UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS E INDUSTRIAL

Diseño y Construcción de Prototipo de Software para Reconocer Lenguaje de Señas de Personas con Discapacidad Auditiva

Nelson Mauricio Ortiz Farfán

Universidad Nacional de Colombia

Facultad de Ingeniería

Maestría en Ingeniería

Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2020

Diseño y Construcción de Prototipo de Software para Reconocer Lenguaje de Señas de Personas con Discapacidad Auditiva

Nelson Mauricio Ortiz Farfán

Trabajo de grado presentado como requisito parcial para optar al título de:

Magíster en Ingeniería - Ingeniería de Sistemas y Computación

Director:

Ing. Jorge Eliecer Camargo Mendoza, PhD

Universidad Nacional de Colombia

Facultad de Ingeniería

Maestría en Ingeniería - Ingeniería de Sistemas y Computación

Departamento de Ingeniería de Sistemas e Industrial

Bogotá, Colombia

2020

A Dios por darme fortaleza en los momentos más difíciles de mi vida y por supuesto a mi familia que han sido un apoyo fundamental durante estos últimos años.

Agradecimientos

Al ingeniero ***Jorge Eliecer Camargo Mendoza*** por su guía y consejos para llevar a cabo este proyecto de la mejor manera y con calidad. De igual manera al ingeniero ***Fabio Augusto González Osorio*** por sus comentarios de mejora al modelo del presente proyecto durante el diplomado Deep Learning: Introducción al Aprendizaje Profundo de la Universidad Nacional.

Un agradecimiento especial a ***Andrea Bautista,*** interprete de lengua 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 el aprendizaje de las mismas. Finalmente, al ingeniero y gran amigo ***Juan Miguel David Becerra Tovar*** por sus comentarios y revisión de este documento para realizar las mejoras correspondientes.

Resumen

En el presente documento se estudia la posibilidad de implementar una herramienta tecnológica de software poco tradicional que permita traducir algunas de las señas usadas por la población sorda colombiana en sus labores cotidianas. Para ello, se usa técnicas del Aprendizaje de Máquina con el fin de reconocer imágenes de manera simple sin tener que realizar un tratamiento de señales tan profundo como se realizaba años atrás. Esto se logra a través de la información que brinden los datos o imágenes fuentes usadas para entrenar este tipo de modelos computacionales. Ante la ausencia de un repositorio abierto de esta índole, se comenzó construyendo uno usando una cantidad limitada de señas. Una vez realizado y obtenido el modelo, se construye un prototipo de software para que una persona experta en Lengua de Señas Colombiana (LSC) pueda dar un veredicto de su aproximación a las tareas realizadas por estos profesionales.

Palabras Clave: Población sorda, Aprendizaje de Máquina, Modelos Computacionales, Lengua de Señas Colombiana (LSC).

Abstract

This document shows the possibility to implement a software to be able to recognize signs from deaf people in Colombia. The process for doing that is to use techniques from Machine Learning able to work with images for building a model which can be used in a software for this complex task. First, it is mandatory to create a repository with a few amount of these signs because today it does not exist. After the model is created with these data, an expert person in deaf signs used in Colombia will give a concept about how good the software is and if it is similar to the task that this kind of professional does.

Keywords: Deaf People, Machine Learning, Colombian Deaf Signs, Model.

Tabla de Contenido

1. Introducción ____ *11*

1.1 Identificación del Problema ____ 11

1.2 Pregunta de Investigación ____ 13

1.3 Objetivo General ____ 14

1.4 Objetivos Específicos ____ 14

1.5 Contribución ____ 15

1.5.1 Tecnología al servicio de la población sorda colombiana ____ 15

1.5.2 Repositorio de imágenes de lengua de señas ____ 15

1.5.3 Modelos computacionales de Machine Learning ____ 15

1.5.4 Artículo de modelo computacional en Machine Learning ____ 16

1.6 Organización del documento ____ 16

2. Estado del Arte ____ *17*

3. Modelo Computacional para Reconocimiento de Señas ____ *30*

3.1. Construcción de Repositorio de Lengua de Señas Colombiano ____ 30

3.2. Procesamiento de Datos ____ 41

3.3. Construcción del Modelo ____ 45

3.4. Evaluación del Modelo ____ 53

4. Prototipo de Software ____ *64*

4.1 Diseño de Prototipo ____ 64

4.1.1 Caso de Uso ____ 64

4.1.2 Diagrama de Secuencias ____ 65

4.1.3 Diagrama de Actividades ____ 66

4.2 Codificación del Prototipo ____ 67

4.3 Prueba de Concepto y Análisis de Resultados ____ 70

5. Conclusiones ____ *77*

6. Bibliografía ____ *80*

Anexo A ____ *82*

Índice de figuras

Fig. 1 - Servicio de Intérprete en Línea SIEL. [3] 12
Fig. 2 - : Estructura general de un modelo de aprendizaje de maquina supervisado. [Autor] 17
Fig. 3 - : Estructura general de una Matriz de Confusión. [6] 18
Fig. 4 -: Registro de señas con sensores y cámaras. [8] 20
Fig. 5 - : Diagrama de bloques de sistema de reconocimiento de señas en India. [9] 20
Fig. 6 - :Entrenamiento de imágenes de lengua de señas sorda en India. [9] 21
Fig. 7 - : Sistema de interpretación de gestos basados en la visión. [10] 21
Fig. 8 - : Sistema de reconocimiento de señas usando Redes Neuronales. [11] 22
Fig. 9 - : Manejo de gestos dinámicos con las manos para entrenar en una CNN. [13] 23
Fig. 10 - : Matriz de Confusión de reconocimiento de gestos estáticos y dinámicos usando la CNN. [13] 23
Fig. 11 - : Lenguaje de señas del alfabeto árabe. [14] 24
Fig. 12 - : Arquitectura del sistema de clasificación de lengua de señas en India usando CNN. [15] 25
Fig. 13 - : Extracción de la seña Victoria de una imagen pre procesada. [15] 25
Fig. 14 - : Arquitectura sistema multimodal para reconocer gestos de la mano usando CNN y SVM. [16] 26
Fig. 15 - : 36 clases de números representados en lenguaje de señas en Bangladesh. [17] 27
Fig. 16 - : Arquitectura de la CNN de clasificación de caracteres en Bangladesh. [17] 27
Fig. 17 - : Esquema general de una CNN para clasificar imágenes. [18] 28
Fig. 18 - : Reto KAGGLE para detectar alfabeto de Lenguaje de Señas Universal. [19] 29
Fig. 19 - : Red CNN con Dropout para 24 clases. [Autor] 29
Fig. 20 - : Página Web Diccionario de Gestos de INSOR. [20] 30
Fig. 21 - : Gesto dinámico usado en Bogotá para indicar el sistema de transporte Transmilenio. [20] 31
Fig. 22 - : Gesto dinámico de traslación lineal para indicar Televisión. [20] 32
Fig. 23 - : Proceso para grabar videos con OpenCV. [Autor] 33
Fig. 24 - : Proceso para obtener Frames de un video. [Autor] 34
Fig. 25 - : Seña estática de la palabra YO. [Autor] 35
Fig. 26 - : Seña estática de la palabra NOMBRE. [Autor] 35
Fig. 27 - : Seña dinámica de HOLA1 con 2 dedos. [Autor] 35
Fig. 28 - : Seña dinámica de HOLA2 con 4 dedos. [Autor] 35
Fig. 29 - : Seña estática de expresión ¿QUÉ PASO? [Autor] 36
Fig. 30 - : Seña dinámica palabra GRACIAS. [Autor] 36
Fig. 31 - : Seña dinámica expresión CON MUCHO GUSTO. [Autor] 36
Fig. 32 - : Seña dinámica expresión SIÉNTESE. [Autor] 36
Fig. 33 - : Seña dinámica palabra BAÑO. [Autor] 37
Fig. 34 - : Seña estática de palabra HOY. [Autor] 37
Fig. 35 - : Seña estática de palabra CASA. [Autor] 37
Fig. 36 - : Seña dinámica palabra ESCUCHAR. [Autor] 37
Fig. 37 - : Seña dinámica palabra ATARDECER. [Autor] 38

Fig. 38 - : Seña dinámica palabra COLOR. [Autor] 38
Fig. 39 - : Seña dinámica palabra PROFESOR. [Autor] 38
Fig. 40 - : Seña estática de palabra UNIVERSIDAD. [Autor] 38
Fig. 41 - : Seña dinámica palabra TELEVISOR. [Autor] 39
Fig. 42 - : Seña dinámica palabra NOVIO. [Autor] 39
Fig. 43 - : Seña dinámica palabra PAPÁ. [Autor] 39
Fig. 44 - : Seña dinámica palabra MAMÁ. [Autor] 39
Fig. 45 - : Seña estática expresión TENGO CURIOSIDAD. [Autor] 40
Fig. 46 - : Seña estática expresión TENER POSESIÓN. [Autor] 40
Fig. 47 - : Imágenes secuenciales obtenidas de los videos grabados de una persona haciendo las señas. [Autor] 42
Fig. 48 - : Proceso de conversión de matriz 2x2 a vector de 1 dimensión usado en imágenes. [Autor] .. 43
Fig. 49 - : Validación en Pandas de archivo de etiquetas asociadas con el nombre de archivo de imagen. [Autor] 43
Fig. 50 - : Distribución uniforme de todas las señas del repositorio. [Autor] 44
Fig. 51 - : Cross Validation con 5 pliegues. Sobre los datos de Entrenamiento se hacen nuevas divisiones y se itera. [Autor] 47
Fig. 52 - : Mapa de calor Grid Search Funciones de Activación vs Numero de Neuronas por Capas. [Autor] 48
Fig. 53 - : Loss del modelo durante el entrenamiento por varias épocas. [Autor] 50
Fig. 54 - : Accuracy del modelo durante el entrenamiento por varias épocas. [Autor] 50
Fig. 55 - : Valores de Kernel de la primera capa convolucional. [Autor] 52
Fig. 56 - : Imagen de prueba antes de aplicar Kernel de primera capa convolucional. [Autor] 52
Fig. 57 - : Imagen de prueba después de aplicar Kernel de primera capa convolucional. [Autor] 52
Fig. 58 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas. [Autor] 54
Fig. 59 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas. [Autor] 54
Fig. 60 - : Valores de Kernel de la primera capa convolucional de modelo con Dropout. [Autor] 55
Fig. 61 - : Imagen de prueba después de aplicar Kernel de primera capa convolucional de modelo con Dropout. [Autor] 56
Fig. 62 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas usando Transfer Learning. [Autor] 59
Fig. 63 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas usando Transfer Learning. [Autor] 60
Fig. 64 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas usando Fine Tuning. [Autor] 62
Fig. 65 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas usando Fine Tuning. [Autor] 62
Fig. 66 - : Caso de Uso del sistema. [Autor] 65
Fig. 67 - : Diagrama de Secuencias del sistema. [Autor] 66
Fig. 68 - : Diagrama de actividades del sistema. [Autor] 67
Fig. 69 - : Estructura folder prototipo usando Framework Flask. [Autor] 68
Fig. 70 - : Página Home del sistema. [Autor] 69
Fig. 71 - : Página Respuesta etiqueta de seña. [Autor] 69

Fig. 72 - : Folder con señas de Test clasificado por gestos. [Autor] 71
Fig. 73 - : Documento Excel de la Matriz de Confusión de las señas de Test. [Autor] 72
Fig. 74 - : Seña de CASA de test clasificada correctamente. [Autor] 73
Fig. 75 - : Seña de ESCUCHA de test clasificada erróneamente. [Autor] 73
Fig. 76 - : Seña de COLOR de persona nueva de test clasificada correctamente. [Autor] 74
Fig. 77 - : Seña de TENERPOSESION de persona nueva de test clasificada erróneamente. [Autor] 74

Índice de tablas

Tabla 1 - Resumen Imágenes por Clase [Autor] 41
Tabla 2 - Resumen de Parámetros de Modelo Obtenido [Autor] 49
Tabla 3 - Resumen de Medidas por Clase sobre los Datos de Test o Prueba [Autor] 51
Tabla 4 - Resumen de Parámetros de Modelo obtenido con Dropout [Autor] 53
Tabla 5 - Resumen de Medidas por Clase sobre los Datos de Prueba en Modelo con Dropout [Autor] 55
Tabla 6 - Resumen de Modelo con 6 Capas Convolucionales con Dropout [Autor] 57
Tabla 7 - Resumen Modelos Loss [Autor] 57
Tabla 8 - Resumen Modelos Accuracy [Autor] 58
Tabla 9 - Nuevo Modelo con los Pesos Cargados del Modelo Base, Técnica de Transfer Learning [Autor] 58
Tabla 10 - Nuevo Modelo con Transfer Learning Entrenando solo las Capas Adicionales [Autor] 59
Tabla 11 - Resumen de Modelo con 6 Capas Convolucionales con Dropout usando Transfer Learning [Autor] 60
Tabla 12 - Resumen Modelos Loss con Transfer Learning [Autor] 61
Tabla 13 - Resumen Modelos Accuracy con Transfer Learning [Autor] 61
Tabla 14 - Resumen Modelos Loss con Fine Tuning [Autor] 62
Tabla 15 - Resumen Modelos Accuracy con Fine Tuning [Autor] 63

1. Introducción

En este capítulo se explica cuál es el problema que dio origen a la pregunta de investigación a solucionar. Para ello se indica el objetivo general a alcanzar por medio de objetivos específicos que permitieron lograrlo y la contribución que el presente trabajo realizó. Finalmente se explica brevemente el contenido de todos los capítulos del documento presente.

1.1 Identificación del Problema

En la sociedad actual colombiana, 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 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 acceda a instituciones educativas universitarias de forma virtual como se indica en [1] basándose en las leyes que se mencionan en el artículo. Pero no solo la educación ha sido abierta a esta inclusión, también en el deporte se ha intentado establecer un lenguaje de señas específicos para esta actividad como se menciona en [2].

Sin embargo, a pesar de los avances obtenidos hasta el momento, mucha población oyente colombiana desconoce la Lengua de Señas Colombiana (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 Min TIC ha desarrollado en los últimos años el

Servicio de Intérprete En Línea SIEL el cual se accede a través del portal web Centro de Relevo como se observa en la (Fig. 1) [3]:

Fig. 1 - Servicio de Intérprete en Línea SIEL. [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. Adicional no 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 y teniendo que establecer una nueva sesión que retrasa el proceso de comunicación.

1.2 Pregunta de Investigación

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 de la lengua de señas con las tecnologías tradicionales conlleva a pensar que otro tipo de solución informática se pueda implementar garantizando fiabilidad en su proceso.

Por esta razón, la pregunta específica que se planteó contestar para este contexto fue:

• ¿Puede implementarse un software que sea capaz de reconocer algunos gestos de la lengua de señas colombiana de manera automática y que sea capaz de detectarlas con el menor número de desaciertos?

Dado los avances tecnológicos de las últimas décadas, nuevas investigaciones en reconocimiento de imágenes y videos y el uso de algoritmos de Inteligencia Artificial 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.

1.3 Objetivo General

Por esta razón el objetivo general del presente proyecto consistió en:

- ✓ Diseñar e implementar un prototipo de software que traduzca automáticamente en texto el lenguaje de señas empleado por la población sorda de Colombia usando técnicas de aprendizaje de máquina.

1.4 Objetivos Específicos

Para ello, este objetivo general se dividió en los siguientes objetivos específicos con el fin de desarrollar esta compleja labor:

- ✓ Analizar diferentes técnicas de aprendizaje de máquina para seleccionar la más apropiada para reconocer gestos del lenguaje de señas.
- ✓ Construir un repositorio de imágenes de lenguaje de señas útiles para entrenar el modelo de aprendizaje de máquina a desarrollar.
- ✓ Implementar un prototipo de software usando el modelo de entrenamiento construido previamente.
- ✓ Realizar una prueba de concepto del prototipo de software con el apoyo de una persona interprete con conocimientos en lenguaje de señas colombiana.

1.5 Contribución

1.5.1 Tecnología al servicio de la población sorda colombiana

El prototipo de software desarrollado capaz de reconocer señas empleadas por las personas sordas para comunicarse en su diario vivir, demuestra que las Tecnologías de la Información (TI) son capaces de resolver no solo problemas que lidian las organizaciones sino además problemáticas sociales. Este prototipo se le puede considerar como un punto de partida de sistemas de reconocimiento más complejos que a futuro permita a la población sorda comunicarse con las entidades y demás personas sin ningún tipo de barrera existente por el desconocimiento de la lengua que manejan.

1.5.2 Repositorio de imágenes de lengua de señas

Se brinda un repositorio de lengua de señas colombianos disponible en el repositorio en línea indicado en el apéndice A con el fin de que a futuro pueda ser usado por otras personas para construir modelos computacionales más precisos y puedan entrenar sistemas inteligentes para problemáticas similares. A diferencia de otros trabajos ya implementados no se tuvo en cuenta solo las manos que simbolizan los gestos del alfabeto de la lengua de señas, sino que además se incluyó otras partes del cuerpo usadas para la comunicación, lo cual hace que sea algo totalmente diferente.

1.5.3 Modelos computacionales de Machine Learning

Los modelos computacionales al igual que el código fuente utilizado para construirlos se comparten en el repositorio indicado en el apéndice A con el fin de que personas interesadas

puedan hacer uso de los mismos para futuras experimentaciones o para conocer el proceso realizado y replicarlo en situaciones parecidas ya sea de tipo académico o aplicado a la industria.

1.5.4 Artículo de modelo computacional en Machine Learning

Como resultado de esta tesis, se generó un artículo científico que se sometió a la revista *TecnoLógicas*:

Ortiz, N., & Camargo, J. (2020). Modelo Computacional Para Reconocimiento De Lenguaje De Señas En Un Contexto Colombiano. *TecnoLógicas*. (En revisión)

1.6 Organización del documento

El presente documento se ha dividido en las siguientes secciones:

Capítulo 2: Estado del Arte de los modelos usados por aprendizaje de máquina para reconocimiento de imágenes y así elegir uno de ellos para implementar. De igual manera determinar las variables que permitan evaluar los resultados del modelo.

Capítulo 3: Construcción del repositorio de señas del LSC, procesamiento de las mismas, implementación de varios modelos usando Redes Neuronales Convolucionales y evaluación de los mismos para determinar los más óptimos a probar.

Capítulo 4: Diseño del prototipo de software, codificación y pruebas del mismo con los modelos óptimos obtenidos para dar una prueba de concepto final por parte de una intérprete.

Capítulo 5: Conclusiones obtenidas del presente trabajo y mejoras a futuro.

2. Estado del Arte

El aprendizaje de maquina supervisado es un nuevo paradigma computacional donde los computadores aprenden a partir de una experiencia con el fin de predecir o clasificar una nueva entrada. En (Fig. 2) el sistema toma datos de entrada conocidos etiquetados con su respetiva salida para construir un modelo capaz de predecir correctamente la entrada de datos similares nunca antes vistos [5]:

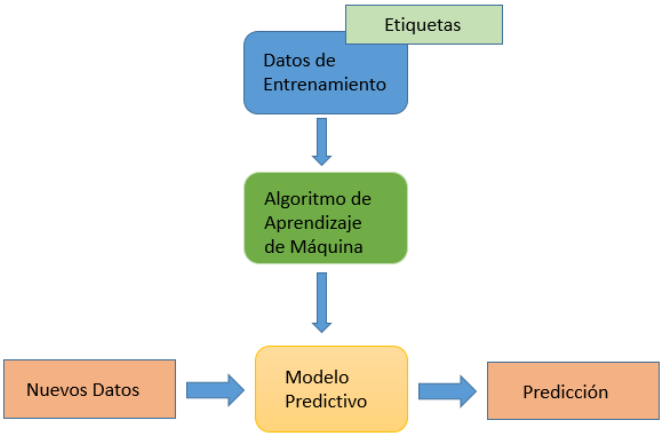


Fig. 2 - : Estructura general de un modelo de aprendizaje de maquina supervisado. [Autor]

El desempeño de estos modelos se mide con varias métricas dependiendo del tipo de salida a obtener. En el caso de los gestos de la lengua de personas sordas, será clasificar una nueva seña antes no vista por el sistema en una de las diferentes clases existentes, razón por la cual se usa una Matriz de Confusión. La matriz de (Fig. 3) define una grilla en la que se compara los valores reales con los valores predichos por el modelo y detectando así cuantos aciertos se obtuvieron realmente [6]:

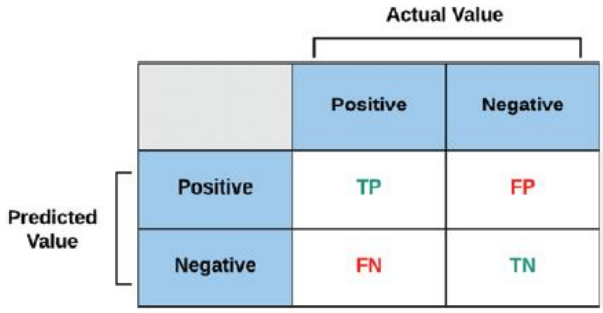


Fig. 3 - : Estructura general de una Matriz de Confusión. [6]

• **TP (*True Positive*):** Positivos Verdaderos (en español) es el número de muestras predichas como una clase a ser verdaderas cuando su verdadero valor de clase es positivo. En otras palabras, es un acierto.

• **FP (*False Positive*):** Falsos Positivos (en español) es el número de muestras predichas como una clase a ser verdaderas cuando su verdadero valor de clase es negativo. Es un desacierto.

• **TN (*True Negative*):** Negativos Positivos (en español) es el número de muestras predichas como una clase a ser falsas cuando su verdadero valor de clase es negativo. Es un acierto.

• **FN (*False Negative*):** Falsos Negativos (en español) es el número de muestras predichas como una clase a ser falsa cuando su verdadero valor de clase es positivo. Es un desacierto.

En adición a la matriz, se tienen 3 medidas más para evaluar el modelo [6]. Estas son:

- ***Accuracy:*** es la fracción de predicciones correctas hechas por el modelo. Se puede entender como la relación entre la suma de TP y TN divididos por el total del conjunto de datos:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- ***Precision:*** Son los valores predichos positivamente. Consiste en la relación de los TP sobre la suma de TP y FP. Es el número de aciertos sobre el total de clasificaciones hechas por el modelo como positiva:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- ***Recall:*** Consiste en la relación de los TP sobre la suma de TP y FN. Es el número de aciertos sobre el total de clasificaciones acertadas hechas por el modelo:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  Otra medida usada es el *F1-Score* que es definido como el promedio pesado de Precisión y Recall [7]:

$$\text{F1} = 2\frac{Precision * Recall}{Precision + Recall}$$

De lo anterior, el desempeño a medir del modelo es el *Accuracy*. El repositorio de datos que se construirá tendrá una distribución balanceada en todas sus clases; en casos donde las clases en el conjunto de datos de entrenamiento no este balanceado, este tipo de métrica no se recomienda.

De acuerdo a la información publicada en 2017 en [8] la lengua de señas no es un estándar universal y cada país puede tener sus propias definiciones e incluso no siempre existe una relación entre las señas y el lenguaje hablado. El sistema de reconocimiento de gestos de manera automática que se menciona en [8] 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 Maquinas de Soporte Vectorial (SVM en inglés) al ingresar una nueva seña se clasifica en las 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 a (Fig. 4):

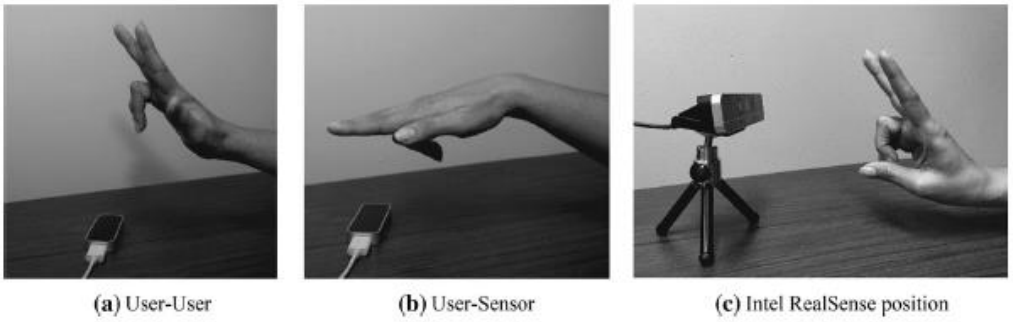


Fig. 4 -: Registro de señas con sensores y cámaras. [8]

Un sistema similar se propone en India en 2016, [9] 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 al diagrama de bloques de (Fig. 5):

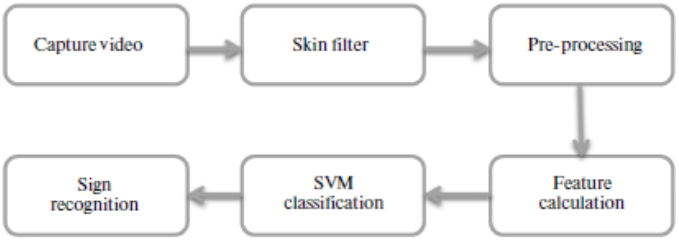


Fig. 5 - : Diagrama de bloques de sistema de reconocimiento de señas en India. [9]

En [9] el entrenamiento de cada seña es realizado y la modificación de cada imagen para obtener las características específicas de acuerdo a (Fig. 6):

Fig. 6 - :Entrenamiento de imágenes de lengua de señas sorda en India. [9]

En 2014 en [10] 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. 7):

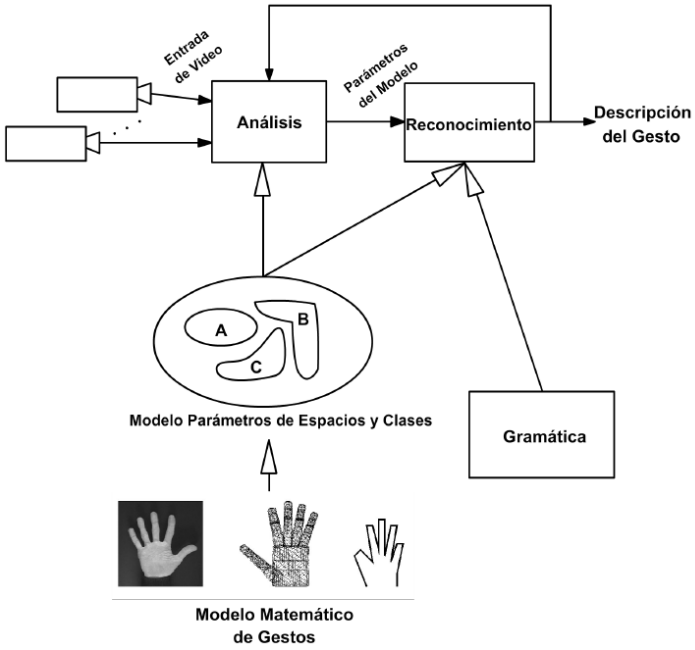


Fig. 7 - : Sistema de interpretación de gestos basados en la visión. [10]

Los resultados y conclusiones obtenidos de [10] determinaron que esta técnica no es adecuada dado a 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 [11] para reconocer señas de la lengua Thai. 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. 8):

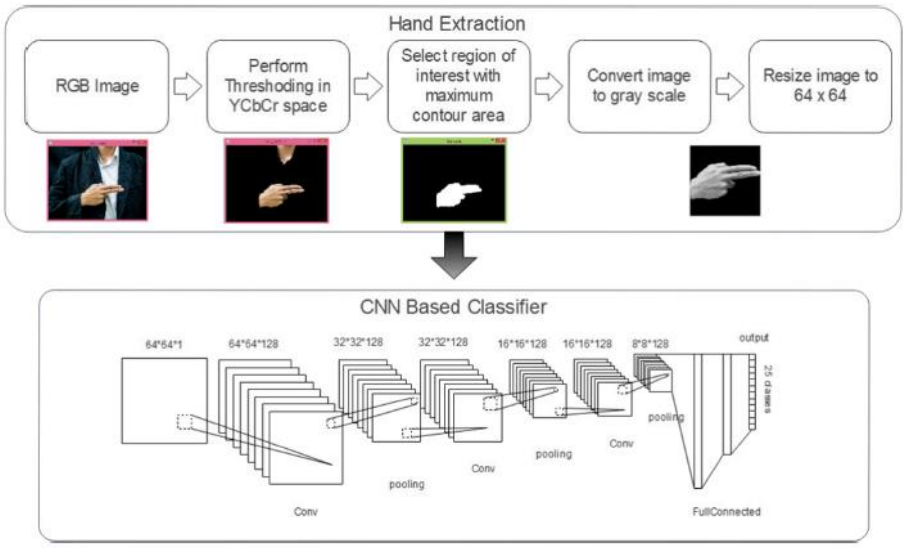


Fig. 8 - : Sistema de reconocimiento de señas usando Redes Neuronales. [11]

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

En el lenguaje de señas, es necesario tener en cuenta la existencia de gestos dinámicos y estáticos. 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 [12].

En el momento de revisar la literatura, se encuentra que en [13] la forma de trabajar los gestos dinámicos es grabar un video del mismo y luego obtener varios *Frames* o capturas de imágenes secuenciales del mismo, 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 realiza limpieza de los mismos para tener los datos de entrenamiento que se usaran en una Red Neuronal Convolucional (CNN *Convolutional Neural Network* en inglés) como se ve en (Fig. 9):

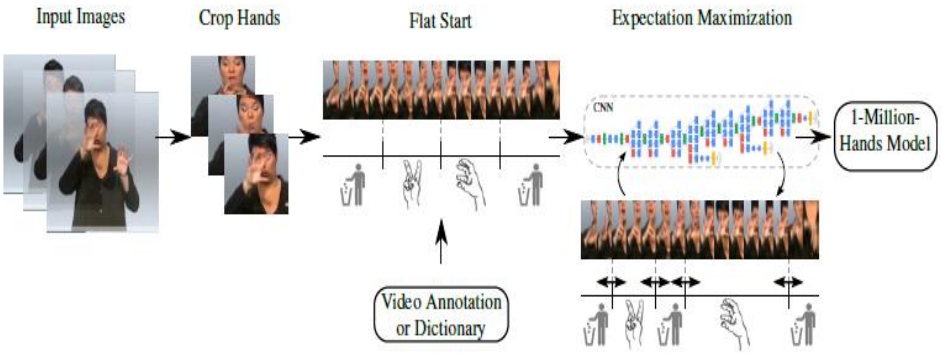


Fig. 9 - : Manejo de gestos dinámicos con las manos para entrenar en una CNN. [13]

Los resultados del experimento muestran en (Fig. 10) una matriz de confusión con porcentajes de aciertos muy altos para la mayoría de señas [13]:

96.5	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0
2.0	90.1	0.8	0.1	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	2.7	94.3	2.3	0.0	0.0	0.0	0.0	0.0	1.9	0.0	0.0	0.0
0.0	0.0	0.0	49.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	18.8	41.7	6.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0
0.0	0.0	0.0	4.1	9.4	81.9	0.0	0.0	0.0	0.5	0.0	0.0	0.0
0.0	0.0	0.0	1.7	0.0	0.0	47.6	2.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.6	0.0	0.0	0.0	95.5	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	3.9	1.3	0.0	0.0	0.0
0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	64.9	0.0	0.0	0.0
0.0	0.0	0.0	3.5	38.1	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.0	0.0	0.2	0.0	100.0	0.0
0.4	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	64.3

Fig. 100 - : Matriz de Confusión de reconocimiento de gestos estáticos y dinámicos usando la CNN. [13]

Recientemente en [14] se realizó un estudio de las diferentes técnicas para clasificar imágenes que se puedan aplicar a un determinado concepto árabe, utilizando técnicas de *Deep Learning* (Fig. 11).

Fig. 111 - : Lenguaje de señas del alfabeto árabe. [14]

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

En [15] 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. 12).

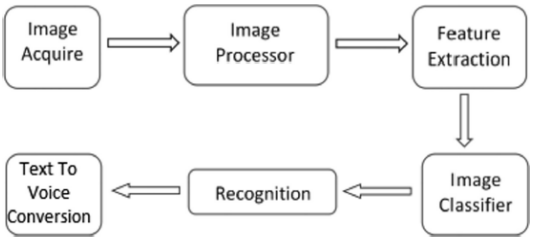


Fig. 122 - : Arquitectura del sistema de clasificación de lengua de señas en India usando CNN. [15]

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%, el cual es bueno teniendo en cuenta además que el sistema permite obtener texto de voces introducidas al sistema (Fig. 13).

Fig. 133 - : Extracción de la seña Victoria de una imagen pre procesada. [15]

Un trabajo similar se tiene para reconocer gestos de manos haciendo uso de aprendizaje multimodal [16]. 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 CNNs y finalmente los resultados se unen para ser la entrada de un clasificador SVM para detectar los gestos de mano presentes (Fig. 14).

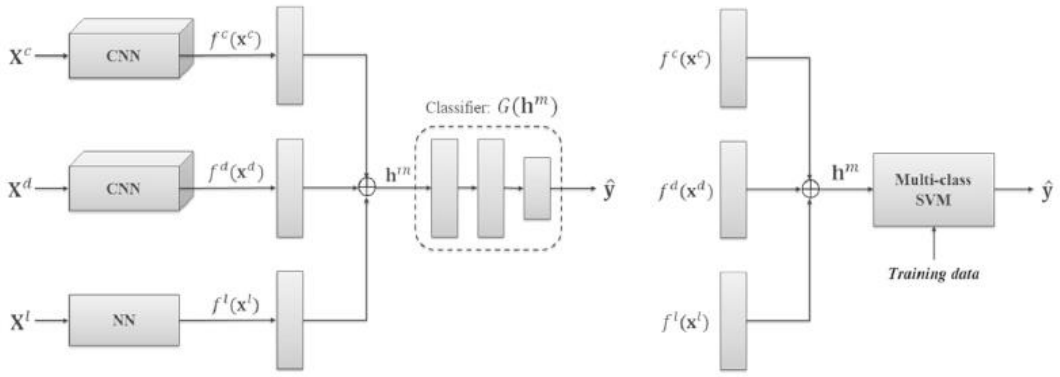


Fig. 144 - : Arquitectura sistema multimodal para reconocer gestos de la mano usando CNN y SVM. [16]

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 Red Neuronal (*Neural Network* en inglés, NN) el *Accuracy* fue de 82.83%, lo que demuestra que las CNN ofrecen mejor clasificación.

En Bangladesh [17] 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. 15).

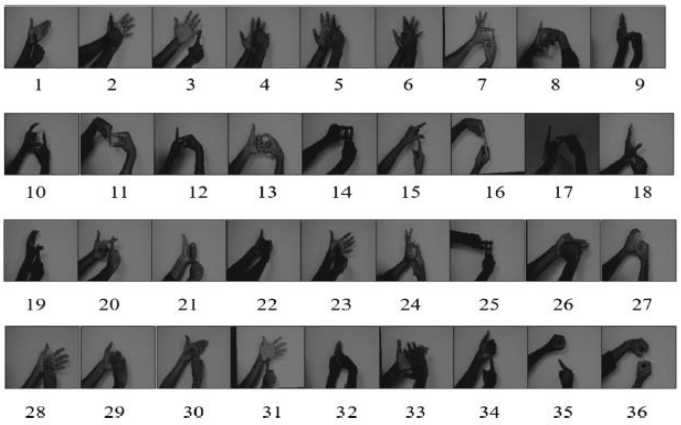


Fig. 155 - : 36 clases de números representados en lenguaje de señas en Bangladesh. [17]

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. 16).

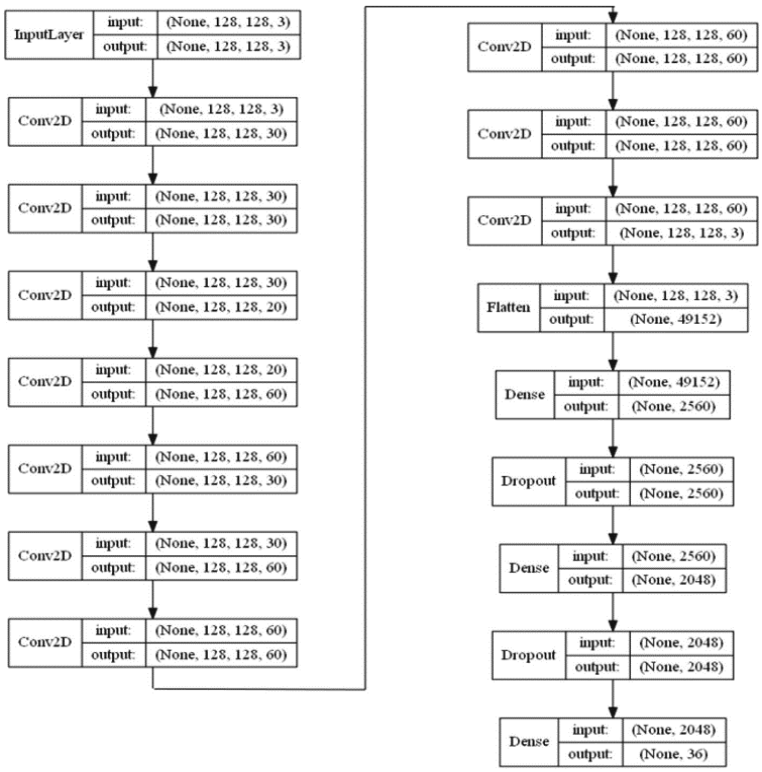


Fig. 166 - : Arquitectura de la CNN de clasificación de caracteres en Bangladesh. [17]

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

Una CNN como en (Fig. 17) es una red neuronal artificial especial diseñada para resolver problemas de clasificación y tratamiento de imágenes [18]. Allí no solo las neuronas están conectadas entre sí por medio de capas intermedias, sino que además usan filtros o matrices denominados *Kernel* que detectan patrones en las imágenes y extraen características de las mismas (proceso denominado Convolución). Dado que los pixeles de una imagen contienen mucha información, esta se promedia o se obtiene su valor máximo por medio de herramientas denominadas *Pooling*. Finalmente se agrega una capa de clasificación para determinar a qué clase de imagen pertenece [18]:

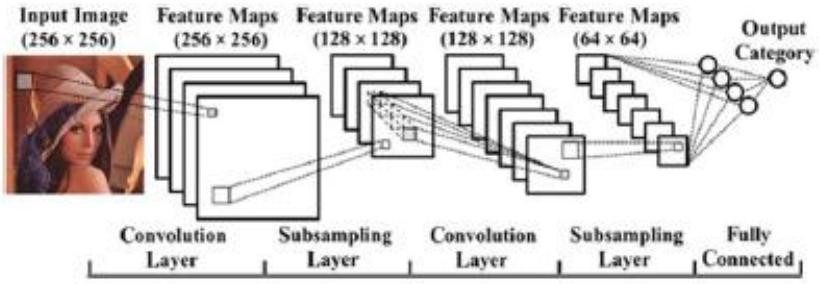


Fig. 177 - : Esquema general de una CNN para clasificar imágenes. [18]

Muchas de las CNN existentes hoy en día se basan en métodos ingenieriles (experimentación, prueba y error) para obtener sistemas que funcionen correctamente. Al buscar una CNN que se relacione con la detección de gestos se encuentra en [19] un foro con la detección del alfabeto universal de lengua de señas que se visualiza en (Fig. 18):

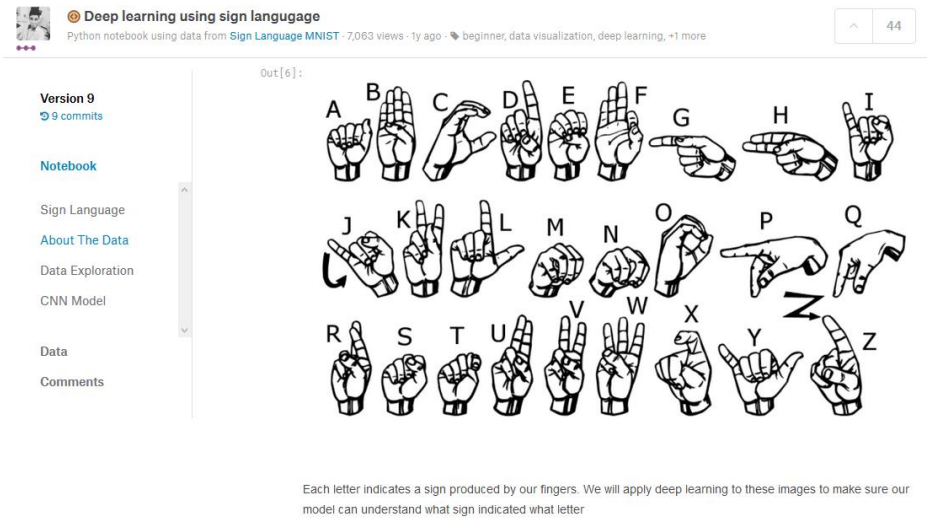


Fig. 188 - : Reto KAGGLE para detectar alfabeto de Lenguaje de Señas Universal. [19]

La (Fig. 19) 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]:

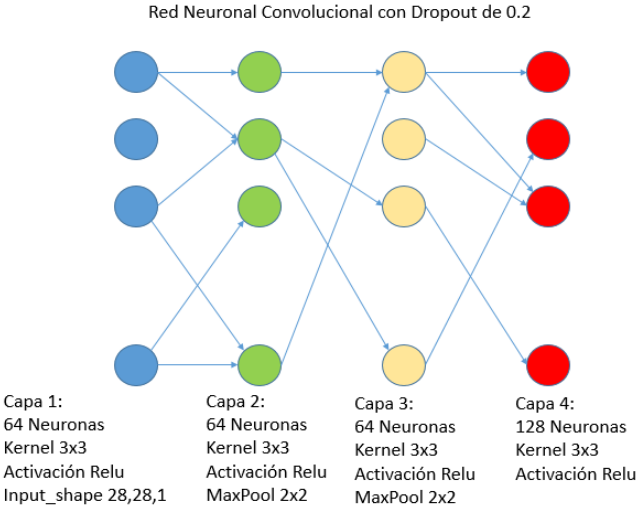


Fig. 199 - : Red CNN con Dropout para 24 clases. [Autor]

Con esto se consigue el primer objetivo, elegir una CNN (técnica de aprendizaje de maquina) para construir el modelo deseado.

3. Modelo Computacional para Reconocimiento de Señas

El presente capitulo describe como se construyó el repositorio de imágenes útiles como datos de entrenamiento del modelo computacional a trabajar. Explica además como estas imágenes son procesadas para ser fuente de información de la CNN a implementar. Se dan los detalles de la construcción de los diferentes modelos a obtener y finalmente la evaluación de cada uno de ellos para seleccionar los mejores a usar.

3.1. Construcción de Repositorio de Lengua de Señas Colombiano

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. 20) y el conocimiento de una persona interprete [20]:

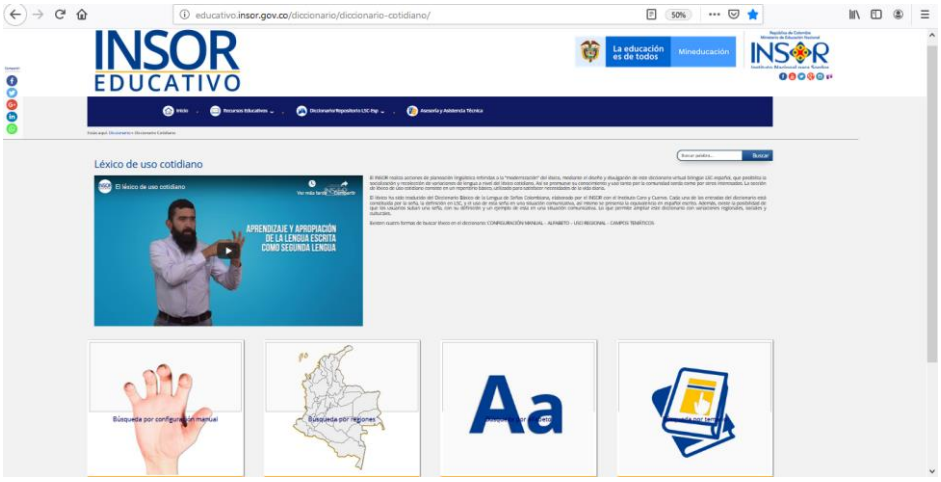


Fig. 20 - : Página Web Diccionario de Gestos de INSOR. [20]

Dependiendo de la región o la categoría que se desee consultar, existen videos multimedia de corta duración donde un intérprete realiza el movimiento correspondiente a una seña. Señas de carácter universal y otras autóctonas y cotidianas en las regiones se incluyen, como es el caso de

la ciudad de Bogotá con la seña del sistema de transporte masivo representativo Transmilenio de (Fig. 21):

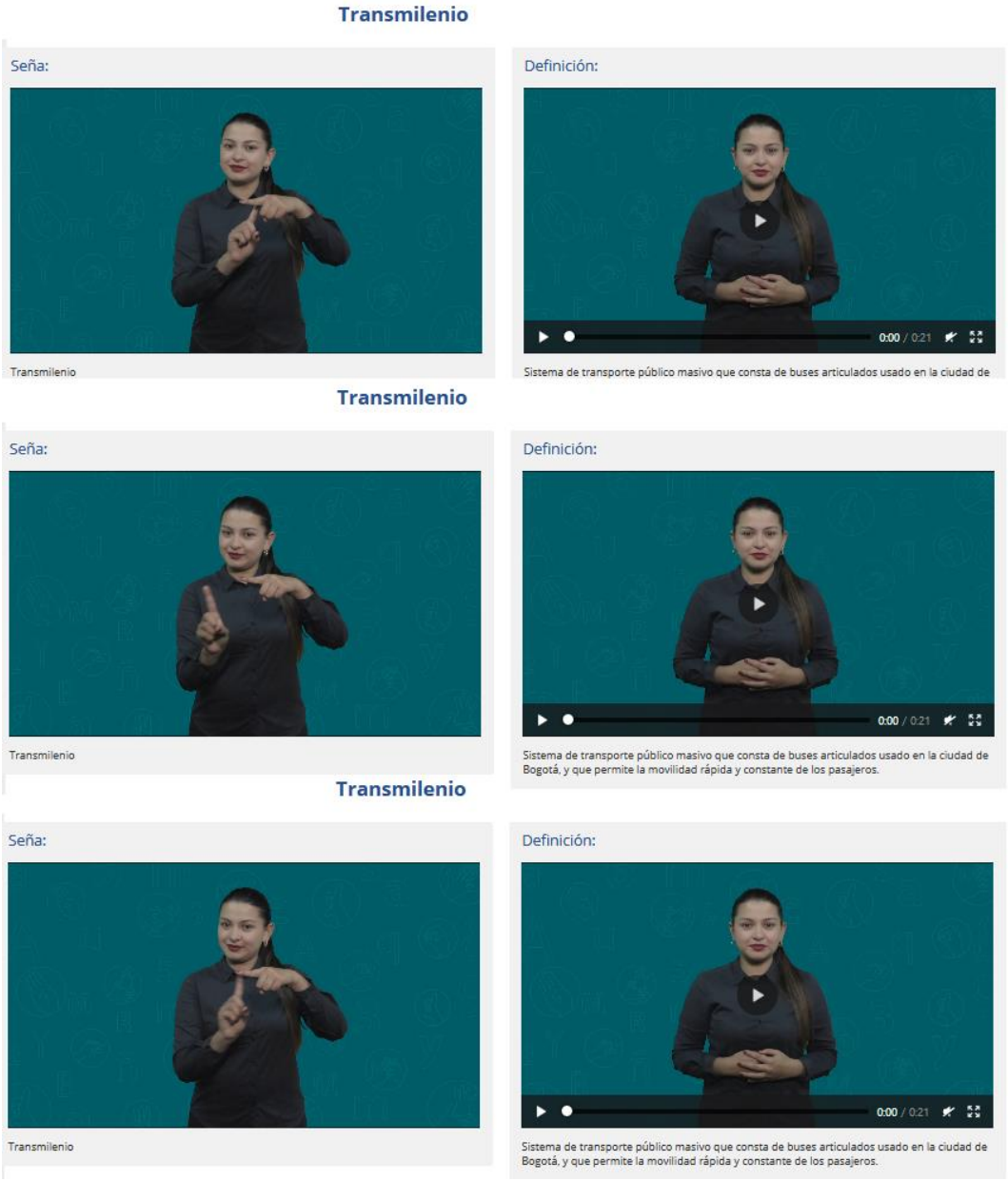


Fig. 21 - : Gesto dinámico usado en Bogotá para indicar el sistema de transporte Transmilenio. [20]

La lengua 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 como el de (Fig. 22), 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 a lo expresado en [20]:

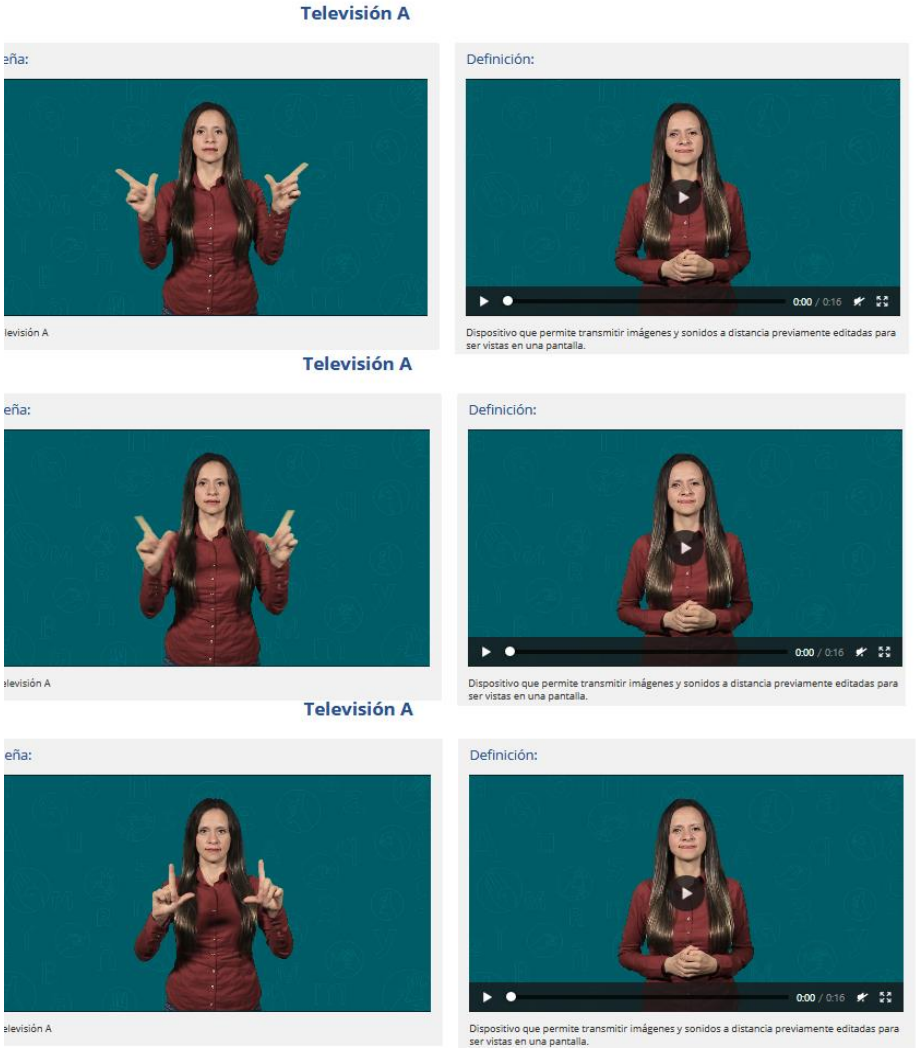


Fig. 22 - : Gesto dinámico de traslación lineal para indicar Televisión. [20]

Con estos criterios establecidos, se construye un repositorio de imágenes con 22 diferentes gestos de la lengua 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 [12] 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 [12] es que solo M-1 serán usadas para la etapa de entrenamiento y la persona restante se incluirá durante la etapa de prueba. 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.

La captura de estas imágenes se hizo con la ayuda de la librería *OpenCV* en *Python*, donde por medio de una cámara se grabó a color en un ambiente controlado (ambientes de luz y manejo de fondos) a la persona realizar los gestos. El proceso para grabar es el de (Fig. 23):

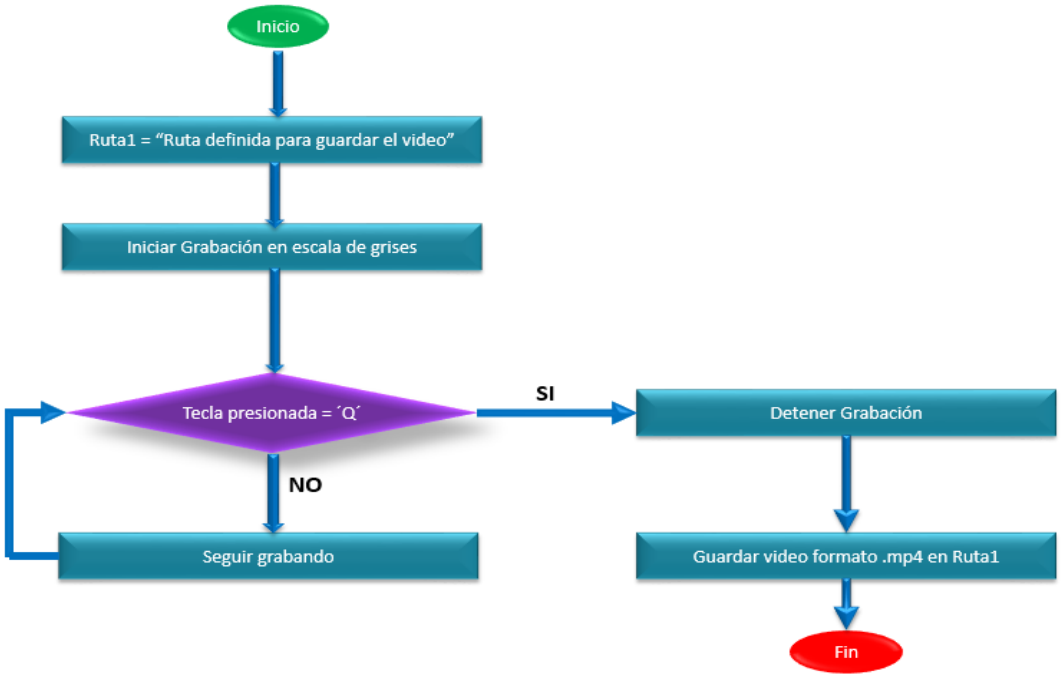


Fig. 23 - : Proceso para grabar videos con OpenCV. [Autor]

Con otro programa (Fig. 24) realizado con la ayuda de *OpenCV* del video se obtienen múltiples *Frames* en escala de grises. Las imágenes que tienen relación con las señales se etiqueten en un documento en formato .csv aparte para tener el aprendizaje supervisado.

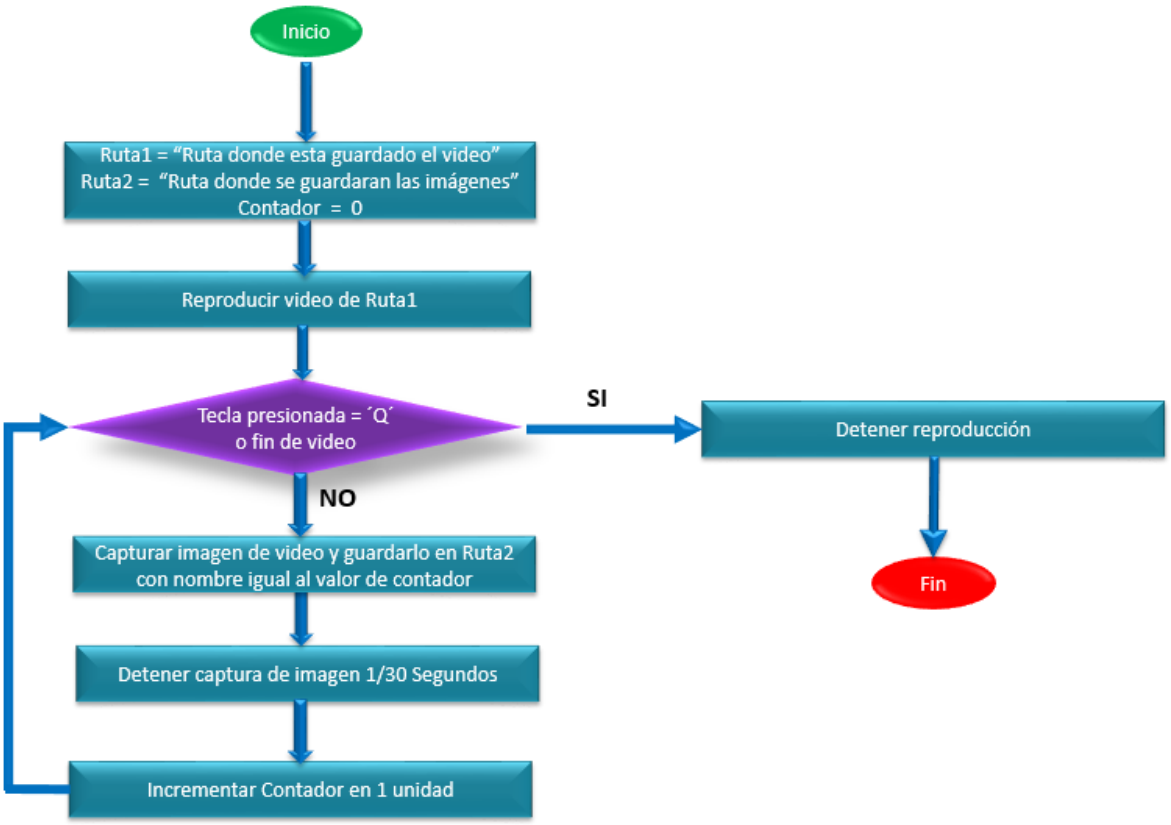


Fig. 24 - : Proceso para obtener Frames de un video. [Autor]

A continuación, se muestran cada una de las 22 señas obtenidas de los videos con las diferentes personas que servirán como insumo para la construcción del repositorio, (Fig. 25 a 46). El total de imágenes del repositorio obtenidas es de 3168 imágenes con una distribución uniforme para las 22 clases.

Fig. 25 - : Seña estática de la palabra YO. [Autor]

Fig. 26 - : Seña estática de la palabra NOMBRE. [Autor]

Fig. 27 - : Seña dinámica de HOLA1 con 2 dedos. [Autor]

Fig. 28 - : Seña dinámica de HOLA2 con 4 dedos. [Autor]

Fig. 29 - : Seña estática de expresión ¿QUÉ PASO? [Autor]

Fig. 30 - : Seña dinámica palabra GRACIAS. [Autor]

Fig. 31 - : Seña dinámica expresión CON MUCHO GUSTO. [Autor]

Fig. 32 - : Seña dinámica expresión SIÉNTESE. [Autor]

Fig. 33 - : Seña dinámica palabra BAÑO. [Autor]

Fig. 34 - : Seña estática de palabra HOY. [Autor]

Fig. 35 - : Seña estática de palabra CASA. [Autor]

Fig. 36 - : Seña dinámica palabra ESCUCHAR. [Autor]

Fig. 37 - : Seña dinámica palabra ATARDECER. [Autor]

Fig. 38 - : Seña dinámica palabra COLOR. [Autor]

Fig. 39 - : Seña dinámica palabra PROFESOR. [Autor]

Fig. 40 - : Seña estática de palabra UNIVERSIDAD. [Autor]

Fig. 41 - : Seña dinámica palabra TELEVISOR. [Autor]

Fig. 42 - : Seña dinámica palabra NOVIO. [Autor]

Fig. 43 - : Seña dinámica palabra PAPÁ. [Autor]

Fig. 44 - : Seña dinámica palabra MAMÁ. [Autor]

Fig. 45 - : Seña estática expresión TENGO CURIOSIDAD. [Autor]

Fig. 46 - : Seña estática expresión TENER POSESIÓN. [Autor]

Todas las imágenes contienen una resolución de 640x380 y guardadas en formato .png. El resumen de cada una de las señas y la cantidad de personas está en la tabla 1:

Tabla 1 - Resumen Imágenes por Clase [Autor]

Nombre de Seña	Cantidad de imágenes	N° de personas que realizan la seña
ATARDECER	144	5
BAÑO	144	5
CASA	144	5
COLOR	144	5
ESCUCHAR	144	5
GRACIAS	144	5
HOLA1	144	5
HOLA2	144	5
HOY	144	5
MAMA	144	5
MUCHOGUSTO	144	5
NOMBRE	144	5
NOVIO	144	5
PAPA	144	5
PROFESOR	144	5
QUEPASO	144	5
SIENTESE	144	5
TELEVISION	144	5
TENERCURIOSIDAD	144	5
TENERPOSESION	144	5
UNIVERSIDAD	144	5
YO	144	5

De esta manera, se consigue el segundo objetivo específico, construir un repositorio de imágenes que contenga señas del LSC. En la siguiente sección se describe el pre procesamiento de estas imágenes antes de ser usadas como datos de entrenamiento del modelo.

3.2. Procesamiento de Datos

Dado que los videos registrados tienen una duración en promedio de 3 a 4 minutos y se obtienen 20 *Frames* por segundo con la ayuda del programa construido con la librería *OpenCV*, la

cantidad de imágenes obtenidas superan en su mayoría el valor de 3000 o 4000 como se ve en (Fig. 47):

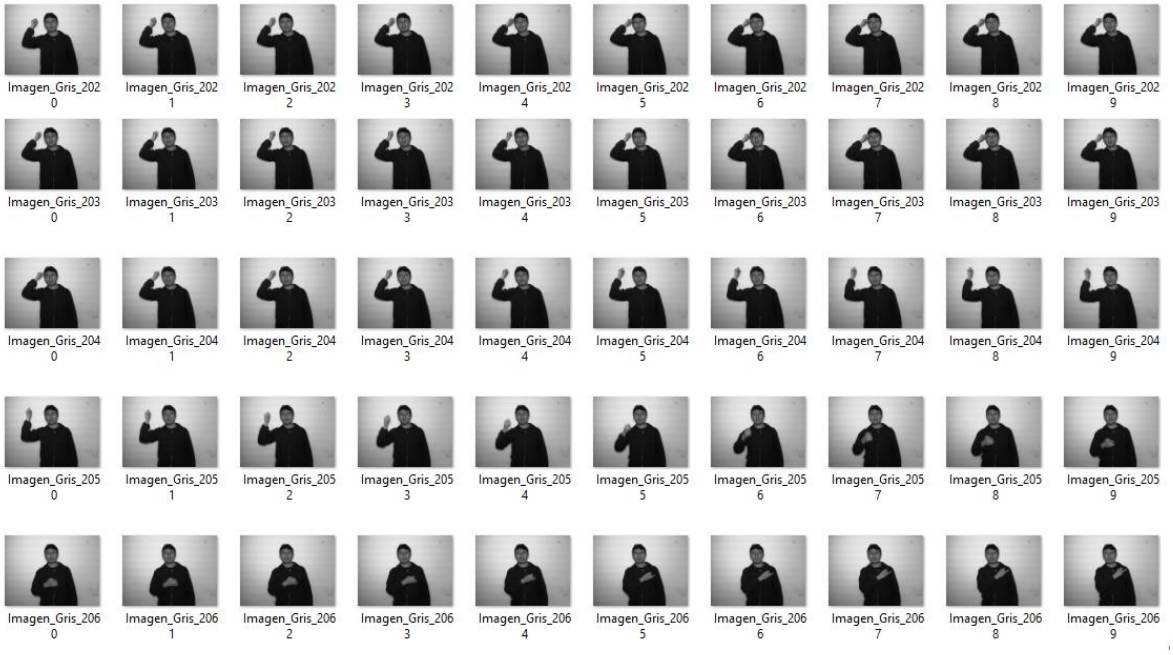


Fig. 47 - : Imágenes secuenciales obtenidas de los videos grabados de una persona haciendo las señas. [Autor]

Por esta razón, de manera manual se tomó cada uno de los 10 videos registrados (2 por persona) para seleccionar las imágenes que son utilizadas para construir el repositorio. Las imágenes se renombran con secuencia de números para evitar que se repitan o renombren entre ellas. Imágenes que no son usadas para el repositorio se borran. Esta etapa es la limpieza de datos.

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. 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 el tamaño de las mismas 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 76800 (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. 48):

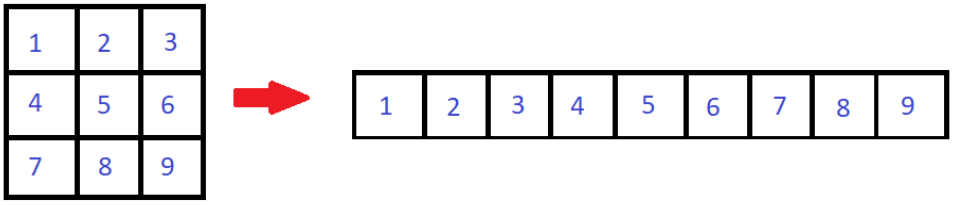


Fig. 48 - : Proceso de conversión de matriz 2x2 a vector de 1 dimensión usado en imágenes. [Autor]

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 Excel (almacenado en formato .csv) y se visualizó los primeros registros de la asociación del nombre de la imagen con su etiqueta como se ve en (Fig. 49):

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

Fig. 49 - : Validación en Pandas de archivo de etiquetas asociadas con el nombre de archivo de imagen. [Autor]

Por medio de la librería *Matplotlib* de *Python* se ve la distribución uniforme de las señas en (Fig. 50):

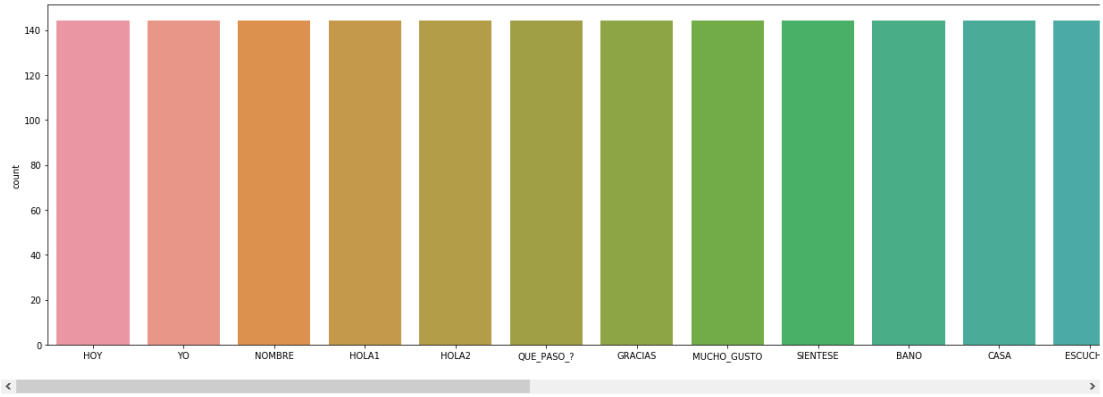


Fig. 50 - : Distribución uniforme de todas las señas del repositorio. [Autor]

Dado que para la CNN se trabajan 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).

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.

Finalmente, se realizó la normalización de los datos para manejar los valores numéricos tanto de los datos de entrenamiento como de prueba entre 0 y 1. Para ello se determinó 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 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

Al haber realizado el procesamiento de las diferentes imágenes que servirán como datos fuente de entrenamiento del modelo, se construye el mismo. La arquitectura de este modelo es la misma planteada heurísticamente en [19], reconocimiento de alfabeto de señas de personas sordas, con la diferencia que no se agregó inicialmente una capa de *Dropout* para ver como este factor altera una CNN full 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* en inglés, 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 describe 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 320x340. El número de neuronas y función de activación de la capa son parámetros a 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 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 capa 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 nuevamente 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. 51). 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):

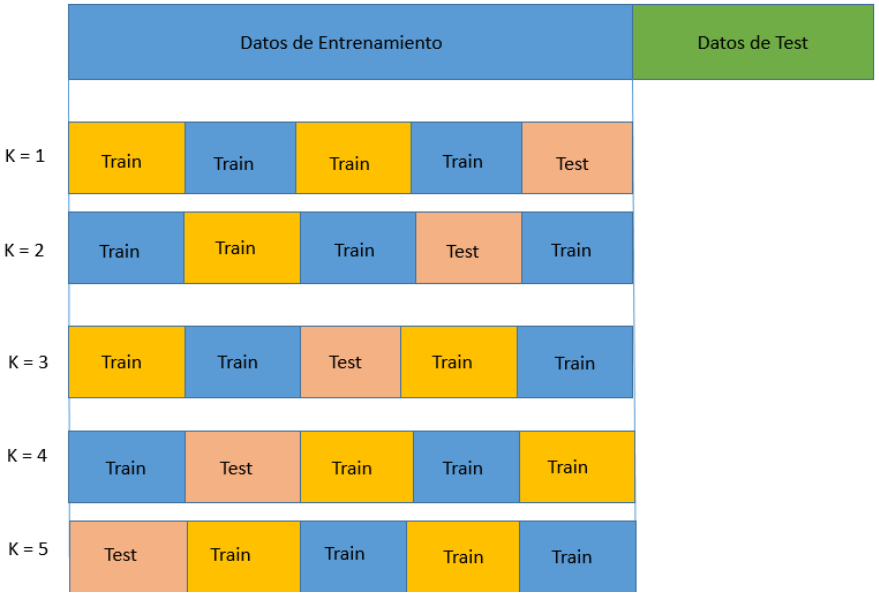


Fig. 51 - : Cross Validation con 5 pliegues. Sobre los datos de Entrenamiento se hacen nuevas divisiones y se itera. [Autor]

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 default un valor de 32. En el momento de realizar esta simulación computacionalmente 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 función de activación que se observa en (Fig. 52):

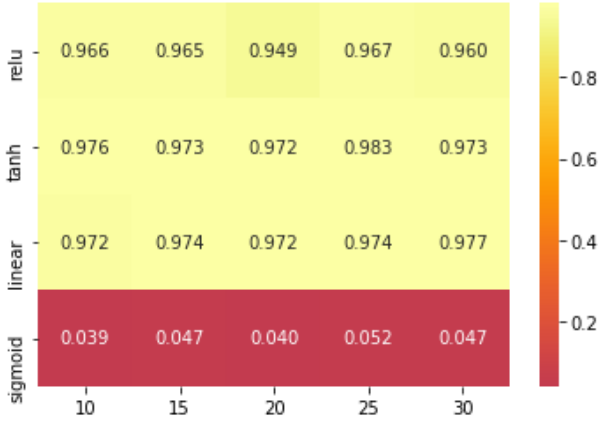


Fig. 52 - : Mapa de calor Grid Search Funciones de Activación vs Numero de Neuronas por Capas. [Autor]

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. Los resultados detallados de esta simulación son almacenados en un archivo Excel que se puede encontrar en el repositorio de material adicional del presente trabajo (ver anexo A).

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.

Una vez construido el modelo con estos parámetros, Keras permitió obtener el número de parámetros por capas y el total de la red. Se observó claramente que las capas *MaxPooling* son bastante útiles para reducir el número de estos. El número total de parámetros de la arquitectura del modelo es 2.494122 de acuerdo a la tabla 2:

Tabla 2 - Resumen de Parámetros de Modelo Obtenido [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Capa 5 Flatten	(None,113100)	0
Capa 6 Densa	(None, 22)	2.488.222
Total Parámetros:	2.494.122	
Parámetros Entrenables:	2.494.122	
Parámetros No Entrenables:	0	

El modelo obtenido se entrena de manera similar que 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. La (Fig. 53) muestra cómo evolucionó el *Loss* en función del número de épocas:

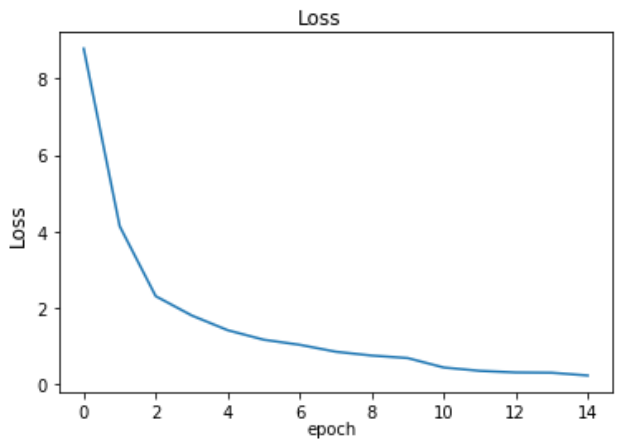


Fig. 53 - : Loss del modelo durante el entrenamiento por varias épocas. [Autor]

Igualmente, en (Fig. 54) el *Accuracy* en función del número de épocas, se ve como este aumenta para llegar a valores cercanos al 0.9:

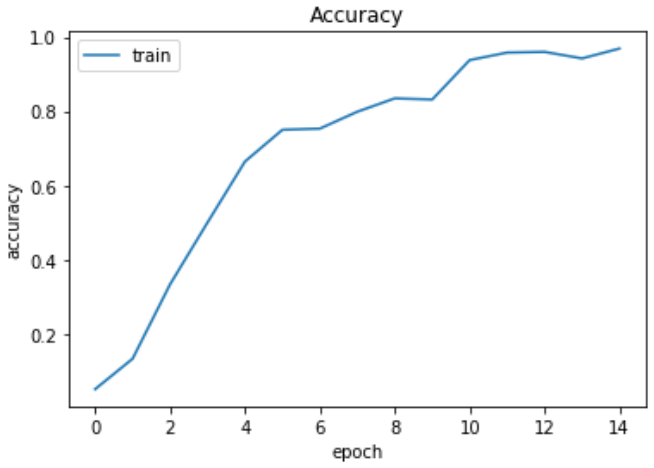


Fig. 54 - : Accuracy del modelo durante el entrenamiento por varias épocas. [Autor]

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

Se usaron los datos de test para hacer las respetivas predicciones y de cada clase sacar las medidas de *precisión*, *recall* y *f1* con la ayuda de la función *classification_report* de Tensorflow

(en este paso, las predicciones son convertidas en formato *One Hot Code* para tener la misma base de comparación). Los resultados de cada clase se muestran en la tabla 3, siendo organizadas por orden alfabético:

Tabla 3 - Resumen de Medidas por Clase sobre los Datos de Test o Prueba [Autor]

Nombre de Seña	Precisión	Recall	F1-Score	Número de imágenes de muestra
ATARDECER	1.0	1.0	1.0	44
BAÑO	0.98	1.0	0.99	43
CASA	1.0	1.0	1.0	43
COLOR	1.0	1.0	1.0	43
ESCUCHAR	1.0	0.95	0.98	44
GRACIAS	0.98	1.0	0.99	43
HOLA1	0.69	0.98	0.81	44
HOLA2	0.84	0.63	0.72	43
HOY	1.0	0.98	0.99	43
MAMA	1.0	1.0	1.0	43
MUCHOGUSTO	1.0	0.98	0.99	44
NOMBRE	1.0	0.91	0.95	43
NOVIO	1.0	1.0	1.0	43
PAPA	0.98	1.0	0.99	43
PROFESOR	1.0	1.0	1.0	43
QUEPASO	0.95	0.93	0.94	44
SIENTESE	1.0	1.0	1.0	43
TELEVISION	0.98	1.0	0.99	43
TENERCURIOSIDAD	1.0	0.98	0.99	43
TENERPOSESION	1.0	1.0	1.0	43
UNIVERSIDAD	1.0	1.0	1.0	43
YO	0.88	0.86	0.87	43

A modo de ejemplo se obtiene el *Kernel* de la primera capa en (Fig. 55):

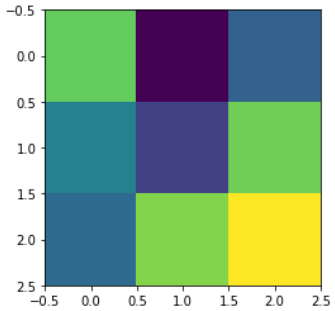


Fig. 55 - : Valores de Kernel de la primera capa convolucional. [Autor]

A continuación, se elige una imagen de muestra que se ve en (Fig. 56):

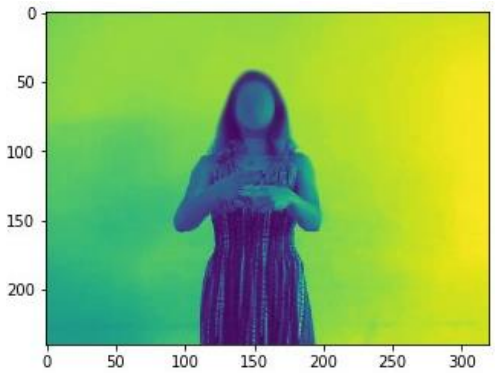


Fig. 56 - : Imagen de prueba antes de aplicar Kernel de primera capa convolucional. [Autor]

Y al realizar la convolución de ese *Kernel* con la imagen, se obtiene la (Fig. 57):

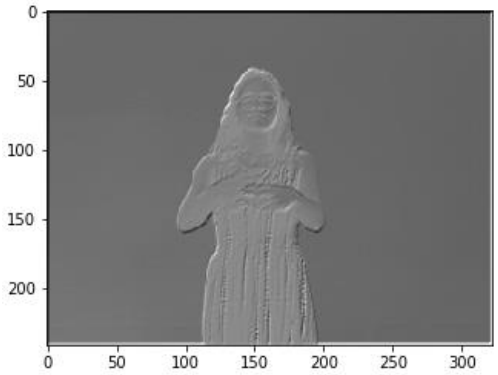


Fig. 57 - : Imagen de prueba después de aplicar Kernel de primera capa convolucional. [Autor]

Se guarda el modelo obtenido para posteriormente ser usado en el prototipo a implementar.

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). El nuevo modelo obtenido no modifica los parámetros del modelo inicial (total de parámetros igual a 2.494.122) como se observa en la tabla 4:

Tabla 4 - Resumen de Parámetros de Modelo obtenido con Dropout [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Capa 5 Flatten	(None,113100)	0
Capa 6 Dropout	(None,113100)	0
Capa 7 Densa	(None, 22)	2.488.222
Total Parámetros:	2.494.122	
Parámetros Entrenables:	2.494.122	
Parámetros No Entrenables:	0	

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. 58 y Fig. 59) respectivamente:

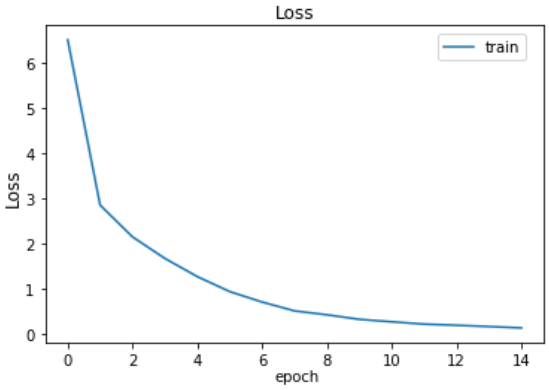


Fig. 58 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas. [Autor]

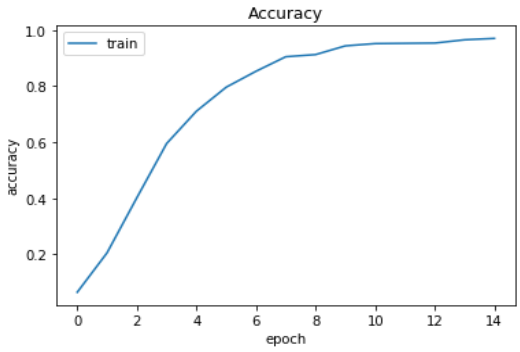


Fig. 59 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas. [Autor]

Para los mismos datos de test, el *Loss* es 0.1516 y el *Accuracy* 0.9642. La medición de *Precisión*, *Recall* y *F1-Score* para cada clase se detalla en la tabla 5:

Tabla 5 - Resumen de Medidas por Clase sobre los Datos de Prueba en Modelo con Dropout [Autor]

Nombre de Seña	Precisión	Recall	F1-Score	Número de imágenes de muestra
ATARDECER	1.0	1.0	1.0	44
BAÑO	1.0	1.0	1.0	43
CASA	1.0	1.0	1.0	43
COLOR	1.0	1.0	1.0	43
ESCUCHAR	1.0	0.98	0.99	44
GRACIAS	0.91	1.0	0.96	43
HOLA1	0.69	1.0	0.81	44
HOLA2	1.0	0.53	0.7	43
HOY	1.0	1.0	1.0	43
MAMA	1.0	1.0	1.0	43
MUCHOGUSTO	1.0	0.95	0.98	44
NOMBRE	0.98	0.98	0.98	43
NOVIO	1.0	1.0	1.0	43
PAPA	1.0	0.95	0.98	43
PROFESOR	1.0	1.0	1.0	43
QUEPASO	0.83	1.0	0.91	44
SIENTESE	1.0	1.0	1.0	43
TELEVISION	1.0	1.0	1.0	43
TENERCURIOSIDAD	1.0	1.0	1.0	43
TENERPOSESION	1.0	1.0	1.0	43
UNIVERSIDAD	1.0	1.0	1.0	43
YO	1.0	0.81	0.9	43

El *Kernel* obtenido de la primera capa es el que se muestra en (Fig. 60):

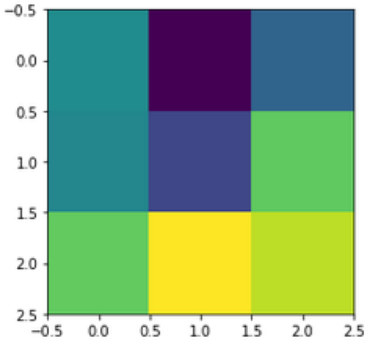


Fig. 60 - : Valores de Kernel de la primera capa convolucional de modelo con Dropout. [Autor]

Al hacer la convolución con la misma imagen de muestra se obtiene la (Fig. 61):

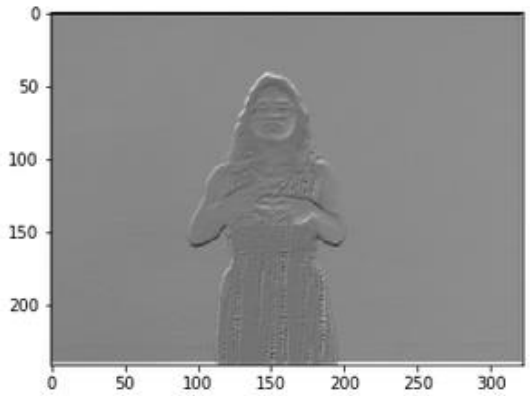


Fig. 61 - : Imagen de prueba después de aplicar Kernel de primera capa convolucional de modelo con Dropout. [Autor]

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.* Un ejemplo de la estructura de uno de estos modelos se observa en la tabla 6, que aplica para el modelo con más capas convolucionales y de *MaxPool*:

Tabla 6 - Resumen de Modelo con 6 Capas Convolucionales con Dropout [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Capa 5 Convolucional 2D	(None,76,56,25)	5650
Capa 6 Max Pooling	(None,38,28,25)	0
Capa 7 Convolucional 2D	(None,36,26,25)	5650
Capa 8 Max Pooling	(None,18,13,25)	0
Capa 9 Convolucional 2D	(None,16,11,25)	5650
Capa 10 Max Pooling	(None,8,5,25)	0
Capa 11 Convolucional 2D	(None,6,3,25)	5650
Capa 12 Max Pooling	(None,3,1,25)	0
Capa 5 Flatten	(None,75)	0
Capa 6 Dropout	(None,75)	0
Capa 7 Densa	(None, 22)	1.672
Total Parámetros:	30.172	
Parámetros Entrenables:	30.172	
Parámetros No Entrenables:	0	

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 7 y los nombres de los modelos usaron acrónimos como CV5 (*Cross Validation* de 5), DP (*Dropout*) y LY (*Layer*) para diferenciarlos:

Tabla 7 - Resumen Modelos Loss [Autor]

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

Por su parte el Accuracy obtenido es el de la tabla 8:

Tabla 8 - Resumen Modelos Accuracy [Autor]

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

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. Al hacer esto, la estructura del nuevo modelo en Keras fue el descrito en la tabla 9:

Tabla 9 - Nuevo Modelo con los Pesos Cargados del Modelo Base, Técnica de Transfer Learning [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Total Parámetros:	5.900	
Parámetros Entrenables:	0	
Parámetros No Entrenables:	5.900	

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

comportamiento. La estructura obtenida en la tabla 10 permitió ver que el entrenamiento solo se haría solo con las nuevas modificaciones y se usaran los pesos del primero:

Tabla 10 - Nuevo Modelo con Transfer Learning Entrenando solo las Capas Adicionales [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Capa 5 Flatten	(None,113100)	0
Capa 6 Dropout	(None,113100)	0
Capa 7 Densa	(None, 22)	2.488.222
Total Parámetros:	2.494.122	
Parámetros Entrenables:	2.488.222	
Parámetros No Entrenables:	5.900	

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. 62 y Fig. 63):

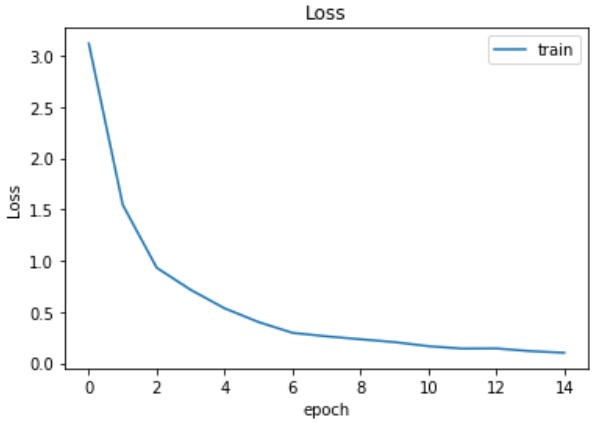


Fig. 62 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas usando Transfer Learning. [Autor]

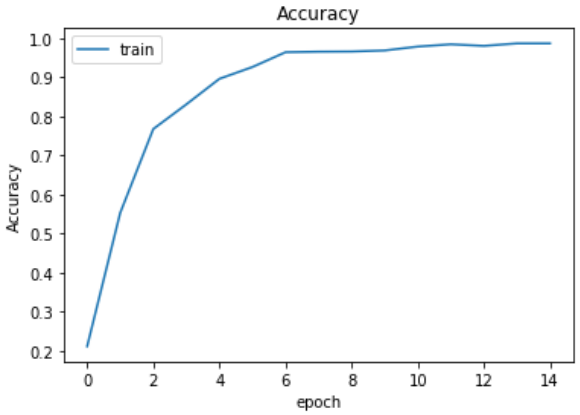


Fig. 63 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas usando Transfer Learning. [Autor]

Se construyen modelos similares a los anteriores con la misma estructura, usando los pesos del modelo base sin modifican durante el entrenamiento. La tabla 11muestra las capas agregadas a entrenar para un modelo con 6 capas convolucionales y *MaxPool* usando *Dropout*:

Tabla 11 - Resumen de Modelo con 6 Capas Convolucionales con Dropout usando Transfer Learning [Autor]

Capa (Tipo)	Forma de Salida	# Parámetros
Capa 1 Convolucional 2D	(None,318,238,25)	250
Capa 2 Max Pooling	(None,159,119,25)	0
Capa 3 Convolucional 2D	(None,157,1119,25)	5650
Capa 4 Max Pooling	(None,78,58,25)	0
Capa 5 Convolucional 2D	(None,76,56,25)	5650
Capa 6 Max Pooling	(None,38,28,25)	0
Capa 7 Convolucional 2D	(None,36,26,25)	5650
Capa 8 Max Pooling	(None,18,13,25)	0
Capa 9 Convolucional 2D	(None,16,11,25)	5650
Capa 10 Max Pooling	(None,8,5,25)	0
Capa 11 Convolucional 2D	(None,6,3,25)	5650
Capa 12 Max Pooling	(None,3,1,25)	0
Capa 5 Flatten	(None,75)	0
Capa 6 Dropout	(None,75)	0
Capa 7 Densa	(None, 22)	1.672
Total Parámetros:	30.172	
Parámetros Entrenables:	24.272	
Parámetros No Entrenables:	5900	

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 12 y 13 respectivamente:

Tabla 12 - Resumen Modelos Loss con Transfer Learning [Autor]

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 13 - Resumen Modelos Accuracy con Transfer Learning [Autor]

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

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 radico 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. 64) se observa el comportamiento del *Loss* al final de las 15 épocas y en (Fig. 65) el *Accuracy* del modelo base con Dropout:

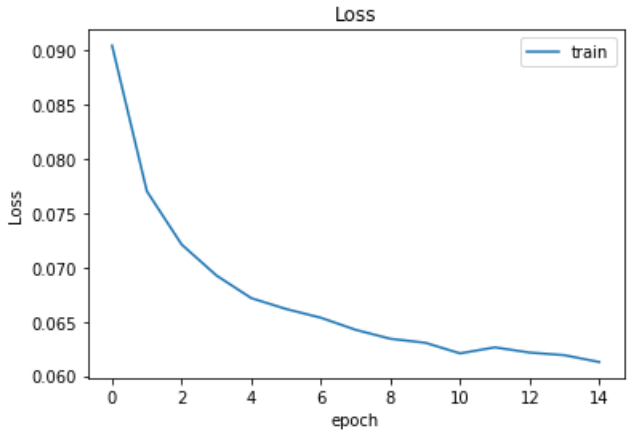


Fig. 64 - : Loss del modelo con Dropout durante el entrenamiento por varias épocas usando Fine Tuning. [Autor]

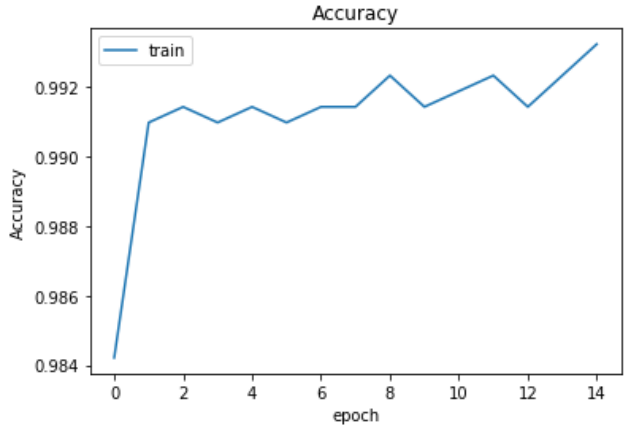


Fig. 65 - : Accuracy del modelo con Dropout durante el entrenamiento por varias épocas usando Fine Tuning. [Autor]

En este caso los nombres de los modelos poseen el acrónimo FT (*Fine Tuning*) y los resultados de *Loss* y *Accuracy* se ven en las tablas 14 y 15:

Tabla 14 - Resumen Modelos Loss con Fine Tuning [Autor]

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 15 - Resumen Modelos Accuracy con Fine Tuning [Autor]

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

Como se pudo observar, modificando el modelo base se obtuvo 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 8, 13 y 15 se decide tomar cuatro modelos para usar en el prototipo y evaluar con nuevas imágenes similares al del proceso de entrenamiento y test. Estos modelos fueron el base (*Accuracy* de 0.9631), el modelo base con 2 capas convolucionales mas 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).

4. Prototipo de Software

Se comenzó a diseñar y construir un prototipo de software en el cual la persona interprete use el modelo obtenido de manera sencilla. Esto se hace usando el Framework de trabajo *FLASK* para construir una aplicación Web manera rápida y con librerías de *Machine Learning* de *Python*. Finalmente se hacen las pruebas sobre el prototipo y se obtiene un concepto del mismo.

4.1Diseño de Prototipo

Basados en [22] se diseña el prototipo usando UML, definiendo el caso de uso específico (que se desea hacer con el software), diagrama de secuencias (interacción entre el usuario y el software a través del tiempo) y diagrama de actividades (algoritmo del prototipo del software). La aplicación Web estuvo local en el PC donde va a realizarse las pruebas funcionales por simplicidad.

4.1.1 Caso de Uso

En (Fig. 66) se define como actor al usuario que usara 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 de las 22 señas es. Al finalizar la tarea la persona intérprete regresa a la página de inicio para comenzar de nuevo:

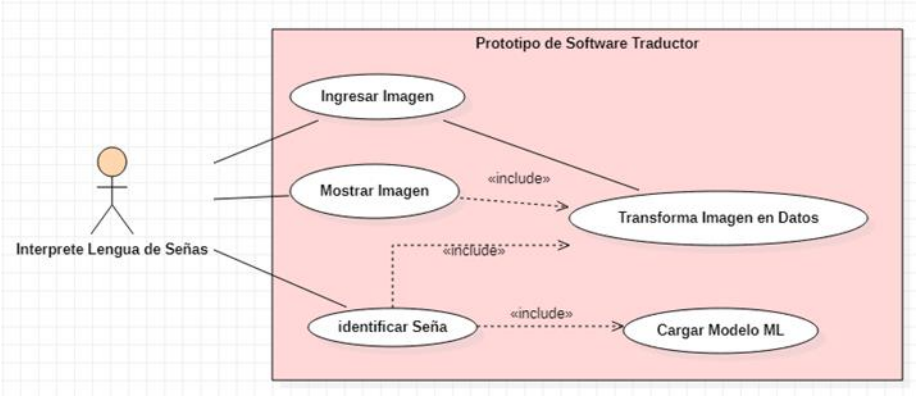


Fig. 66 - : Caso de Uso del sistema. [Autor]

4.1.2 Diagrama de Secuencias

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. 67). El primero de ellos es el intérprete que realiza una interacción con la Interfaz Gráfica de Usuario (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.

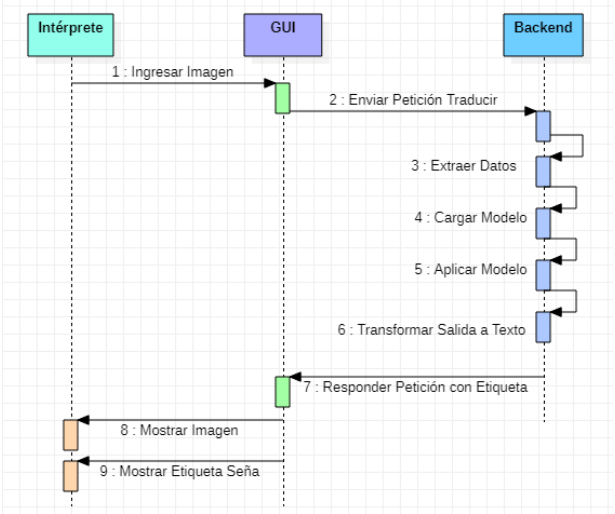


Fig. 67 - : Diagrama de Secuencias del sistema. [Autor]

4.1.3 Diagrama de Actividades

Para finalizar, se detalla la lógica del software con una descripción de las actividades secuenciales de la lógica del mismo y de forma visual en (Fig. 68). El proceso de clasificación de imágenes en una de las señas establecidas es:

1. Cargue de la imagen de la señal de la página principal
2. Cargar el modelo de entrenamiento obtenido
3. Almacenamiento de la imagen a matriz en escala de grises
4. Almacenamiento de la imagen en formato RGB
5. Transformación de la matriz de escala de grises a vector de una dimensión
6. Ajuste de vector en formato de ingreso para la red neuronal
7. Aplicar modelo a vector y guardar resultado en variable numérica de salida

8. Convertir variable de salida en etiqueta de clase

9. Enviar etiqueta de salida a página de respuesta

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

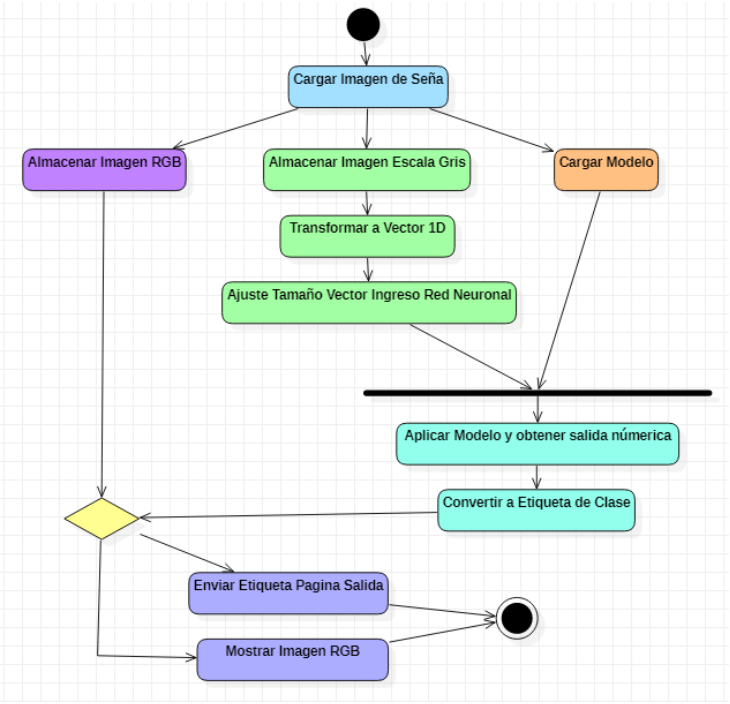


Fig. 68 - : Diagrama de actividades del sistema. [Autor]

4.2 Codificación del Prototipo

Una vez definida la arquitectura y lógica del prototipo de software se comenzó la codificación del mismo. El primer paso fue construir un ambiente en el software *Anaconda Python* con el fin de tenerlo exclusivamente para esta funcionalidad. El nombre de este ambiente se denominó Prototipo y se le instaló el Framework de *FLASK* junto con las siguientes versiones de las librerías de Keras, Tensorflow y PIL para que no se presenten inconvenientes:

- Python 3.7
- Pillow 6.2.1
- Scipy 1.1.0
- Tensorflow 1.14
- Keras 2.2.5

Una de las ventajas de usar un Framework es que permiten un desarrollo ágil ya prestablecido haciendo uso de plantillas o estructuras existentes. Para el prototipo en cuestión, se manejó una carpeta donde se encuentran los *Templates* HTML que servirán como *Front-End* hacia el usuario y además contendrá una carpeta *static* con la hoja de estilos de cascada (CSS) para definir la GUI a trabajar. La parte de la lógica del software se definió en funciones escritas en lenguaje de *Python* en el archivo app.py y el modelo se almacenó en el folder model, como se observa en (Fig. 69):

Nombre	Fecha de modifica...	Tipo	Tamaño
model	16/11/2019 8:25 p....	Carpeta de archivos	
static	11/11/2019 8:20 p....	Carpeta de archivos	
templates	11/11/2019 8:20 p....	Carpeta de archivos	
app.py	18/11/2019 4:37 p....	Archivo PY	4 KB
README.md	11/11/2019 1:44 p....	Archivo MD	1 KB
requirements	11/11/2019 1:44 p....	Documento de tex...	1 KB

Fig. 69 - : Estructura folder prototipo usando Framework Flask. [Autor]

Una vez modificada la página de inicio, se tiene la visual de la (Fig. 70) como página *Home* del prototipo:

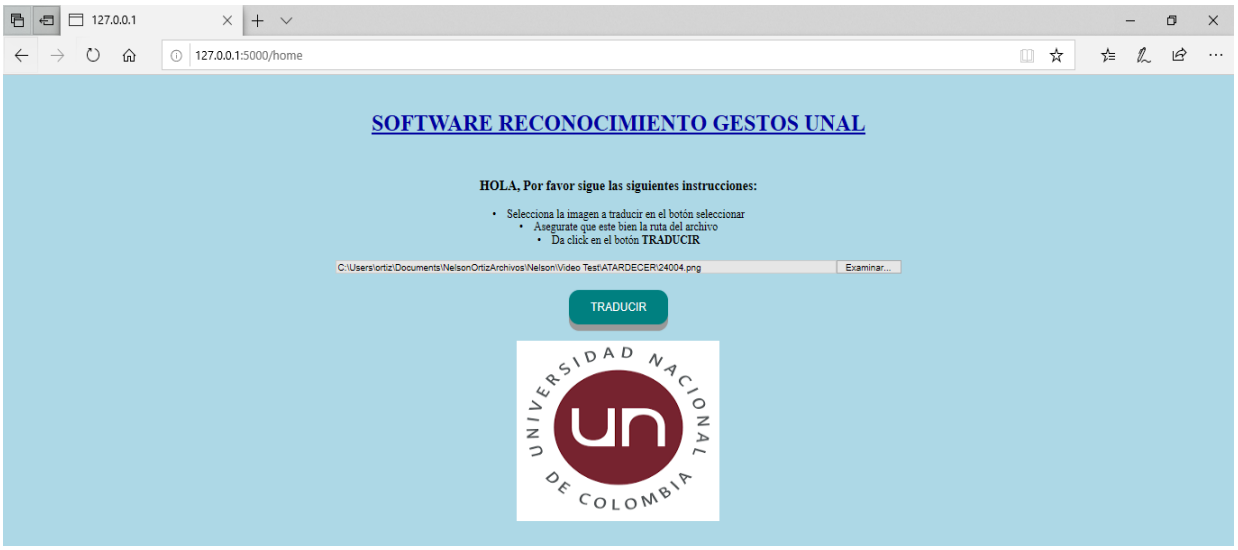


Fig. 70 - : Página Home del sistema. [Autor]

Y la página de respuesta se diseña para que la respuesta sea sencilla y fácil de entender, de acuerdo a (Fig. 71):

Fig. 71 - : Página Respuesta etiqueta de seña. [Autor]

De esta manera se concluye el tercer objetivo específico que consiste en la implementación de un prototipo de software que haga uso del modelo de entrenamiento obtenido. Para mayor detalle del software ver información en el anexo.

4.3 Prueba de Concepto y Análisis de Resultados

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. Con esta finalidad, se desarrolla una lista de actividades y condiciones que se deben cumplir para poder realizar la evaluación del modelo. Estas son:

1. **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.

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

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

4. **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.

5. **Actividad 5:** Construir en un archivo Excel una matriz de confusión con la seña a realizar y la seña reconocida por el software. Adicional incluir el nombre de la persona que realiza el gesto. Separar por cada uno de los modelos a evaluar.

6. **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.

7. **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.

8. **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 construyó los fólderes de (Fig. 72) para que la evaluación sea más organizada y transversal para todos los modelos, teniendo así el mismo punto de comparación de los modelos obtenidos:

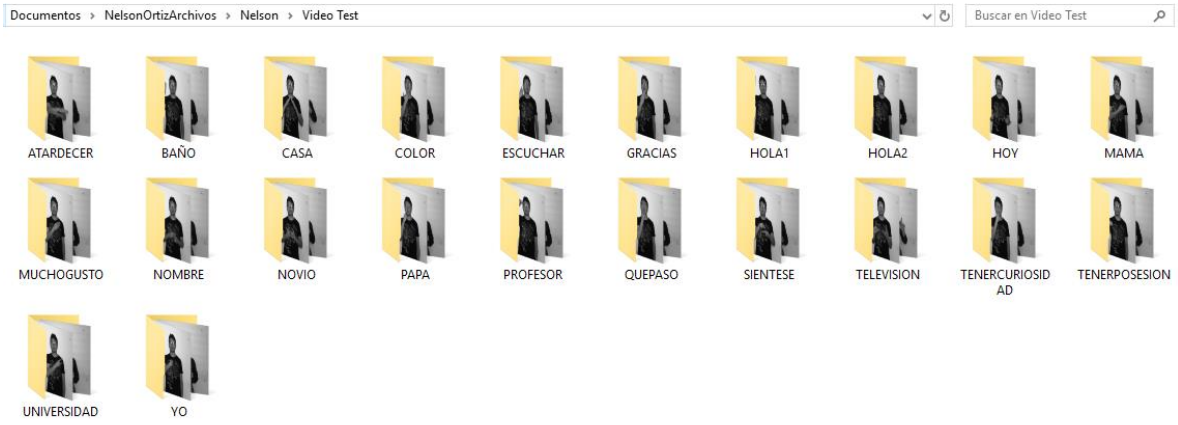
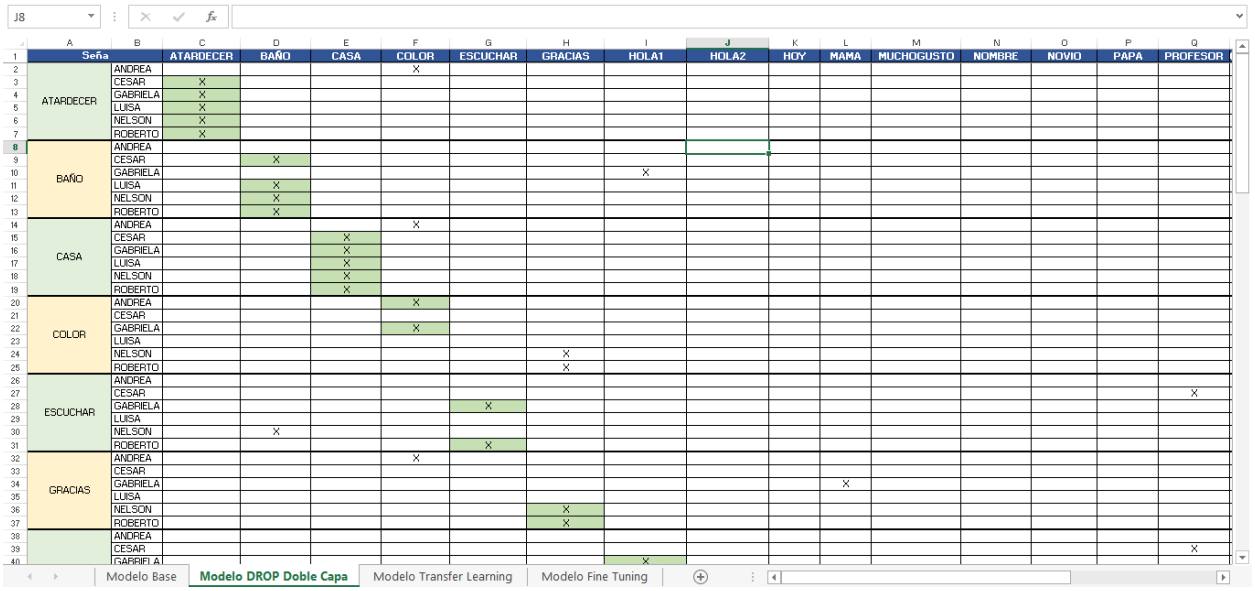


Fig. 72 - : Folder con señas de Test clasificado por gestos. [Autor]

De igual forma se construye el archivo en Excel de (Fig. 73), donde cada pestaña incluye una descripción del modelo usado y adicional el nombre del mismo se incluye en la información usada (ver detalles en el anexo):



Seña		ATARDECER	BAÑO	CASA	COLOR	ESCUCHAR	GRACIAS	HOLA1	HOLA2	HOY	MAMA	MUCHOGUSTO	NOMBRE	NOVIO	PAPA	PROFESOR
ATARDECER	ANDREA				X											
	CESAR	X														
	GABRIELA	X														
	LUISA	X														
	NELSON	X														
	ROBERTO	X														
BAÑO	ANDREA															
	CESAR		X													
	GABRIELA							X								
	LUISA		X													
	NELSON		X													
	ROBERTO		X													
CASA	ANDREA				X											
	CESAR			X												
	GABRIELA			X												
	LUISA			X												
	NELSON			X												
	ROBERTO			X												
COLOR	ANDREA				X											
	CESAR															
	GABRIELA				X											
	LUISA															
	NELSON						X									
	ROBERTO						X									
ESCUCHAR	ANDREA															
	CESAR															X
	GABRIELA					X										
	LUISA															
	NELSON		X													
	ROBERTO					X										
GRACIAS	ANDREA				X											
	CESAR															
	GABRIELA										X					
	LUISA															
	NELSON						X									
	ROBERTO						X									
	ANDREA															
	CESAR															X
	GABRIELA							X								

Modelo Base | Modelo DROP Doble Capa | Modelo Transfer Learning | Modelo Fine Tuning

Fig. 73 - : Documento Excel de la Matriz de Confusión de las señas de Test. [Autor]

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 a (Fig. 74):

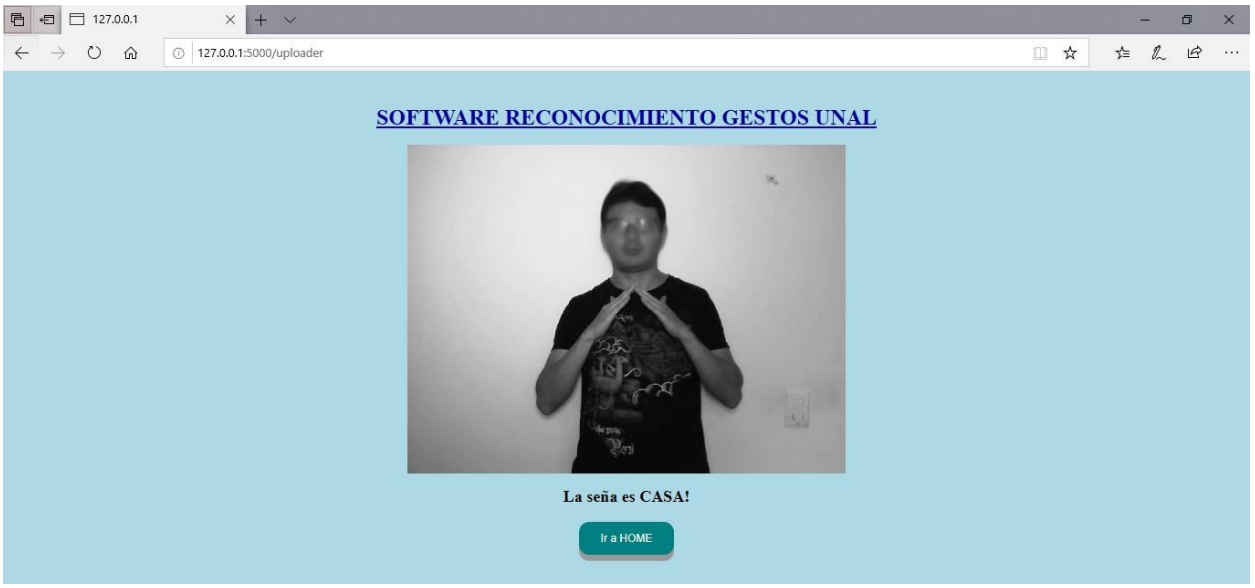


Fig. 74 - : Seña de CASA de test clasificada correctamente. [Autor]

Sin embargo, si se evalúa una seña como escuchar se observa en la (Fig. 75) que clasificó una imagen específica con otra clase de manera errónea:

Fig. 75 - : Seña de ESCUCHA de test clasificada erróneamente. [Autor]

Por su parte, al evaluar una nueva imagen de otra persona se obtuvo aciertos como en la clase COLOR de la (Fig. 76), mientras que con la clase TENERPOSESION se equivocó al clasificarla como YO en (Fig. 77):

Fig. 76 - : Seña de COLOR de persona nueva de test clasificada correctamente. [Autor]

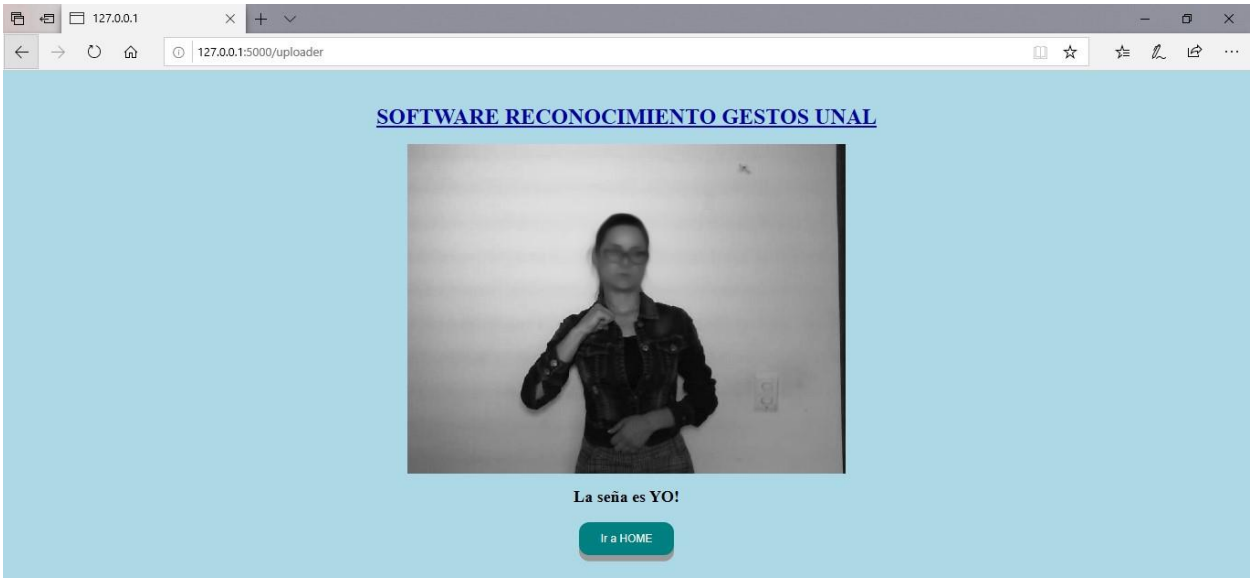


Fig. 77 - : Seña de TENERPOSESION de persona nueva de test clasificada erróneamente. [Autor]

Los resultados obtenidos para cada modelo se resumen a continuación.

• **Modelo Inicial:** El modelo inicial pudo reconocer 5 de 22 clases y entre ellas puede 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 es 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 es capaz de detectarla correctamente. Los desaciertos obtenidos los relacionó con diferentes 5 clases 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 puede 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 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. De los resultados obtenidos, el mejor modelo es el que tomó el modelo base y le agrego un *Dropout* al igual que 2 capas convolucionales mas. Al tenerse 15 de 22 clases, la eficiencia del prototipo de software es de un 68% lo cual es mejor a 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 interprete no recomienda aun 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, si puede el aprendizaje de maquina 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. Así mismo, 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.

Dado 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 se tenga una barrera de comunicación entre estas personas y los sistemas de información con ambientes de trabajo similares a los del INSOR.

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 con 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 full conectada o 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.

Técnicas como *Transfer Learning* y *Fine Tuning* son usadas hoy en día en las CNN como una alternativa para agilizar tiempos de entrenamiento al hacer uso de los pesos obtenidos en datos de entrenamiento de la misma naturaleza al problema que se quiere solucionar. Sin embargo, no siempre los resultados de *Accuracy* van a ser buenos como se evidencio en la construcción de los

diferentes modelos en este trabajo y que incluso entrenar un modelo sin pesos iniciales puede ayudar a construir un sistema más eficiente a pesar del tiempo de entrenamiento que conlleve.

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 demás 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 iluminación de las mismas, la luz solar del ambiente logra generar ruido e impactar en los resultados. Se requeriría contar con un espacio mucho más cerrado y controlado de los ofrecidos en sets de grabación para obtener una mejora considerable.

En este proyecto se usó una pequeña muestra de gestos y con los resultados obtenidos en el momento de pensar de manera general con muchas más señas se requerirá muchos recursos mas no solo a nivel computacional (procesamiento en paralelo de diversas maquinas) sino también de personal, factor que muchas veces es difícil conseguir por el desconocimiento de la LSC a nivel poblacional en general. Adicionalmente, la poca voluntad de las personas a ser partícipes de conocer esta lengua 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.

Con los avances computacionales del *Machine Learning* que se están presentando en los últimos años y con los nuevos desarrollos que se esperan por venir, no es descabellado llegar a pensar que los sistemas de cómputo puedan hacer tareas de reconocimiento de imágenes que el

cerebro de los seres humanos realiza en su día a día. Basados en esta aproximación, también es válido pensar en prototipos de Software que puedan incluir no solo un modelo en su lógica de negocio sino abarcar más tecnologías. Por ejemplo, se puede pensar en un modelo que sea capaz de transformar las imágenes en etiquetas y a su vez una tecnología de Procesamiento de Lenguaje Natural (NLP en inglés) tome estas etiquetas y les den un significado semántico que pueda ser entendible para las personas. Por ejemplo, las señas YO, PROFESOR y UNIVERSIDAD podrían revisarse y clasificarse en un contexto académico para expresar no solo la profesión de una persona sino su labor día a día y demás actividades que ello conlleva.

Finalmente se ve que este tipo de aprendizaje computacional no es ajeno a las tecnologías tradicionales que se manejan hoy en día en las organizaciones y pueden quedar más allá del ámbito investigativo. Frameworks como *FLASK* permite esta actividad de manera rápida y a su vez con la complejidad que un sistema de información requiera. El prototipo aquí implementado puede tener mejoras añadiendo una capa de persistencia con motores de bases de datos comerciales hoy en día para llevar registro de la información predicha o clasificada y con ese ánimo realizar la retroalimentación necesaria al sistema en el momento que se requiera. De igual manera, haciendo uso adecuado de procesamiento de imágenes y video en tiempo real un trabajo interesante a futuro es permitir que en tiempo real la persona pueda comunicarse y obtener el reconocimiento en línea, para lo cual no solo se requeriría trabajar con librerías especiales de video (como OpenCV) sino además un muy buen hardware que permita tener latencias muy bajas entre el paso del video a imagen, clasificación de la misma a etiqueta y respuesta al usuario instantáneamente.

6. Bibliografía

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, n.o 1, p. 1, oct. 2017.

2. A. G. B. Muñoz, «Yenny Milena Cortés Bello**2», vol. 26, p. 22.

3. «Servicio de Interpretación en línea SIEL». [En línea]. Disponible en: https://centroderelevo.gov.co/632/w3-propertyvalue-15254.html. [Accedido: 06-may-2019].

4. «Manual uso Centro de Relevo v5 (1).pdf», Google Docs. [En línea]. Disponible en: https://drive.google.com/file/d/1swrQp_skuDd_fBbVHI0Vu7EWwp4C9UZp/view?usp=sharing&usp=embed_facebook. [Accedido: 06-may-2019].

5. Neural Network Programming with Java. Fabio M. Soares, Alan M.F. Souza. Packt Publishing. 2016

6. Building Machine Learning and Deep Learning Models on Google Cloud Platform. Ekaba Bisong. Apress. Canada. 2019.

7. Programa de formación Machine Learning and Data Science MLDS, [En línea]. Disponible en: https://ingenieria.bogota.unal.edu.co/uec/?p=10947 [Accedido: 10-ago-2019].

8. 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, n.o 4, pp. 625-635, ago. 2017.

9. J. L. Raheja, A. Mishra, y A. Chaudhary, «Indian sign language recognition using SVM», Pattern Recognit. Image Anal., vol. 26, n.o 2, pp. 434-441, abr. 2016.

10. García Cortes Diana Carolina, Reconocimiento de Gestos de Manos como Mecanismo de Interacción Humano – Computador, 2014, Trabajo de Grado Maestría Ingeniería Sistemas y Computación Universidad Nacional de Colombia.

11. P. Nakjai y T. Katanyukul, «Hand Sign Recognition for Thai Finger Spelling: An Application of Convolution Neural Network», J. Signal Process. Syst., vol. 91, n.o 2, pp. 131-146, feb. 2019.

12. Ronchetti Franco, Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas, 2016, Trabajo de Grado Doctorado en Ciencias Informáticas Universidad Nacional de la Plata, Argentina.

13. O. Koller & R. Bowden. «Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled», RWTH Aachen University, German & University of Suervy UK, 2016.

14. Mustafa, Mohammed. «A Study on Arabic Sign Language Recognition for Differently Abled Using Advanced Machine Learning Classifiers». Journal of Ambient Intelligence and Humanized Computing, mar. 2020.

15. Mishra, Shubham Kr., Sheona Sinha, Sourabh Sinha, y Saurabh Bilgaiyan. «Recognition of Hand Gestures and Conversion of Voice for Better-ment of Deaf and Mute People». En Advances in Computing and Data Sciences, editado por Mayank Singh, P.K. Gupta, Vipin Tyagi, Jan Flusser, Tuncer Ören, y Rekha Kashyap, 1046:46-57. Singapore: Springer Singapore, 2019.

16. Ferreira, Pedro M., Jaime S. Cardoso, y Ana Rebelo. «On the Role of Multimodal Learning in the Recognition of Sign Language». Multimedia Tools and Applications 78, n.o 8, abr. 2019.

17. Islam, Md. Sanzidul, Sadia Sultana Sharmin Mousumi, AKM Shahariar Azad Rabby, y Syed Akhter Hossain. «A Simple and Mighty Arrow-head Detection Technique of Bangla Sign Lan-guage Characters with CNN». En Recent Trends in Image Processing and Pattern Recognition, editado por K. C. Santosh y Ravindra S. Hegadi, 429-37. Communications in Computer and In-formation Science. Singapore: Springer, 2019.

18. Practical Machine Learning and Image Processing. Himanshu Signgh. Apress. India. 2019.

19. Sign Deaf Recognition in Kaggle. [En línea]. Disponible en: https://www.kaggle.com/datamunge/sign-language-mnist [Accedido: 15-abr-2019].

20. [En línea]. Disponible en: http://educativo.insor.gov.co/diccionario/diccionario-cotidiano/ [Accedido: 20-oct-2019].

21. A Gentle Introduction to k-fold Cross-Validation [En línea]. Disponible en: https://machinelearningmastery.com/k-fold-cross-validation/ [Accedido: 19-sep-2019].

22. "Planos Arquitectónicos: El modelo de 4 + 1 Vistas de la Arquitectura de Software", Philippe Kruchten. IEEE Marzo 2006. [En línea]. Disponible en: http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf [Accedido: 15-nov-2019].

Anexo A

En el siguiente repositorio de *GitHub* se encuentra el material usado para construir los modelos, así como las simulaciones realizadas en *Anaconda*, el software prototipo implementado en *FLASK*, los datos de prueba del prototipo y los resultados.

https://github.com/nmortizf/TrabajoMestriaDev.git

En la siguiente URL, se encuentra un video explicando cómo inicializar el prototipo de software y su uso con las diferentes imágenes usadas para la prueba de concepto, así como una breve explicación de la estructura y funcionamiento del proyecto:

https://youtu.be/SMHFznqLuS8

Por último y no menos importante, en el CD adjunto a este trabajo de grado se encuentra las contribuciones realizadas que se mencionaron al comienzo incluyendo el artículo, los modelos computacionales, el prototipo de software y el repositorio de imágenes.