**TecnoLógicas**
ISSN-p 0123-7799
ISSN-e 2256-5337
Vol. 23, No. 48, pp. 197-232
Mayo-agosto de 2020

Artículo de Investigación/Research Article

# Modelo computacional para reconocimiento de lenguaje de señas en un contexto colombiano

## Computational Model for Sign Language Recognition in a Colombian Context

Nelson Ortiz-Farfán [1]
Jorge E. Camargo-Mendoza [2]

Recibido: 27 de enero de 2020
Aceptado: 17 de abril de 2020

Cómo citar / How to cite

N. Ortiz-Farfán, J. E. Camargo-Mendoza, "Modelo computacional para reconocimiento de lenguaje de señas en un contexto colombiano", *TecnoLógicas,* vol. 23, no. 48, pp. 197-232, 2020.
https://doi.org/10.22430/22565337.1585



[1] MSc en Ingeniería de Sistemas y Computación, Departamento de Ingeniería de Sistemas e Industrial, Universidad Nacional de Colombia, Bogotá-Colombia, nmortizf@unal.edu.co
[2] PhD en Física, Departamento de Física, Universidad Nacional de Colombia, Bogotá-Colombia, jecamargom@unal.edu.co

**Resumen**

Este documento presenta la implementación de un software de reconocimiento de lenguaje de señas colombiano para personas sordas. Para este propósito, el Aprendizaje de Máquina será usado como base del sistema específico. Hoy en día no existe un repositorio público de imágenes o video que contenga estas señas ni la información necesaria para alcanzar esta meta, siendo uno de los principales impedimentos para iniciar la tarea. Por esta razón, se comenzó con la construcción de un repositorio. Pese a las dificultades de tiempo de los participantes, cinco personas realizaron las señas ante una cámara de video, de donde se obtuvieron las imágenes que compondrían el repositorio. Una vez hecho esto, las imágenes se usaron como datos de entrenamiento de un modelo computacional óptimo que puede predecir el significado de una nueva imagen presentada. Evaluamos el rendimiento del método utilizando medidas de clasificación y comparando diferentes modelos. La medición conocida como Accuracy fue un factor importante para medir los diferentes modelos obtenidos y así elegir el más adecuado. Los resultados muestran que es posible proporcionar nuevas herramientas a las personas sordas para mejorar la comunicación con otras personas que no conocen el lenguaje de señas. Una vez que se han elegido los mejores modelos, se prueban con nuevas imágenes, similares a las del entrenamiento, donde se puede ver que el mejor modelo logra una tasa de éxito de alrededor del 68 % de las 22 clases utilizadas en el sistema.

**Palabras clave**

Personas sordas, Aprendizaje de Maquina, modelo computacional, lenguaje de señas.

**Abstract**

This document presents the implementation of a Colombian sign language recognition software for deaf people. For this purpose, Machine Learning will be used as the basis of the specific system. Today there is no public repository of images or video that contains these signs or the information necessary to achieve this goal, being one of the main obstacles to undertake the task. For this reason, the construction of a repository was started. Despite the time constraints of the participants, five people carried out the signs in front of a video camera, from which the images that would make up the repository were obtained.Once this was done, the images were used as training data for an optimal computer model that can predict the meaning of a new image presented. We evaluated the performance of the method using classification measures and comparing different models. The measurement known as Accuracy was an important factor in measuring the different models obtained and thus choosing the one most suitable. Results show that it is possible to provide new tools to deaf people to improve communication with others who do not know sign language. Once the best models have been chosen, they are tested with new images, similar to those in the training, where it can be seen that the best model achieves a success rate of around 68 % of the 22 classes used in the system.

**Keywords**

Deaf people, Machine Learning, computational model, sign language.

## 1. INTRODUCCIÓN

La necesidad de realizar el presente trabajo surge a raíz de problemáticas de comunicación entre las personas sordas y las oyentes en diferentes entidades de servicios, ya sean de carácter público o privado. Hoy en día se cuenta con una plataforma en línea para tener acceso de un intérprete de lenguaje de señas, pero su disponibilidad y confidencialidad de los datos no es del todo completa, por lo que se requiere buscar nuevas estrategias informáticas que abarquen una solución más automática.

En la sociedad colombiana actual, la población sorda ha mejorado su calidad de vida con el apoyo de las tecnologías de la información. Parte de este proceso se debe a las normas y leyes establecidas por la Constitución Política y decretos de diferentes ministerios que buscan dar igualdad de oportunidades a las personas, sin importar si presenta una discapacidad.

Por ejemplo, hoy día es posible que la población sorda pueda acceder a instituciones educativas universitarias de forma virtual, como se indica en [1], basándose en las leyes que se mencionan en el artículo. En este trabajo, los autores Luz Myriam Rojas Rojas, Néstor Arboleda Toro, Leidy Johanna Pinzón Jaime mencionan cómo el uso de la tecnología ha permitido que la población discapacitada acceda a programas de la facultad de la Universidad Pedagógica y Tecnológica de Colombia (UPTC), de manera remota y con apoyo de traductores digitales.

Pero no solo la educación ha sido abierta a esta inclusión; también en el deporte se ha intentado establecer un lenguaje de señas específico, como se menciona en [2]. En el artículo, se logra evidenciar, a través de un estudio en 11 ciudades, cómo el lenguaje de señas puede tener diferentes gestos y no se tiene un estándar como se espera. Ejemplo de esto, como se ve en la (Fig. 1), es la representación de una tarjeta roja con cuatro gestos diferentes, que podrán ser entendidos dependiendo de la edad, la región o los antecedentes educativos de las personas que la usan. En esta investigación se declara que el Lenguaje de Señas Colombiano (LSC) es una lengua nueva con no más de 90 años y considerada minoritaria.

Sin embargo, a pesar de los avances obtenidos hasta el momento, mucha población oyente colombiana desconoce el LSC, haciendo que en interacciones con personas sordas la comunicación no pueda ser efectiva. Esta problemática se puede presentar con mayor recurrencia en entidades públicas y privadas, que prestan servicios en general y que atienden público.

Para solventar esta problemática, el Ministerio de Tecnologías de la Información y las Comunicaciones de Colombia (MinTIC) ha desarrollado en los últimos años el Servicio de Intérprete En Línea (SIEL), al cual se accede a través del portal web Centro de Relevo como se observa en la (Fig. 2) [3].

Fig. 1. Lengua de señas para representar una tarjeta roja en Cali. Fuente: [1].


Fig. 2. Servicio de Intérprete en Línea, SIEL. Fuente: [3].

A través de este portal, las personas sordas u oyentes crean una cuenta con la cual solicitan el servicio del intérprete.

Deben contar además con ciertos requerimientos técnicos de hardware y de permisos de red o uso de navegadores específicos, que en algunas entidades puede simbolizar un riesgo de seguridad [4].

A pesar de ser un buen servicio, uno de los mayores inconvenientes es que depende de la renovación del convenio anual con el Estado, lo cual no ofrece cobertura total.

Tampoco se ofrece soporte ni garantía de la confidencialidad de los datos que se tratan, afectando la privacidad de la población sorda. Los servicios poseen una duración de solo 30 minutos, lo cual puede no cubrir toda la necesidad de la persona sorda que lo solicita, teniendo que establecer una nueva sesión que retrasa el proceso de comunicación.

Dadas las problemáticas existentes, se desea investigar la existencia de un software que sea capaz de reconocer estas señas de manera automática y que se encuentre disponible la mayor cantidad del tiempo posible del año en las entidades colombianas para dar atención a la población sorda. Sin embargo, la complejidad de tener un sistema que sea capaz de reconocer los gestos del lenguaje de señas con las tecnologías tradicionales conlleva a pensar que otro tipo de solución informática se pueda implementar garantizando la fiabilidad en su proceso.

Dados los avances tecnológicos de las últimas décadas, nuevas investigaciones en reconocimiento de imágenes y videos y el uso de algoritmos de inteligencia artificial (IA) por grandes empresas como Google, Amazon, Facebook, entre otros, proponen el uso del aprendizaje de máquina, o Machine Learning (ML) en inglés, para solucionar este tipo de problemas. Prueba de ello es el uso que hace Google de este tipo de tecnología para detectar la presencia de cáncer de seno con mayor precisión que los métodos, que utilizan una gran cantidad de imágenes históricas que poseen la enfermedad para luego compararlas con el estado actual del paciente [5].

Es por esa razón, que el objetivo principal de este trabajo consistió en determinar qué tan exactos pueden ser los modelos construidos con aprendizaje de máquina, más específicamente con Deep

Learning (DL) o Aprendizaje Profundo para reconocer imágenes de señas empleadas por personas sordas en Colombia y si pueden ser usados en la industria para dar apoyo a esta población en sus necesidades, especialmente cuando requieren servicios públicos y/o privados de las diferentes entidades a nivel nacional.

## 2. ESTADO DEL ARTE

En algunos países es común ver la construcción de sistemas que permiten traducir texto a señas usadas por personas sordas para establecer una comunicación desde el oyente hacia esta comunidad.

Un ejemplo de ello es el software construido en [6], en el que se evidencia cómo a través de una aplicación web, luego de seleccionar una expresión o palabra, aparece una imagen de un avatar o una persona pregrabada representando en el lenguaje de señas el respectivo concepto (Fig. 3).

En Brasil, el uso de sistemas de reconocimiento de lenguaje de señas es más común, y se busca que desde diferentes fuentes se pueda transformar la información hacia las señas que son usadas por las personas sordas. En [7] se describe cómo el uso de la multiplataforma llamada VLibras es usada para transformar texto y video para que un avatar exprese lo que se quiere transmitir hacia la persona sorda (Fig. 4). Sin embargo, este tipo de comunicación es de persona oyente hacia persona sorda, pero no contempla el otro sentido de la comunicación, es decir, desde persona sorda hacia el oyente, quien en la mayoría de los casos desconoce el lenguaje de señas.

Así mismo, en muchas situaciones cotidianas la comunicación es principalmente verbal y uno de los mayores inconvenientes para establecer comunicación entre el lenguaje verbal y el lenguaje de señas de las personas sordas es que no existe un estándar definido.

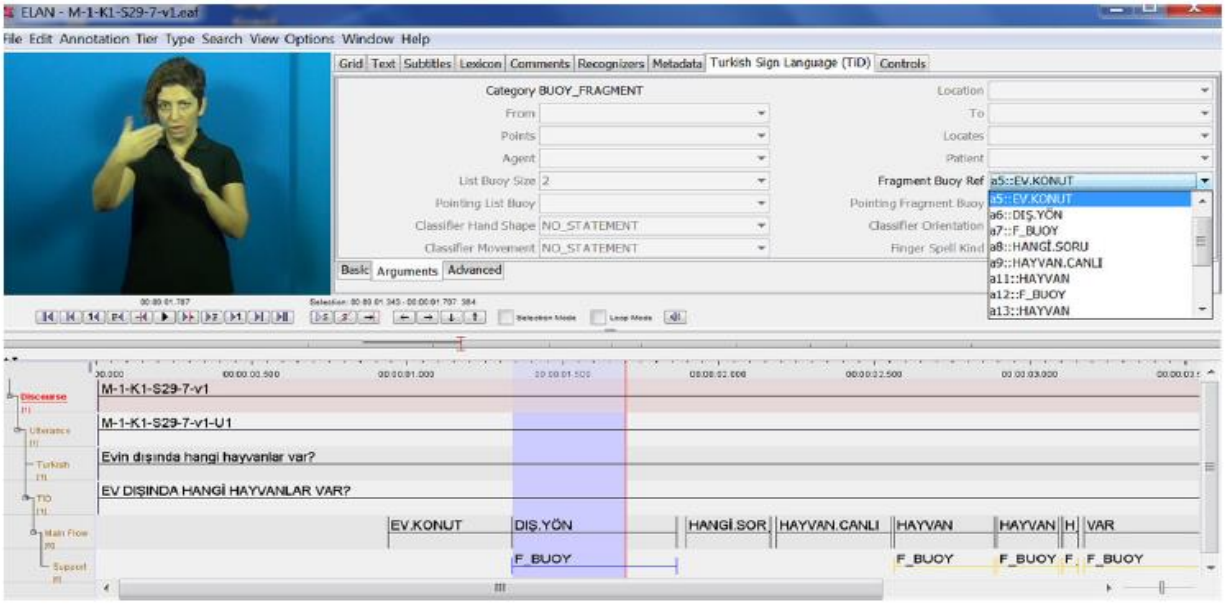


Fig. 3. Video de expresión: "¿Qué animales están fuera de la casa?" en idioma turco
Fuente: [6].

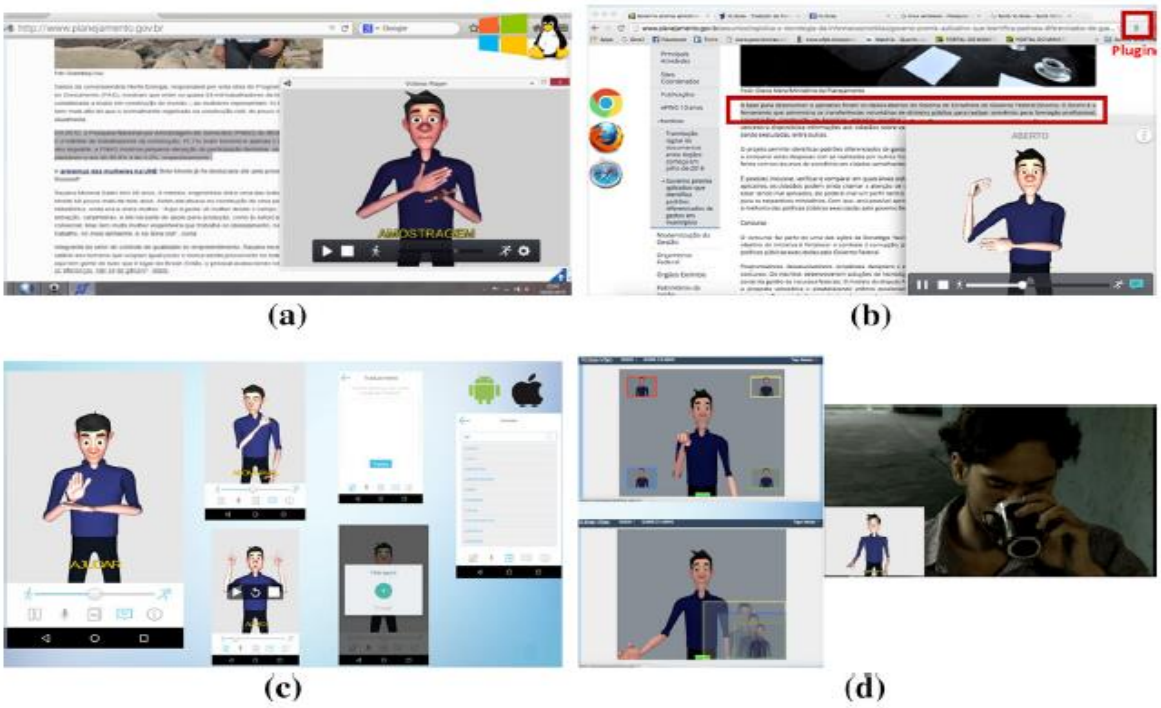


Fig. 4. Aplicación VLibras para traducir textos y videos en diferentes plataformas hacia avatares que realizan señas. Fuente: [7].

Esto sucede ya sea a nivel universal o local como se describe en [8], en donde se plantea una posible solución para mitigar esta problemática en la India. Los autores consideraron el uso de una notación escrita intermedia que permita asociar estos 2 componentes (Fig. 5).

No obstante, esto no resuelve la problemática inicial correspondiente entre la comunicación de una persona hacia un oyente, ya que el uso de estas notaciones tampoco está centralizado y generalizado para todo el público.

Otro de los inconvenientes existentes consiste en que cada país puede tener sus propias definiciones e incluso no siempre existe una relación entre las señas y el lenguaje hablado [9].

Incluso en Colombia, cada región, zona o departamento cuenta con algunas señas propias, aunque eso sí, otras de las que se usan son universales. Debido a esto, se decide trabajar con las señas ofrecidas en la página web del Instituto Nacional de Sordos de Colombia (INSOR) y las manejadas por un intérprete de señas de la ciudad de Bogotá en sus labores diarias.

Se comienza investigando sistemas automáticos capaces de reconocer señas digitalmente. El primero de ellos que se menciona en [9] consiste en tomar muestras de las señas que se usan para construir un repositorio de datos de entrenamiento y con la ayuda de la técnica de aprendizaje de maquina denominada Support Vector Machine (SVM) o Maquinas de Soporte Vectorial, se clasifica en las diferentes clases que se tienen en este repositorio. La recolección de estas señales se hace con el apoyo de un sensor de movimiento o de tecnología de cámaras de video, de acuerdo con (Fig. 6).

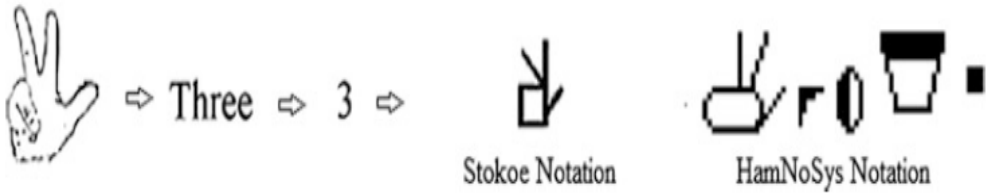


Fig. 5. Uso de notaciones para relacionar el lenguaje verbal y el lenguaje de señas
Fuente: [8].

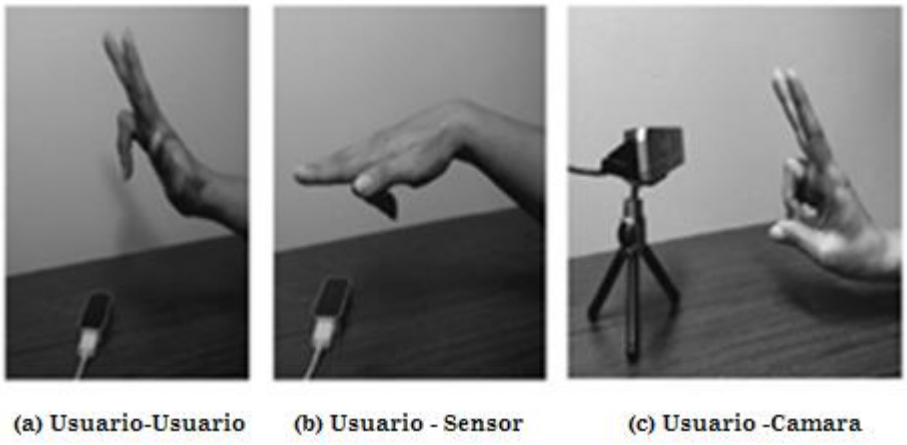


Fig. 6. Registro señas con sensores y cámaras. Fuente: [9].

Un sistema similar se propone en India en 2016, [10] donde a una imagen se le hace un procesamiento con un filtro y se obtienen los patrones a reconocer (manos, figura de cabeza, torso, etc.). El sistema propuesto consiste en adquirir imágenes, aplicarles filtros en blanco y negro para evitar interferencia de los colores, detectar las características de las formas de cada señal y asignar una clasificación, de acuerdo con el diagrama de bloques de (Fig. 7).

En [10] el entrenamiento de cada seña es realizado y la modificación de cada imagen para obtener las características específicas de acuerdo con (Fig. 8).

En [11] se construyó un modelo en un ambiente controlado, donde la información capturada por vídeo se procesó a través de parámetros espaciales y temporales como se observa en (Fig. 9).

Los resultados y conclusiones obtenidos de [11] determinaron que esta técnica no es adecuada, dado que parámetros como luz, locación de la persona, fondos dinámicos y no uniformes, tono de piel de la persona y calidad de la imagen procesada varían los resultados.

Un sistema más complejo y robusto se propone en 2018 en [12] para reconocer señas de la lengua tailandesa. Este sistema es similar a los descritos, solo que el clasificador es una red neuronal de muchas capas que da una gran precisión, según (Fig. 10).

Este tipo de sistemas es más robusto, pero requiere un mayor entrenamiento y mayor cantidad de datos de entrenamiento.

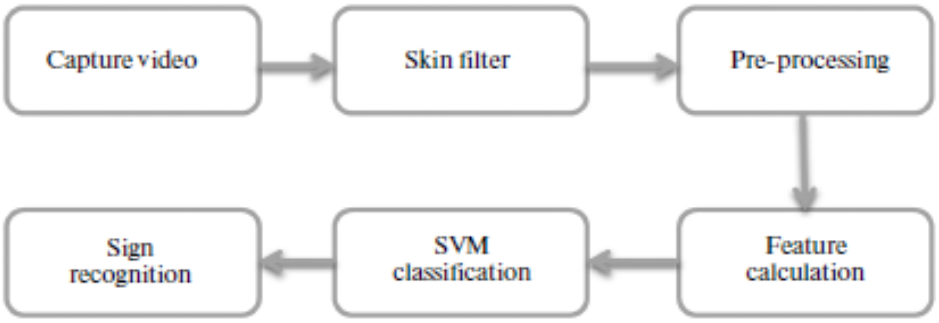


Fig. 7. Diagrama de bloques de sistema de reconocimiento de señas en India. Fuente: [10].

Fig. 8. Entrenamiento de imágenes de lenguaje de señas sorda en India. Fuente: [10].

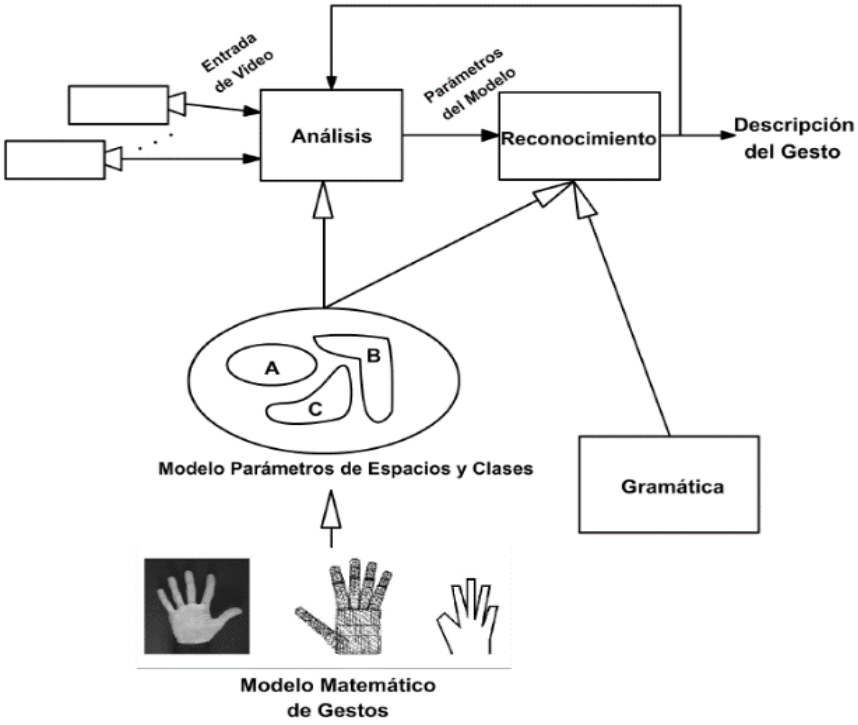


Fig. 9. Sistema de interpretación de gestos basados en la visión. Fuente: [11].

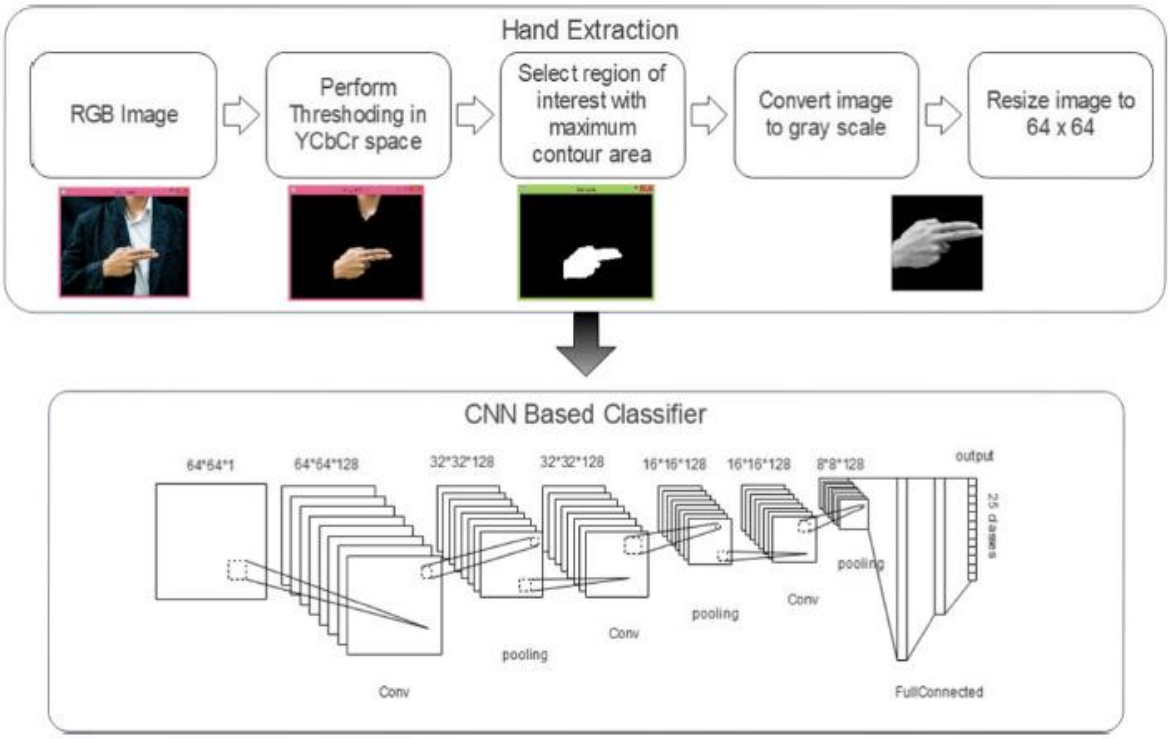


Fig. 10. Sistema de reconocimiento de señas usando redes neuronales. Fuente: [12].

En el lenguaje de señas, es necesario tener en cuenta la existencia de gestos dinámicos y estáticos. Con base en el trabajo de doctorado de Ronchetti Franco en la Universidad Nacional de la Plata, Argentina [13], desde el punto de vista informático, una seña o gesto dinámico es aquel que requiere del movimiento de alguna parte del cuerpo para dar un significado. Por su parte, la seña o gesto estático solo necesita una pose en un solo instante para proporcionar un significado o idea [13].

En el momento de revisar la literatura, se encuentra que en [14] la forma de trabajar los gestos dinámicos es grabarlos en video y luego obtener varias *frames* o capturas de imágenes secuenciales, etiquetando todo el conjunto de imágenes como una sola clase. Se debe tener en cuenta que muchas de las imágenes obtenidas del video pueden ser ruido o información que no será útil para clasificar, por lo que se hace una limpieza para tener los datos de entrenamiento que se usaran en una Convolutional Neural Network (CNN) o Red Neuronal Convolucional, como se ve en (Fig. 11).

Los resultados del experimento muestran en (Fig. 12) una matriz de confusión con porcentajes de aciertos muy altos para la mayoría de las señas [14].

Recientemente en [15] se realizó un estudio de las diferentes técnicas para clasificar imágenes que se puedan aplicar a un determinado concepto árabe, utilizando técnicas de DL (Fig. 13).

En esta investigación se encontró que el mejor *Accuracy* se obtenía al aplicar CNN con un valor de 97.82 % sobre un conjunto de datos relacionados con el alfabeto de señas americano. En el contexto del alfabeto árabe se obtuvo un *Accuracy* de 98.05 %, definiendo esta técnica como la que mejores resultados brinda.

En [16] se realizó un trabajo de reconocimiento de lengua de señas usando técnicas de CNN, en el cual se adquiere la imagen a color, se procesa a escala de grises y se realiza la extracción de la seña en la

imagen para poder clasificarla dentro de la categoría para que luego sea predicha por el sistema (Fig. 14).

En este sistema se extraen características de cada una de las imágenes examinadas de prueba. El *Accuracy* obtenido para el sistema es cercano al 86 %, lo cual es bueno teniendo en cuenta además que el sistema permite obtener texto de voces introducidas al sistema (Fig. 15).

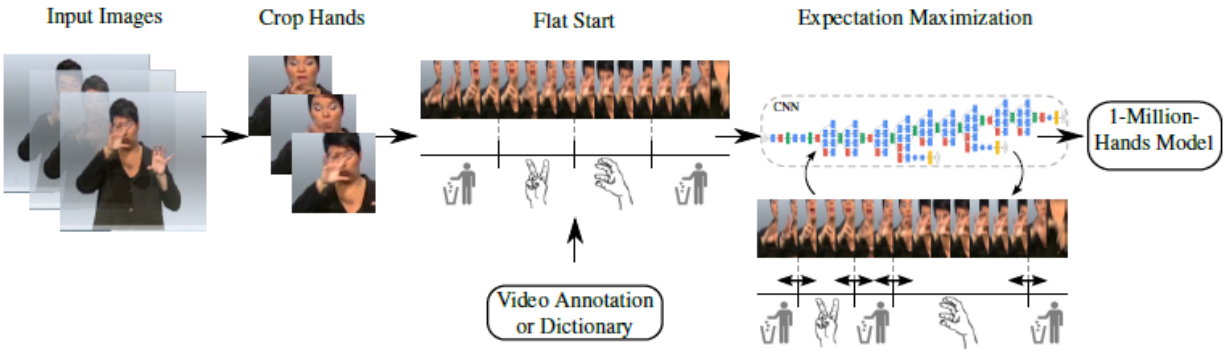


Fig. 11. Manejo de gestos dinámicos con las manos para entrenar en una CNN
Fuente: [14].


Fig. 12. Matriz de Confusión reconocimiento de gestos estáticos y dinámicos usando CNN
Fuente: [14].

Fig. 13. Lenguaje de señas del alfabeto árabe. Fuente: [15].

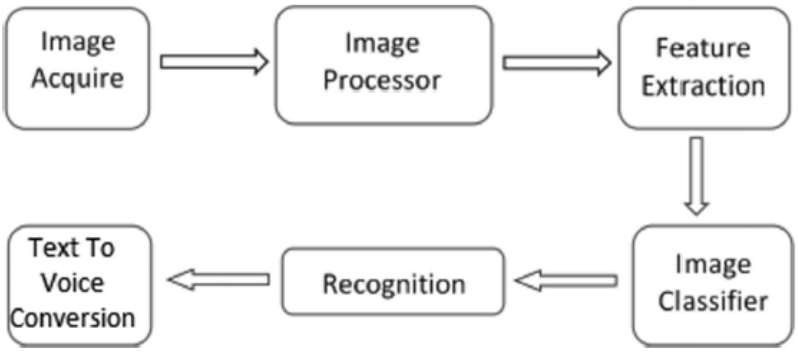


Fig. 14. Arquitectura del sistema de clasificación de lengua de señas en India usando CNN
Fuente: [16].


Fig. 15. Extracción de la seña Victoria de una imagen pre procesada. Fuente: [16]

Un trabajo similar se tiene para reconocer gestos de manos haciendo uso de aprendizaje multimodal [17]. En este se toman como insumo imágenes en 3 diferentes ambientes, a color, escala de grises con profundidad y video. Cada una de las entradas es tratada con CNN y finalmente los resultados se unen para ser la entrada de un clasificador SVM para detectar los gestos de mano presentes (Fig. 16).

El *Accuracy* obtenido por este sistema fue 97.66 % usando 1100 imágenes como entrenamiento y 300 de prueba, con una validación cruzada con 5 pliegues.

Se observa que las CNN independientes de las imágenes a color y en escala de grises dan valores de *Accuracy* de 93.17 % y 92.61 %, mientras que para las de movimiento que usaron solo una Neural Network (NN) o Red Neuronal el *Accuracy* fue de 82.83 %, lo que demuestra que las CNN ofrecen mejor clasificación.

En Bangladesh [18] una situación similar a la que se plantea en el presente artículo se intentó solucionar, dando como resultado que después de revisar varias técnicas de aprendizaje computacional del estado del arte, las CNN son las más eficientes para reconocer este tipo de gestos.

En esta investigación se construye el repositorio de datos desde cero, utilizando los números del 0 al 35, tratando como tal un problema de clasificación de 36 clases.

Cada clase tenía un total de 50 imágenes y el repositorio en su totalidad tenía 1800 imágenes, cada una de ellas en formato JPG y con dimensiones de 128x128 pixeles (Fig. 17).

La arquitectura del sistema construido constó de 10 capas de convolución con función de activación *Relu*, un *Kernel* de 3x3, capas de *Dropout* con valores entre 25 y 50 % para evitar *Overfitting* y una capa final con función de activación *Softmax* (tasa de *Learning Rate* se deja con valor de 0.001 con un optimizador Adam) (Fig. 18).

Como resultado, el sistema obtuvo un Accuracy de 92.65 % para los datos de entrenamiento y de un 92.74 % para los datos de prueba.

Con base en lo anterior y dado que las redes neuronales con DL han tomado gran fuerza en la industria y la investigación, se elige un modelo de una CNN para este proyecto.

Muchas de las CNN existentes hoy en día se basan en métodos ingenieriles (experimentación, prueba y error) para obtener sistemas que funcionen correctamente. Cuando se busca una CNN que se relacione con la detección de gestos se encuentra en [19] un foro con la detección del alfabeto universal de lenguaje de señas que se visualiza en (Fig. 19).

La (Fig. 20) muestra la implementación de una CNN usando 2 capas convolucionales y 2 capas de *Pooling* como capas intermedias, haciendo uso de un *Kernel* o filtro de 3x3 (estándar manejado en CNN) y logrando un *Accuracy* de 85 % para un total de 24 clases [19].

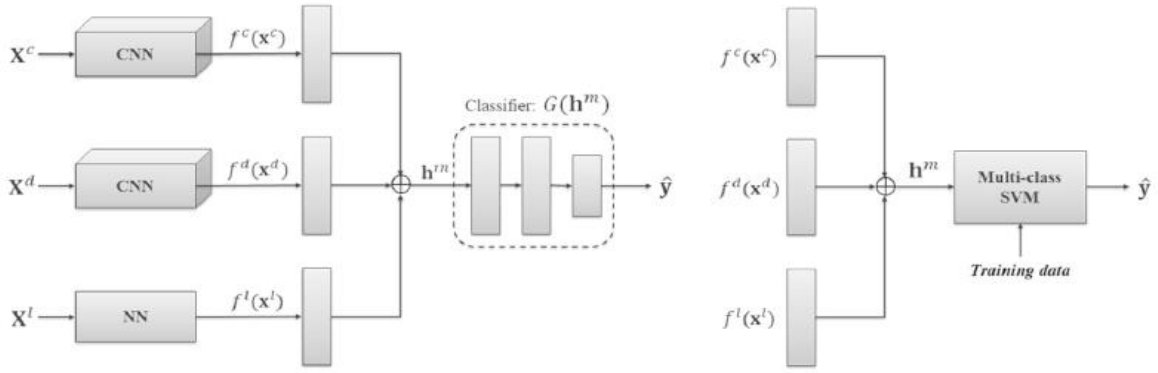


Fig. 16. Arquitectura sistema multimodal para reconocer gestos de la mano usando CNN y SVM
Fuente: [17]

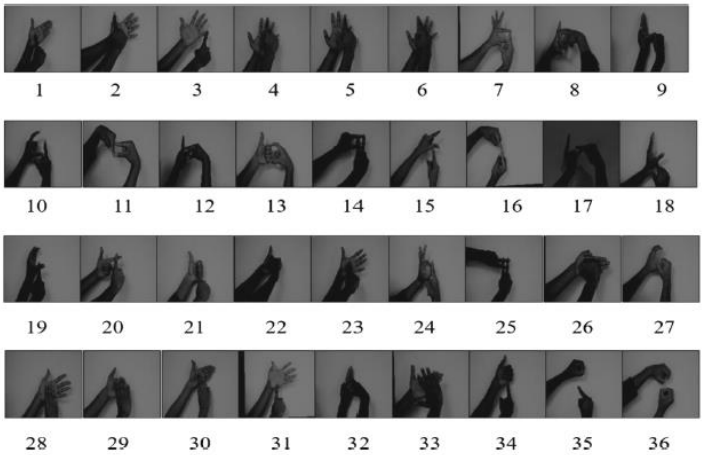


Fig. 17. 36 clases de números representados en lenguaje de señas en Bangladesh
Fuente: [18].

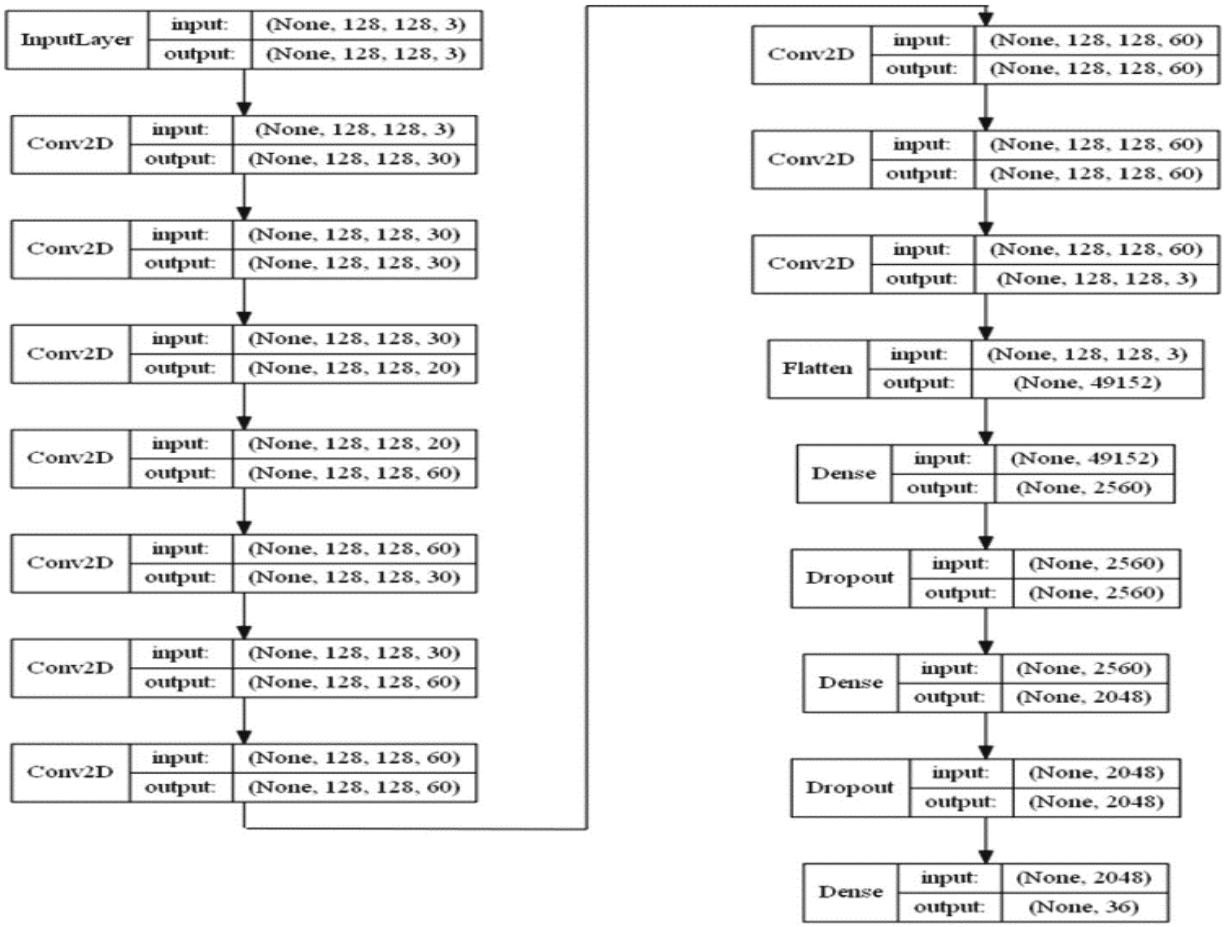


Fig. 18. Arquitectura de la CNN de clasificación de caracteres en Bangladesh
Fuente: [18].


Fig. 19. Reto KAGGLE para detectar alfabeto de Lenguaje de Señas Universal. Fuente: [19].

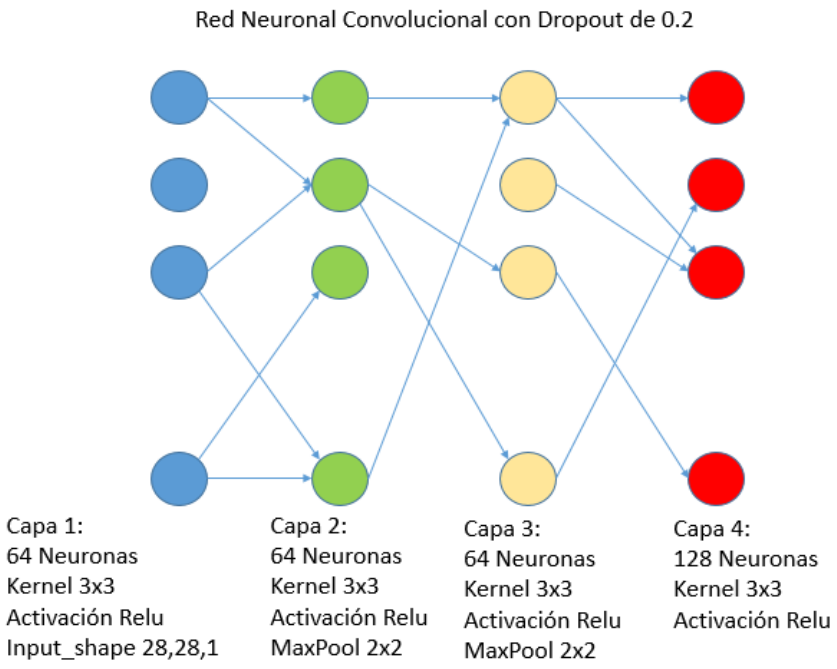


Fig. 20. Red CNN con *Dropout* para 24 clases. Fuente: elaboración propia.

## 3. IMPLEMENTACIÓN

### 3.1 Construcción del Repositorio de Imágenes

Dado que el INSOR aún no cuenta con un repositorio digital de lenguaje de señas, se construye uno para el sistema. Se usa como guía el contenido de la página web de la entidad que se detalla en (Fig. 21) y el conocimiento de una interprete [20].

El lenguaje de señas se constituye de gestos dinámicos y estáticos. De los videos y el apoyo de la intérprete de señas se revisa gestos estáticos fáciles de replicar. Con respecto a los gestos dinámicos, algunos se conforman de gestos estáticos que poseen desplazamientos lineales durante el tiempo (es decir, la postura de las manos o el torso es el mismo, pero existe un movimiento de translación). De este último grupo, se seleccionarán algunas señas y se etiquetarán en diferentes momentos temporales con la misma palabra o expresión, de acuerdo con lo expresado en [20].

Por ejemplo, en la siguiente seña (Fig. 22) presente en la página, se ve que las manos de la persona que realiza el gesto se mueven horizontalmente, mientras que su torso se mantiene estático. La idea es tomar gestos donde parte de las manos se vea relacionada con el torso o incluso partes de la cabeza como el mentón o la sien [20].

Con estos criterios establecidos, se construye un repositorio de imágenes con 22 diferentes gestos del lenguaje de señas de la página Web del INSOR y la experiencia del intérprete. Cada uno de estos gestos serán las diferentes clases del sistema. Se toma la idea de un procedimiento hecho en Argentina [13] en la que se trabajan con M intérpretes o personas que conocen el lenguaje de señas para tener más datos que entrene el modelo.

De esta cantidad de personas, la sugerencia en [13] es que solo M-1 serán usadas para la etapa de entrenamiento y la persona restante se incluirá durante la etapa de prueba.


Fig. 21. Página Web Diccionario de Gestos de INSOR. Fuente: [20].

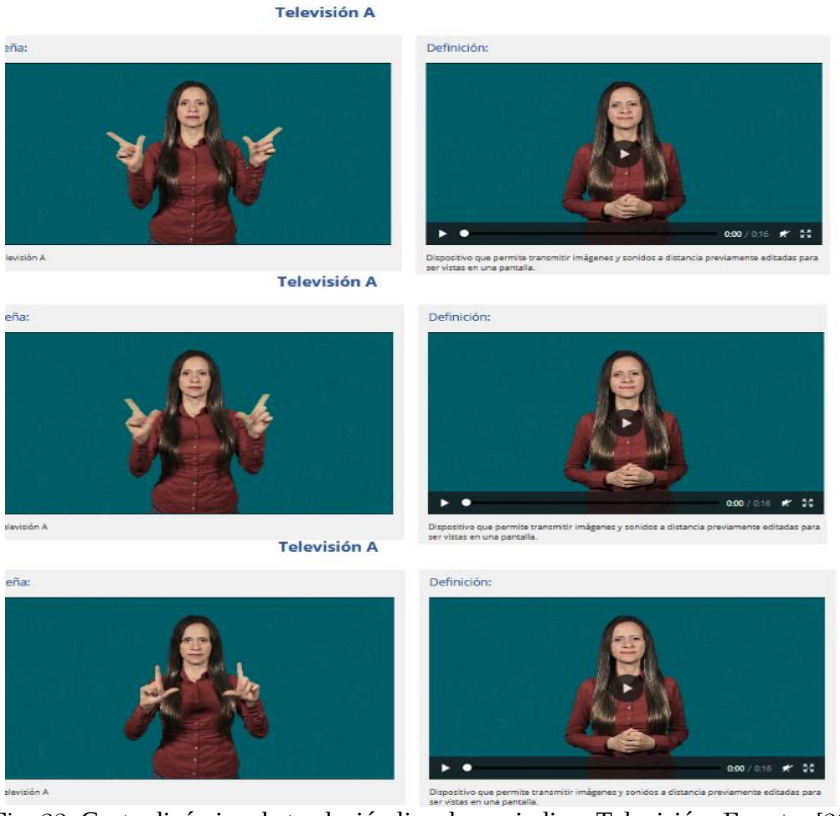


Fig. 22. Gesto dinámico de traslación lineal para indicar Televisión. Fuente: [20]

En el presente caso, se entrenó el modelo con 5 personas y la última persona (interprete) se incluyó con los otros para evaluar el prototipo de software.

El criterio para elegir las señas se basa en su facilidad para replicar y, adicionalmente, que los movimientos no involucren complejidad para ser detectadas.

El listado de las señas correspondientes a las diferentes clases del sistema es el siguiente:

- Atardecer
- Baño
- Casa
- Color
- Escuchar
- Gracias
- Hola1 (usando 2 dedos)
- Hola2 (usando 4)
- Hoy
- Mamá
- Mucho gusto
- Nombre
- Novio
- Papá
- Profesor
- ¿Qué paso?
- Siéntese
- Televisión
- Tener curiosidad
- Tener algo como posesión
- Universidad
- Yo

A continuación, se muestran algunas de las 22 señas obtenidas de los videos con las diferentes personas que servirán como insumo para la construcción del repositorio, (Fig. 23 a 25). El total de imágenes del repositorio obtenidas es de 3168 con una distribución uniforme para las 22 clases.

Todas las imágenes contienen una resolución de 640x380 y guardadas en formato PNG. Estas imágenes son obtenidas como *frames* cada 0.0033 segundos de videos en blanco y negro donde se grabó a cada una de las personas realizándolas. Luego, cada imagen se guardó con un título numérico y se asoció con una etiqueta en un listado completo del repositorio en un archivo de Excel.

Fig. 23. Seña estática de la palabra YO
Fuente: elaboración propia.

Fig. 24. Seña estática de la palabra CASA
Fuente: elaboración propia.

Fig. 25. Seña dinámica de la palabra TELEVISOR. Fuente: elaboración propia.

### 3.2 Procesamiento de imágenes

La ventaja de construir el repositorio de señas es la de tener una distribución uniforme de las diferentes clases que la conforman. En total, las 3168 imágenes construidas poseen un total de 144 imágenes por cada una de las expresiones o palabras a clasificar (o clases del sistema).

Con esto en mente, se construyó una matriz de 2 dimensiones en Python del tipo *Array*, donde estas imágenes puedan ser representadas numéricamente con el fin de ser tratada por la CNN.

Dado que las imágenes capturadas poseen una resolución alta, gracias a la cámara comercial con la que fue adquirida, el primer paso es ajustar su tamaño sin que se pierda calidad. Con un ajuste en el tamaño de 320x240 pixeles se observa poca modificación de las imágenes originales.

Dado que computacionalmente una imagen es una matriz de números enteros, cada imagen se ajusta a un vector de tamaño 76 800 (valor resultante de multiplicar 320x240) y de esa manera todas las imágenes se almacenan en una matriz más fácil de manejar (3368 x 76800), justo como se detalla en (Fig. 26).

A su vez, en un archivo de Excel cada nombre de archivo de imagen es asignado con una etiqueta de su correspondiente significado o expresión de seña, en el mismo orden en que la matriz tiene guardada la imagen. Con la librería Pandas de Python, se cargó este archivo de Excel (almacenado en formato CSV) y se visualizaron los primeros registros de la asociación del nombre de la imagen con su etiqueta como se ve en (Fig. 27).

Dado que para la CNN se trabaja con datos numéricos, la variable de salida se representa por medio de '0' y '1' con la técnica conocida como *One Hot Code*.

En este caso, Python permite realizar esta transformación dando como resultado una matriz de 3168 registros (cantidad de imágenes) por 22 columnas (las 22 clases del sistema, donde cada salida es un conjunto de 21 '0' y un '1' que representa esa clase en determinada posición).

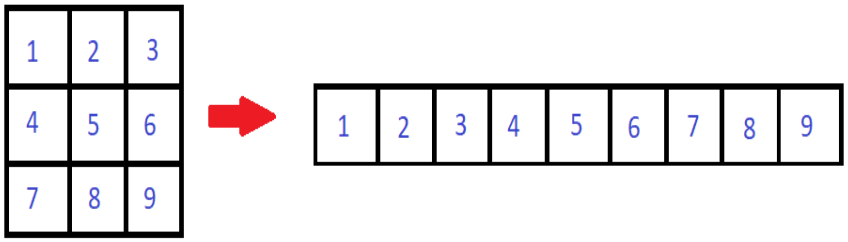


Fig. 26. Proceso de conversión de matriz 2x2 a vector de 1 dimensión usado en imágenes
Fuente: elaboración propia.

| | Secuencia | Nombre Archivo | Etiqueta |
|---|---|---|---|
| **0** | 1 | 1000 | HOY |
| **1** | 2 | 10001 | YO |
| **2** | 3 | 10002 | YO |

Fig. 27. Validación en Pandas de archivo de etiquetas asociadas con el nombre de archivo de imagen. Fuente: elaboración propia.

A continuación, se separaron los datos en aquellos que se usan para entrenar y los que se usan para probar el modelo a construir. Se indica que el 70 % de los datos es para entrenamiento y de manera estratificada, con el fin de conservar el balanceo de clases. Se adiciona también un valor de semilla en la aleatoriedad en estos datos con el fin de que en una próxima simulación se tengan las mismas imágenes y etiquetas elegidas aleatoriamente para la separación. Se valida nuevamente la distribución de las señas después de la separación y esta fue de 100 imágenes para los datos de entrenamiento.

Se aplicó un proceso de normalización sobre los datos para manejar los valores numéricos, tanto de los datos de entrenamiento como de prueba entre 0 y 1.

Para ello se determinaron los valores máximo y mínimo de las imágenes que son 255 y 0, respectivamente. Dada la simplicidad de estos valores, la normalización consistió en dividir todos los datos por el valor de 255, dando como resultado que para los datos de entrenamiento o *Train* el valor máximo y mínimo son 1 y 0, mientras que para los datos de Test son 1 y 0.003, resultado de la división flotante. Con esto, los datos ya están procesados y listos para construir el modelo.

### 3.3 Construcción del Modelo

El modelo se construye luego de haber realizado el procesamiento de las diferentes imágenes que servirán como datos fuente de entrenamiento. La arquitectura de este modelo es la misma planteada heurísticamente en [19], reconocimiento de alfabeto de señas de personas sordas, con la diferencia de que no se agregó inicialmente una capa de *Dropout* para ver como este factor altera una CNN totalmente interconectada.

En una red neuronal son muchos los hiperparámetros que varían para obtener diferentes resultados, haciendo que sea complejo cambiarlos todos a la vez (costo computacional). Los hiperparámetros de una red neuronal más comunes son el número de neuronas por capa, las funciones de activación presentes, el tipo de optimizador, la tasa de aprendizaje o *Learning Rate*, el número de épocas o interacciones de entrenamiento, momento, entre otros.

Por fortuna, la sencillez del modelo a usar permitió variar el número de neuronas existentes en las capas convolucionales junto con sus funciones de activación correspondientes. Para encontrar los parámetros óptimos se usó la técnica *Grid Search*, donde se variaron estos parámetros dentro un rango establecido y se combinaron los mismos. La arquitectura y funcionamiento básico de la red neuronal a implementar se describen a continuación.

La primera capa es convolucional con un *Kernel* o filtro de 3x3 ejecutado sobre cada una de las imágenes de entrada. Es la responsable de recibir los datos por lo que se indica que estos deben venir en una

dimensión de 320 x3 40. El número de neuronas y función de activación de la capa son parámetros por encontrar.

La capa que procede es de *MaxPooling* cuya matriz es de 2x2 y permite reducir el número de parámetros de salida de la capa de entrada. Esta capa toma el máximo valor de la imagen de cada 4 pixeles cercanos organizados en 2x2.

La tercera capa es similar a la primera con la diferencia de que no recibe la misma cantidad de entradas, ya que por la capa de *MaxPooling* el número es reducido.

Nuevamente, las neuronas y función de activación se definieron variables en esta capa.

La cuarta es una capa *MaxPooling* similar a la segunda, usada para reducir aún más el número de parámetros de la CNN. No tuvo neuronas al igual que la segunda.

Luego de pasar por las 2 capas de Convolución y *MaxPooling*, se usó una capa *Flatten* para que todas las salidas convergieran hacia una salida. Finalmente, se agregó una capa de 22 neuronas igual al número de clases existentes en el sistema, con una función de activación *Softmax,* asegurando que las probabilidades obtenidas den una sola clase predicha como salida.

Por simplicidad, se dejó como optimizador el valor de "*Adam*" y la pérdida o *Loss* de la red neuronal es categórica *Crossentropy*. La medida para ver la eficiencia del algoritmo es el *Accuracy*.

El proceso de *Grid Search* se acompañó de una técnica denominada *Cross Validation* [21]. Esta técnica consiste en que de los datos de entrenamiento de los datos originales se vuelve a hacer otra división. Esta división transforma todo el segmento de datos en conjuntos de datos más pequeños (N conjuntos) para entrenar N-1 y el restante para validar el modelo obtenido en el entrenamiento como se observa en (Fig. 28).

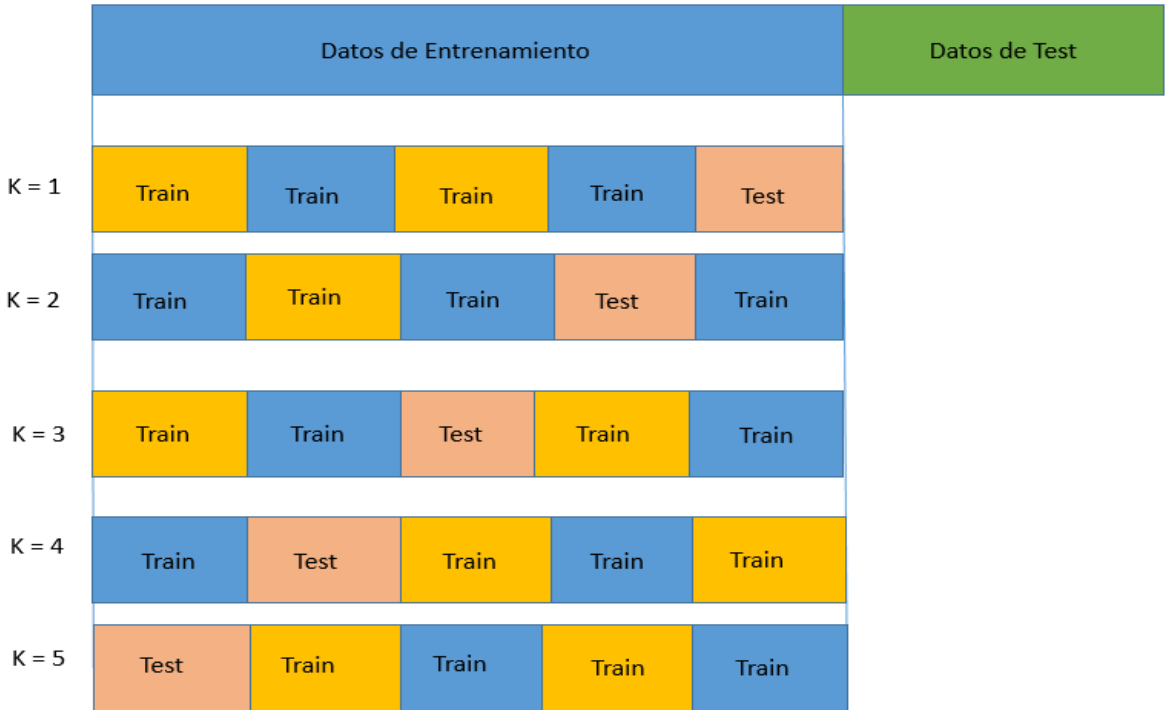


Fig. 28. *Cross Validation* con 5 pliegues. Sobre los datos de entrenamiento se hacen nuevas divisiones y se itera. Fuente: elaboración propia.

Este proceso es iterativo y se realiza N veces haciendo que cada conjunto al final sea usado como datos de validación.

Los resultados obtenidos se promedian para cada una de las combinaciones de hiperparámetros (número de neuronas y funciones de activación en este caso).

Los parámetros que se variaron para los hiperparámetros son los siguientes:

**Numero de neuronas** = [10, 15, 20, 25 y 30].

**Funciones de Activación** = [Relu, Tanh, Linear y Sigmoid] (más usados en la industria).

Una vez establecido, se entrenó el modelo indicando que se pueden hacer múltiples *jobs* en paralelo y usar un valor de 15 épocas. No se incluye un numero de tamaño de *batch* por lo que el sistema tomó por defecto un valor de 32. En el momento de realizar computacionalmente esta simulación, los recursos de Memoria y CPU tomaron valores picos cercanos al 100 %.

Dado que se tienen 5 diferentes números de neuronas, 4 diferentes funciones de activación y un valor de *Cross Validation* igual a 5 pliegues para mejores resultados, el *Grid Search* realizó 100 procesos de entrenamiento sobre la CNN con 2217 datos de entrenamiento. Después de unas cuantas horas de simulación, los resultados obtenidos se grafican en un mapa de calor de número de neuronas vs. la función de activación que se observa en (Fig. 29).

De los resultados obtenidos, la mejor combinación ocurrió cuando las capas convolucionales tienen un numero de 25 neuronas cada una y sus funciones de activación son *Tanh*, con un *Accuracy* en promedio de 0.983. En general, los Accuracy para las funciones de activación *Relu, Tanh* y *Linear*, sin importar la cantidad de neuronas en capas, se encuentra por encima del 90 %, siendo óptimas para trabajar.

Por el contrario, la función de activación *Sigmoid* ofrece resultados muy bajos inferiores al 5 % por lo que es descartada en su totalidad.

En el momento de evaluar los datos de prueba con el mejor resultado, el *Accuracy* obtenido es de 0.988, razón por la cual se construyó el modelo base especifico con estos parámetros.

El modelo obtenido se entrena de manera similar a como se hizo con el *Grid Search*, solo que sin usar *Cross Validation* y procesos en paralelo. Al usar 15 épocas y un tamaño de *batch* de 128 sobre las 2217 imágenes de entrenamiento, el *Loss* del modelo es 0.2348 y el *Accuracy* de 0.9693.

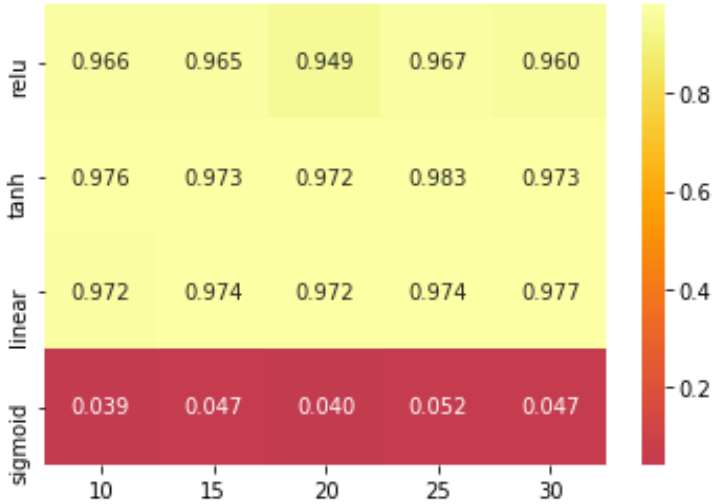


Fig. 29. Mapa de calor *Grid Search*. Funciones de activación vs número de neuronas por capas
Fuente: Fuente: elaboración propia.

La (Fig. 30) muestra cómo evolucionó el *Loss* en función del número de épocas.

Igualmente, en (Fig. 31) el *Accuracy* en función del número de épocas se ve como este aumenta para llegar a valores cercanos al 0.9.

Nuevamente se evaluaron los datos de test (951 imágenes) con un *Loss* igual al 0.2498 y el *Accuracy* de 0.9631, valores muy buenos para que el modelo sea usado.

### 3.4 Evaluación del Modelo

Del modelo obtenido, se agregó una capa *Dropout* similar a la inicial del modelo de [19] (valor de 0.2).

Al realizar el entrenamiento, el *Loss* es 0.1396 y el *Accuracy* 0.9707 sobre las mismas imágenes de entrenamiento. Las curvas de *Loss* y *Accuracy* con respecto a las mismas 15 épocas son (Fig. 32 y Fig. 33), respectivamente. Para los mismos datos de test, el *Loss* es 0.1516 y el *Accuracy* 0.9642.

A continuación, se tomó el modelo base inicial y se le agregaron 2 capas más convolucionales y de *MaxPool* y se construyeron 2 modelos adicionales con y sin *Dropout*. Se obtuvo otro modelo adicional, agregando 2 capas más convolucionales (para un total de 6) y de *MaxPool* con *Dropout*.

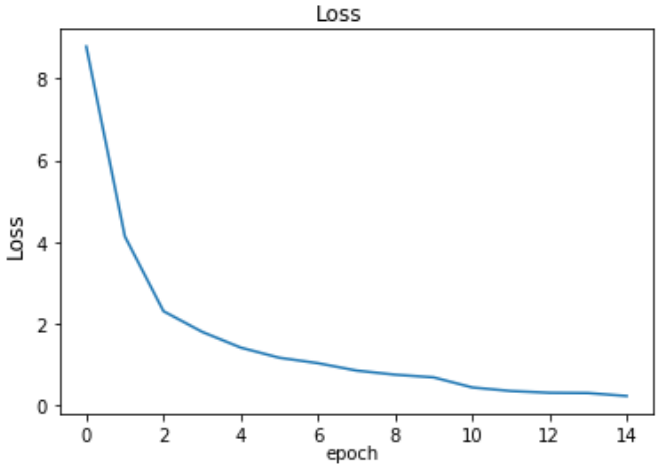


Fig. 30. *Loss* del modelo durante el entrenamiento por varias épocas
Fuente: elaboración propia.

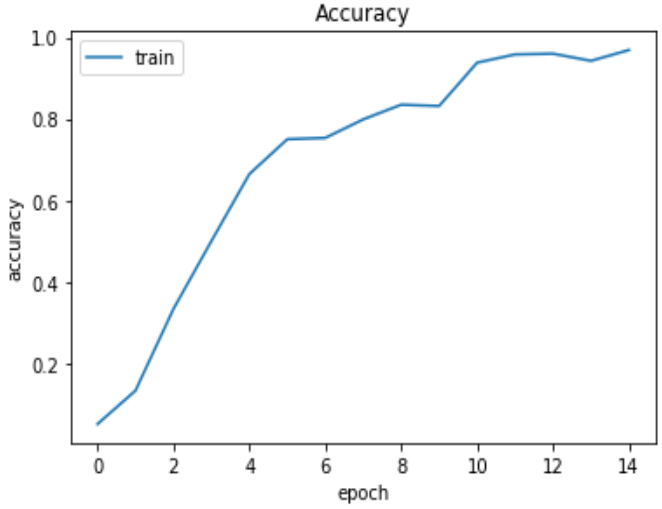


Fig. 31. *Accuracy* del modelo durante el entrenamiento por varias épocas
Fuente: elaboración propia.

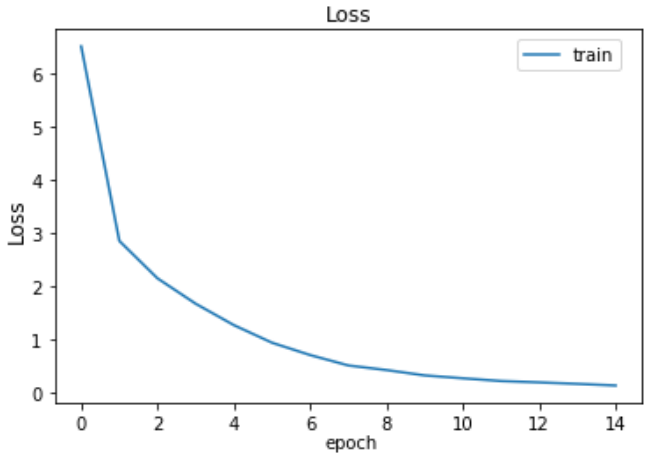


Fig. 1 *Loss* del modelo con *Dropout* durante el entrenamiento por varias épocas
Fuente: elaboración propia.

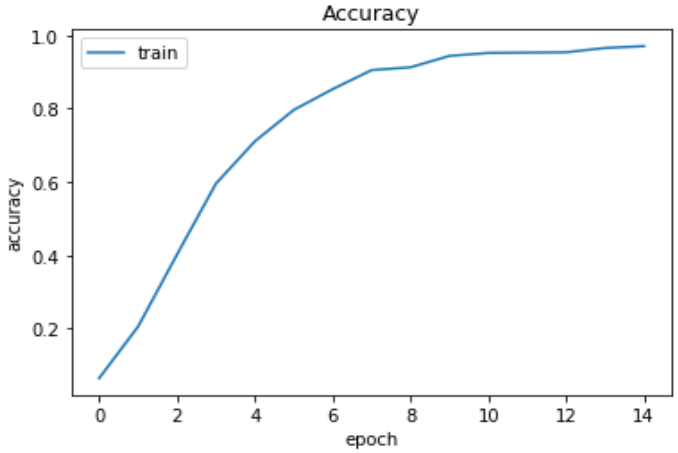


Fig. 33. *Accuracy* del modelo con *Dropout* durante el entrenamiento por varias épocas
Fuente: elaboración propia.

Todos estos modelos son construidos desde ceros y se entrenan nuevamente sus pesos desde ceros. Los resultados alcanzados en *Loss* son los de la Tabla 1 y los nombres de los modelos usaron acrónimos como CV5 (*Cross Validation* de 5), DP (*Dropout*) y LY (*Layer*) para diferenciarlos. Por su parte el *Accuracy* obtenido es la Tabla 2.

Como siguiente paso, se usó la técnica de *Transfer Learning*. Se usó el modelo inicial obtenido con los pesos obtenidos en las capas convolucionales y *MaxPool* dentro de un nuevo modelo. Esto permitió agregar más capas a su arquitectura para que solo entrenaran las adicionales.

Esta técnica es muy útil ya que permitió ahorrar tiempos en el proceso de entrenamiento. Se revisó el *Accuracy* generado por las nuevas capas o modificaciones a la arquitectura de la red CNN del modelo base. En este caso, se adicionaron capas de *Dropout*, *Flatten* y *Softmax* para ver su comportamiento.

Al hacer el proceso, se observó que el *Accuracy* de entrenamiento es de 0.9869 y el de Test es 0.9831, valores superiores al del modelo inicial. Por su parte el *Loss* de entrenamiento es 0.1019 y la de Test es 0.1316, menores al del modelo base. Las gráficas de *Accuracy* y *Loss* en los datos de entrenamiento son las (Fig. 34 y Fig. 35).

Se construyen modelos similares a los anteriores con la misma estructura, usando los pesos del modelo base sin modifican durante el entrenamiento. Para el caso de *Transfer Learning*, los modelos obtenidos tienen en su nombre el acrónimo TL (*Transfer Learning*) para indicar que se usó esta técnica. El *Loss* y *Accuracy* de estos modelos se ven en las Tablas 3 y 4 respectivamente

Finalmente, se aplicó *Fine Tuning* tomando como referencia el procedimiento realizado con *Transfer Learning*. En este proceso, se tomaron los pesos de las capas convolucionales del modelo base y se construyeron modelos similares al procedimiento anterior. La diferencia radicó en que en las capas adicionales se comenzó con un valor de *Learning Rate* de 0.001 y entrenando cada modelo durante 5 épocas para hacer un calentamiento. Luego se disminuyó el *Learning Rate* a 0.00001 y se entrenó con 15 épocas. En (Fig. 36) se observa el comportamiento del Loss al final de las 15 épocas y en (Fig. 37) el *Accuracy* del modelo base con *Dropout*.

Tabla 1. Resumen Modelos *Loss* agregando capas y *Dropout.* Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Loss Train* | *Loss Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.2348 | 0.2498 |
| modeloDPFinalCV5 | Base con *Dropout* 0.2 | 0.1396 | 0.1516 |
| modeloDP2LYFinalCV5 | Base con *Dropout* 0.2 y 2 capas más | 0.1265 | **0.1091** |
| modelo2LYFinalCV5 | Modelo Base con 2 capas más | 0.1180 | 0.1450 |
| modeloDP3LYFinalCV5 | Base con *Dropout* 0.2 y 4 capas más | 0.8524 | 0.8105 |

Tabla 2. Resumen Modelos *Accuracy* agregando capas y *Dropout.* Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Accuracy Train* | *Accuracy Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.9693 | 0.9631 |
| modeloDPFinalCV5 | Base con *Dropout* 0.2 | 0.9707 | 0.9642 |
| modeloDP2LYFinalCV5 | Base con *Dropout* 0.2 y 2 capas más | 0.9698 | **0.9758** |
| modelo2LYFinalCV5 | Modelo Base con 2 capas más | 0.9711 | 0.9558 |
| modeloDP3LYFinalCV5 | Base con *Dropout* 0.2 y 4 capas más | 0.8169 | 0.8548 |

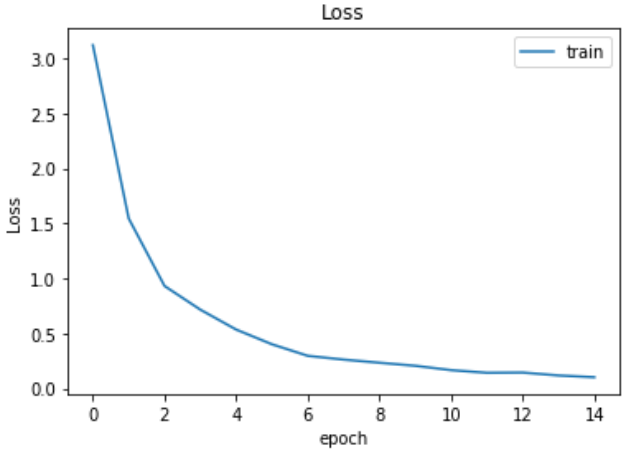


Fig. 34. *Loss* del modelo con *Dropout* durante el entrenamiento por varias épocas usando *Transfer Learning* Fuente: elaboración propia.

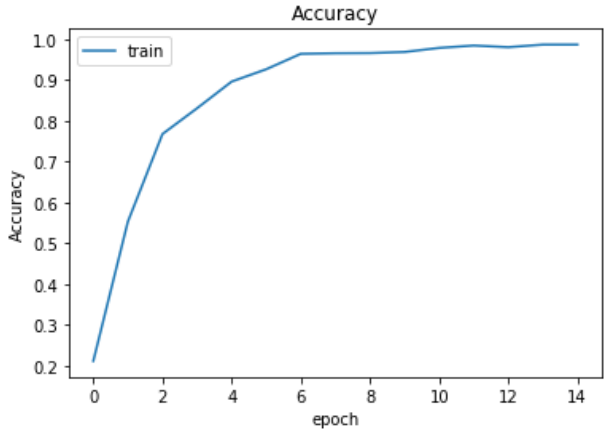


Fig. 35. *Accuracy* del modelo con *Dropout* durante el entrenamiento por varias épocas usando
Fuente: elaboración propia.

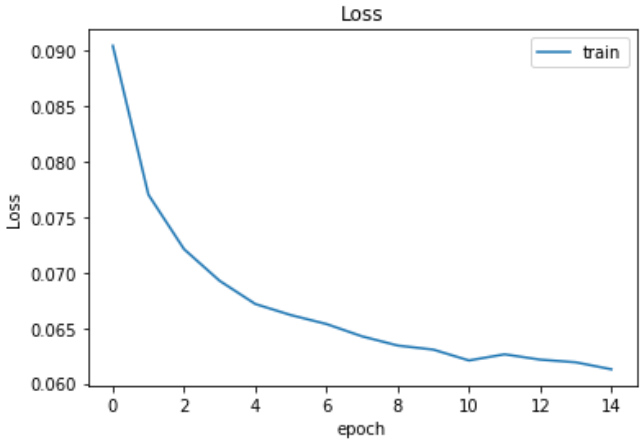


Fig. 36. *Loss* del modelo con *Dropout* durante el entrenamiento por varias épocas usando
Fuente: elaboración propia.

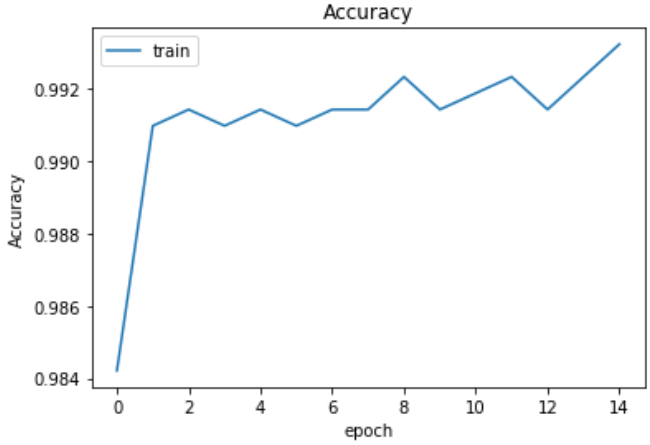


Fig. 37. *Accuracy* del modelo con *Dropout* durante el entrenamiento por varias épocas usando *Fine Tuning*.
Fuente: elaboración propia.

Tabla 1. Resumen Modelos *Loss* con *Transfer Learning*.
Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Loss Train* | *Loss Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.2348 | 0.2498 |
| modeloDPFinalCV5TL | Base con *Dropout* 0.2 | 0.1019 | **0.1316** |
| modeloDPFinalCV52LTL | Base con *Dropout* 0.2 y 2 capas más | 0.2866 | 0.2917 |
| modeloFinalCV52LTL | Modelo Base con 2 capas más | 0.1909 | 0.2137 |
| modeloDPFinalCV54LTL | Base con *Dropout* 0.2 y 4 capas más | 0.9976 | 0.9705 |

Tabla 4. Resumen Modelos *Accuracy* con *Transfer Learning*
Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Accuracy Train* | *Accuracy Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.9693 | 0.9631 |
| modeloDPFinalCV5TL | Base con *Dropout* 0.2 | 0.9869 | **0.9831** |
| modeloDPFinalCV52LTL | Base con *Dropout* 0.2 y 2 capas más | 0.9310 | 0.9190 |
| modeloFinalCV52LTL | Modelo Base con 2 capas más | 0.9666 | 0.9568 |
| modeloDPFinalCV54LTL | Base con *Dropout* 0.2 y 4 capas más | 0.7740 | 0.7991 |

En este caso, los nombres de los modelos poseen el acrónimo FT (*Fine Tuning*) y los resultados de *Loss* y *Accuracy* se ven en la Tabla 5 y Tabla 6.

Como se pudo observar, modificando el modelo base se obtuvieron diferentes modelos modificando su arquitectura, agregando más capas convolucionales o *Dropout* e, incluso, usando los pesos de las capas iniciales y entrenando las capas adicionales usando *Transfer Learning* y *Fine Tuning*. Dado que el *Accuracy* es la medida que se tomó para medir la eficiencia de los modelos y tomando en cuenta los resultados de las Tablas 2, 4 y 6 se decide tomar cuatro modelos para usar en el prototipo y evaluar con nuevas imágenes similares al del proceso de entrenamiento y test. Estos fueron el modelo base (*Accuracy* de 0.9631), el modelo base con 2 capas convolucionales más, junto con un *Dropout* y con pesos iniciales sin entrenar (*Accuracy* de 0.9758), al usar *Transfer Learning* el modelo base con los mismos pesos de las capas convolucionales y aplicando *Dropout* (*Accuracy* de 0.9831) y finalmente con *Fine Tuning* el modelo base con *Dropout* teniendo en cuenta que se modificó el *Transfer Learning* y se hizo un proceso de calentamiento de 5 épocas (*Accuracy* de 0.9831).

En la Tabla 7 se resumen los mejores modelos obtenidos de las técnicas empleadas y que se usaron para construir y evaluar el prototipo tomando como referencia el *Accuracy*.

Por su parte, en la Tabla 8 se muestran los valores de errores de estos mismos modelos usando el parámetro *Loss*.

Tabla 5. Resumen Modelos Loss con Fine Tuning. Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Loss Train* | *Loss Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.2348 | 0.2498 |
| modeloDPFinalCV5TLFT | Base con *Dropout* 0.2 | 0.0613 | **0.0791** |
| modeloDPFinalCV52LTLFT | Base con *Dropout* 0.2 y 2 capas más | 0.7268 | 0.8139 |
| modeloFinalCV52LTLFT | Modelo Base con 2 capas más | 0.7156 | 0.8346 |
| modeloDPFinalCV54LTLFT | Base con *Dropout* 0.2 y 4 capas más | 2.2961 | 2.2951 |

Tabla 6. Resumen Modelos *Accuracy* con *Fine Tuning*. Fuente: elaboración propia.

| Nombre del Modelo | Descripción | *Accuracy Train* | *Accuracy Test* |
|---|---|---|---|
| modeloFinalCV5 | Modelo Base | 0.9693 | 0.9631 |
| modeloDPFinalCV5TLFT | Base con *Dropout* 0.2 | 0.9932 | **0.9894** |
| modeloDPFinalCV52LTLFT | Base con *Dropout* 0.2 y 2 capas más | 0.8466 | 0.8128 |
| modeloFinalCV52LTLFT | Modelo Base con 2 capas más | 0.8805 | 0.8286 |
| modeloDPFinalCV54LTLFT | Base con *Dropout* 0.2 y 4 capas más | 0.3717 | 0.4100 |

Tabla 7. Resumen Modelos *Accuracy*. Fuente: elaboración propia.

| Nombre del Modelo | Técnica usada | Descripción | *Accuracy Train* | *Accuracy Test* |
|---|---|---|---|---|
| modeloFinalCV5 | Modelo Base Obtenido | Modelo Base | 0.9693 | 0.9631 |
| modeloDP2LYFinalCV5 | Modificación del modelo base agregando Capas y *Dropout* | Base con *Dropout* 0.2 y 2 capas más | 0.9698 | 0.9758 |
| modeloDPFinalCV5TL | Modificación del modelo base agregando *Dropout* y usando *Transfer Learning* | Base con *Dropout* 0.2 | 0.9869 | 0.9831 |
| modeloDPFinalCV5TLFT | Modificación del modelo base agregando *Dropout* y usando *Fine Tuning* | Base con *Dropout* 0.2 | 0.9932 | 0.9894 |

Tabla 8. Resumen Modelos *Loss*. Fuente: elaboración propia.

| Nombre del Modelo | Técnica usada | Descripción | *Accuracy Train* | *Accuracy Test* |
|---|---|---|---|---|
| modeloFinalCV5 | Modelo Base Obtenido | Modelo Base | 0.2348 | 0.2498 |
| modeloDP2LYFinalCV5 | Modificación del modelo base agregando Capas y *Dropout* | Base con *Dropout* 0.2 y 2 capas más | 0.1265 | 0.1091 |
| modeloDPFinalCV5TL | Modificación del modelo base agregando *Dropout* y usando *Transfer Learning* | Base con *Dropout* 0.2 | 0.1019 | 0.1316 |
| modeloDPFinalCV5TLFT | Modificación del modelo base agregando *Dropout* y usando *Fine Tuning* | Base con *Dropout* 0.2 | 0.0613 | 0.0791 |

## 4. RESULTADOS

### 4.1 Construcción del prototipo de software

Como siguiente paso, se construyó un prototipo de software con la ayuda del *Framework FLASK* que permita hacer uso de los modelos construidos. De esta manera, un usuario final puede interactuar con los mismos de manera sencilla y a través de una Interfaz Gráfica de Usuario (GUI). Es importante resaltar que esta actividad se hizo con el fin de que los modelos fueran más sencillos de evaluar. Primero, se construye pensando en el intérprete como usuario experto de señas y conocedor de las necesidades de la población sorda.

En (Fig. 38) se define como actor al usuario que usará el prototipo de software, en este caso la persona intérprete de lengua de señas. El actor carga una imagen que contenga a una persona haciendo un gesto

de las 22 señas establecidas. Luego ejecutara la orden para que el sistema sea capaz de mostrar la imagen cargada junto con una etiqueta que indica explícitamente cuál es de las 22 señas. Al finalizar la tarea, la persona intérprete regresa a la página de inicio para comenzar de nuevo.

El prototipo de software, al tratarse de una aplicación Web, consta de diferentes objetos que interactúan entre sí de manera dinámica, como se aprecia en (Fig. 39).

El primero de ellos es el intérprete que realiza una interacción con la GUI, donde carga la información y ve la respuesta del sistema. A su vez, esta GUI interactúa con el *Backend* o lógica del sistema que transforma la información recibida, la pasa al modelo obtenido y brinda una respuesta hacia la GUI. Dado que no se almacena esta información en una base de datos por el momento, no se cuenta con un objeto de persistencia de la información.

A continuación, se detalla la lógica del software con una descripción de las actividades secuenciales de su lógica y de forma visual en (Fig.40). El proceso de clasificación de imágenes en una de las señas establecidas es:

-Carga de la imagen de la señal de la página principal

-Carga del modelo de entrenamiento obtenido

-Almacenamiento de la imagen a matriz en escala de grises

-Almacenamiento de la imagen en formato RGB

-Transformación de la matriz de escala de grises a vector de una dimensión

-Ajuste de vector en formato de ingreso para la red neuronal

-Aplicar modelo a vector y guardar resultado en variable numérica de salida

-Convertir variable de salida en etiqueta de clase

-Enviar etiqueta de salida a página de respuesta

-Mostrar imagen almacenada en formato RGB en página de respuesta.

### 4.2 Definición de pruebas de modelos

Para validar la eficiencia del prototipo, más específicamente del modelo construido, se realizó una prueba de concepto con la persona intérprete de LSC, con el fin de que desde el área usuaria se tenga una calificación objetiva del sistema. Para ello, basados en la información de [22] los 4 pasos de la validación del prototipo fueron los siguientes:

-**Comprender el problema a resolver**: se deseaba tener un software que fuera capaz de reconocer automáticamente un determinado conjunto de señas usado por las personas sordas en su comunicación diaria.

-**Definir un *target***: las personas que harán uso de este software son aquellas que conocen este lenguaje ya que son las expertas y podrán decidir qué tan bueno es el prototipo para reconocer las expresiones o palabras realizadas.

-**Mapear el recorrido:** se desarrollarán una serie de actividades para que la persona que conoce el lenguaje de señas interactúe con el prototipo y de acuerdo con unas condiciones establecidas de su evaluación del producto.

-**Bocetar la solución:** este paso se omite, ya que el prototipo de software construido no varía por su arquitectura lógica o física establecida, sino por la calidad de los datos que se usan para construir el modelo. No obstante, de acuerdo con los resultados la retroalimentación obtenida es de gran valor para construir a futuro mejores modelos que soporten el prototipo y con ello un producto final de calidad.

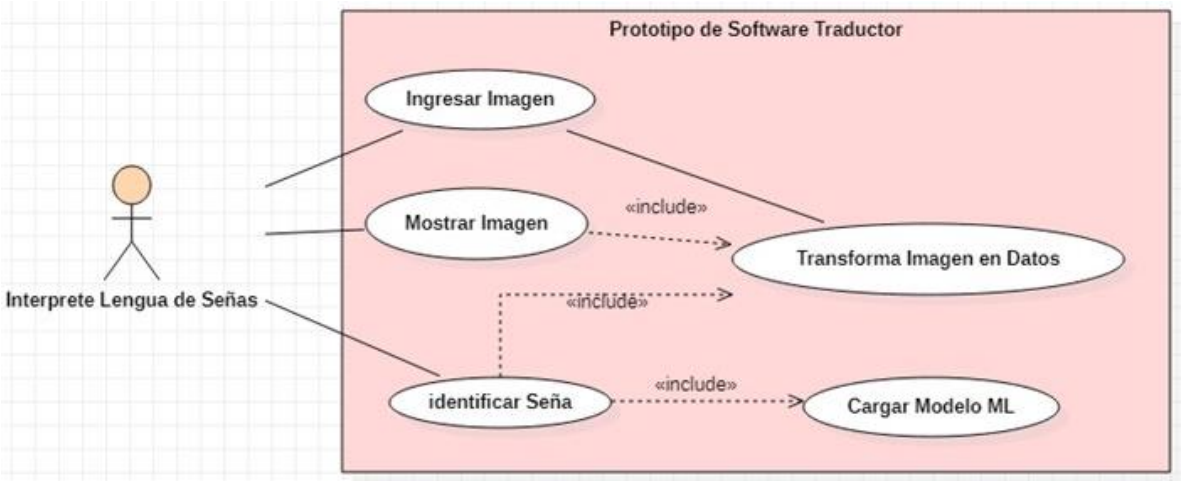


Fig. 38. Caso de uso del sistema. Fuente: elaboración propia.

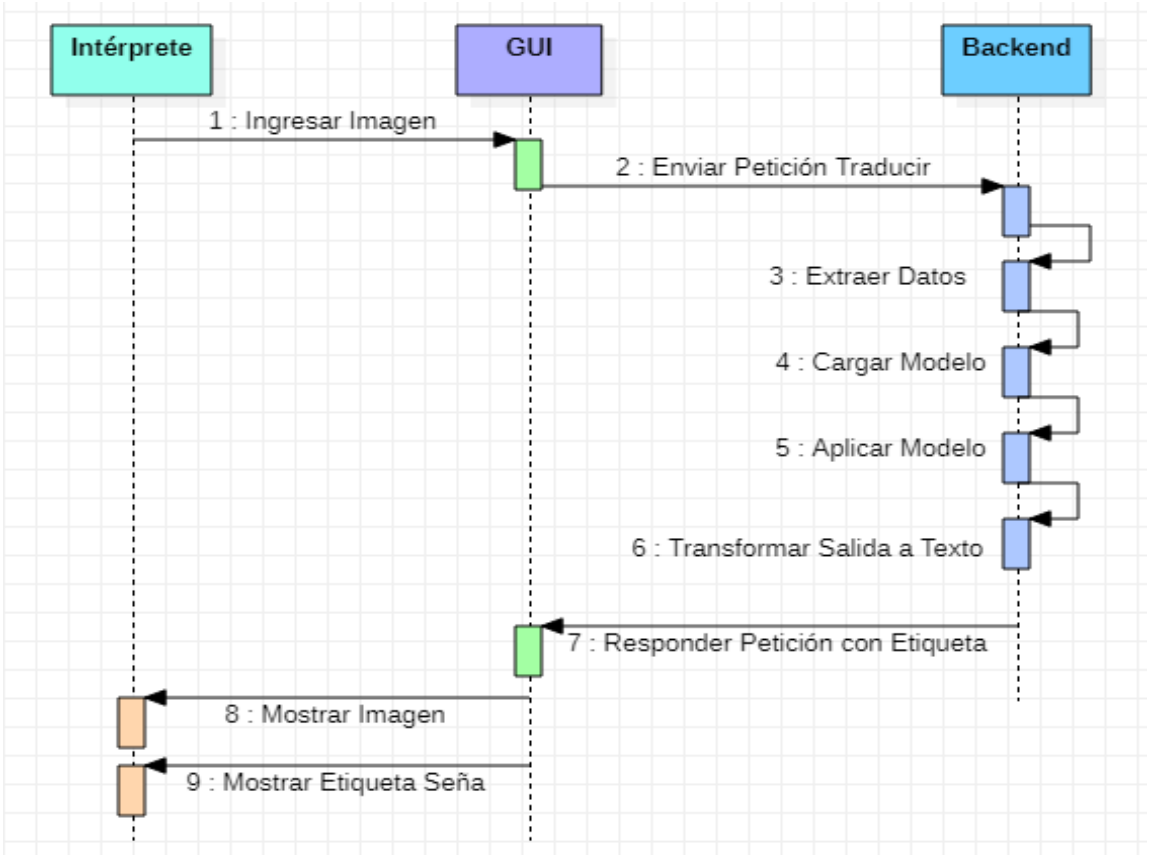


Fig. 39. Diagrama de secuencias del sistema. Fuente: elaboración propia.

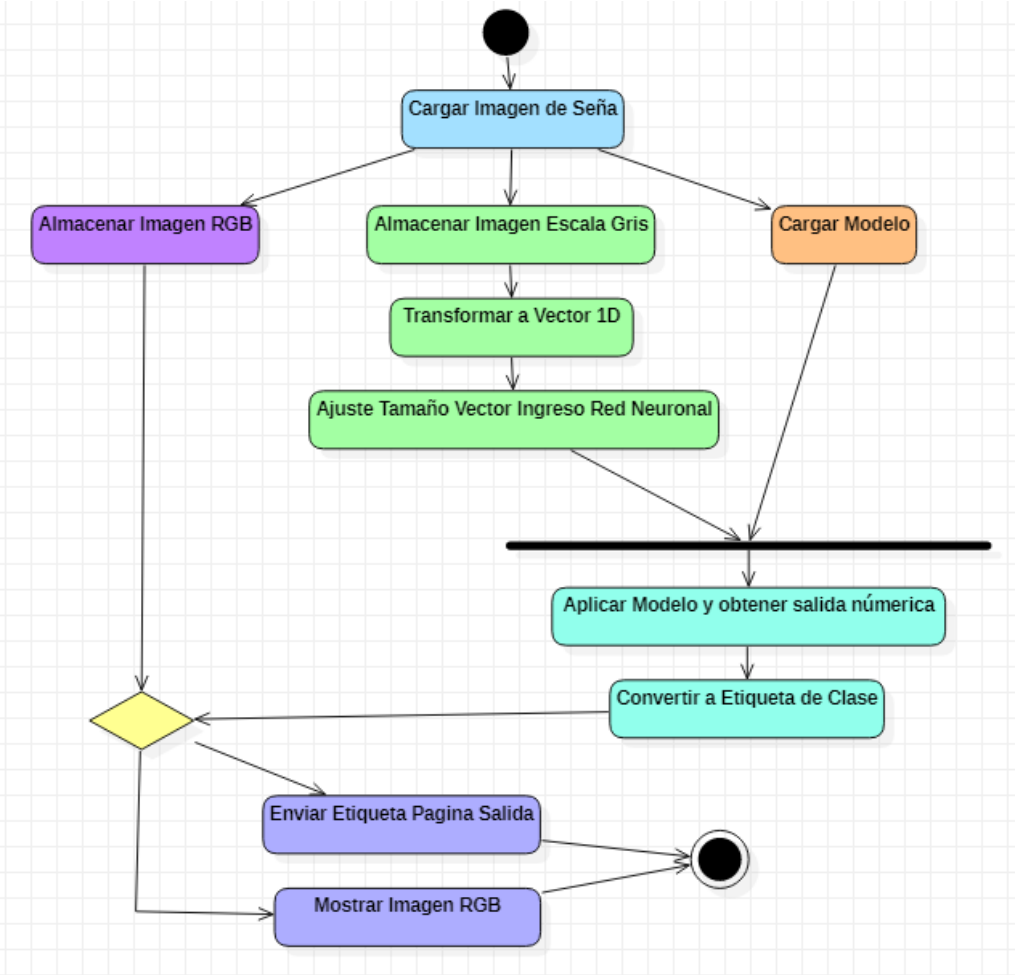


Fig. 40. Diagrama de actividades del sistema. Fuente: elaboración propia.

De igual manera, con base en lo establecido en [23], se complementó la validación haciendo que la persona que conocía el lenguaje de señas interactuara con un software de fácil accesibilidad, de acuerdo con sus recomendaciones, manteniéndolo lo más simple posible. Dado que se probaría con nuevos datos y los resultados esperados podían variar, las hipótesis que se tenían fueron que algunas señas serían acertadas en su totalidad y otras no, razón por la cual en la validación se definieron condiciones para determinar mejor el asertividad del sistema.

Finalmente, la persona intérprete realizó las pruebas de manera objetiva, y una vez se obtuvieron los resultados, se le consultó su opinión sobre la viabilidad del uso de este modelo en un sistema de producción.

Con esta finalidad, se elaboró una lista de actividades y condiciones que se debían cumplir para poder realizar la evaluación del modelo. Estas fueron:

-**Actividad 1:** tomar nuevas imágenes de las personas que ayudaron a construir el repositorio, dentro de un ambiente similar al previamente usado, haciendo de nuevo las señas.

-**Actividad 2:** incluir a la persona intérprete en estas nuevas imágenes realizando las 22 señas.

-**Actividad 3:** en un fólder especifico de Test, crear para cada seña un fólder con el nombre de cada seña y guardar en ellas las imágenes de las 6 personas haciendo estos gestos.

-**Actividad 4:** para simplificar la evaluación, tanto en gestos estáticos como dinámicos, se toma una imagen de cada persona y en secuencia que completen la seña.

-**Actividad 5:** construir en un archivo Excel una matriz de confusión con la seña a realizar y la seña reconocida por el software.

Adicionalmente, incluir el nombre de la persona que realiza el gesto. Separar por cada uno de los modelos a evaluar.

-**Condición 1:** para cada seña evaluada, dado que se trabaja con 6 personas, se aceptará que el software es capaz de reconocer una seña si y solo si el 50 % o más de imágenes evaluadas son acertadas; en otras palabras, si 3 o más personas haciendo la seña son reconocidas correctamente, se puede decir que el software es capaz de reconocer correctamente esa seña.

-**Condición 2:** si una seña es categorizada erróneamente en otra clase que se repite 3 o más veces en las diferentes personas, la clasificación del sistema es errada. Si entra en conflicto con 3 personas categorizadas como correctas se considera una clase errónea.

-**Evaluación:** se considera el porcentaje de asertividad del modelo como la relación entre el número correcto de clases clasificadas sobre el total de clases existentes.

Con base en estas actividades y criterios, se construyeron los fólderes de (Fig. 41) para que la evaluación fuera más organizada y transversal para todos los modelos, teniendo así el mismo punto de comparación de los modelos obtenidos.

Al realizar las pruebas de concepto, se observó que se tienen aciertos y clasificaciones erróneas en las diferentes imágenes. Por ejemplo, al pasar una de las imágenes con la seña de CASA se tiene un acierto correcto de acuerdo con (Fig. 42).

Sin embargo, si se evalúa una seña como escuchar, se observa en la (Fig. 43) que se clasificó una imagen específica con otra clase de manera errónea.

Por su parte, al evaluar una nueva imagen de otra persona se obtuvieron aciertos como en la clase COLOR de la (Fig. 44), mientras que con la clase TENERPOSESION se equivocó al clasificarla como YO en (Fig. 45).

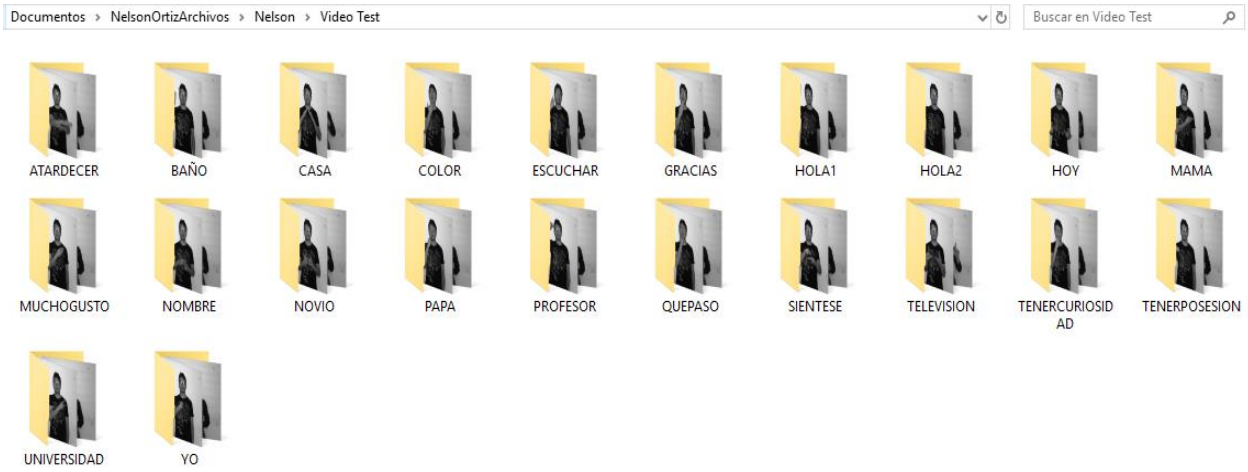


Fig. 41. Folder con señas de Test clasificado por gestos. Fuente: elaboración propia.


Fig. 42. Seña de CASA de test clasificada correctamente
Fuente: elaboración propia.


Fig. 43. Seña de ESCUCHA de test clasificada erróneamente
Fuente: elaboración propia.


Fig. 44. Seña de COLOR de persona nueva de test clasificada correctamente.
Fuente: elaboración propia.

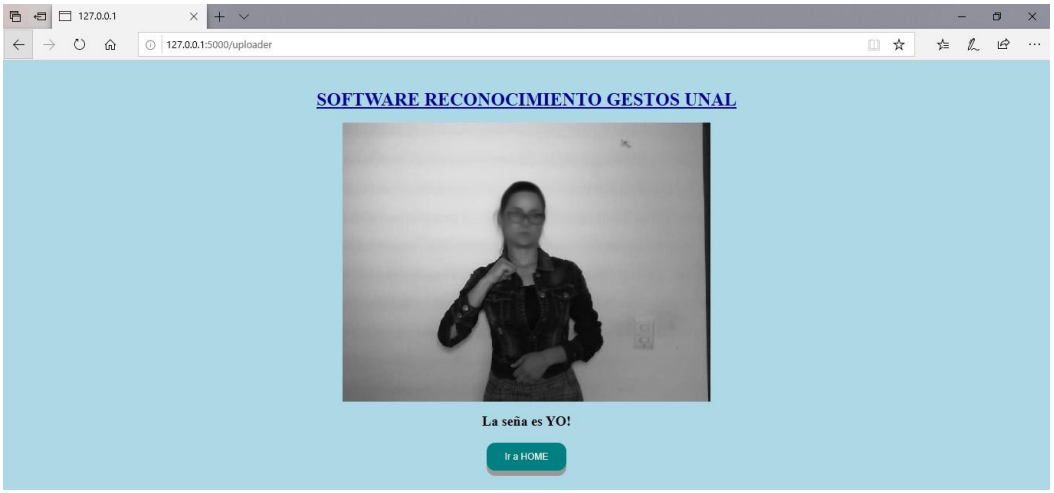


Fig. 45. Seña de TENERPOSESION de persona nueva de test clasificada erróneamente. Fuente: elaboración propia.

### 4.3 Pruebas de los modelos

Los resultados obtenidos para cada modelo se resumen a continuación:

-**Modelo Inicial:** el modelo inicial pudo reconocer 5 de 22 clases y entre ellas pudo predecir con una nueva persona. Algunas señas en donde la cantidad de dedos era la diferencia, tenían ambigüedad ("Hola1", "Hola2"). Muchas señas se confundieron por la posición diagonal de los brazos y no fue capaz de realizar diferencias en la parte de las manos. La seña "Gracias" es confundida como imagen base debido a que su pose es un patrón que se repite en las demás.

-**Modelo Inicial añadiendo *Dropout* y 2 Capas (convolucionales y *MaxPool*) más:** este modelo reconoció 15 de 22 clases. Al reconocer una nueva persona la asoció con las señales "Color" y "Yo", por ende, no fue capaz de detectarla correctamente. Los desaciertos obtenidos los relacionó con 5 clases diferentes de manera distribuida ("Color", "Profesor", "TenerPosesión" y "Universidad" y "Yo").

-**Modelo Inicial con *Dropout* y *Transfer Learning*:** este modelo reconoció 7 señas de 22 clases. No pudo reconocer nuevo rostro y lo asocia con la clase "Yo". Los demás desaciertos obtenidos se confunden de manera distribuida entre todas las clases del sistema, presentando mayor inclinación con la clase "Yo".

-**Modelo con *Dropout* y *Fine Tuning:*** este modelo tuvo como acierto solo 5 clases de 22. Para una nueva persona la clasificó entre las clases "Gracias" y "Yo". La clase "Novio" se clasifico la mitad de las imágenes tanto correctas como erróneas por lo que de acuerdo a la condición establecida no fue acertada. En general, el desacierto de imágenes se estableció entre las clases "Gracias", "Hola2" y "MuchoGusto".

Un experimento adicional realizado consistió en tomar las imágenes de prueba y borrar los rostros de las personas, obteniendo que las clases reconocidas son los mismos resultados y descartando el hecho de que los rostros puedan jugar un factor clave en la predicción.

### 4.4 Resultados de las pruebas

De los resultados obtenidos, el mejor modelo es el que tomó el modelo base y le agrego un *Dropout* con 2 capas convolucionales más. Al tenerse 15 de 22 clases clasificadas correctamente, la eficiencia del prototipo de software es de un 68 %, lo cual es mejor que un modelo de probabilidad de distribución uniforme donde la eficiencia seria 1 dividido por el número de clases (22 en este caso que es

igual a 4,5 %). Aunque este número es muy alto, no llega a un umbral aceptable como lo es igual o superior a 80 %, razón por la cual la persona intérprete no recomienda aún utilizarlo en ambientes productivos hasta que se mejore este número.

A pesar del resultado obtenido, el hecho de que el sistema sea capaz de clasificar nuevas imágenes con un porcentaje mayor al 50 % permite contestar la pregunta planteada para este proyecto. En definitiva, el aprendizaje de maquina sí puede ser usado para construir un software que sea capaz de reconocer señas de la LSC; sin embargo, su calidad dependerá de los datos que se usen para su modelo. Pese a que la cantidad de datos obtenidos con el apoyo de 5 personas dio buenos resultados, se debe contar con más imágenes de muchas más personas, factor muy crítico, ya que la disponibilidad de personas que se presten para este proyecto es compleja.

En comparación con la técnica SVM usada en el trabajo descrito en [11], se observa que el uso de CNN es mejor, ya que no se realiza tanto preprocesamiento de las imágenes, sino que, por el contrario, el sistema es capaz de detectar automáticamente en los patrones de gestos de las manos y otras partes del cuerpo la definición de cada clase o seña. Comparten en común que las calidades de las imágenes son muy importantes para la calidad del trabajo y que a mayor cantidad de clases la tarea se hace más compleja.

Por su parte, comparado con el trabajo desarrollado en [13], la cantidad de clases es menor en un tercio. Se comparte en común el construir un repositorio desde ceros, pero la cantidad de cálculos matemáticos más el uso de tecnología de sensores que se utilizaron generan que la técnica empleada en el trabajo descrito sea más engorrosa. Ambos trabajos ofrecen buenos resultados, por lo que hacer mezclas de las técnicas empleadas más los utensilios adicionales empleados permitirán realizar un mayor reconocimiento de señas de este tipo de lenguaje.

Dados los resultados obtenidos de predicción con personas conocidas, este software podría ser usado por entidades que en sus organigramas cuenten con personal de la población sorda. Incluso permitiría que las mismas entidades ya puedan ser más incluyentes con esta población en el mercado laboral sin que hayauna barrera de comunicación entre estas personas y los sistemas de información con ambientes de trabajo similares a los del INSOR.

Por ahora, la opción de SIEL es la más indicada para trabajar la interacción entre personas oyentes y sordas de público en general.

Dado que el lenguaje de señas por su propia naturaleza posee regionalismos en diferentes partes del país, uno de los mayores retos consistió en obtener un repositorio con señas representativas para el contexto colombiano. Para ello se debía contar con el apoyo de una institución gubernamental, como por ejemplo el INSOR. Lo primero que se debía establecer era a qué nivel de detalle se tomarían las muestras de las regiones colombianas, y si se hiciera por regiones naturales o si se hiciera por departamentos.

Sin importar el nivel de detalle escogido, el procedimiento a realizar era el mismo para la construcción del modelo computacional: primero construir el repositorio de imágenes con sus respetivas etiquetas de las palabras o expresiones autóctonas o específicas de la región. Posteriormente, realizar la construcción del modelo usando los mismos algoritmos de entrenamiento que se utilizaron para el prototipo y validar el mismo. Con base en la necesidad del software a implementar se puede construir un prototipo para cada región o uno central donde en su navegabilidad permita seleccionar la región donde se usará. Sin embargo, si una persona de otra región lo usara y ejecuta una seña propia de su región que no se encuentra registrada el sistema, no sería capaz de reconocerlo.

Dado lo anterior, el mejor método a usar es primero designar las regiones de interés y con base en ello seleccionar personal en cada una de ellas encargado de tomar los diferentes datos con los mismos estándares (tamaño de imágenes, luz, resolución de cámara, etc.) y posteriormente, de manera centralizada, construir un solo modelo que aplique para todo el país. Este mismo procedimiento podrá repetirse en otros países hacia futuro.

## 5. CONCLUSIONES

La construcción de un modelo CNN depende de muchos factores, dada su complejidad de manejar múltiples hiperparámetros (número de neuronas por capa, funciones de activación, numero de capas convolucionales, pesos iniciales de las capas y variación del *Learning Rate* modificados en el presente trabajo). En el entrenamiento de las imágenes creadas para el repositorio del presente trabajo, se observó que al aplicar el proceso de *Grid Search* con múltiples pliegues y *Cross Validation*, no todas las funciones de activación responden adecuadamente y que incluso la linealidad no es un factor determinante que asegure la construcción de un buen modelo, como se vio en el caso de la función *Sigmoid* que dio pobres resultados de *Accuracy* mientras que la *Relu* ofreció mejores.

En el momento de evaluar los diferentes modelos, se evidenció que el uso del factor de *Dropout* mejora la medición del *Accuracy* y *Loss* del modelo, demostrando que una CNN totalmente conectada o que una mayor cantidad de conexiones entre capas de las neuronas no aseguran un mejor aprendizaje. También fue muy evidente que, en todos los modelos construidos experimentalmente, el variar el número de capas tiene un comportamiento en común y es que tener más número de capas adicionales al modelo original no fue garantía para mejorar el *Accuracy* y *Loss* iniciales, sino que, por el contrario, lo degradaban.

Durante la prueba de concepto muchas señas llegaron a confundirse porque sus patrones son muy similares. A diferencia de estados del arte tradicionales, donde lo que se evalúa son las expresiones hechas con las manos, en el LSC el uso de otras partes del cuerpo como los codos, posiciones cruzadas de los brazos, contacto con el mentón o la frente puede llegar a ser un desafío mayor para la detección de patrones usando las técnicas de convolucionales de imágenes o filtros. Aquí juega un papel importante el ambiente construido para obtener las imágenes, ya que a pesar de que se trabajó con luz artificial, la luz solar del ambiente logra generar ruido e impactar en los resultados. Se requeriría contar con un espacio mucho más cerrado y controlado como los ofrecidos en sets de grabación para obtener una mejora considerable.

En este proyecto se usó una pequeña muestra de gestos. Con los resultados obtenidos, y cuando se piense en agregar muchas más señas se requerirán muchos recursos más, no solo a nivel computacional (procesamiento en paralelo de diversas maquinas), sino también de personal, factor que muchas veces es difícil de conseguir por el desconocimiento de la LSC, a nivel poblacional en general. Adicionalmente, la poca voluntad de las personas para conocer este lenguaje y ayudar a construir un repositorio de datos, que en Colombia no existe a la fecha, es una de las mayores barreras que podrían ser solucionadas con programas del Gobierno que permitan sacar adelante este tipo de iniciativas.

A pesar de que cada país, o incluso regiones, cuente con señas propias que impiden que sean universales para todas las personas sordas, el procedimiento aplicado en el presente trabajo es transversal y puede ser replicado en cada contexto social. De acuerdo con la semántica de la lengua, se construirá en cada región el respectivo repositorio, pero los algoritmos usados y simulaciones

ejecutadas sí serán las mismas, dando a entender que lo que varía es la fuente de los datos, cuyo insumo es uno de los mayores componentes de este tipo de tecnologías.

A futuro, se espera que este trabajo pueda ser usado para construir no solo reconocimiento de imágenes, sino que además posibilite que las palabras o etiquetas obtenidas permitan obtener una semántica con Procesamiento Natural de Lenguaje PLN con el fin de mejorar la comunicación entre la población sorda y los oyentes.

## 6. AGRADECIMIENTOS

En este espacio se da un agradecimiento especial a Andrea Bautista, interprete de lenguaje de señas colombianas, por compartir conmigo su conocimiento, no solo para la construcción del repositorio, sino su paciencia en esta labor social. A Luisa María Gómez Pinto y Roberto German Ameria por su tiempo y disposición participando en la generación de las señas, así como su aprendizaje.

## 7. REFERENCIAS

[1] L. M. Rojas-Rojas, N. Arboleda-Toro, y L. J. Pinzón-Jaime, "Caracterización de población con discapacidad visual, auditiva, de habla y motora para su vinculación a programas de pregrado a distancia de una universidad de Colombia", *Rev. Electrónica Educ.,* vol. 22, no. 1, pp. 1-28, Jan. 2018. https://doi.org/10.15359/ree.22-1.6

[2] Y. M. Cortés Bello, A. G. Barreto Muñoz, "Variación sociolingüística en la lengua de señas colombiana: observaciones sobre el vocabulario deportivo, en el marco de la planificación lingüística" *Forma y Función*, vol. 26, no. 2 pp. 149-170. Disponible en: URL

[3] Centro de relevo Colombia, Ministerio de Tecnologías de la Información y las Comunicaciones "Servicio de Interpretación en línea SIEL" (s/f). Disponible en: URL

[4] Centro de relevo Colombia, "Instructivo para la implementación de los servicios de centro de relevo," (s/f) Google Docs. [En línea]. Disponible en: URL

[5] H. Ziady, CNN Business "Google's AI system can beat doctors at detecting breast cancer," Jan. 2020. Accedido: 07-mar-2020. Disponible en: URL

[6] G. Eryiğit, et al. "Building the first comprehensive machine-readable Turkish sign language resource: methods, challenges and solutions", *Lang Resources & Evaluation.* Vol. 54. pp. 97–121, Apr. 2019. http://doi.org/10.1007/s10579-019-09465-5

[7] R.E.O. Costa, et al. "Towards an open platform for machine translation of spoken languages into sign languages". *Machine Translation,* vol. 33, pp. 315–348, Aug. 2019. https://doi.org/10.1007/s10590-019-09238-5

[8] V. Kumar Vivek, y S. Srivastava, "Toward Machine Translation Linguistic Issues of Indian Sign Language". En: Agrawal S., Devi A., Wason R., Bansal P. (eds) Speech and Language *Processing for Human-Machine Communications. Advances in Intelligent Systems and Computing*, vol. 664. Springer, Singapore. https://doi.org/10.1007/978-981-10-6626-9_14

[9] L. Quesada, G. López, y L. Guerrero, "Automatic recognition of the American sign language fingerspelling alphabet to assist people living with speech or hearing impairments", J. *Ambient Intell. Humaniz. Comput*., vol. 8, no. 4, pp. 625-635, Mar. 2017. https://doi.org/10.1007/s12652-017-0475-7

[10] J. L. Raheja, A. Mishra, y A. Chaudhary, «Indian sign language recognition using SVM», *Pattern Recognit. Image Anal*., vol. 26, no. 2, pp. 434-441, Jun. 2016. https://doi.org/10.1134/S1054661816020164

[11] D. C. García Cortes, "Reconocimiento de Gestos de Manos como Mecanismo de Interacción Humano – Computador" (Tesis de Maestría), Facultad de Ingeniería, Universidad Nacional de Colombia, 2014. Disponible en: URL

[12] P. Nakjai y T. Katanyukul, "Hand Sign Recognition for Thai Finger Spelling: An Application of Convolution Neural Network", *J. Signal Process. Syst*., vol. 91, no. 2, pp. 131-146, Apr. 2018. https://doi.org/10.1007/s11265-018-1375-6

[13] F. Ronchetti, "Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas", (Tesis Doctoral), Facultad de Informática, Universidad Nacional de la Plata, Argentina, 2017. Disponible en: URL

[14] O. Koller, H. Ney and R. Bowden, "Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016, pp. 3793-3802. http://dx.doi.org/10.1109/CVPR.2016.412

[15] M. Mustafa. "A Study on Arabic Sign Language Recognition for Differently Abled Using Advanced Machine Learning Classifiers". *Journal of Ambient Intelligence and Humanized Computing*, Mar. 2020. https://doi.org/10.1007/s12652-020-01790-w

[16] S. K. Mishra, S. Sinha, S. Sinha, y S. Bilgaiyan, "Recognition of Hand Gestures and Conversion of Voice for Betterment of Deaf and Mute People," en *International Conference on Advances in Computing and Data Sciences*, Singapure, 2019, pp. 46–57. https://doi.org/10.1007/978-981-13-9942-8_5

[17] P. M. Ferreira, J. S. Cardoso, y A. Rebelo, "On the role of multimodal learning in the recognition of sign language," *Multimed. Tools Appl.*, vol. 78, no. 8, pp. 10035–10056, Sep. 2018. https://doi.org/10.1007/s11042-018-6565-5

[18] Md. Sanzidul Islam, S. S. Sharmin Mousumi, AKM. S. Azad Rabby, y S. Akhter Hossain. "A Simple and Mighty Arrowhead Detection Technique of Bangla Sign Language Characters with CNN". En *Recent Trends in Image Processing and Pattern Recognition*, Singapore, 2019. https://doi.org/10.1007/978-981-13-9181-1_38

[19] Kaggle Inc "Sign Language MNIST Drop-In Replacement for MNIST for Hand Gesture Recognition Tasks version 1", 2019. Accedido: 15-abr-2019.Disponible en: URL

[20] Instituto Nacional para sordos, Insor educativo "Léxico de uso cotidiano" Accedido: 20-oct-2019. Disponible en: URL

[21] J. Brownlee "A Gentle Introduction to k-fold Cross-Validation" *Machine Learning Mastery Pty,* 2018. Accedido: 19-sep-2019. Disponible en: URL

[22] A. Schelstraete, "4 Principios para validar cualquier prototipo," Medium, Accedido: 11-Abr-2019. Disponible en: URL

[23] M. Timney, "Building Better Products through Prototype Validation", InVisionApp Inc. 2015. Disponible en: URL

## 8. CONTRIBUCION DE LOS AUTORES

[1] Se encargó del desarrollo y definición de la metodología, adquisición de los datos, construcción de los modelos, evaluación y prueba de los mismos al igual que el desarrollo del prototipo a utilizar. Coordinación y gestión del equipo de personas que ayudaron a construir el repositorio de imágenes correspondientes.

[2] Se encargó del apoyo en la conceptualización de la problemática, definición y evaluación de la metodología. Revisión de la escritura y correcciones técnicas correspondientes con el desarrollo de los modelos a desarrollar, así como sugerir técnicas para la mejora de imágenes y datos adquirir. Apoyo en las correcciones de metodología, presentación y estilo en el manuscrito.