Convergencia Mecatrónica, Capítulo 39, pp. 553 - 566.
ISBN: 978-607-9394-30-1, 2024.

Diseño de Sistema Domótico con Visión Artificial para el Reconocimiento de Lengua de Señas Mexicanas

Enoc Isai Martinez Jimenez, Carlos Alberto Toro Arcila, Josué Gómez Casas, Jesús Salvador Galindo Valdés, Daniela Estefanía Ortiz Ramos, Oziel Gómez Casas, Jesús Fernando Martínez Villafañe y Carlos Rodrigo Muñiz Valdez.

Facultad de Ingeniería de la Universidad Autónoma de Coahuila.
*J. Gómez Casas: jogomezc@uadec.edu.mx.

Resumen

Actualmente, las personas sordas toman un papel importante en nuestra comunidad, hoy existe un gran número de ellas en el mundo y la mayoría no tienen acceso eficiente a la tecnología que les pueda ayudar a comunicarse y/o mejorar su calidad de vida. Debido a lo anterior, el objetivo de esta investigación es diseñar e implementar un sistema domótico que mediante el uso de la lengua de señas mexicana y la visión artificial pueda comunicarse con dispositivos electrónicos (un foco y ventilador), considerando cualquier tipo de iluminación y ambiente, es decir, en condiciones con iluminación natural o eléctrica de día, así como también en condiciones de noche, en el cual la iluminación es nula o de baja intensidad. La identificación utilizará cámaras tipo IP "Internet Protocol" (Protocolo de Internet) para interiores y equipadas con leds infrarrojos para visión nocturna. Mediante este diseño se obtuvo una precisión del sistema al momento de detectar las señas dadas de un 94% en condiciones de día y un 90% en la noche, además de comunicaciones eficientes y rápidas con los dispositivos electrónicos. Con estas precisiones se pudieron detectar señas de forma confiable reduciendo el porcentaje de error en cualquier condición de iluminación.

Palabras clave: Visión artificial, domótica, lengua de señas mexicana, personas sordas, machine learning.

1. Introducción

De acuerdo con la Organización Mundial de la Salud (OMS) [1], 430 millones de personas sufren discapacidad auditiva y sordera según los datos reportados hasta el año 2024, por otra parte, los datos de la Organización Mundial de la Propiedad Intelectual (OMPI) [2], reportan que solo un 2% de esta población tiene acceso a tecnología para comunicarse y mejorar su calidad de vida. De igual manera, de acuerdo con las cifras más recientes del Instituto Nacional de Geografía y Estadística (INEGI) registradas en 2020 [3], revelan que en México hay un millón 350 mil 802 personas sordas, de las cuales 710 mil 405 son hombres y 640 mil 397 mujeres. En México, aproximadamente 2.3 millones de personas padecen discapacidad auditiva, de las cuales más del 50% son mayores de 60 años; poco más del 34% tienen entre 30 y 59 años y cerca del 2% son niñas y niños. Ahora bien, según datos proporcionados por el Centro de Rehabilitación y Educación Especial (CREE) localizado en Saltillo Coahuila, en la actualidad existe un 4.3% (134,816) de personas con discapacidad en Coahuila, el 20.1% (27,098) de personas corresponden a personas sordas.

Considerando lo anterior, la lengua de señas sirve como un medio de comunicación con personas sordas. Sin embargo, se carece de la implementación tecnológica para aplicaciones de comunicación con personas sordas que faciliten su cotidianidad y accesibilidad. Esta problemática motiva a la creación de sistemas de comunicación basados en herramientas de ingeniería para su interacción con su entorno de forma adecuada. La propuesta de esta investigación radica en el diseño e implementación de un sistema domótico capaz de interpretar la lengua de señas mexicano, sin la necesidad de emitir sonidos o interpretaciones vocales para controlar aparatos electrodomésticos y/o electrónicos, utilizando herramientas de visión por computadora y algoritmos de inteligencia artificial. Actualmente, los sistemas

domóticos comerciales (Alexa, Google Home, Siri, etc.) se activan mediante comunicación verbal, sin embargo, en la actualidad, no existen sistemas capaces de reconocer la lengua de señas mexicana para controlar un entorno o vivienda para el caso específico de personas sordas. El diseño de un sistema domótico usando la lengua de señas pretende mejorar la cálida de vida de las personas sordas mediante la asistencia tecnológica de comunicación y control de entornos. La hipótesis de la investigación se basa en el uso de técnicas de Machine Learning (Aprendizaje Automático) utilizando visión artificial, para controlar dispositivos electrónicos mediante el uso de la lengua de señas mexicana para personas sordas.

Según investigaciones realizadas en los últimos años, diseñar sistemas que reconozcan o interpreten la lengua de señas ha sido un desarrollo con avances importantes, como es el caso de Mundewadi y colabroradores [4] quienes realizaron un sistema domótico basado en la interfaz de un guante en la mano que controlaba distintos dispositivos electrónicos. Por otro lado, autores como Chithra y colaboradores [5], que, en el año 2020, utilizaron un guante especial con sensores que detectan números realizados por las manos para controlar dispositivos electrónicos. En el caso del trabajo reportado por Oz & Leu en 2011 [6], los cuales utilizaron un guante especializado de la marca Cyberglove entrenado para el reconocimiento de palabras en lengua de señas mediante el uso de redes neuronales artificiales.

Aunque estos sistemas funcionan de forma correcta, las personas que lo utilizan dependen del uso de guantes en sus manos en todo momento, es por ello que pasando a investigaciones enfocadas en el reconocimiento de la lengua de señas usando visión artificial los investigadores Naranjo, Alarcon-O, Amancha-P, Ortiz-Espinosa, y Naranjo [7] realizaron un sistema a bajo costo para la detección de letras del alfabeto del idioma latín y funcionar como un traductor de esta lengua a texto, éste sistema detecta mediante el uso de visión artificial cada letra del alfabeto, dando como resultado un promedio de 905.88 milisegundos para el reconocimiento de cada letra. Igualmente, en 2015 en el trabajo reportado por Galicia, R. y otros investigadores [8] utilizaron un sensor de movimiento detectando el esqueleto de la mano humana mediante imágenes y obteniendo la identificación de cada letra en la lengua mexicana, con este sistema se precisó el 76.19% en el reconocimiento de cada señal.

En los últimos años, el uso de las redes neuronales se ha convertido una de las herramientas principales para este tipo de sistemas de reconocimiento de lenguajes o gestos, tales como el trabajo reportado por Agnihotri & Arora en el año 2023 [9] quienes, mediante una red neuronal convolucional y segmentación de la mano en escala de grises, detectan gestos realizados por las manos, obteniendo una precisión del 94% para gestos estáticos y un 92% con gestos en movimiento de la lengua de señas.

Otra de las herramientas más importantes en esta área, fue el uso de la librería MediaPipe como fue reportado por Tomar, D. y otros colaboradores [10], quien a través de esta librería fueron capaz de detectar letras del alfabeto mexicano y Farhan & Madi en 2022 [11] detectaron 12 señas con movimiento en la legua de señas americano con una precisión del 97.2%, sin embargo, el uso de esta detección de gestos y lenguajes no solo se limita al control de sistemas domóticos y el reconocimiento para traductores. En el caso de Grif & Turc, 2018 [12] y Grif & Farcas en el 2016 [13] quienes usaron el reconocimiento de gestos por medio de la librería MediaPipe para el control de un *mouse* virtual para la mejora y eficiencia del uso de computadoras o laptops, en su investigación, mediante gestos controlaron parámetros como los niveles de volumen, selección de texto, *zoom* en pantalla, entre otros.

El objetivo de esta investigación es crear un asistente domótico para personas sordas mediante el reconocimiento de la lengua de señas mexicana mediante el uso de técnicas de Machine Learning (Aprendizaje Automático) y visión artificial para controlar dispositivos electrónicos. Por lo que se requiere del diseño e implementación de una interfaz hombre-maquina capaz de interpretar la lengua de señas mexicana para la comunicación de dispositivos electrónicos dirigido a personas sordas, lo anterior, utilizando un modelo de redes neuronales para la identificación de señas en entorno bajo condición de iluminación de día y de noche.

Las secciones del artículo se dividen de la siguiente manera: en la Sección 2 se presenta la metodología utilizada en este proyecto, la Sección 3 se abordan los detalles de la red neuronal en el

entrenamiento, la Sección 4 muestran los resultados experimentales obtenidos en entornos reales y finalmente en la Sección 5 se discuten las conclusiones de este trabajo de investigación.

2. Metodología

En la Figura 1, se presenta el esquema de la metodología implementada para la realización de esta investigación.

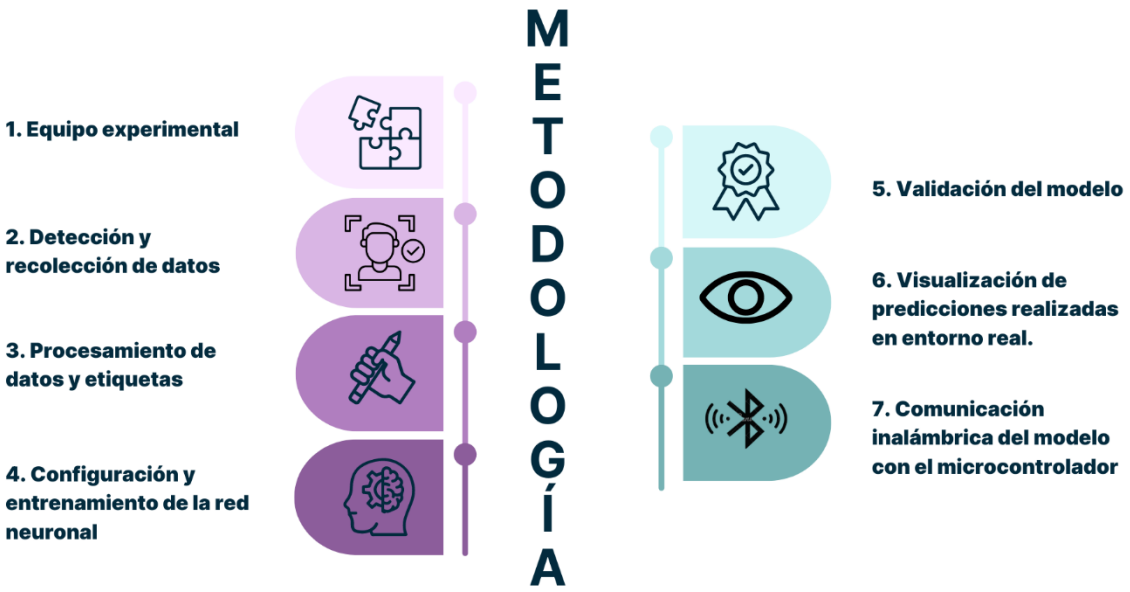

Figura 1: Diagrama de metodología para el diseño y la implementación del sistema domótico con visión artificial para el reconocimiento de la lengua de señas mexicana.

2.1 Equipo experimental

Para esta investigación se utilizó una Laptop Hewlett-Packard (HP) con procesador *11th Gen Intel® Core™ i3-1115G4 @ 3.00GHz*, 8GB de memoria *RAM*, con sistema operativo Windows 11 Home Single Language version 22H2 de 64 bits y procesador x64.

Para la adquisición de imagenes se usaron tres cámaras, una webcam y dos cámaras IP "Internet Protocol" (Protocolo de Internet), la primera es una webcam de la marca ACTECK modelo WM20, ésta cámara cuenta con una resolución de video de 720p (1280*720)/30 fps (frames por segundo), contiene un sensor H62 1.0M con un enfoque fijo y una alimentación de DC 5V 120mA, es importante mencionar que cuenta con tecnología Plug & Play (conectar y usar), con conectarla a un dispositivo, como una laptop o computadora de escritorio (PC) funciona correctamente.

La primera cámara IP usada es de la marca Icsee, cuenta con 4MP y FullHD de resolución, esta cámara incluye visión nocturna infrarroja y a color, además, es óptima para entornos interiores.

También se utilizó una cámara IP de la marca IMOU modelo DK2 2MP, con calidad FullHD, 2MP de resolución, visión nocturna infrarroja, la cual, también es óptima para entornos interiores únicamente, las cámaras mencionadas se observan en la Figura 2.

Figura 2: Cámaras utilizadas en el desarrollo experimental para la implementación del sistema domótico por visión artificial.

Los entornos usados para la implementación del sistema domótico fueron los siguientes: el primero emulando un entorno de oficina de trabajo, que recibía luz solar y luz artificial a través de lámparas fluorescentes y el otro entorno, emulando una habitación en el que se trabajó por la noche para evitar luz que afectara las pruebas realizadas en horas de noche.

2.2 Detección y recolección de datos

Para detectar los puntos característicos (*keypoints*) del cuerpo humano, el cual será el encargado de realizar las señas correspondientes, se usó la biblioteca MediaPipe (versión 0.10.9), OpenCV (versión 4.9.0) como bibliotecas principales y Python (versión 3.11.7) como lenguaje de programación. Hoy en día estas bibliotecas son muy populares en temas de visión artificial y detección de características del movimiento del cuerpo humano. MediaPipe es una librería open source (de código abierto) creada por la empresa Google LLC, la cual puede detectar puntos importantes del cuerpo para la detección de poses, señas, movimientos, entre otras acciones. Para esta investigación se usó el modelo *Holistic*, este modelo es el más completo ya que puede detectar 543 puntos en todo el cuerpo humano, estos puntos se dividen en:

- 468 puntos de referencias en el rostro.
- 33 puntos de referencia de postura.
- 21 puntos de referencias en cada mano.

Además, se usó la función *drawing_utils*, con la que se muestran e identifican de forma visual y distinguible los puntos de referencia ya mencionados.

OpenCV, también de código abierto, fue desarrollado por Intel, esta biblioteca se utilizó para la visualización en tiempo real de los movimientos corporales mediante el procesamiento y la visualización de las imágenes adquiridas por la cámara, al igual que para la visualización de los resultados, esta biblioteca tiene el beneficio de poder conectarse mediante cámaras USB o mejor conocidas como Webcams y también a cámaras por medio de IP.

Para la recolección de datos se empleó OS, un módulo integrado por el lenguaje Python, capaz de interactuar con el sistema operativo en el que se está programando, este módulo se usó para crear las carpetas en las que se almacenaron los datos de la recolección de las señas realizadas por la persona. Las carpetas antes mencionadas fueron etiquetadas por un nombre correspondiente a cada acción con la librería Numpy a través de una lista de elementos que podríamos definir como un estante de almacenamiento. Para este proyecto fueron cuatro las carpetas creadas, las cuales corresponden a las señas “LigthOn, LigthOff, AirOn, AirOff” que en español se traduce como “Luz prendida, Luz apagada, Aire prendido, Aire apagado”, respectivamente. Las señas se escribieron en la pantalla de

visualización en idioma inglés por temas de facilidad de visualización en el proceso de predicción con el modelo en tiempo real.

Una vez que se obtuvo el conjunto de datos, con un número de secuencia de 100 y un número de 40 frames por cada secuencia, a continuación de define detalladamente el procedimiento.

Primero se obtiene una secuencia de 100 acciones, esto significa que cada carpeta con el nombre de la señal tendrá internamente 100 carpetas, y por último dentro de cada carpeta internas habrá 40 archivos de tipo Numpy, el cual tendrá las coordenadas de los puntos clave detectados por MediaPipe en el espacio de la grabación, estas carpetas serán las que se usarán para el entrenamiento del modelo. Teniendo el conjunto de datos con todos los ejemplos de las señales se pasa al procesamiento de los datos y las etiquetas.

2.3 Procesamiento de datos y etiquetas

Para poder procesar los datos almacenados se utilizó la función *sklearn.model selection* de la librería scikit-learn, de la cual se importa la función *train_test_split*, la cual permite dividir el conjunto en dos partes, la primera es parte del entrenamiento con un 80% (320 secuencias) y la segunda que será la de la validación con el 20% (80 secuencias). Además, se empleó la librería keras.utils en conjunto con la función *to_categorical*, que se utiliza para convertir matrices de etiquetas en una representación de tipo categórico. Esta función es comúnmente utilizada en problemas de clasificación multiclase cuando las etiquetas se representan como enteros. La función convierte estos enteros en una codificación “one-hot”, el cual es un formato binario en el que cada clase está representada por un vector binario único y facilita el aprendizaje automático para ciertos algoritmos.

Con los datos divididos en entrenamiento y validación, además de tener binarizada cada etiqueta, se procedió a la configuración y el entrenamiento de la red neuronal.

2.4 Configuración y entrenamiento de la red neuronal

Para la construcción de la red se utilizó TensorFlow.keras (tf.keras) en su versión 2.16.1. TensorFlow es un entorno (framework) para el aprendizaje automático desarrollado por la compañía Google LLC. Por otro lado, también se utilizó Keras, que es una interfaz de programación de aplicaciones (API) del framework TensorFlow, además, es una biblioteca de alto nivel para redes neuronales la cual fue diseñada para ser fácil de manejar, modular y extender. Con las herramientas descritas anteriormente, se construirán y entrenarán los modelos utilizando la API de Keras, pero con la potencia y flexibilidad de TensorFlow.

Para esta red se utilizó el módulo *keras.models* del cual se importa el modelo Sequential, el cual organiza las capas en una lista lineal, donde cada capa tiene exactamente un tensor de entrada y un tensor de salida. Es el tipo de modelo más simple y comúnmente utilizado en Keras para construir redes neuronales.

También se usó el módulo *keras.layers* y se importó LSTM y Dense. De acuerdo con [14] LSTM (Long short-term memory) “*es un tipo de red neuronal recurrente (RNN) que se utiliza en el aprendizaje profundo para procesar y predecir secuencias de datos. La LSTM fue diseñada para abordar el problema del desvanecimiento del gradiente en las redes neuronales recurrentes tradicionales, que se producen cuando se retropropaga el error a través de múltiples capas y se pierde información importante en el proceso. La LSTM usa una estructura de celda con puertas y permite a la red controlar la información almacenada y olvidada en cada paso de tiempo, especialmente adecuada para procesar secuencias de datos a largo plazo.* La capa Dense se conectó completamente con cada neurona anterior, para realizar transformaciones lineales y no lineales en los datos. Por último, de *keras.callbacks* se importó TensorBoard el cual ayuda a la visualización, mediante gráficas, de los valores obtenidos durante el entrenamiento en cuestión de precisión y perdida.

Antes de pasar a la configuración de la red, se creó una carpeta en la que se almacenan los parámetros de entrenamiento y validación de la red, lo anterior, usando TensorBoard. Además de almacenar los parámetros, ayudará a visualizar las gráficas de entrenamiento antes mencionadas.

La configuración de la red se compone de tres capas, la capa de entrada, las capas ocultas y por último la capa de salida, estas capas se agregaron usando el método *model.add*. Para el modelo se utilizaron dos funciones de activación, ReLU (Rectified Lineal Unit) que en español se traduce como Unidad Lineal Rectificada y SoftMax. La activación de tipo ReLU umbraliza los valores de entrada en 0, devolviendo 0 para los valores negativos y para los mayores a 0 se les mantiene la propia entrada como se muestra en la Figura 3, lo cual permite no modificar la escala de los valores de entrada positivos y permite que el gradiente trabaje sin cambios durante la retropropagacion, mitigando así el desvanecimiento del gradiente, esta explicación se puede expresar mediante la Ecuación 1.

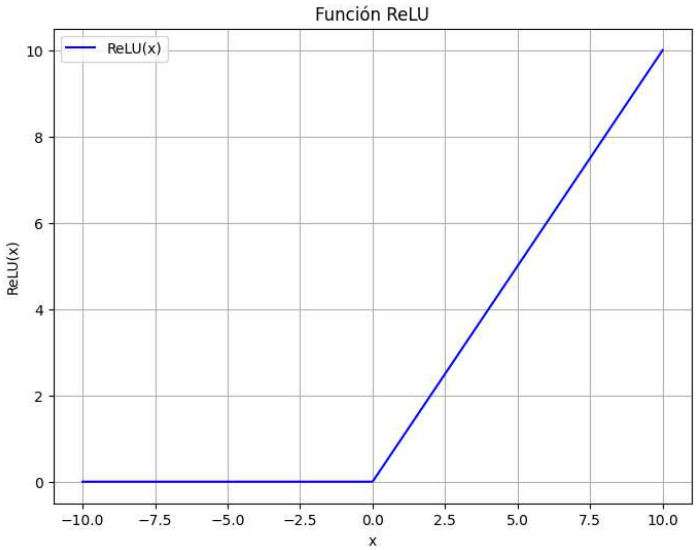

Figura 3: Diagrama de activación de la función ReLU.

$$f(x) = max(0, x) \tag{1}$$

La activación SoftMax, también conocida como función exponencial normalizada, por el tipo de gráfica que la representa (ver Figura 4), se utilizó solamente para la capa de salida, esta función es muy útil en problemas de clasificaciones multiclase, por lo tanto, se optó por esta activación en la capa de salida, la activación de la función se rige mediante la Ecuación 2.

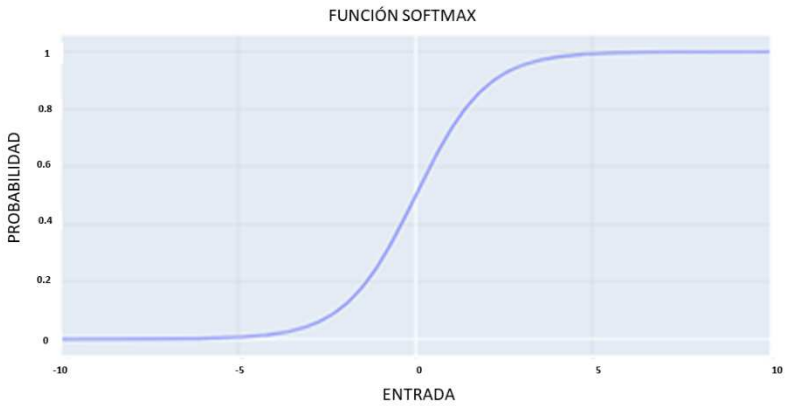

Figura 4: Diagrama de activación de la función SoftMax [15].

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

2.4.1 Configuración de las capas

Para la capa de entrada del tipo LSTM, se usaron 64 neuronas, como se muestra en la Figura 5, se le aplicó un retorno de secuencia, se usó una activación ReLU con un tamaño de entrada de 40,1662, estos valores corresponden a los 40 frames y la cantidad de puntos clave.

En las capas ocultas se usaron 5 capas, con las siguientes configuraciones: para la primera capa, una red de tipo LSTM con 64 neuronas con un retorno de secuencia y una activación tipo ReLU, esta activación se utilizó en cada capa oculta, la segunda capa, una red tipo LSTM con 32 neuronas sin retorno de secuencia, para la tercera, cuarta y quinta capa se usó una red tipo Dense con 32 neuronas.

Por último, para la capa de salida se usó un tipo de red Dense, donde el número de neuronas depende de las señales que serán reconocidas, para el objeto de estudio de esta investigación, fueron 4 y, para terminar, se utilizó SoftMax como función de activación.

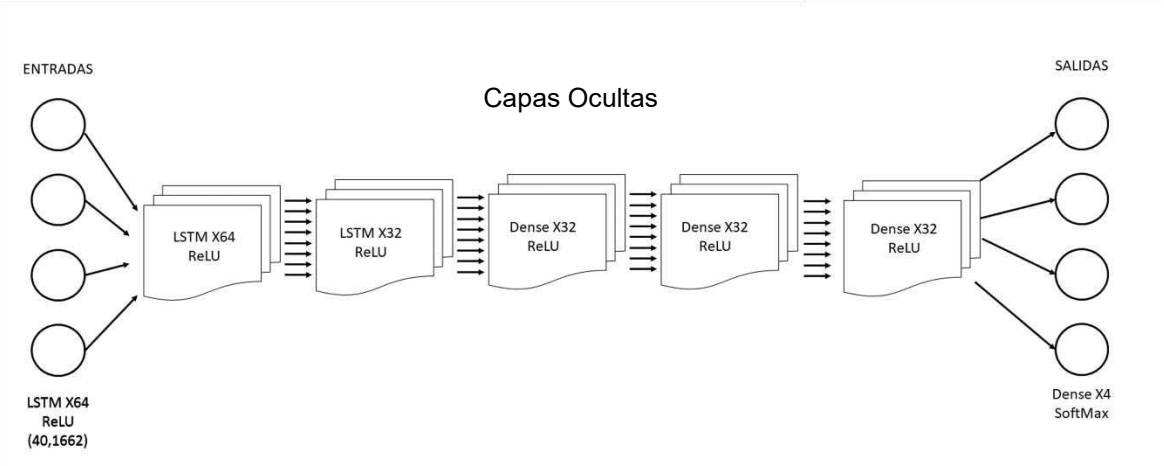

Figura 5: Diagrama de la red neuronal usada para el entrenamiento.

2.4.2 Configuración del proceso de entrenamiento

Para este paso se utilizó el módulo *compile* con el cual se configura el procesamiento de la red, utilizando como parámetros, un optimizador Adam (Adaptive Moment Estimation) traducido como Estimación Adaptativa de Momentos en español. Este optimizador es un algoritmo flexible y adaptativo que ha demostrado que funciona correctamente en problemas de aprendizaje profundo, útil en problemas donde los datos son escasos o ruidosos o donde las tasas de aprendizaje adaptativas son beneficiosas. Como función de perdida, se utilizó la función *categorical_crossentropy* ,la cual es una función de pérdida adecuada para problemas de clasificación multiclase con clases mutuamente excluyentes, donde las etiquetas están codificadas en un formato one-hot. Su objetivo es minimizar la discrepancia entre las predicciones del modelo y las etiquetas verdaderas durante el entrenamiento. Por último, se utilizó *categorical_accuracy*, con este parámetro de métrica se calculó y se registraron los valores de precisión del modelo durante el entrenamiento.

2.4.3 Entrenamiento de la red

Como ultimo método para entrenar la red, se utilizó *fit,* el cual es el método encargado de entrenar la red, en este método se le entregan como parámetros, los valores de entrenamiento de entrada y salida, así como un batch size (Tamaño de lote) de 30 muestras, el cual se encargó de pasar este batch size a la red para calcular el gradiente en cada iteración, esto ayuda a aprovechar la eficiencia computacional. Para las épocas se le asigna un valor de 80, esto significa que los datos de entrenamiento dieron 80 iteraciones completas por la red. Para evaluar que la red aprenda y no memorice, se usaron los datos de entrada, los datos de salida de la validación y se accedió a la carpeta con los parámetros de entrenamiento y validación final para visualizarse gráficamente.

2.5 Validación del modelo

Finalizado el entrenamiento del modelo, se pasó a la etapa de visualización, primero se guardó el modelo mediante el método save, el cual debe tener un nombre (en nuestro caso lleva el nombre “Prueba”) seguido de la extensión .keras, y mediante la importación del método load_model se manda llamar el modelo entrenado para su validación.

Con el método *predict* se manda llamar a las predicciones realizadas en los datos de entrada de validación, estas predicciones se almacenan en una variable, la cual se usó para verificar que los valores de entrada y salida de validación estuvieran correctos, así como también, obtener las probabilidades con mayor porcentaje dentro de las opciones mediante el uso de la función *argmax*.

2.6 Visualización de predicciones realizadas en entorno real

Mediante un entorno visual, desarrollado con OpenCV, se visualiza el porcentaje de confiabilidad de la predicción del modelo en un entorno real, estos porcentajes se representan mediante una barra de llenado en la ventana de muestreo que dependiendo de la probabilidad, rellena el espacio de la señal que se está realizando como se puede observar en la Figura 6, se estableció un umbral (treshold) de más del 80% del nivel de confiabilidad de la predicción para tomar una señal como valida, de lo contrario es una señal no valida, lo anterior, como medida de seguridad y evitar errores o confusiones de señas, terminado este umbral, se escribe el nombre de la señal en la parte superior de la ventana. Como otra medida de seguridad y pensando en si una persona realizara dos o más veces la misma señal, se estableció otra regla; si la señal realizada es igual la última predicción detectada, no se realizará ninguna instrucción posterior.

Convergencia Mecatrónica, Capítulo 39, pp. 553 - 566.
ISBN: 978-607-9394-30-1, 2024.

Figura 6: Visualización del llenado de la barra de probabilidad.

2.7 Comunicación inalámbrica

Para la comunicación de las predicciones hechas por el modelo y los dispositivos que se controlaron, se usaron los siguientes componentes:

- Un módulo ESP32 Bluetooth y Wifi.
- Un relevador (relé) de dos módulos Arduino.
- Cableado eléctrico.
- Una roseta con un foco.
- Un ventilador de aire doméstico.

Para la programación de la tarjeta ESP32 y los relevadores se usó el entorno de Arduino (versión 1.8.18), en este entorno solo se controlaron los datos obtenidos mediate la librería BluetoothSerial.h, la cual se encargó de comunicarse vía Bluetooth. Para esta comunicación entre Python y Arduino se usó la librería Serial en Python y la librería BluetoothSerial.h, en lo que se refiera a la programación desde Python, se accedió a los índices de cada señal, al ser almacenados en una lista que contienen un número de índice, los cuales son:

- Para LigthOn el número 0.
- Para LigthOff el número 1.
- Para AirOn el número 2.
- Para LigthOff el número 3.

Una vez obtenidos los índices, dependiendo de la probabilidad mediante una validación condicional, se determina que, si se obtiene el número 0, mande una letra, en el caso de este índice se mandó la letra A, mediante la función *write* y el método *encode* que se encarga de codificar la letra y mandarla para ser recibida por Arduino usando la función *read*. Una vez enviada la letra a Arduino, con una estructura condicional tipo switch, se enciende o apaga el aparato deseado.

Para las conexiones eléctricas de los relevadores con el módulo ESP32 y los dispositivos eléctricos, se siguió el diagrama de conexiones mostrado en la Figura 7.

Convergencia Mecatrónica, Capítulo 39, pp. 553 - 566.
ISBN: 978-607-9394-30-1, 2024.

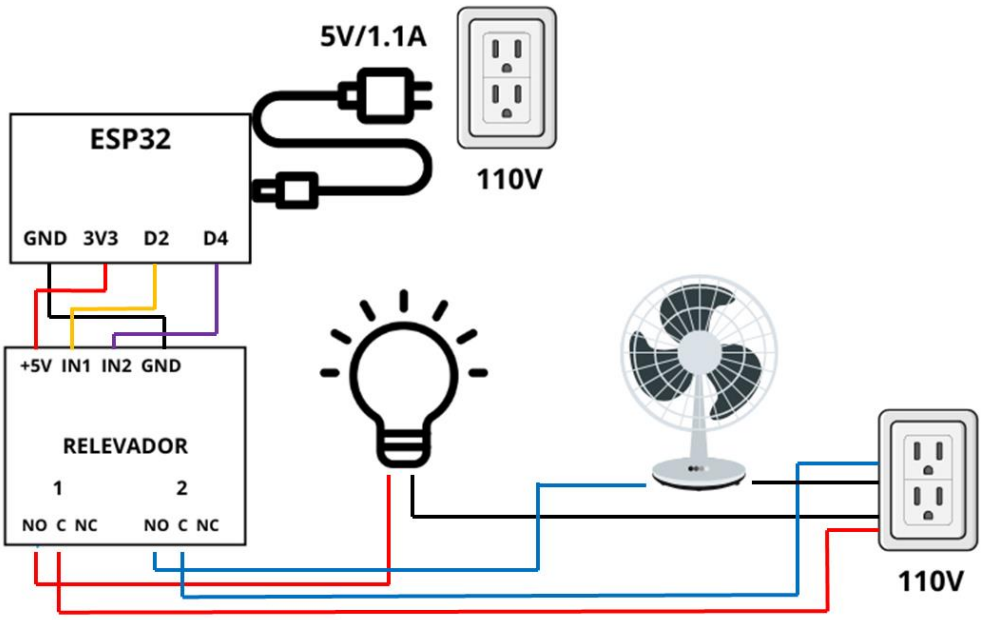

Figura 7: Diagrama eléctrico para la conexión inalámbrica.

Como se muestra en el diagrama anterior se procede a explicar las conexiones de los dispositivos eléctricos con el relevador, el relevador tiene nomenclaturas tales como NO, C y NC, NC (Normally Close) significa normalmente cerrado, este tipo de contacto es el que permite el paso del flujo de energía al unir sus dos platinos interno, NO (Normally Open) es un contacto normalmente abierto que a diferencia de NC impide el paso de la energía y por último C (common), que significa común, su función es actuar como el punto de contacto central que se conecta con uno de los otros dos terminales (NO o NC) dependiendo del estado del relé.

Después, sigue la conexión de los dispositivos, como se mencionó anteriormente, se trabajó con un módulo relé doble, en el diagrama se ve un número 1 y 2 en el recuadro del relevador, especificando el número del módulo. El módulo 1 se conectó al foco mediante una roseta eléctrica, un cable de línea (mostrado en color rojo), sale del foco para ser conectado a NO, el terminal común va directamente a una conexión eléctrica domestica de 110 o 220 V (Voltios) en su terminal de fase. Y por último del foco sale un cable de tipo neutro (color negro) a la conexión eléctrica doméstica. Este mismo proceso se repite para el módulo 2 que va conectado al aire.

Para las conexiones del relevador con la tarjeta ESP32 se utilizaron los pines digitales 2 y 4 de la tarjeta y fueron conectados respectivamente a los IN1 e IN2 (entradas) del relevador que son los que se muestran en color amarillo y morado. Se conectaron el pin GND (negativo) al GND del relevador y el pin de 3.3V a el pin de voltaje del relevador, este relevador puede trabajar con 3.3v y 5v DC (corriente directa) ya que permite controlar un circuito de alta potencia o corriente con una señal de baja potencia. Por último, se conecta la tarjeta mediante un transformador de voltaje que reduce un voltaje de 110/220V a uno de 5V.

3. Resultados del entrenamiento de la red neuronal

Analizando las gráficas dadas para el proceso de entrenamiento de la red, se definieron dos métricas de interés, la primera “epoch categorical accuaracy” (precisión categórica por épocas), la cual permitió observar la precisión categórica de la red durante cada época de entrenamiento y validación. Esto significa, que se evalúa la precisión de clasificación para cada categoría conforme aumentan las

épocas, esta gráfica asciende conforme pasan las épocas y el comportamiento ideal debe ser una función creciente y al final las curvas de entramiento y validación deben tener valores cercanos a 1 como se muestra en la Figura 8 la cual en la parte del eje vertical “y” muestra el porcentaje que se obtienen en cada época la cual es mostrada en el eje horizontal “x”. En este caso, el modelo entrenado obtuvo una precisión categórica de 94% en su última época. Se puede observar también, que los datos de entrenamiento (línea de color naranja) y los datos de validación (línea color azul) tienen un comportamiento creciente y su forma es muy parecida, lo cual indica que se está evitando el sobreajuste (overfitting, en inglés), el cual sucede cuando los datos de validación tienden a ir de forma descendente y los de entrenamiento ascendente. Además, se está evitando el subajuste (underfitting, en inglés), el cual se presenta cuando ambas curvas presentan comportamiento descendente.

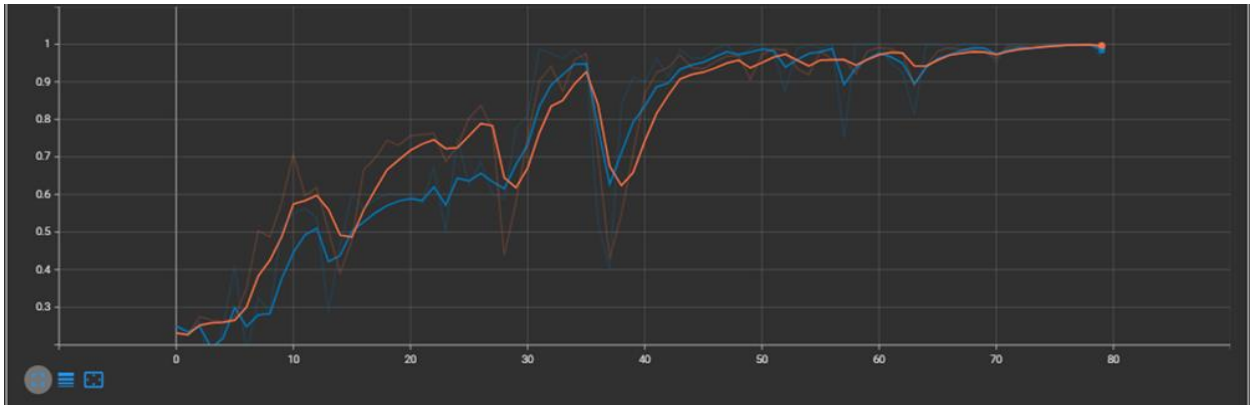

Figura 8: Gráfica de precisión categórica en cada época.

Para la función de costo se utilizó la gráfica de “epoch loss” (perdida por época), los resultados de los datos tanto de validación (color azul) como entrenamiento (color naranja) presentan un comportamiento decreciente hasta llegar a un valor cercano a cero en la época 78, lo que indica que el modelo está obteniendo el comportamiento deseado como se muestra en la Figura 9.

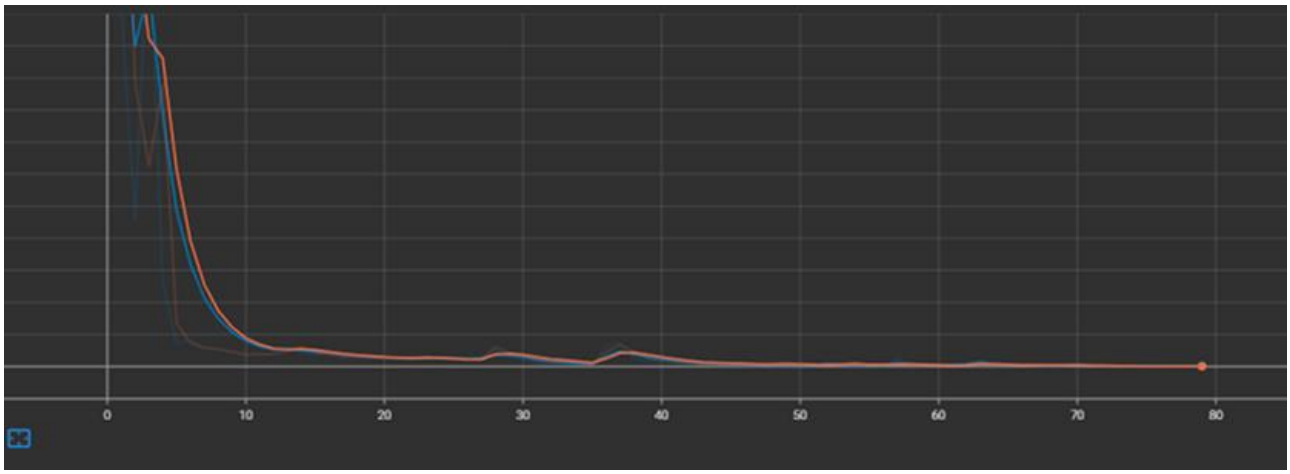

Figura 9: Grafica de perdida por época.

Convergencia Mecatrónica, Capítulo 39, pp. 553 - 566.
ISBN: 978-607-9394-30-1, 2024.

4. Resultados del modelo en un entorno real

Para validar el desempeño del modelo, se realizaron pruebas en los dos entornos mencionados en la Sección 2.1. El primer experimento se realizó en las instalaciones del Laboratorio de Automatización de la Facultad de Ingeniería de la Universidad Autónoma de Coahuila, en horas de la tarde (12:00 a 14:00 hora Monterrey, México) cuando aún se cuenta con iluminación solar, además se utilizaron las luminarias del laboratorio. Se realizaron pruebas con las cuatro señales: encender luz, apagar luz, encender aire y apagar aire. Para este experimento se utilizó la cámara Webcam. Según el análisis, el modelo realiza correctamente la predicción de las señas ejecutadas por el usuario y la comunicación inalámbrica con los dispositivos funciona adecuadamente. En la Figura 10 se muestra al usuario ejecutando la seña de "Encender Luz" (LigthOn), también se muestra la barra de porcentaje de confiabilidad de la seña y, por último, la seña definitiva tomada por al algoritmo.

Figura 10: Resultado para la señal encender luz (LigthOn).

En la Figura 11, se presentan los resultados en el mismo escenario, pero ahora para la seña de “Encender Aire” (AirOn).

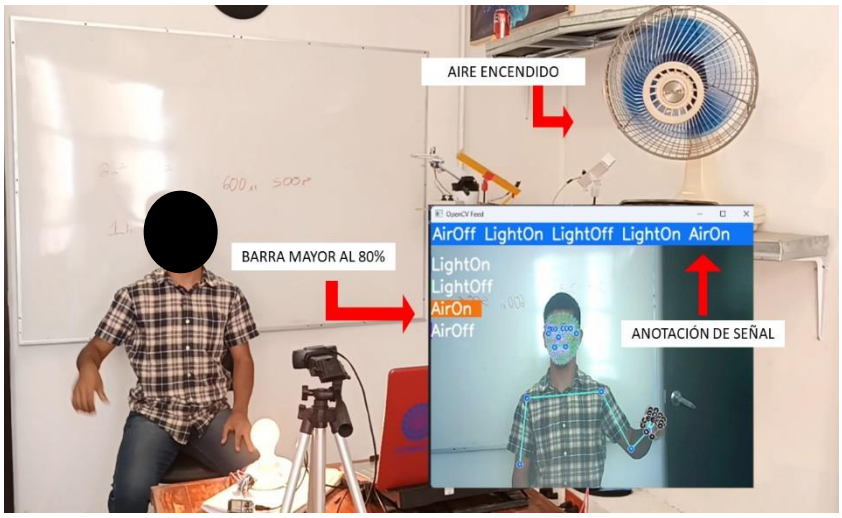

Figura 11: Resultado para la señal encender aire (AirOn).

Convergencia Mecatrónica, Capítulo 39, pp. 553 - 566.
ISBN: 978-607-9394-30-1, 2024.

El segundo experimento se desarrolló en una casa habitación, en horas de la noche (10:00 pm), sin luz solar ni iluminación artificial (luminarias, focos, etc.). Para este experimento se utilizaron las cámaras IP con la función de visión nocturna (infrarroja). Los resultados son similares a los del primer experimento descrito anteriormente, por lo que el modelo funciona en entornos tanto para el día y la noche, con iluminación y sin iluminación. En la Figura 12 se muestra que el modelo reconoció la seña de “Encender Luz” (LigthOn) y la comunicación con el dispositivo (foco), funcionó adecuadamente.

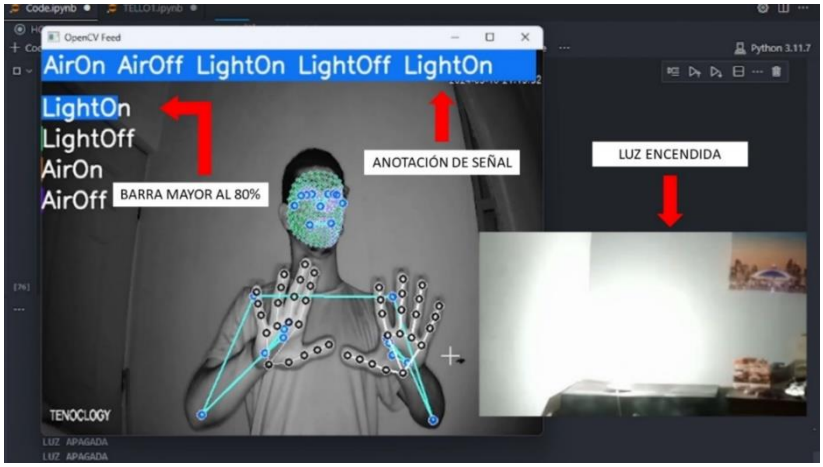

Figura 12: Resultado en entorno oscuro para la señal encender luz (LigthOn).

Los videos de la ejecución de los dos experimentos se pueden ver en el siguiente enlace: *https://youtu.be/BYTHWg-TJVk.*

5. Conclusiones

Según los resultados del sistema, se obtuvo un diseño innovador y único donde se realizó una interfaz hombre-máquina para comunicarse con dispositivos electrónicos, cabe destacar que funciona tanto con luz de día o ambiental como de noche o en oscuridad y promete ser una tecnología revolucionaria para las personas sordas.

En primer lugar, se estableció el equipo requerido para diseñar el sistema domótico con visión artificial para lograr el reconocimiento de la lengua de señas mexicana, como se menciona en el apartado 2.1, se debe destacar que se utilizaron diferentes entornos para la implementación del sistema domótico, en los cuales se aplicó la luz solar, luz artificial y oscuridad. Primero se realizó la detección y recolección de los datos para posteriormente realizar el procesamiento de datos y etiquetas, seguido se realizó la configuración y entrenamiento de la red neuronal.

En el caso de la red neuronal, la configuración se compone de tres capas: la capa de entrada, del tipo LSTM se usaron 64 neuronas y se le aplicó un retorno de secuencia, se usó una activación ReLU con un tamaño de entrada de 40,1662, estos valores corresponden a los 40 frames y la cantidad de puntos clave; las capas ocultas, se utilizaron cinco capas, las cuales se encuentran descritas en el apartado 2.4.1 y finalmente la capa de salida, se usó un tipo de red Dense, donde el número de neuronas depende de las señales que serán reconocidas, para el objeto de estudio de esta investigación, fueron 4 y, para terminar, se utilizó SoftMax como función de activación. Posteriormente se realizó un entrenamiento de la red y validación del modelo.

Los resultados obtenidos muestran que el modelo entrenado obtuvo una precisión categórica de 94% en su última época, y a su vez los datos de validación tienen un comportamiento creciente y similar

al modelo entrenado, lo anterior indica que se evita el sobreajuste y el subajuste. Para el entorno real se realizó en un ambiente de luz ambiental y artificial obtenido la respuesta esperada por el sistema domótico según la aplicación de las cuatro señales aplicadas: encender luz, apagar luz, encender aire y apagar aire; finalmente se realizó el mismo experimento pero cambiando la luz a un entorno de oscuridad, donde la respuesta del sistema a las mismas cuatro señales fue favorable, obteniendo que la red neuronal pudo detectar correctamente las señas, obteniendo porcentajes de precisión del 94% en promedio para iluminaciones en el día y un 90% para condiciones en la noche, además de comunicarse con los dispositivos electrónicos de forma inmediata y sin complicaciones.

Con esto se confirma que es posible diseñar sistemas domóticos usando visión artificial y la lengua de señas mexicana. Este sistema podrá ayudar a personas sordas a mejorar su calidad de vida, además de brindar una mayor autonomía diaria.

REFERENCIAS

World Health Organization: WHO. (2024, February 2). *Sordera y pérdida de la audición*. https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss

BrightSign: el guante inteligente que da voz a quienes no pueden hablar. (n.d.). https://www.wipo.int/wipo_magazine/es/2019/05/article_0005.html

Esto, R. P., Esto, R. P., & Esto, R. P. (2022, April 24). *¿Quiénes padecen más el problema de la Sordera en México? Esto dice el Inegi. Por Esto!* https://www.poresto.net/mexico/2022/4/24/quienes-padecen-mas-el-problema-de-la-sordera-en-mexico-esto-dice-el-inegi.html

Mundewadi, S. A., Dindore, P. N., Kamurti, L. B., Arole, K. V., & Gurav, V. V. *SIGN LANGUAGE TO SPEECH CONVERSION AND HOME AUTOMATION.*

Chithra Apoorva, D. A., Gowtham, B. S., Charishma, K., Teja, K. K., & Kumar, G. S. C. (2020). *Smart Glove: Sign to Speech Conversion and Home Automation Control for Mute Community*. International Journal of Engineering and Advanced Technology., 9.

Oz, C., & Leu, M. C. (2011). *American sign language word recognition with a sensory glove using artificial neural networks*. Engineering Applications of Artificial Intelligence, 24(7), 1204-1213.

Naranjo, A. E., Alarcon-O, A., Amancha-P, G., Ortiz-Espinosa, J., & Naranjo, J. E. (2021). *Low-cost assistive system for deaf people based on artificial vision*. Advances and Applications in Computer Science, Electronics and Industrial Engineering: Proceedings of CSEI 2020, 249-264.

Galicia, R., Carranza, O., Jiménez, E. D., & Rivera, G. E. (2015, June). *Mexican sign language recognition using movement sensor*. In 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE) (pp. 573-578). IEEE.

Agnihotri, A. R., & Arora, D. (2023, July). *Vision based Interpreter for Sign Languages and Static Gesture Control using Convolutional Neural Network*. In 2023 2nd International Conference on Edge Computing and Applications (ICECAA) (pp. 1611-1615). IEEE.

Tomar, D., Nauni, D., Zaidi, A. M., & Kaur, M. (2023, November). *Gesture-Controlled Home Automation for the Differently Abled: Enhanced Accessibility and Independence*. In 2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS) (pp. 1560-1564). IEEE.

Farhan, Y., & Madi, A. A. (2022, December). *Real-time dynamic sign recognition using mediapipe*. In 2022 IEEE 3rd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS) (pp. 1-7). IEEE.

Grif, H. S., & Turc, T. (2018). *Human hand gesture based system for mouse cursor control*. Procedia Manufacturing, 22, 1038-1042.

Grif, H. S., & Farcas, C. C. (2016). *Mouse cursor control system based on hand gesture*. Procedia Technology, 22, 657-661.

Qué es LSTM: Long short-term memory Concepto y definición. Glosario. (n.d.). GAMCO, SL. https://gamco.es/glosario/lstm-long-short-term-memory/

Ali, M. (2024, April 15). *Diagrama de la función de activación SoftMax* [Figura]. https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks