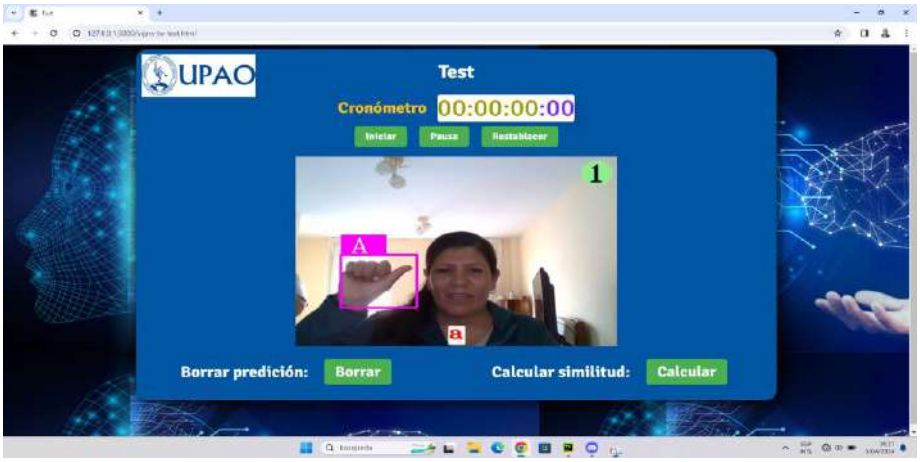
Anexo 8: Abecedario del Lenguaje de Señas Peruano

Nota. Alfabeto dactilológico peruano. Tomado de (Dirección General de Servicios Educativos Especializados, 2019)

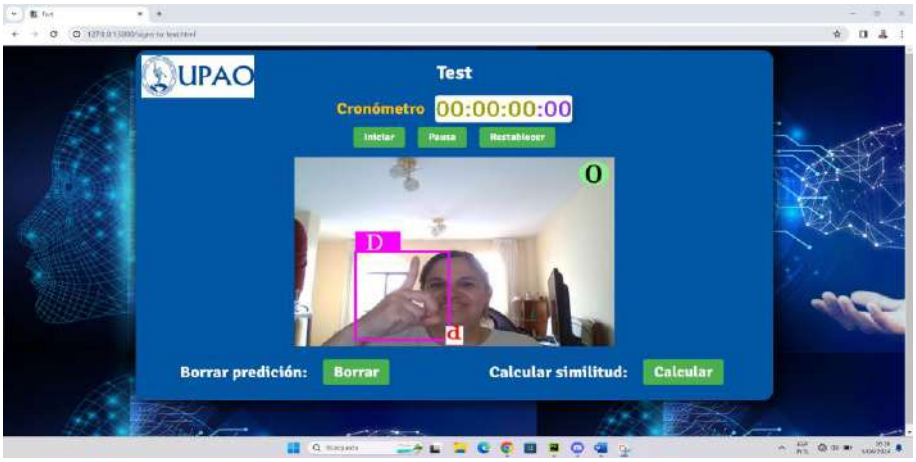
Anexo 9: Traduciendo a texto mediante el sistema traductor de comunicación bidireccional la seña realizada de la letra H por el Sr. Renan Miguel Begazo Torres frente a una cámara web



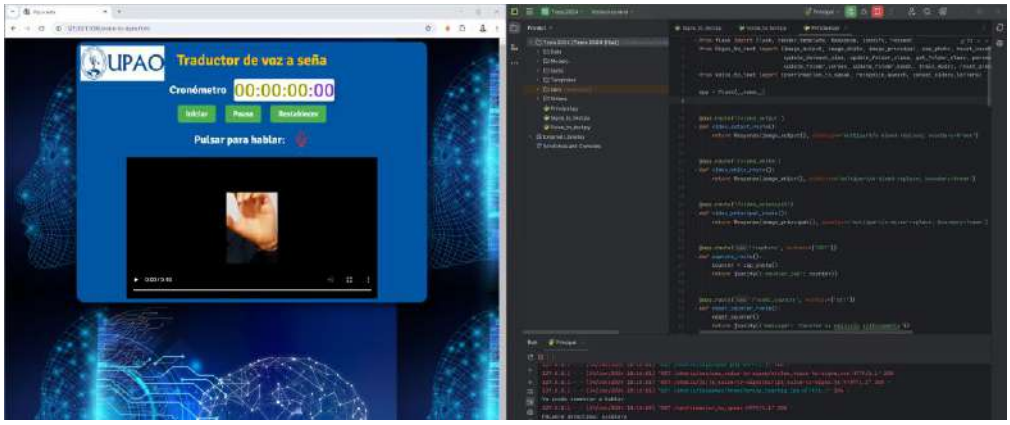
Anexo 10: Traduciendo a texto mediante el sistema traductor de comunicación bidireccional la seña realizada de la letra A por la Sra. Sofía Raquel León Saldaña frente a una cámara web.



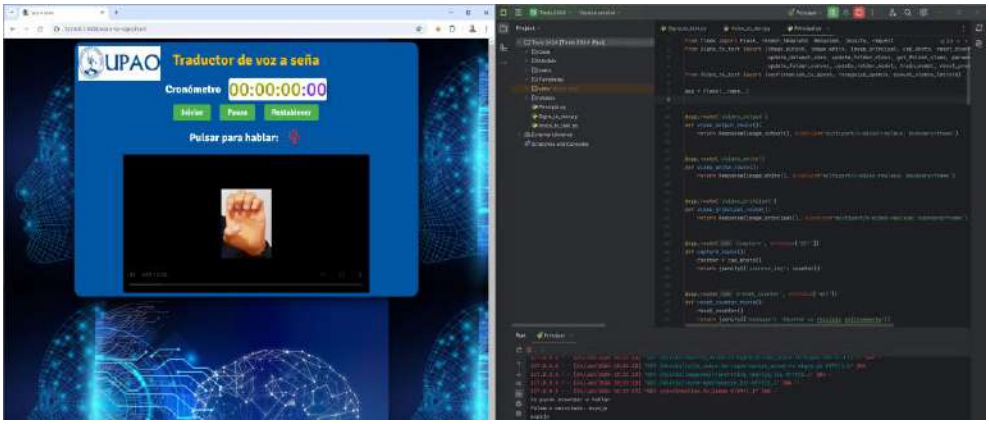
Anexo 11: Traduciendo a texto mediante el sistema traductor de comunicación bidireccional la seña realizada de la letra D por la Sra. Castillo Abad Mirla Consuelo frente a una cámara web.



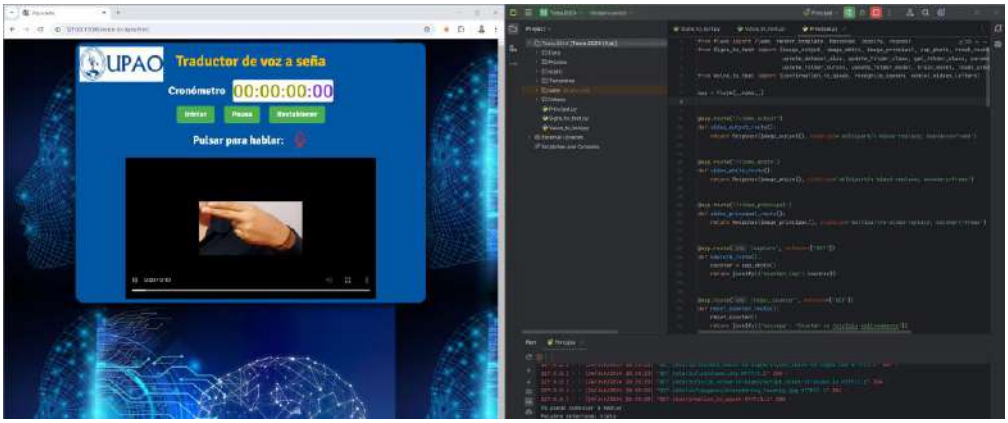
Anexo 12 Traduciendo a señas (video) mediante el sistema traductor de comunicación bidireccional la palabra pronunciada (“escalera”) por el Sr. Anthony Paúl Panta Rivera a través de un micrófono de auricular



Anexo 13 Traduciendo a señas (video) mediante el sistema traductor de comunicación bidireccional la palabra pronunciada (“espejo”) por la Sra. Milagros Stefany Sánchez Aguilar a través de un micrófono de auricular



Anexo 14 Traduciendo a señas (video) mediante el sistema traductor de comunicación bidireccional la palabra pronunciada (“hielo”) por el Sr. Marcos Miguel Paredes Carrasco a través de un micrófono de auricular



ANEXO 15: CUESTIONARIO DE ENCUESTA PARA DETERMINAR LA REALIDAD PROBLEMÁTICA DE LAS PERSONAS SORDAS EN LA ASOCIACIÓN DE SORDOS DE LA LIBERTAD

Buenos tardes: Soy encuestador de opinión data (Nombre: Kristhian Martin Panta Rivera, estudiante de la carrera Ingeniería Electrónica (UPAO), ID:000201211), hoy junto con mi compañero (Nombre: Jordan Luiggi Millán Bermejo, estudiante de la carrera Ingeniería Electrónica (UPAO), ID: 000202894) estamos haciendo una encuesta para nuestro trabajo de investigación (tesis): “Sistema de traductor de comunicación bidireccional para mejorar la comunicación con las personas sordas de la Asociación De Sordos De La Libertad” y nos gustaría contar con su opinión.

EVALUACIÓN DEL CONCEPTO

A continuación, le voy a mostrar una serie de preguntas en las cuales se busca determinar la realidad problemática de las personas con discapacitad auditiva y comunicativa de la Asociación De Sordos De La Libertad. Quisiera que la lea con detenimiento para que luego responda de manera correcta.

¿Cuál es tu rol en la asociación?

RESPUESTA: Soy información

¿Cuál es la misión y visión de la Asociación De Sordos De La Libertad?

RESPUESTA: Videos las lenguas señas enseñar a ellos sordos.

¿Es usted una persona sorda/a con discapacidad auditiva total? (Marca con una X)

SI (X)

NO ( )

¿Utiliza la lengua de señas peruana? (Marca con una X)

SI (X)

NO ( )

¿Aparte del lenguaje de señas, que otros métodos usas para comunicarte con las personas que no son sordas?

RESPUESTA: Labios, pero costar difícil entender muchas veces.

¿Con qué nivel de dificultad diría usted que puede comprender el significado de lo que le dicen los demás? (Marca con una X)

Con dificultad moderada ( )

Con dificultad severa (X)

No puede realizar la actividad ( )

¿Con qué nivel de dificultad diría que puede comprender y expresarse a través del lenguaje escrito? (Marca con una X)

Con dificultad moderada ( )

Con dificultad severa (X)

No puede realizar la actividad ( )

¿Cómo te enteras de las noticias e información relevante en tu comunidad? ¿Te apoyas de algún medio de comunicación o de alguien para esto?

RESPUESTA: Si a todos información, pero a veces otros sordos no entiendo.

¿Cuáles son los principales desafíos que enfrentas en tu vida diaria como persona sordomuda? ¿y cómo afecta tu sordera a tus relaciones familiares y personales?

RESPUESTA: Si labios la familia tal vez.

¿Sientes que tienes igualdad de oportunidades en comparación con las personas oyentes?

RESPUESTA: Tal vez diferente.

¿Qué obstáculos has encontrado en tu experiencia educativa como persona sordomuda?

RESPUESTA: Si algunos la profe habla no escuchaba, a compañero presta cuaderno copia después.

¿Cuáles son las barreras que has enfrentado al buscar empleo?

RESPUESTA: Muchas, no es fácil y trabajos no iguales que oyente.

¿Has vivido experiencias de discriminación por parte de otras personas o servicio público?

RESPUESTA: Si falta respeto a ellos oyente.

¿Has tenido alguna dificultad para comunicarte cuando has acudido a un servicio públicos (bancos, supermercados, restaurantes, etc.)? ¿En cuál de ellos se te dificulta más poder comunicarte? Explica mediante un ejemplo o un caso que has podido experimentar.

RESPUESTA: Si muchas veces, falta interprete señas.

¿Has recibido apoyo en situaciones de emergencia o desastres naturales?

RESPUESTA: Yo solo y tal vez ayuda mi hijastra.

¿Has tenido dificultades para encontrar intérpretes de lengua de señas cuando los necesitas?

RESPUESTA: Si, ninguna institución pública interprete hay.

Supóngase que usted acude a un servicio público, ¿cuánto tiempo aproximadamente le toma poder comunicarse o expresarle algo a la otra persona que no es sorda?

RESPUESTA: Si poco habla y no entiendo labios rápido oyentes.

¿Tienes dificultad para entender a otra persona de tu misma condición?

RESPUESTA: No entiendo las personas oyentes.

¿Qué palabras son las que te cuesta mucho reconocer cuando una persona oyente se trata de comunicar contigo?

RESPUESTA: aeropuerto, almohada, anfitrión, ardilla, armario, biblioteca, crucigrama, disimular, encuesta, escalera, espejo, fábrica, galleta, hielo, hierbabuena, hospital, impresora, jardín, lámpara, lista, mariposa, montaña, película, relámpago, semáforo, simulacro, sistema, USB, zapato.

¿Qué palabras son las que les cuesta mucho a las personas oyentes en reconocer cuando te expresas en el lenguaje de señas peruana?

RESPUESTA: aceptar, aguantar, árbol, buscar, canción, comprender, contador, día, diccionario, ensayo, equivocar, examen, explicar, iglesia,

manta, navidad, país, panadería, piedra, quien, refrigerador, siempre, sufrir, tentación, terremoto, trapear, universidad, vacaciones, verdad, visitar.

¿Qué opinas sobre el uso de la tecnología para mejorar la vida de las personas sordas? ¿Has utilizado alguna de estas tecnologías?

RESPUESTA: No conocer, solo ver frente labios.

¿Prefieres deletrear las palabras en lenguaje de señas? O ¿en qué caso deletrearías alguna palabra mediante lenguaje de señas peruanas?

RESPUESTA: Para difíciles palabras o nueva.

DATOS DE CONTROL:

Nombre del encuestado: Renan Miguel Begazo Torres

Dirección: José Castelli 698, La Esperanza 13012

Teléfono/Celular: +51 967 882 112

Código del entrevistador: 000201211 Código del supervisor: 000201211

000202894 000202894

Fecha: 15/07/2023

Nota: Se resalta que, durante la realización de la encuesta dirigida a las tres personas sordas de la Asociación De Sordos De La Libertad, únicamente una de ellas, el señor Renan Miguel Begazo Torres pudo proporcionar respuestas. Esto se debió a que él es el único de los tres individuos que posee conocimientos rudimentarios en lectura y escritura, aunque requirió asistencia (en este caso, la ayuda de ambos) para completar la encuesta. Por lo que, no fue posible realizar la encuesta a las demás personas sordas.