SISTEMA EXPERTO PARA LA IDENTIFICACIÓN DE GESTOS DEL LENGUAJE DE SEÑAS COLOMBIANO

MOISÉS NÚÑEZ LÓPEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS
ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2022

SISTEMA EXPERTO PARA LA IDENTIFICACIÓN DE GESTOS DEL LENGUAJE DE SEÑAS COLOMBIANO

MOISÉS NÚÑEZ LÓPEZ

Trabajo de Grado para optar al título de
Ingeniero Electrónico

Director
Mag. Jaime Guillermo Barrero Pérez.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICOMECÁNICAS
ESCUELA DE INGENIERÍAS
ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2022

CONTENIDO

LISTA DE FIGURAS

pág.

LISTA DE TABLAS

pág.

RESUMEN

TITLE: Sistema Experto para la Identificación de Gestos del Lenguaje de Señas Colombiano [*]

AUTHOR: MOISÉS NÚÑEZ LÓPEZ [**]

PALABRAS CLAVE: Red, Neuronal, CNN, LSTM, YOLO, Lenguaje, Señas, MAIX II, Sistema, Embebido, Python, Tensorflow, Keras.

DESCRIPCIÓN:

Este trabajo es una propuesta para la aplicación de las redes neuronales en problemas de la sociedad, como es el caso de la comunicación para la población no oyente, por lo tanto, se exploraron algunas técnicas necesarias en el campo de la inteligencia artificial para diseñar alternativas de algoritmos con los que una Se puede construir un sistema inteligente, basado en la implementación del dispositivo *Sipeed Maix-II*, para identificar los gestos de la lengua de señas colombiana, esto, a través de visión por computadora y nuevos avances en procesamiento de imágenes y reconocimiento de patrones.

---

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Mag. Jaime Guillermo Barrero Pérez.

ABSTRACT

TITLE: Sistema Experto para la Identificación de Gestos del Lenguaje de Señas Colombiano [*]

AUTHOR: MOISÉS NÚÑEZ LÓPEZ [**]

KEYWORDS: Red, Neuronal, CNN, LSTM, YOLO, Lenguaje, Señas, MAIX II, Sistema, Embebido, Python, Tensorflow, Keras.

DESCRIPTION:

This work is a proposal for the application of neural networks in problems of society, as is the case of communication for the non-hearing population, therefore, some necessary techniques in the field of artificial intelligence were explored to design alternatives of algorithms with which an intelligent system can be built, based on the implementation of the *Sipeed Maix-II* device, to identify the gestures of the Colombian sign language, this, through computer vision and new advances in image processing and pattern recognition.

---

[*] Bachelor Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Director: Mag. Jaime Guillermo Barrero Pérez.

Introducción

Según estudios realizados por la Organización Mundial de la Salud (OMS), se prevé que para el año 2050 alrededor de 700 millones de personas padecerán perdida de la audición, lo que significa que de cada diez personas al menos una tendrá problemas auditivos. Las personas que sufren de sordera en ciertos casos se debe a causas relacionadas con la genética o complicaciones de gestación, debido a esto, estas personas desde su nacimiento se encuentran incapacitadas para realizar acciones cotidianas relacionadas con la comunicación en su entorno. Teniendo en cuenta que el 80 % de estos casos de perdida de audición ocurre en países en vías de desarrollo es común encontrarse con pocas, o si no, nulas oportunidades ante las dificultades que se pueden presentar en el contexto, tales como acceso a la educación, conseguir empleo o el simple hecho de comunicarse con otra persona, acarreando discriminación OMS (2018). *Sordera y pérdida de la audición*. [En linea] Disponible en:
`https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss`.

Se evidencia con la población del departamento de Santander que padece de discapacidad auditiva, que pese a los esfuerzos para promover la inclusión, hay cifras que demuestran la realidad que estos afrontan en situaciones como la búsqueda de empleo, siendo que un 34,5 % de la población no oyente de Santander está incapacitada para trabajar ocasionando que su índice de pobreza multidimensional sea mayor comparado con una persona sin discapacidad INSOR (2018). *REALIDADES DE LA POBLACIÓN SORDA EN 10 CIUDADES DEL PAÍS*. [En linea] Disponible en:
`https://www.insor.gov.co/home/realidades-de-la-poblacion-sorda-en-10-ciudades-del-pais/`.

Teniendo en cuenta lo anterior, por medio de este trabajo es pertinente que se alcan-

ce una oportunidad para desarrollar y visibilizar un dispositivo que ayude a disminuir la brecha comunicacional que existe entre personas con discapacidad auditiva y personas oyentes. Diseñando un dispositivo que permita identificar los gestos de la lengua de señas colombiano. Este objetivo se pretende alcanzar mediante la implementación de las redes neuronales convolucionales y recurrentes, después de investigar y recopilar técnicas de procesamiento de imágenes, se entrenaron las redes propuestas para clasificar y predecir las 27 señas del abecedario perteneciente al lenguaje de señas colombiano y posteriormente acondicionar el código para ser ejecutado desde el sistema embebido *Sipeed Maix-II*.

1. Objetivos

Objetivo General

- Identificar los gestos que componen al lenguaje de señas colombiano mediante un algoritmo basado en la implementación de una red neuronal convolucional.

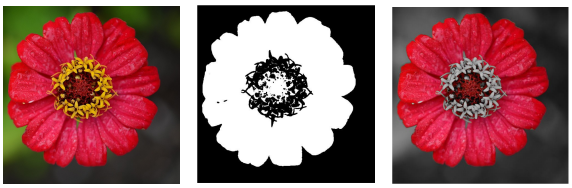
Objetivos Específicos

- Entrenar la red neuronal convolucional de identificación de gestos utilizando la base de datos completa actualizada con todos los gestos del abecedario del lenguaje de señas colombiano.Daniel Mauricio Avila Rey. *Construcción y Entrenamiento de una Red Neuronal para la Identificación de Gestos. [recurso electronico]*. spa. Bucaramanga: UIS, 2021.
- Implementar la red neuronal convolucional como un sistema experto que logre identificar y categorizar distintos gestos del lenguaje de señas colombiano.
- Realizar pruebas de campo en un sistema embebido con la red neuronal convolucional entrenada para determinar su desempeño y optimizar el funcionamiento del sistema.

2. Metodología

En esta sección se abordaron todas las partes que conforman el proceso de ejecución del trabajo, desde las investigaciones realizadas, como los teoremas, códigos y conceptos utilizados para conseguir el objetivo del proyecto.
En primer lugar, se presentaran de manera amplia los conceptos que son necesarios para comprender lo planteado y proseguir en la implementación de algoritmos.

2.0.1. Procesamiento de Imágenes El procesamiento digital de imágenes es una rama de las ciencias de la computación, estrictamente relacionada con el tratamiento de señales, la estadística y el álgebra lineal. Haciendo uso de estas tres disciplinas de la ingeniería se han constituido una serie de técnicas y métodos con los que se puede adquirir, procesar y manipular imágenes, con el fin de observarlas, almacenarlas, transmitirlas o analizarlas Said Pertuz. *0.1 presentación DIP*. Presentación del curso de Procesamiento Digital de Imágenes del programa de ingeniería electrónica de la Universidad Industrial de Santander. 2020. Ya que existen diferentes tipos de imágenes también se crearon un amplia gama de estrategias con las que se pueden abordar de manera adecuada, para este trabajo se aplico el uso del filtrado lineal también llamado procesamiento en bloques.
El procesamiento en bloques consiste en un filtro o kernel, que se puede considerar como una ventana de tamaño $N \times N$ con $N < M$ (en donde N es la dimensión del kernel y M la dimensión de la imagen), que se aplica a cada píxel de la imagen en un proceso llamado convolución o correlación, obteniendo así una transformación como se observa en la Fig. 1, generando una imagen con características diferentes dependiendo de los valores del kernel.



(a) (b) (c)

Figura 1. Ejemplo de procesamiento en una imagen, en este caso segmentación basado en color. (a) Imagen de interés. (b) Mascara de segmentación basada en color. (c) Imagen procesada. **Tomado de:** Said Pertuz. *0.1 presentación DIP*. Presentación del curso de Procesamiento Digital de Imágenes del programa de ingeniería electrónica de la Universidad Industrial de Santander. 2020

2.0.2. Machine Learning El *machine learning* es un campo de la computación que mediante el uso de algoritmos le concede a un programa de computadora la habilidad de aprender, sin necesidad de ejecutar una programación explicita Boston Farnham et al. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*. Son todas las teorías y técnicas que permiten mediante la recopilación de datos y la estadística brindarle un entorno de aprendizaje inteligente a la computadora Steven Cooper. *Data Scienve from Scratch*.

Hay tres tipos de *machine learning*:

- Cualquier entrenamiento que requiere o no de la supervisión de un humano (aprendizaje supervisado, no supervisado, semi supervisado y por refuerzo).
- Pueden o no aprender de forma incremental sobre la marcha (aprendizaje en línea versus aprendizaje por lotes).
- Funcionan comparando nuevos puntos de datos con otros puntos de datos conocidos, o que detecten patrones en los datos de entrenamiento y construyan un modelo predictivo (aprendizaje basado en instancias versus aprendizaje basado en modelos).

Cada uno de los tipos de machine learning tiene sus propias técnicas y métodos con los que se puede llegar a un resultado aceptable en el aprendizaje con datos, sin embargo, en algunas ocasiones no va a ser suficiente o será muy complicado conseguir una solución reproducible ante el problema planteado. Por esta razón existe el subcampo derivado del machine learning llamado deep learning.

2.0.3. Deep Learning Inspirándose en las redes neuronales humanas y partiendo de las teorías ya estudiadas en las demás practicas del *machine learning*, fue elaborado un modelo de aprendizaje llamado redes neuronales artificiales, que es la base fundamental *deep learning* Farnham et al., *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

Las redes neuronales artificiales principalmente están compuestas por este nuevo concepto de neurona, donde se puede hacer el símil con una función matemática de regresión lineal como la mostrada en la ecuación (1), conformada por una operación interna dependiente de los pesos que se le asigna a cada entrada de la neurona y dando como resultado una salida, tal como se observa en la Fig. 2. Estos pesos van a ser modificados en el entrenamiento por la propia red neuronal para así conseguir la salida esperada, también dependiendo del problema es necesario aplicar una función de activación en la red neuronal para convertir las operación matemática en una función no lineal que permita resolver problemas con una dispersión particular DotCSV. *¿Qué es una Red Neuronal? Parte 1 : La Neurona | DotCSV*. En esta serie de vídeos vamos a aprender cuál el funcionamiento de las potentes Redes Neuronales. En esta primera parte veremos el funcionamiento de una neurona y cuál es su relación con el modelo de regresión lineal. 2020.

$$F_a(Y_1) = F_a(W1 \cdot X_1 + W2 \cdot X_2 + b) \tag{1}$$

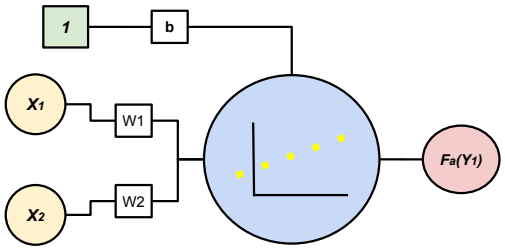


Figura 2. Representación de una neurona con dos variables de entrada, una entrada de bias y una salida, con una aproximación de la operación que ocurre en su interior. **Tomado de:** Elaboración propia.

El uso más común de las redes neuronales es relacionado con problemas de clasificación, predicción y agrupamiento. Asimismo, existen aplicaciones específicas dentro de las redes neuronales, dos de estos tipos específicos fueron utilizados en la realización de este trabajo, tales como las redes neuronales convolucionales y las redes neuronales recurrentes.

2.0.4. Redes Neuronales Convolucionales Las redes neuronales convolucionales se crearon con la primicia de poder realizar el procedimiento de aprendizaje con imágenes, dado que en una red neuronal convencional a los píxeles de una imagen no se les da la importancia que tienen en conjunto con los demás píxeles y los interpretaría como un vector. En concordancia con lo anterior, se pensó en una forma de imitar la visión cortex humana, que consiste en la detección de patrones, esquinas, intensidad y relieve para determinar cual es el objeto que se está observando Farnham et al., *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

Para abordarlo desde el campo de la computación fue necesario agregar un concepto del procesamiento de imágenes llamado procesamiento en bloques, en donde por medio de un kernel y la convolución con cada píxel de la imagen se obtienen todas

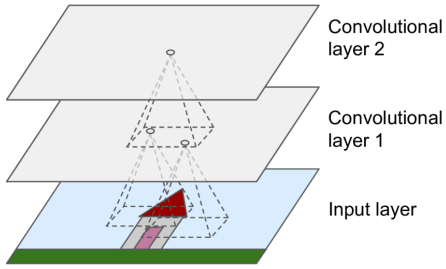


Figura 3. Aplicación de capas convolucionales a una imagen de entrada. **Tomado de:** Boston Farnham et al. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

las características que permitan identificar su contenido. Teniendo en cuenta lo anterior, se agregaron a la red neuronal nuevas capas llamadas capas convolucionales, en donde cada una de estas aplica un kernel diferente para obtener un mapa de características como se observa en la Fig. 4, así, teniendo una imagen de entrada se le pueden aplicar varias capas convolucionales para extraer los patrones que ayudarán a que la red neuronal identifique y aprenda del contenido de la imagen Soroush Nasiriany et al. *A Comprehensive Guide to Machine Learning*. 2019.

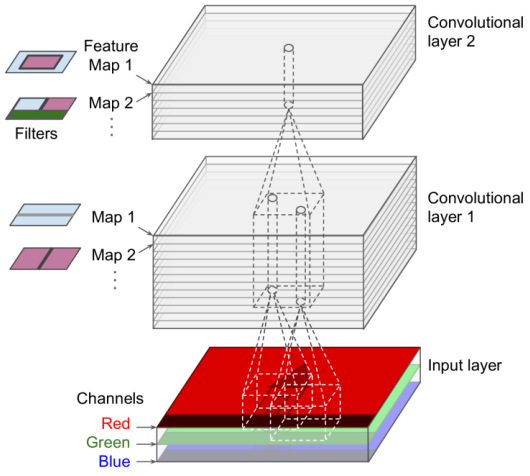


Figura 4. Representación de los mapas de características obtenidos en dos capas de convolución. **Tomado de:** Boston Farnham et al. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

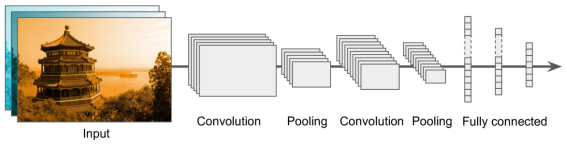


Figura 5. La arquitectura de red neuronal convolucional típica. **Tomado de:** Boston Farnham et al. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION*.

2.0.5. Redes Neuronales Recurrentes Las redes neuronales recurrentes fueron diseñadas teniendo en cuenta todas las situaciones en donde se requiere analizar datos que siguen una secuencia recurrente, es decir, como en los casos de una cadena de caracteres o en un vídeo, donde el valor actual de la entrada esta relacionado con valores anteriores. A fin de encontrar una arquitectura que desarrolle las competencias deseadas para la problemática, se partió desde la red neuronal tradicional Fig. 2 y se le otorgo la habilidad de almacenar el estado anterior de la entrada, asemejando una pequeña memoria como se ilustra en la Fig. 6.

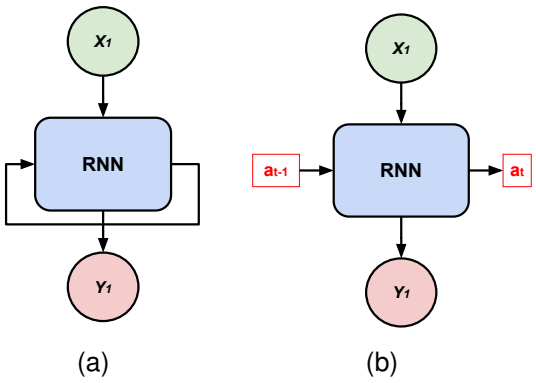


Figura 6. Esquema de una neurona en las arquitecturas de redes neuronales recurrentes. (a) Ciclo de las variables en una red neuronal recurrente. (b) Representación de la memoria almacenando el estado anterior de una variable interna relacionada a la entrada. **Tomado de:** Elaboración propia.

Dentro de las redes neuronales recurrentes existen diferentes tipos dependiendo de

la cantidad de módulos repetidos que se utilizan para realizar el seguimiento temporal de la entrada. En este trabajo se usó un modelo de red neuronal recurrente llamado *LSTM* (de sus siglas en inglés *Long Short-Term Memory*), esta red se diseño con el fin de resolver una problemática común de las redes recurrentes, conocida como el problema del *Long-Term*. En secuencias largas en donde se necesite hacer una predicción o aproximación al final de la cadena de entrada, es probable que requiera de información del inicio que ya no se tiene almacenada como se ´puede observar en la Fig. 7, esto es estrictamente solucionado por las redes *LSTM* ya que se agregaron nuevas estructuras al módulo ya conocido de la red neuronal recurrente Christopher Olah. *Understanding LSTM Networks*. You can email me at christopherolah.co@gmail.com. 2015.

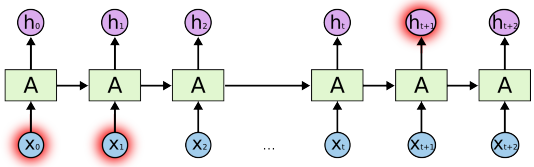


Figura 7. Problemática de *Long-Term* con x como entrada y h como la salida temporal en cada paso de la red. **Tomado de:** Christopher Olah. *Understanding LSTM Networks*. You can email me at christopherolah.co@gmail.com. 2015.

3. Soluciones Propuestas

Después de realizar la investigación de los conceptos principales necesarios para el trabajo, se propusieron dos soluciones ante la problemática planteada. En cada uno de los modelos se apuntó a una implementación diferente para tener alternativas y lograr la clasificación de las vocales del lenguaje de señas colombiano mostradas en la Fig. 8, teniendo como primicia utilizar las redes neuronales recurrentes, dado a que el reto principal es lograr identificar los gestos con movimiento (las letras G, J, Ñ, S y Z). Para que no ocurrieran errores en el código de *python* se cambio la letra Ñ por NE como su representación en los *labels* de cada modelo. Así mismo, se exploró el uso de la herramienta novedosa *MaixHub*, proporcionada por *Sipeed* la empresa del dispositivo *Maix-II*, siendo una muy buena opción para los interesados en incursionar en el desarrollo con *deep learning* sin mucho conocimiento.

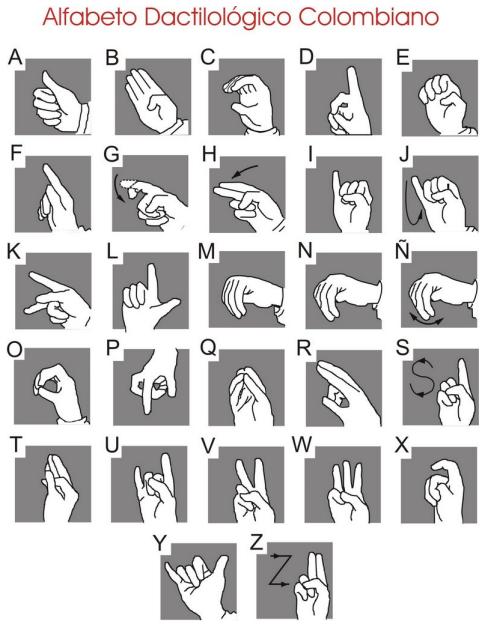


Figura 8. Alfabeto del lenguaje de señas colombiano. **Tomado de:** Autor desconocido, 2018, http://aprendiendolsc.blogspot.com/2018/05/alfabeto-dactilologico-colombiano.htmlEnlace.

3.0.1. Modelo CNN + LSTM En el desarrollo del primer modelo inicialmente se añadieron las 5 letras que faltaban a la base de datos conformada por 22 de los 27 gestos del lenguaje de señas colombiano creada por Daniel Ávila Rey, *Construcción y Entrenamiento de una Red Neuronal para la Identificación de Gestos. [recurso electronico]*, por ende, se grabaron 25 vídeos por cada letra y por medio de un código en *python* se extrajeron 10 *frames* de cada vídeo en los cuales se concentra la información importante de cada seña, como se ilustra en la Fig. 9. Estos *frames* se les modificó el tamaño a 250x250 para que coincidieran con las imágenes que ya se tenían de la base de datos mencionada, la base de datos se puede encontrar en este repositorio de drive: https://drive.google.com/drive/folders/1oidCccYqb6CQ$_{a}blYcdk2jsMaG0go$
$sharingBasededatosDrive.$

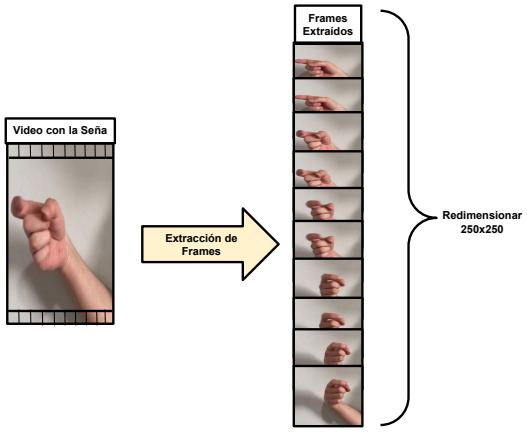


Figura 9. Proceso en el que se extrajeron los *frames* para cada vídeo de las señas en movimiento. **Tomado de:** Elaboración propia.

En razón a lo investigado sobre las redes neuronales convolucionales y recurrentes se planteó comenzar con una implementación que pudiera combinar ambas redes, utilizando la red convolucional para extraer características de las imágenes y la red recurrente *LSTM* para que detectara la relación secuencial que existe entre cada *frame* en el caso de las señas con movimiento. Se estableció que el programa estaría dirigido a realizar clasificación teniendo 27 clases que representan las 27 letras

del abecedario, cada clase tendrá un promedio de 250 imágenes separadas en grupos de 10, haciendo referencia a que cada vídeo tendrá una relación de 10 *frames* por gesto.

Para realizar el algoritmo del modelo se decidió utilizar *tensorflow* y *keras* por su facilidad de programación y sus múltiples librerías, de esta forma más adelante cuando se necesite implementar y entrenar el modelo se van a usar las funciones ya establecidas en el *API* (por sus siglas en inglés interfaz de programación de aplicaciones) importada Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. Ya teniendo el directorio con las imágenes y *frames* almacenados, era necesario crear un *dataset* con la información de esas imágenes para que el modelo pudiera interpretarlas, por medio de la librería *OpenCV* se cargaron todas las imágenes. De este proceso se obtuvieron los datos de las imágenes y los *labels* a los que están relacionadas. Los parámetros que se usaron para el código se pueden observar en la Tabla 1, en donde tocó reducir el tamaño de las imágenes a 64x64, debido a que la creación del *dataset* compatible con una red neuronal recurrente requiere un gran consumo de *RAM*. Se utilizó una función de costo de *categorical_crossentropy* en donde se calcula la entropía cruzada entre las predicciones y sus *labels*, esto se decidió por el enfoque categórico de la red, así mismo se utilizó un optimizador de *SGD*, es decir, el descenso del gradiente con *momentum* para poder encontrar máximos y mínimos locales y globales con mayor facilidad, el *momentum* y *learning rate* se configuraron para el optimizador y en base a lo establecido en el trabajo previo realizado por Daniel Ávila Rey, *Construcción y Entrenamiento de una Red Neuronal para la Identificación de Gestos. [recurso electronico]*. Mediante el comando *train_test_split* se separó el *dataset* en un 80 % para el entrenamiento y un 20 % para la validación.

Por consiguiente, en el diseño del modelo se tomó como guía el esquema que se muestra en la Fig. 10, donde se puede observar que se heredó el modelo pre-

Tabla 1. Tabla de parámetros modelo CNN + LSTM

Parámetro	**Valor**
Dimensiones Imagen	64x64
Épocas	50
Tamaño de sequencia	10
Learning Rate	0.001
Momentum	0.9
Optimizador	*SGD* (Descenso del Gradiente con momentum)
Función de costo	*categorical_crossentropy*

entrenado de *MobileNet_v2* para la parte convolucional de extracción de características manteniendo las primeras capas con sus pesos originales y dejando las ultimas 30 para ser modificadas Mark Sandler et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. En: (2018). DOI: 10.48550/ARXIV.1801.04381. Por ello, es primordial que las capas que se tienen en el modelo de *MobileNet_v2* tengan la misma relación de tiempo que se necesita para la capa de la red *LSTM*, en consecuencia se utilizó la función de *tensorflow TimeDistributed()*, así los grupos de 10 *frames* son interpretados de manera secuencial desde las capas convolucionales. Finalmente, al modelo se agregó una nueva capa densa con la función de activación *"softmax"*, permitiendo proporcionar una salida con la probabilidad en relación a la entrada con alguna de las 27 clases.

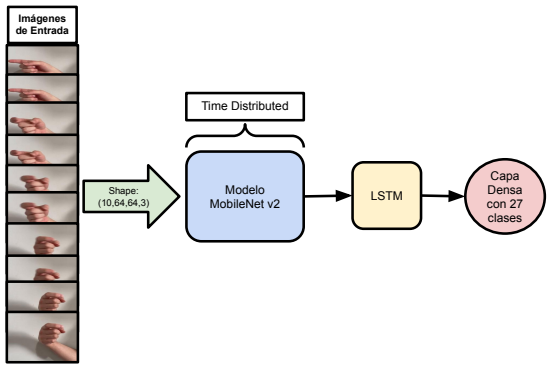


Figura 10. Diagrama que representa el flujo de ejecución que sigue el modelo implementado de *CNN + LSTM*. **Tomado de:** Elaboración propia.

3.0.2. Modelo MediaPipe + LSTM En la segunda alternativa de solución se utilizó una nueva *API* para *python* llamada *MediaPipe*, con esta librería es posible detectar puntos clave de referencia en varias partes y posiciones del cuerpo, por lo tanto para el caso de este trabajo se utilizó la información proporcionada por los trazos de la mano. Los datos que se obtienen de la mano son la posición en el espacio de los puntos que la representan, entonces cada marca tiene unas coordenadas X, Y y Z asociadas con las que se puede saber que gesto esta haciendo la mano. Con el fin de extraer suficiente información de cada seña se utilizó la cámara del computador para recopilar 30 vídeos compuestos por secuencias de 10 *frames* como se muestra en la Fig. 11, todas estas coordenadas se almacenan en una carpeta en formato de arreglos *numpy* separados por sus respectivas clases, esto, se hizo para poder armar unas variables en formato de tensores con forma: (810, 10, 126) para la entrada de características.

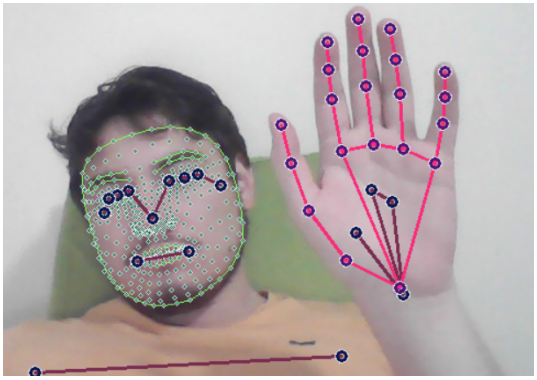


Figura 11. Puntos clave referencia de la mano izquierda. **Tomado de:** Elaboración propia

En consecuencia y teniendo el tensor de características y el vector con los *labels* relacionados a las clases, se separaron la información de los arreglos *numpy* usando la función *train_test_split* para conseguir una entrada de entrenamiento del 95 % y otra para validación del 5 %. Con los datos de entrada ya preparados se definieron los parámetros que iban a regir la red neuronal, mostrados en la Tabla 2, el optimiza-

dor que se escogió es el optimizador *Adam*, un algoritmo de descenso del gradiente que esta basado en estimaciones adaptativas para procesos requerimientos de memoria limitados. Para la función de perdida se escogió igual que en el primer modelo la *categorical_crossentropy* por su buen funcionamiento en casos de clasificación. Se diseñó el modelo de la red neuronal *LSTM* teniendo en cuenta el esquema de la Fig. 12, se utilizaron 3 capas *LSTM* con diferentes tamaños de parámetros y 3 capas densas, 2 para hacer una conexión completa de toda la red y una de salida para asignar la probabilidad de predicción a cada clase.

Tabla 2. Tabla de parámetros modelo *MediaPipe + LSTM*

Parámetro	**Valor**
Forma de entrada	(810,10,128)
Épocas	5000
Tamaño de secuencia	10
Optimizador	*Adam* (Decenso del Gradiente adaptativo)
Función de costo	*categorical_crossentropy*

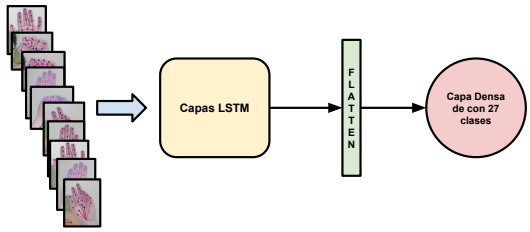


Figura 12. Diagrama que representa el flujo de ejecución que sigue el modelo implementado de *MediaPipe + LSTM*. **Tomado de:** Elaboración propia.

3.0.3. MaixHub Cabe considerar que para implementar el programa de identificación de gestos en un sistema embebido, se consideraron dos factores principales al seleccionar el dispositivo, los cuales son, la repercusión económica y la utilidad específica hacia el proyecto. En atención al trabajo predecesor a este, en donde se utilizó el dispositivo de la empresa *Sipeed Maix-I* Rey, *Construcción y Entrenamien-*

Tabla 3. Tabla de características comparando *Maix-I vs Maix-II* Sipeed. *Maix-II Dock*. 2021

Característica	MAIX-I (K210)	MAIX-II (V831)
CPU del chip de control principal	400∼600MHz	800∼1000Mhz
Codificador de vídeo	Ninguna	H.264, hasta 1080p@30fps H265, hasta 1080p@30fps JPEG, hasta 1080p@30fps
Acelerador de IA NPU	0.23TOPS apoyo Conv+BN+ACT+POOL	0.2TOPS admite Conv,Inner_Product, Pool,Eltwise,ACT,BN, Split,Concat
Memoria	memoria RAM de 8 MB	SIP 64 MB DDR2
Almacenamiento	16MB SPI Ni Flash	Flash opcional de 16M
Cámara	DVP, entrada máxima 30W píxeles	MIPI de 2 carriles, hasta 1080P@60fps
Pantalla	MCU LCD de 8 bits	LCD MCU de 8 bits, con placa adaptadora se puede conectar hasta LCD RGB de 10 pulgadas
SDIO	Ninguna	SMHC x2 (SDC0, SDC1)
SPI	SPIx3	SPIx2 (SPI0, SPI1)
2C	I2C x3	I2C x4 (TWI0, TWI1, TWI2, TWI3)
I2S	8 bits I2S	I2S x1 (I2S0)
Ethernet	Ninguna	Puerto Ethernet 10/100 Mbit/s con interfaz RMII
ADC	Ninguna	LRADC de 6 bits de 1 canal para clave
Audio	Ninguna	LINEOUTP + MICIN1P/N
Software desarrollador	maixpy/c	MaixPy3/linux

to de una Red Neuronal para la Identificación de Gestos. [recurso electronico], se indagaron los otros productos de la empresa que incurrieran en un mejor rendimiento, por ende, se escogió el dispositivo *Maix-II* teniendo en cuenta todas las ventajas y utilidades añadidas respecto a su predecesora, como se puede observar en la Tabla 3, unas de los principales mejoras es la mayor capacidad en cuanto a memoria y velocidad del micro-procesador, como también, la compatibilidad con nuevas herramientas de *Sipeed*, tales como, *MaixHub* y *MaixPy3*.

Se revisaron todas las oportunidades de desarrollo que proporciona la herramienta *MaixHub* para implementar redes neuronales en la *Maix-II*, de este modo se creo un nuevo proyecto dentro de la plataforma en donde lo primero que se necesitaba era cargar las imágenes del *dataset* en la página, para esto, fue necesario adaptar la base de datos mencionada en el modelo *CNN + LSTM*. Se cambiaron los *frames* de

las letras compuestas por movimiento (G, J, Ñ, S ,Z) por imágenes que representen estos movimientos y que no se confunda con las imágenes de las demás letras, ya que las implementaciones en *MaixHub* no contemplan el uso de redes neuronales recurrentes y funcionan sólo con redes neuronales convolucionales. Después por medio de la herramienta *labelImg* Tzutalin. *LabelImg*. Git code. 2015, se hicieron anotaciones de cada imagen para ser guardadas en archivos con formato *xml*, estas anotaciones conforman la información del tamaño del recuadro que contiene una sección de la imagen en donde se encuentra la seña que se quiere reconocer, como se muestra en la Fig. 13, en donde para este caso se encerró en un recuadro la seña que representa la letra A y se le dio este *label* a esa parte de la imagen.

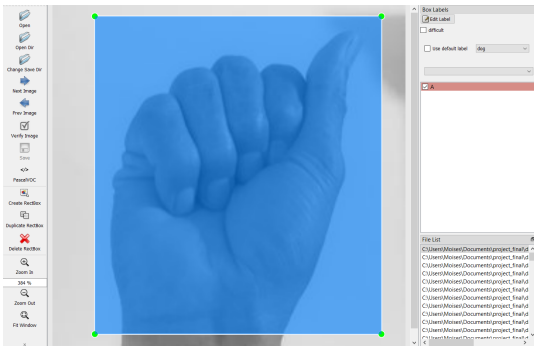


Figura 13. Anotación de la seña que representa la letra A, por medio de la herramienta labelImg. **Tomado de:** Elaboración propia.

Posteriormente teniendo la base de datos con todas las anotaciones completas (puede encontrar la base de datos utilizada para este modelo en esta carpeta de *drive*: https://drive.google.com/drive/folders/1zyb1uVotdL16UpI3rHDXPjj$_fMG7FHNV?usp=sharingEnlacedeDrive)seconfigurlaherramientaMaixHubsegnlosparmetrosdeseadosparalaimp$
En definitiva, se pudo entrenar el modelo y generar un código QR para poder montarlo en la *Maix-II* por medio de la misma aplicación *MaixHub*. También existe la opción de cargar el modelo al dispositivo descargando dos archivos de formato *bin* y *param* que representan los pesos y la arquitectura de la red, para después por medio de un código de *python* montarlos igualmente al dispositivo.

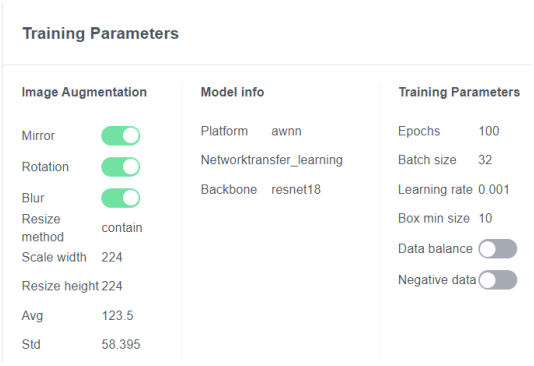


Figura 14. Configuración de los parámetros utilizados para el entrenamiento del modelo *MaixHub*. **Tomado de:** Elaboración propia.

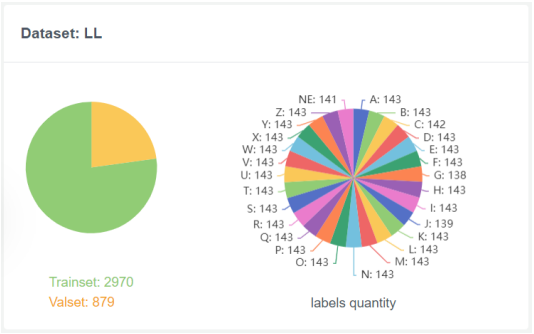


Figura 15. Separación del *dataset* para el modelo *MaixHub* y la cantidad de imágenes utilizadas por *label*. **Tomado de:** Elaboración propia.

Todos los códigos con los que se llevo a cabo la implementación de los dos modelos presentados se encuentran en un repositorio de *GitHub* para que puedan ser revisados por cualquier persona interesada. https://github.com/moshe64/CNN$_LSC.gitRepositoriodelt$

4. Resultados

En esta sección se presentan todos los resultados obtenidos con cada modelo, con sus respectivas pruebas en tiempo real y gráficas de los valores alcanzados en el entrenamiento.

Inicialmente se obtuvieron los resultados del modelo *CNN + LSTM*. La gráfica que se observa en la Fig. 16 representa el valor que tomó la precisión en entrenamiento y validación durante cada época del ajuste de pesos en la red, de igual manera en la Fig. 17 se muestra la variación de la perdida en la predicción con respecto a cada época. Debido a que fue necesario reducir el tamaño de las imágenes del *dataset*, el modelo pre-entrenado de *MobileNet_v2*, que en este caso extrae las características de las imágenes, no está funcionando con un desempeño óptimo ya que requiere un tamaño de imagen mínimo de 224x224, por lo tanto en esta situación se puede determinar que el modelo en conjunto tiene buenos resultados porque está realizando un proceso de memorizar la información del *dataset*. En la tabla 4 están los valores de los resultados de este modelo.

Tabla 4. Tabla con los datos obtenidos en el entrenamiento del modelo *CNN + LSTM*.

Resultado	**Valor**
Precisión en entrenamiento	99.600 %
Perdida en entrenamiento	0.143
Precisión en validación	98.000 %
Perdida en validación	0.187
Épocas	24

Posteriormente, los resultados que se obtuvieron fueron los del modelo *MediaPipe + LSTM*, igual que en el anterior modelo se lograron gráficas de la precisión y la perdida durante cada época del ajuste de los pesos de la red. Se puede observar en las Fig. 18 y 19 cómo el ajuste de los datos es disperso en algunas ocasiones y con

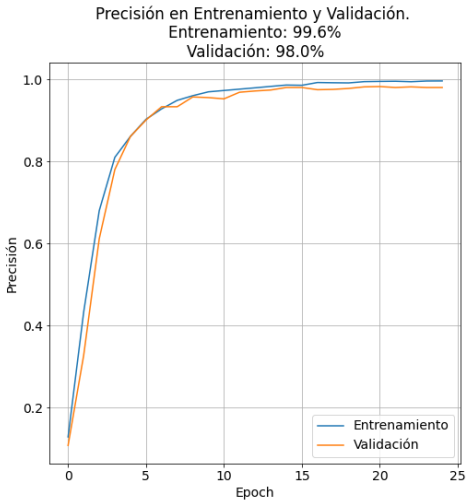


Figura 16. Gráfica con la precisión de entrenamiento y validación en el modelo *CNN + LSTM*. **Tomado de:** Elaboración propia.

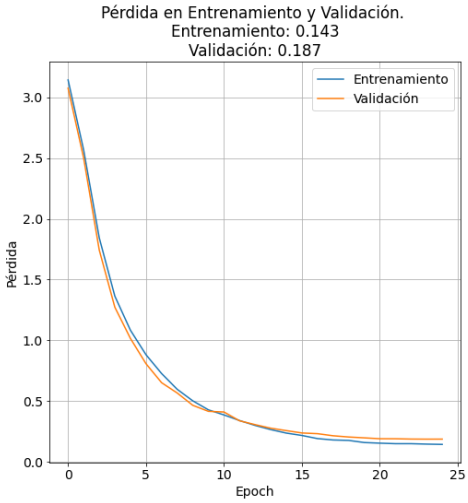


Figura 17. Gráfica de la perdida en las predicciones del modelo *CNN + LSTM*. **Tomado de:** Elaboración propia.

numerosas variaciones abruptas durante el proceso pero llegando a estabilizarse entre el valor 0.8 y 1 para el caso de la precisión. En la tabla 5, se encuentran los valores de precisión y perdida que se alcanzaron al finalizar el entrenamiento de este modelo.

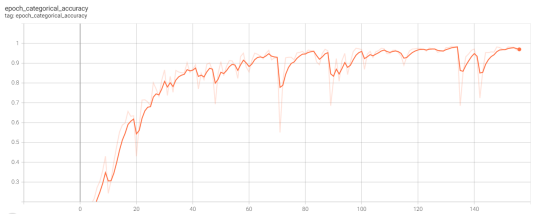


Figura 18. Gráfica con la precisión en el modelo *MediaPipe + LSTM*. **Tomado de:** Elaboración propia.

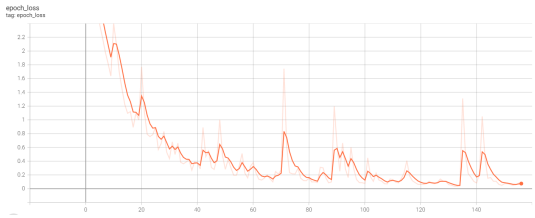


Figura 19. Gráfica de la perdida en las predicciones del modelo *MediaPipe + LSTM*. **Tomado de:** Elaboración propia.

Tabla 5. Tabla con los datos obtenidos en el entrenamiento del modelo *MediaPipe + LSTM*.

Resultado	Valor
Precisión en entrenamiento	0.9636
Perdida en entrenamiento	0.0833
Épocas	156

Para el caso de la implementación con *MaixHub* se generaron de igual manera las gráficas de precisión y perdida durante cada una de las épocas como se muestra en las Fig. 20 y 21 respectivamente, para este caso la precisión se estima de manera diferente dado que el modelo es de detección de objetos, en el entrenamiento la ejecución principal es estimar la predicción en cuanto a la clase correcta y adicional la correcta creación de un recuadro que contenga la seña correspondiente a esa clase. Así mismo, el valor de la perdida en este entrenamiento es un total en relación a la suma de la perdida en cuanto a la estimación de la clase y al perdida de la posición y tamaño del recuadro que contiene la señal. Los valores conseguidos para su mejor resultado en la época 149, se encuentran en la tabla 6.

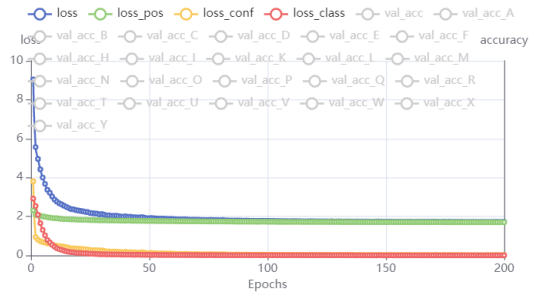


Figura 20. Gráfica con la precisión de validación en el modelo *MaixHub*. **Tomado de:** Elaboración propia.

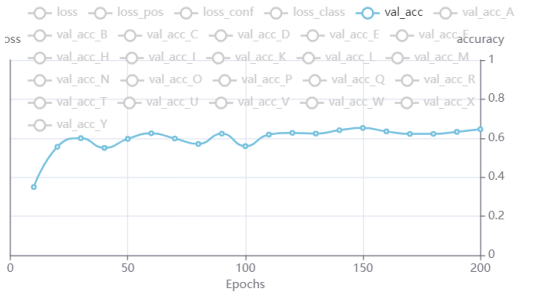


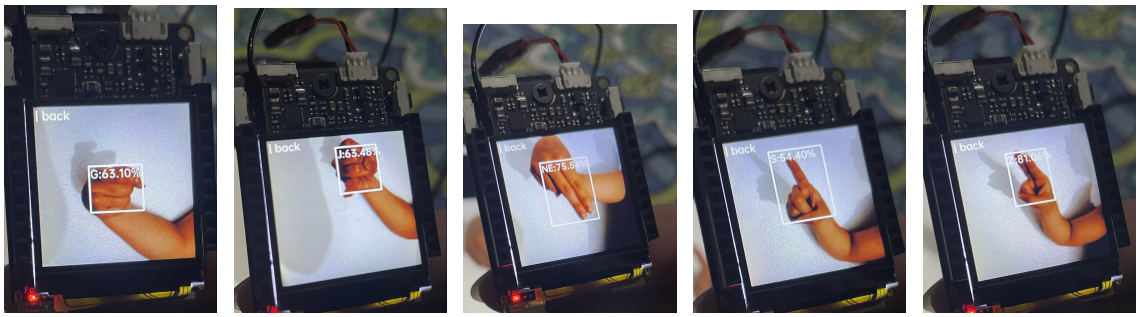
Figura 21. Gráfica de la perdida en las predicciones del modelo *MaixHub*. **Tomado de:** Elaboración propia.

Tabla 6. Tabla con los datos obtenidos en el entrenamiento de *MaixHub*.

Resultado	Valor
Precisión en validación	0.653
Perdida total	1.736
Perdida posición del recuadro	1.712
Perdida tamaño del recuadro	0.018
Perdida predicción clase	0.004
Épocas	149

En definitiva, se hicieron pruebas en tiempo real para predecir de las 5 señas compuestas con movimiento (G, J, Ñ, S, Z).En la Fig. 22 se encuentran imágenes de las señas marcadas con recuadro de detección reconocidas por el desarrollo implementado en *MaixHub* ya montado en el sistema embebido *Maix-II* con el porcentaje de certeza que se tuvo en cada predicción.

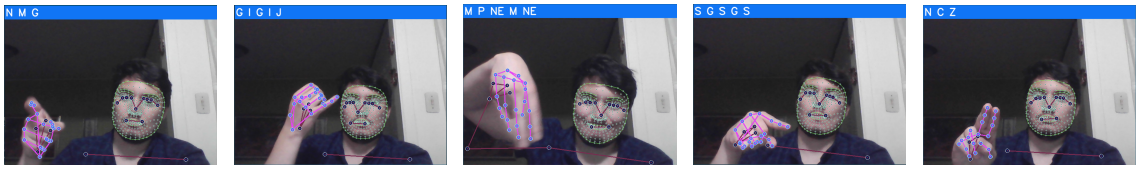
En la Fig. 23 están las predicciones realizadas con el modelo de *MediaPipe + LSTM*



(a) (b) (c) (d) (e)

Figura 22. Imágenes con las pruebas en tiempo real del modelo *MaixHub* con la *Maix-II*. (a) Predicción de la letra G con su cuadro de detección y un porcentaje de certeza del 63.10 %. (b) Predicción de la letra J con su cuadro de detección y un porcentaje de certeza del 63.48 %. (c) Predicción de la letra Ñ con su cuadro de detección y un porcentaje de certeza del 75.54 %. (d) Predicción de la letra S con su cuadro de detección y un porcentaje de certeza del 54.40 %. (e) Predicción de la letra Z con su cuadro de detección y un porcentaje de certeza del 81.06 %. **Tomado de:** Elaboración propia.

la primera letra que se observa de derecha a izquierda en el rectángulo azul es la ultima estimación realizada para la seña que se este mostrando, así, al principio hace valoraciones diferentes interpretando otras señas, pero al final consigue dar con la clase del gesto correcta.



(a) (b) (c) (d) (e)

Figura 23. Imágenes con las pruebas en tiempo real del modelo *MediaPipe + LSTM*. (a) Predicción de la letra G acertando con la estimación correcta en el tercer intento. (b) Predicción de la letra J acertando con la estimación correcta en el quinto intento. (c) Predicción de la letra Ñ acertando con la estimación correcta en el quinto intento. (d) Predicción de la letra S acertando con la estimación correcta en el quinto intento. (e) Predicción de la letra Z acertando con la estimación correcta en el tercer intento. **Tomado de:** Elaboración propia.

5. Conclusiones

En este trabajo de investigación se presentaron dos modelos con redes neuronales recurrentes como principal propuesta de solución, esto apuntando al objetivo de proponer diversas estrategias ante la problemática del proyecto.

Para el primer modelo *CNN + LSTM*, según los resultados encontrados se puede concluir que combinar la extracción de características que proporcionan las redes neuronales convolucionales y el seguimiento de secuencias entrelazadas que aporta las redes neuronales recurrentes es una solución que permite dar avance ante cualquier implementación que requiera el entrenamiento por medio de vídeos. La recomendación para este modelo es buscar alternativas para aumentar el tamaño de las imágenes o también aumentar las cantidad de imágenes en la base de datos para que exista una mejor estimación.

En el segundo modelo *MediaPipe + LSTM*, se demuestra el potencial que tienen las redes neuronales recurrentes, ya que, con tan solo información almacenada en vectores se llegó a unos resultados en cuanto a la predicción bastante prometedores, siendo un beneficio principal el bajo consumo en memoria y la facilidad de entrenamiento. En cuanto a la principal limitante es claramente compatibilidad de un sistema con la librería *MediaPipe*, al momento de quererse implementar.

La plataforma de desarrollo *MaixHub* es un avance en el desarrollo de la inteligencia artificial al momento de usar los dispositivos de la empresa *Sipeed*, ya que es una opción fácil de utilizar e intuitiva para cualquier desarrollador que requiere una implementación rápida. Hay que tener en cuenta que a pesar de las notorias ventajas es todavía muy limitada sobre todo ante modelos entrenados con bases de datos propias, ya que no permite bases de datos de mas de 2000 elementos.

Pese a las limitantes de cada modelo se pudo constatar cómo la precisión en las dos alternativas siempre fue mayor al 80 % y el alcance propuesto se cumplió teniendo

en cuenta que se lograron identificar todas las señas pertenecientes al abecedario del lenguaje de señas colombiano.

Las aportaciones principales de este trabajo apuntan directamente a la exploración en cuanto a las ventajas y alternativas que pueden generarse en la implementación de redes neuronales, como también los esfuerzos de conseguir dispositivos inteligentes que ayuden a la inclusión en el entorno social de las personas con capacidades diversas.

Bibliografía

(2018), INSOR. *REALIDADES DE LA POBLACIÓN SORDA EN 10 CIUDADES DEL PAÍS*. [En linea] Disponible en: https://www.insor.gov.co/home/realidades-de-la-poblacion-sorda-en-10-ciudades-del-pais/ (vid. pág. 11).

(2018), OMS. *Sordera y pérdida de la audición*. [En linea] Disponible en: https://www.who.int/es/news-room/fact-sheets/detail/deafness-and-hearing-loss (vid. pág. 11).

Abadi, Martín et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015 (vid. pág. 23).

Cooper, Steven. *Data Scienve from Scratch* (vid. pág. 15).

DotCSV. *¿Qué es una Red Neuronal? Parte 1 : La Neurona | DotCSV*. En esta serie de vídeos vamos a aprender cuál el funcionamiento de las potentes Redes Neuronales. En esta primera parte veremos el funcionamiento de una neurona y cuál es su relación con el modelo de regresión lineal. 2020 (vid. pág. 16).

Farnham, Boston et al. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems SECOND EDITION* (vid. págs. 15-19).

Nasiriany, Soroush et al. *A Comprehensive Guide to Machine Learning*. 2019 (vid. pág. 18).

Olah, Christopher. *Understanding LSTM Networks*. You can email me at christopherolah.co@gmail.com. 2015 (vid. pág. 20).

Pertuz, Said. *0.1 presentación DIP*. Presentación del curso de Procesamiento Digital de Imágenes del programa de ingeniería electrónica de la Universidad Industrial de Santander. 2020 (vid. págs. 14, 15).

Rey, Daniel Mauricio Avila. *Construcción y Entrenamiento de una Red Neuronal para la Identificación de Gestos. [recurso electronico]*. spa. Bucaramanga: UIS, 2021 (vid. págs. 13, 22, 23, 26).

Sandler, Mark et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". En: (2018). DOI: 10.48550/ARXIV.1801.04381 (vid. pág. 24).

Sipeed. *Maix-II Dock*. 2021 (vid. pág. 27).

— *MaixHub*. 2021 (vid. pág. 28).

Tzutalin. *LabelImg*. Git code. 2015 (vid. pág. 28).