Universidad Nacional de San Juan

Facultad de Ciencias Exactas, Físicas y Naturales, Departamento de Informática

Desarrollo de un Sistema Web intérprete de la Lengua de Señas Argentina

Tesis presentada para obtener el grado en: Licenciatura en Ciencias de la Computación

Autor: Facundo Nicolas Recabarren Martin - 41253120

Asesora: Mg. María Isabel Masanet

Asesor: Lic. Fabrizio Amaya

Co-asesor: Prog. Gustavo Emilio Conturzo

San Juan

2024

Resumen

Se propone el desarrollo de un sistema web que permita interpretar un conjunto de señas pertenecientes a la Lengua de señas Argentina (LSA), ya sean estas realizadas en tiempo real o mediante un video pregrabado. Encargándose el sistema web de encontrar la correspondencia entre una secuencia de imágenes (video) y su apropiado texto en español.

Para esto se plantea un modelo de árboles de decisión optimizado y dos modelos de Redes Neuronales Recurrentes, utilizando un conjunto de datos preexistentes sobre el que se realizan tareas de preprocesamiento, muestreo y almacenamiento para entrenar y optimizar estos modelos.

Luego se realiza el planteo, definición, construcción del sistema web e integración del modelo seleccionado.

Palabras clave: Lengua de Señas Argentina, LSA, Machine Learning, Data Science, MediaPipe, XGBoost, RNN, LSTM, GRU.

Índice

1. Motivación

La presente investigación surge de una primera aproximación del estudiante, autor de este trabajo, hacia la accesibilidad en páginas web donde se pretendía que el usuario pudiera desplazarse a través del sitio o aplicación mediante el uso de gestos realizados por una mano frente a la cámara del dispositivo en uso.

Teniendo como base el movimiento o gesto de las manos se propuso otro tipo de accesibilidad, ya no en páginas web, la cual cuenta con un amplio espectro de herramientas para su uso, sino en el desarrollo de una aplicación que permita la comunicación entre personas que usan la Lengua de Señas Argentina como lengua de comunicación.

Con el desarrollo de este sistema y haciendo uso de los avances tecnológicos y científicos de los últimos años, se pretende facilitar la comunicación e interacción entre ambas comunidades haciendo uso de técnicas de Machine Learning.

2. Introducción y Contexto

Las lenguas de señas son lenguas que utilizan la modalidad visual-gestual para transmitir significado a través de articulaciones manuales en combinación con elementos no manuales como el rostro y el cuerpo. Sirven como el principal medio de comunicación para numerosas personas sordas y con problemas de audición. Al igual que las lenguas orales, las lenguas de señas son lenguas naturales regidas por un conjunto de reglas lingüísticas y a diferencia de las lenguas orales, la Lengua de Señas de un idioma en particular, no es una forma visual de ese idioma, sino que es una lengua propia.

El procesamiento del lenguaje de señas es un campo emergente de la inteligencia artificial, centrándose su investigación en los aspectos visuales de estos lenguajes, cubriendo un subcampo tanto del procesamiento de lenguaje natural (NLP) como de la visión por computadora (CV). [1]

La Comunidad Sorda es una comunidad minoritaria que poco a poco ha logrado integrarse en las comunidades de todo el mundo. Si bien en algunos países su integración está muy avanzada, en nuestro país aún queda mucho por hacer para lograrlo.

Tal es el caso de Estados Unidos, donde existe la Universidad de Gallaudet, única universidad en el mundo destinada a la educación avanzada para personas sordas y con problemas de audición.

La provincia de San Juan cuenta con dos escuelas para niños sordos: la Escuela José A. Terry (oralista) y la Escuela Bilingüe para Sordos donde se utiliza la Lengua de Señas y la escritura.

La Comunidad Sorda de la provincia de San Juan, conjuntamente con familiares, estudiantes de Lengua de Señas Argentina (LSA) y personas interesadas en apoyar y colaborar con dicha comunidad, iniciaron un proyecto de Ley en la primera década del año 2000, que fue aprobado como Ley Provincial N° 7412 [2] en el año 2004.

Esta ley reconoce en todo el ámbito de la provincia de San Juan, la Lengua de Señas Argentina y el lenguaje oral utilizado por la Comunidad Sorda e Hipoacúsica, intentando establecer disposiciones tendientes a remover barreras comunicacionales en distintos ámbitos de la provincia.

Si bien en los años subsiguientes la ley no fue implementada, en el año 2014 se retoma este proyecto y actualmente se encuentra vigente como Ley Provincial N° 761-S [3]. No se ha logrado mucho en la provincia de San Juan. Es por ello que esta propuesta de trabajo final tiene como objetivo reforzar, cumplir e implementar lo dispuesto en el Artículo 5, inciso ‘a’:

“Estar provistos de sonorización, avisos, información visual y sistemas luminosos de alarma o de emergencia para el reconocimiento por parte de personas con discapacidad auditiva.”

Lo que innovaría lo propuesto en esta ley que fue redactada en el 2004.

En la provincia no se cuenta con la carrera de Intérpretes de Lengua de Señas, pero sí se dictan cursos en las Facultades de Filosofía e Ingeniería de la Universidad Nacional de San Juan, en la Municipalidad de Rawson y Capital, y en forma particular a cargo de grupos de instructores sordos.

El objetivo que persigue la comunidad sorda/hipoacúsica es lograr que poco a poco la gente aprenda la LSA para poder vencer las imposibilidades comunicacionales entre personas sordas y oyentes.

Debido a que la lengua de señas se entiende por el sentido de la vista (contexto "visual"), las personas sordas hacen un gran uso de la tecnología a través de videollamadas, mensajes de texto, teléfonos móviles, notebooks y todo aquello que les permite comunicarse y darse a entender.

San Juan carece de los medios y tecnologías adecuadas para ellos, y no cuenta con intérpretes de LSA en casi ningún ámbito, ya sea hospitales, bancos, edificios públicos, incluso tampoco los hay en escuelas, universidades y demás.

"El principal problema radica en el abordaje que se le da al niño sordo y al entorno en el cual pueda desenvolverse de manera favorable. Actualmente, se entiende por "inclusión" como poner a niños sordos en colegios de oyentes sin pensar estrategias específicas para ellos. Esa "inclusión" no implica "ponerlos" en un lugar determinado, sino que debe existir una experiencia de participación en la enseñanza/aprendizaje en su lengua… Existen investigaciones que demuestran que los niños sordos que reciben educación de calidad desde un abordaje multilingüe (por ejemplo, en una lengua de señas y una lengua escrita/oral) tienen mayores probabilidades de progresar académicamente y de convertirse en ciudadanos activos, que participan de forma activa y plena en la vida social." [4].

Los proyectos y trabajos que fueron leídos aportan un enfoque para poder realizar la interpretación de señas, ya sean estas estáticas o en movimiento, pero al existir una gran

cantidad de Lengua de Señas que se hablan en el mundo, un aproximado de 270 según se informa en [5], los datos y sistemas no son aplicables en la provincia de San Juan.

Aunado a lo nombrado, cada región de la Argentina ha desarrollado su propia lengua, debido a esto se propone este trabajo con el objetivo de desarrollar un sistema que interprete la Lengua de Señas Argentina y que pueda ser usado por las personas que utilizan la lengua de señas en la Provincia de San Juan.

2.1 Planteamiento del Problema

El problema que se propone resolver consiste en interpretar una secuencia de imágenes y que corresponda a señas provenientes de la Lengua de Señas Argentina.

2.2 Solución Propuesta

La solución propuesta al problema descrito consiste en diseñar y entrenar modelos de redes neuronales que puedan interpretar un conjunto de datos como una seña perteneciente a la Lengua de Señas Argentina. Luego implementar un sistema web que, mediante el acceso a la cámara del dispositivo permita la captura de imágenes en tiempo real (video) y realice la interpretación de la seña realizada mediante los modelos construidos.

2.3 Objetivos

2.3.1 Objetivo General

- Desarrollar un sistema web capaz de interpretar una secuencia de imágenes que contienen a personas realizando movimientos pertenecientes a la Lengua de Señas Argentina.

2.3.2 Objetivos Específicos

- Explorar e Investigar arquitecturas de Redes Neuronales Recurrentes (RNN) para el procesamiento de secuencias.
- Explorar e Investigar sobre Árboles de Decisión para el procesamiento de secuencias.

- Diseñar y Entrenar modelos de redes neuronales capaces de interpretar una secuencia de imágenes en tiempo real e informar la seña realizada.
- Optimizar los modelos entrenados.
- Comparar el/los modelos construidos con RNN y Árboles de Decisión.
- Prototipar e Implementar la interfaz gráfica de un sistema web.
- Implementar una API backend para realizar la interpretación.

3. Marco Conceptual

En el presente trabajo de investigación se espera que el sistema desarrollado permita facilitar la comunicación e interacción entre personas oyentes y personas que utilizan la Lengua de Señas Argentina.

A continuación, se presentan los conceptos que dan soporte al proceso de desarrollo del sistema. Para ello se tratarán técnicas de Machine Learning (ML) o Aprendizaje Automático como lo son las Redes Neuronales y su optimización, Visión Artificial (Computer Vision) y Lengua de Señas Argentina.

3.1 Machine Learning

Dentro del área de la Inteligencia Artificial (IA) se desprende una rama específica conocida como Machine Learning, Aprendizaje Automático o Aprendizaje de Máquina.

El ML es una rama de algoritmos computacionales diseñados para emular la inteligencia humana mediante el aprendizaje del entorno [6].

“El Aprendizaje Automático consiste en programar ordenadores para que optimicen un criterio de rendimiento a partir de datos de ejemplo o de experiencias pasadas. Tenemos un modelo definido hasta unos parámetros, y el aprendizaje es la ejecución de un programa informático para optimizar los parámetros del modelo utilizando datos de entrenamiento o

experiencias pasadas. El modelo puede ser predictivo para hacer pronósticos en el futuro, o descriptivo para obtener conocimientos de los datos, o ambas cosas" [7].

El presente desarrollo se ha basado en el análisis de series temporales, específicamente secuencias de imágenes, y la clasificación de esa secuencia como una palabra perteneciente a un conjunto de palabras aprendidas.

El Aprendizaje Automático puede clasificarse en 3 categorías:

- Aprendizaje Supervisado
- Aprendizaje No Supervisado
- Aprendizaje por Refuerzo

3.1.1 Aprendizaje Supervisado

"En el aprendizaje supervisado, el conjunto de entrenamiento consiste en pares de entrada y salida deseada, y el objetivo es aprender una correspondencia entre los espacios de entrada y salida" [8].

El tipo de aprendizaje que se utiliza en el presente trabajo es el Aprendizaje Supervisado ya que se conoce la clase o palabra a la que corresponde cada video o secuencia de imágenes.

3.1.2 Aprendizaje No Supervisado

"En el aprendizaje no supervisado, el conjunto de entrenamiento consiste en entradas no etiquetadas, es decir, entradas sin ninguna salida deseada asignada" [8].

Al no tener disponible la salida esperada para los datos de entrenamiento el aprendizaje que realiza la máquina es a través del análisis de la estructura de los datos que no se encuentran etiquetados.

Este tipo de aprendizaje sirve para extraer características o patrones similares entre los datos, permitiendo formar grupos o clusters.

3.1.3 Aprendizaje por Refuerzo

"El aprendizaje por refuerzo se sitúa, en cierto sentido, entre el aprendizaje supervisado y el no supervisado. A diferencia del aprendizaje no supervisado, existe algún tipo de supervisión, pero no en forma de especificación de una salida deseada para cada entrada de datos. En cambio, un algoritmo de aprendizaje por refuerzo recibe información del entorno sólo después de seleccionar una salida para una entrada u observación dada. La retroalimentación indica el grado en que la salida, conocida como acción en el aprendizaje por refuerzo, cumple los objetivos del alumno. El aprendizaje por refuerzo se aplica a problemas de toma de decisiones secuenciales en los que el alumno interactúa con un entorno mediante acciones secuenciales - las salidas -en función de sus observaciones - sus entradas - mientras recibe retroalimentación sobre cada acción seleccionada." [8].

3.2 Redes Neuronales

Las redes neuronales son potentes modelos de aprendizaje que logran resultados de vanguardia en una amplia gama de tareas de aprendizaje automático (ML) supervisado y no supervisado [9].

Las Redes Neuronales están motivadas en modelar la forma de procesamiento de la información en sistemas nerviosos biológicos. Una red neuronal es un procesador de información, de distribución altamente paralela, constituido por muchas unidades sencillas de procesamiento llamadas neuronas [9].

3.2.1 Perceptrón

En un principio con el comienzo de las Redes Neuronales en los años 50, se introdujo el modelo del Perceptrón, nacido como un sistema capaz de realizar clasificaciones. El Perceptrón representa una estructura monocapa, compuesta por un conjunto de líneas de entrada y una o varias salidas [11](página 26).

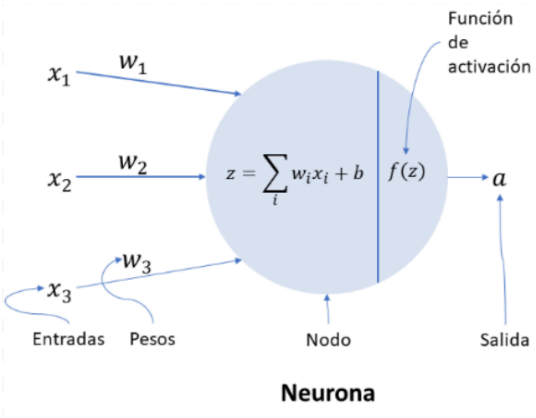


Figura 1: Neurona con salida previa 'z' que atraviesa por el cálculo de la función de activación f(z). Imagen: Jahaziel Ponce (10 Julio 2021).

La figura 1 puede interpretarse como un Perceptrón, con líneas de entrada {X1, X2, X3}, pesos {W1, W2, W3}, salida 'α' y valor de sesgo o bias 'b' un valor constante. En esta neurona la salida es hallada como la suma ponderada de los pesos de todas las entradas:

$$z = \sum_{i=1}^{3} w_i * x_i$$

La salida definitiva es aquella que se obtiene al calcular el valor obtenido 'z' a través de la función de activación f(z).

La función de activación en el perceptrón es la Función Escalón o Step Function (Figura 2) cuya representación gráfica se muestra en la Figura 3.

$$\alpha = f(z, b) = \begin{cases} 1 \text{ si } z >= b \\ 0 \text{ en caso contrario} \end{cases}$$

Figura 2: Función de activación Escalonada.

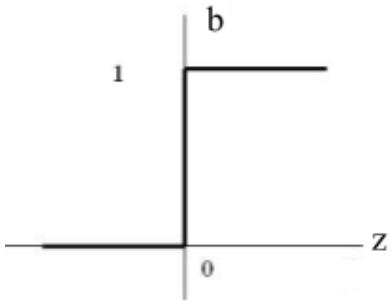


Figura 3: Gráfica de la *función de Activación escalonada. (Autor: Víctor Julio S. Moreno. Simulador de Redes Neuronales Artificiales)*

Esto puede escribirse como:

$$\alpha = f\left(\sum_i w_i * x_i + b\right)$$

La salida de esta función, como puede observarse, es una salida binaria, lo que nos da paso a comentar la limitación del perceptrón simple. La limitación consiste en que este permite realizar clasificaciones binarias, por lo que en los casos donde se necesiten realizar clasificaciones en más de 2 clases no es posible realizarlas con un perceptrón simple.

Para atacar esta limitación se puede utilizar una combinación de neuronas para lograr estructuras más complejas y así mejores predicciones, esta combinación de neuronas, estructuradas en capas, es lo que se conoce como Perceptrón Multicapa y permite resolver problemas que no son linealmente separables (limitación nombrada anteriormente).

3.2.2 Arquitectura de una Red Neuronal

La arquitectura de una Red Neuronal refiere a su estructura, es decir, la cantidad de capas que posee, la cantidad de unidades o neuronas que conforman cada una de sus capas y las conexiones que existen entre ellas (Figura 4).

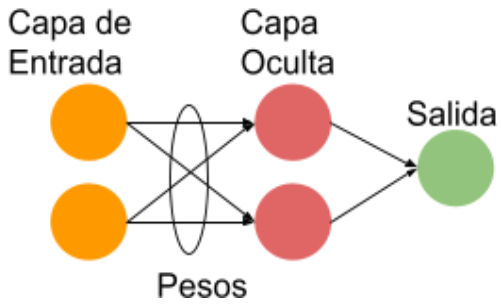


Figura 4: Arquitectura de una Red Neuronal básica.

La arquitectura de una Red Neuronal no solo indica las capas, unidades por capa y conexiones entre capas, sino que también refiere a la forma en la que son entrenadas y al tipo de neuronas usadas.

3.2.3 Función de Activación

Las funciones de activación sirven para limitar la salida de las neuronas a un cierto intervalo, es decir, la salida de cada neurona es evaluada sobre una función de activación.

La forma en la que la salida cambia según las entradas depende de qué función de activación se esté usando y según la función seleccionada el proceso de aprendizaje y performance variará.

Existe un amplio número de funciones de activación, entre las más comunes se encuentran las siguientes:

- Función Sigmoidea
- Tangente Hiperbólica (tanh)
- Relu
- Softmax

3.2.3.1 Función Sigmoidea.

Similar a la Función Escalón observada previamente, la Función Sigmoidea acota los resultados entre 0 y 1, pero con una curva suave, permitiendo el cálculo de derivadas diferente a 0 lo cual no es posible de hallar en la Función Escalón. Al tener siempre una salida de

valores positivos en el intervalo de [0, 1] en ocasiones limita la velocidad de aprendizaje al no contar con valores negativos.

La ecuación que representa a esta función es la siguiente:

$$sig\ (x)\ =\ \frac{1}{1+e^{-x}}$$

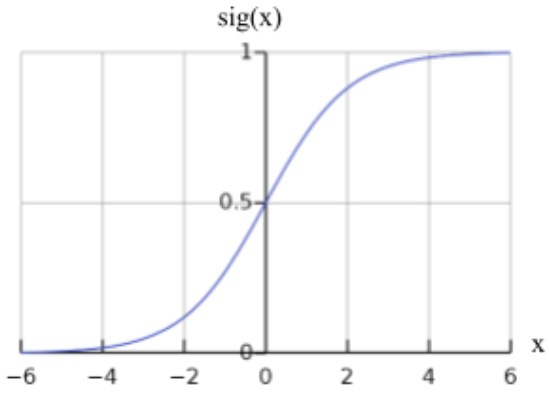


Figura 5: Gráfica de la función Sigmoidea.

3.2.3.2 Tangente Hiperbólica (Tanh).

Continuando con la siguiente función, se tiene a la función Tangente Hiperbólica, esta acota los resultados en un intervalo [-1, 1], permitiendo la existencia de resultados negativos.

Se encuentra definida por la ecuación:

$$tanh\ (x)\ =\ \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

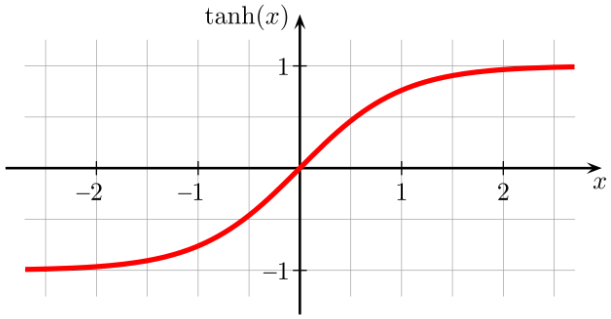


Figura 6: Gráfica de la función tanh. © 2008 Usuario Geek3, Wikipedia.

Un problema general que ocurre con la función Sigmoidea y Tangente Hiperbólica es el "Desvanecimiento del Gradiente", como puede observarse en las Figuras 5 y 6 correspondientes a Sigm. y Tanh, en sus valores extremos la gráfica es prácticamente horizontal, lo que genera valores para las derivadas muy pequeños, que luego en procesos de retropropagación harán más lento el aprendizaje incluso siendo capaz de no aprender en las primeras capas.

3.2.3.3 ReLU.

La Función ReLU o Unidad Lineal Rectificada (por sus siglas en Inglés Rectified Linear Unit).

$$relu(x) = max(0,\ x)$$

Como se puede observar, la función relu(x) es un cálculo sencillo que retorna directamente el valor 'x' entregado a la función como entrada (no está acotada para valores positivos) o retorna el valor 0 si la entrada 'x' es 0 o menor. Debido a esto para valores mayores a 0 tenemos una función lineal, no siendo este el mismo caso para valores de entrada menores a 0.

En comparación con las funciones de activación previas, esta no necesita realizar cálculos complejos como las exponenciaciones que tienen un alto coste computacional, siendo así más veloz.

Además, como se observa en la Figura 7, la derivada para valores positivos es 1 y 0 para valores negativos, siendo un cálculo sencillo. El inconveniente que se halla aquí es que el resultado de 0 para valores negativos puede generar "neuronas muertas" entorpeciendo así el aprendizaje. Han surgido otras funciones de activación como Leaky ReLU, PReLU o GELU para mejorar y atacar el problema de "neuronas muertas" [12].

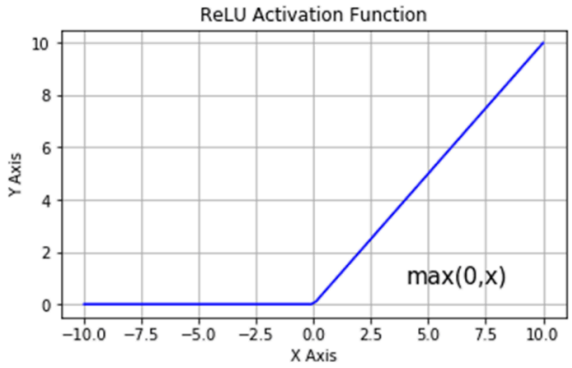


Figura 7: Gráfica función de activación ReLU. (Autor: Satya Mallick).

3.2.3.4 Softmax.

El objetivo de la función de activación Softmax consiste en retornar un conjunto de probabilidades de pertenencia a un conjunto de clases tal que la suma de estas probabilidades es 1. Matemáticamente expresada como:

$$softmax\ (x)_j = \frac{e^{x_j}}{\sum_{K=1}^{K} \left(e^{x_j}\right)}$$

En el presente trabajo esta función será de ayuda en la última capa de la red para tener un vector de probabilidades, donde la componente con el mayor valor será la clase a la que pertenece la seña interpretada.

En la Figura 8 puede interpretarse visualmente su uso:

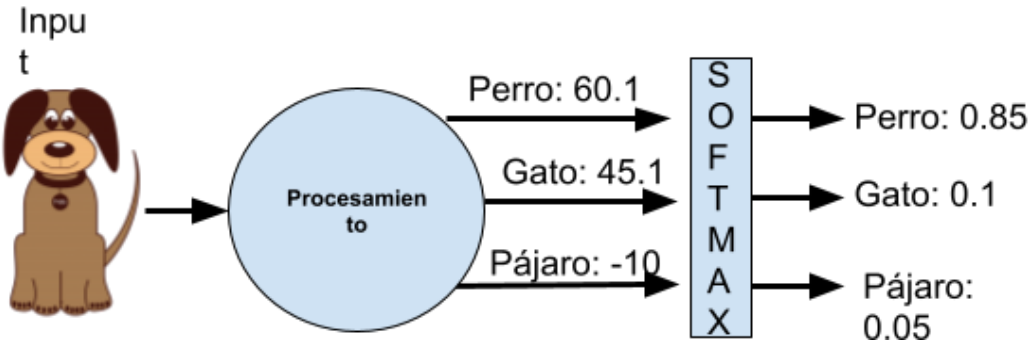


Figura 8: Ejemplo de entrada y salida de la función Softmax.

3.2.4 Aprendizaje

Como se comenta en [11], "La regla o algoritmo de aprendizaje es el mecanismo mediante el cual se van adaptando y modificando todos los parámetros de la red.". En aquellos casos donde el aprendizaje es supervisado la modificación efectuada sobre los parámetros se realiza para que la salida de la red sea lo más próxima a la salida esperada.

3.2.5 Función de Costo

Al tener como objetivo el acercamiento o aproximación entre los valores obtenidos y las salidas deseadas, se puede formular un problema de minimización:

$$Min_w E$$

Aquí *E* representa una función de error que evalúa la diferencia entre las salidas de la red y las salidas deseadas, mientras que W el conjunto de pesos y umbrales.

$$E = \frac{1}{N} \sum_{n=1}^{N} e(n)$$

N indica la cantidad de muestras y e(n) el error cometido por la red para el patrón n, siendo e(n):

$$e(n) = \frac{1}{2} \sum_{i=1}^{n_c} (s_i(n) - y_i(n))^2$$

Donde s_i e y_i son componentes pertenecientes a los vectores S(n) e Y(n) respectivamente, ambos vectores de las salidas de la red esperadas y obtenidas.

El error a hallar en *E* es un error total, pero existen otros procedimientos como lo es el "Descenso del Gradiente" el cual consiste en una sucesiva minimización de los errores para cada patrón, e(n), en lugar de minimizar el error total.

Cabe aclarar que el proceso de aprendizaje se realiza a través de sucesivas iteraciones y no es un paso que se realice una única vez, se parte de una configuración inicial para W

(conjunto de pesos y umbrales) -posiblemente valores aleatorios- y a medida que se presentan las salidas estos irán ajustándose según el error mínimo hallado.

Al alejarse las salidas obtenidas de las esperadas, mayor será la pérdida. Este promedio hallado en (2.2.1) es a lo que llamamos función de Costo, y estrictamente, esta función de costo es la que tratamos de minimizar.

3.2.6 Entropía Cruzada o Cross Entropy

La Entropía Cruzada o Cross-Entropy (CE) es una función de Costo que permite hallar cuantitativamente qué tan adecuados son los parámetros W de la red.

CE es una medida de la diferencia entre 2 distribuciones de probabilidad P y Q para una determinada variable aleatoria o conjunto de sucesos [13].

Se entiende por P a la distribución de probabilidad que siguen los datos de entrenamiento x_i de los cuales conocemos las salidas y_i, y Q la distribución de probabilidad que la red entrega $\hat{y}_i$ para estos datos x_i, tal que:

$$CE = \; - \sum_{i}^{C} y_i \; * \; log\,(\hat{y}_i)$$

Donde $P(x_i) = y_i$, $Q(x_i) = \hat{y}_i$ y C el número de clases a los que se pueden clasificar las salidas.

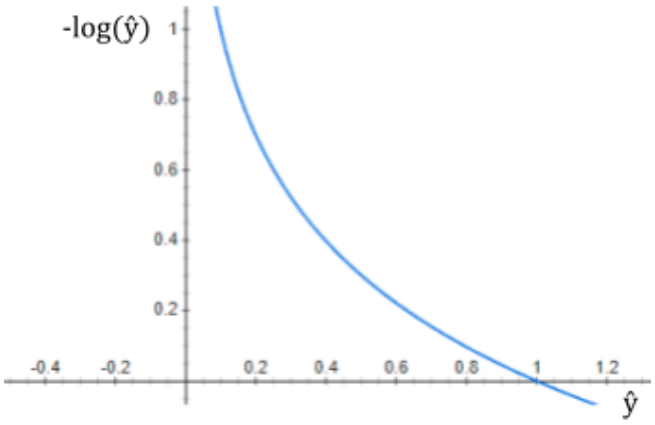


Figura 9: Gráfica de la función -log(x).

Como se observa en la Figura 9, aquellos valores para los que la probabilidad $\hat{y}_i$ o $Q(x_i)$ es cercano a 1, el error $-log(\hat{y}_i)$ será un valor cercano a 0, mientras que una probabilidad alejada del valor 1 entregará mayor valor de error.

Al trabajar con problemas de clasificación, las entradas de los datos de entrenamiento están asociadas con probabilidad de 1 a la clase correspondiente de salida (en aprendizaje supervisado), por ejemplo, para una clasificación en 3 clases se dispone de vectores de la forma [1, 0, 0], [0, 1, 0], [0, 0, 1] para las clases 'A', 'B'. 'C' correspondientemente, conocidos como "one-hot-encoding".

El problema que se trata en el presente trabajo es un problema de clasificación multiclase, por lo que se pretende hacer uso de la Entropía Cruzada.

3.2.7 Descenso del Gradiente

Como se nombró previamente no trata de minimizar el error global, sino que este puede entenderse como un optimizador local en un espacio de búsqueda continuo.

"El objetivo del aprendizaje, es decir del proceso de actualización de pesos, consiste en encontrar la configuración de los mismos que corresponde al mínimo global de la función de error o función de coste definida" [16].

Una forma de visualizar este proceso se encuentra en la Figura 10, donde se visualiza una hipersuperficie que representa a la función de coste f(x). Partiendo de un punto aleatorio indicado por el valor de los pesos, se obtiene la pendiente en la posición actual con el objetivo de encontrar la mayor pendiente, que será la dirección que se tomará para realizar el descenso, este proceso se realiza de manera iterativa hasta haber hallado el mínimo local y/o en ciertos casos un mínimo global.

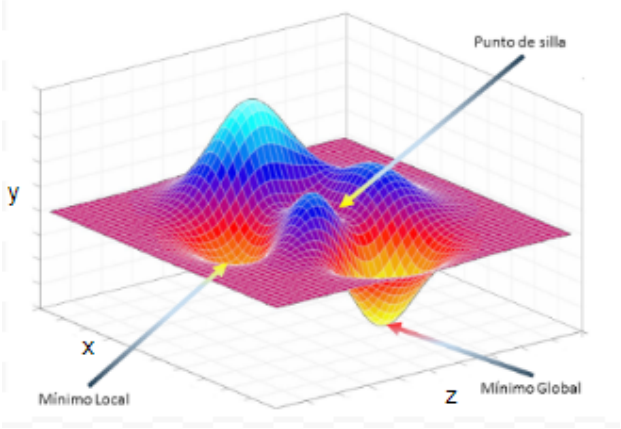


Figura 10: Problema del descenso del gradiente. (Autor: Numeruntur.org)

Desde el punto de vista matemático la búsqueda de la pendiente se hace a través del uso de las derivadas parciales para cada punto. Como los parámetros de la red pueden variar en cantidad ya que $f\ (x)$: $R^n \rightarrow R$, en la Figura 10 se puede identificar una función definida en $f\ (x)$: R^3 , se deberán hallar las derivadas parciales para X y Z, este conjunto de derivadas parciales componen un vector ascendente por lo que se invierte su signo positivo para tomar el sentido contrario y así lograr descender.

$$x_{t+1} = x_t - \eta * f'(x)$$

En la ecuación se indica que el valor de 'x' en la iteración t+1 estará dado por el valor de 'x' en la iteración 't' menos la derivada de la función de coste en el punto 'x' por 'η', donde 'η' representa el tamaño del paso que se tomará para descender, este parámetro es identificado como "Learning Rate", "Ratio de Aprendizaje" o porcentaje de cambio con el que se actualizan los pesos en cada iteración. La elección del valor de Ratio de Aprendizaje debe ser cuidadosa ya que, dependiendo de ésta, el aprendizaje puede ser demasiado lento para valores bajos o estancarnos en un mínimo local ineficiente mientras que si el valor es demasiado grande puede dificultar la convergencia del método dificultando caer en zonas de mínimos y/o generando bucles infinitos.

Los inconvenientes que se encuentran al usar el Descenso del Gradiente pueden ser la obtención de puntos mínimos locales y que aún existan otros mínimos mejores, también es posible encontrarse puntos sillas, como el que se observa en la Figura 10, que se encuentran ubicados entre 2 puntos mínimos, ¿cuál será de los 2 el mejor camino?.

3.2.8 Retropropagación o Backpropagation

Una vez que se obtuvieron las salidas en una primera fase, comienza la fase de corrección, de propagación hacia atrás o de Backpropagation.

Como se introdujo en un principio, las Redes Neuronales están compuestas por distintas capas y cada capa a su vez está compuesta por cierto número de neuronas, cada una con sus pesos, parámetros, funciones de activación, etc.

El algoritmo de Retropropagación actúa del siguiente modo, descrito en [11]: "Cada neurona de salida distribuye hacia atrás su error a todas las neuronas ocultas que se conectan a ella, ponderado por el valor de la conexión. De este modo, cada neurona oculta recibe una proporción del error de cada neurona de salida, y la suma de estas cantidades es el término **δ** asociado a la neurona oculta. Dichos valores permiten, a su vez, obtener los valores **δ** de las neuronas ocultas de la capa anterior y así sucesivamente hasta llegar a la primera capa oculta.". Es decir el error de las neuronas ubicadas en una capa C, dependen del error de las neuronas en la capa C + 1.

Se utiliza el Descenso del Gradiente para optimizar la función de coste haciendo uso de la Retropropagación. El proceso matemático puede hallarse desarrollado en [11] (página 54), en resumen, éste consiste en aplicar la Regla de la Cadena para el cálculo de las derivadas parciales para cada capa y neurona.

En aquellos casos donde la red cuenta con cientos o miles de parámetros, puede aplicarse una alternativa al Descenso del Gradiente para evitar calcular el error sobre todo el conjunto de datos, esta alternativa es conocida como Descenso del Gradiente Estocástico y su

objetivo es hallar el error sobre subconjuntos de los datos de entrenamiento, aumentando la velocidad en el proceso de corrección.

3.2.9 Matriz de Confusión

Una Matriz de Confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Esta contiene información sobre las clasificaciones realizadas y las salidas previstas o esperadas por el modelo o red de clasificación. Los valores presentes en una matriz de confusión permiten el cálculo y evaluación de ciertas métricas como lo pueden ser: Precisión, Recall, F1 Score, ROC – AUC y Accuracy.

		Valor Predicho	
		Positivo	Negativo
Valor Real	Positivo	Verdadero Positivo	Falso Negativo
	Negativo	Falso Positivo	Verdadero Negativo

Figura 11: Matriz de Confusión sobre 2 clases

En la Figura 11 las filas de la matriz indican la clase a la que realmente pertenece cada salida mientras que cada columna indica la clase predicha por el modelo, por ello es que se puede ver en la diagonal principal la cantidad de aciertos por clase.

- Verdadero Positivo (VP): Ejemplificando con las clases de la Figura 11: se predijo positivo y el valor real es positivo.
- Falso Negativo (FN): El valor real es positivo y el modelo predice un resultado negativo.
- Falso Positivo (FP): El valor real es negativo, pero se predijo como positivo.
- Verdadero Negativo (VN): El valor real es negativo y se obtuvo como resultado un negativo.

3.2.10 Métricas de Evaluación de Modelo

3.2.10.1 Accuracy.

Exactitud o Accuracy en inglés, refiere al número de predicciones correctas realizadas sobre el número total de registros. No es una buena medida cuando se tienen datos desequilibrados, es decir, mayor porcentaje de datos de una clase sobre las demás. Por ejemplo, al tener un total de 100 datos de los cuales 5 son Positivos y 95 son Negativos, es posible que todos los casos negativos sean predichos de manera correcta pero no así los de la clase positiva. En este caso el modelo tiene un Accuracy del 95%, pero no es un buen clasificador.

$$Accuracy = \frac{VP+VN}{VP+FN+FP+VN}$$

Esta métrica es un parámetro global del modelo, mientras que las métricas siguientes son calculadas para cada clase.

3.2.10.2 Precisión.

Se representa por la proporción de Verdaderos Positivos dividido en todos los resultados positivos, ya sean Verdaderos Positivos o Falsos Positivos.

$$Precisión = \frac{VP}{VP+FP}$$

3.2.10.3 Recall.

Sensibilidad o Recall en inglés, se calcula como el número de predicciones positivas correctas (VP) dividido por el número total de positivos. Es la proporción de resultados positivos identificados correctamente.

$$Recall = \frac{VP}{VP+FN}$$

3.2.10.4 F1 Score.

Esta métrica reúne los valores de Precisión y Sensibilidad en una sola métrica. Es el promedio ponderado o media armónica de ambas métricas. Esta es útil si se tienen datos

desequilibrados y facilita la comparación del rendimiento entre modelos combinando Precisión y Sensibilidad.

$$F1 = \frac{2*Precisión*Sensibilidad}{Precisión+Sensibilidad}$$

3.2.10.5 ROC - AUC.

En casos de conjuntos de datos desequilibrados, las métricas como ROC son determinantes más efectivos. Este se basa en curvas ROC y el área bajo la curva para AUC (Area Under the Curve). Informa que tan bien puede distinguir el modelo entre dos cosas, es por esto que se realiza sobre clasificaciones binarias, en aquellos casos donde se tiene un problema de clasificación multiclase se pueden utilizar ciertos métodos como OvR (One vs Rest) o OvO (One vs One) donde se compara cada clase con las demás, obteniendo una comparación binaria.

Las curvas ROC son una herramienta estadística utilizada en clasificaciones binarias para evaluar la capacidad discriminativa de una prueba dicotómica. Estas curvas representan la sensibilidad en función de los falsos positivos y para construirlas es necesario calcular la sensibilidad (Recall) y la especificidad [14].

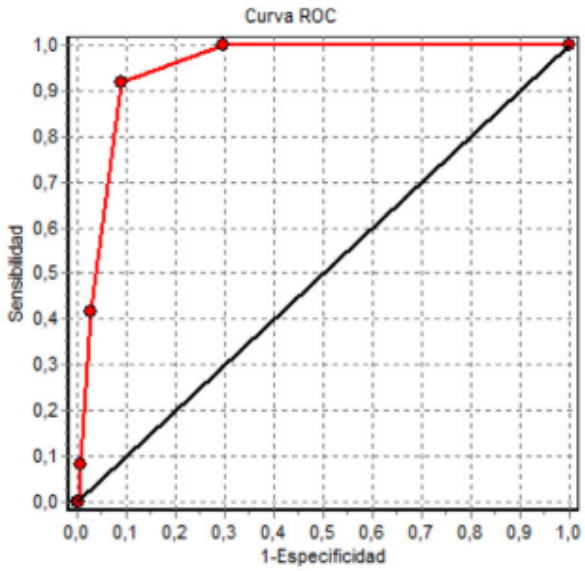


Figura 12: Ejemplo de Curva ROC. [14]

3.3 Redes Neuronales Recurrentes

Las redes neuronales estándar tienen limitaciones. La más notable es que se basan en la suposición de independencia entre los datos de entrenamiento y los datos de prueba. Después de cada ejemplo o dato, se pierde todo el estado de la red [8].

Las RNN son conocidas por su capacidad para procesar y obtener información de datos secuenciales. Por lo tanto, el análisis de vídeo, la subtitulación de imágenes, el procesamiento del lenguaje natural (PLN) y el análisis de la música dependen de las capacidades de las redes neuronales recurrentes. A diferencia de las redes neuronales artificiales estándar, que asumen la independencia entre los datos de entrada, las RNN capturan activamente sus dependencias secuenciales y temporales [1].

Las RNN cuentan con un mecanismo a través del cual regulan sus valores con el objetivo de realizar el aprendizaje (BTT backpropagation through time), el problema que presentan las RNN al momento de aprender es el de la explosión y desvanecimiento del gradiente (gradient vanishing) asociado con el aprendizaje de dependencias de largo alcance [17]. Debido a la existencia del problema nombrado surgen las Redes de Memoria Corta y Larga o Long-Short Term Memory (LSTM) y las redes Gated Recurrent Units (GRU), ambas tienen la capacidad de mantener y/o borrar la información que se crea necesaria.

En pocas palabras, las RNN permiten la persistencia de cierta información de tal manera que esta puede ser usada en etapas siguientes, simulando la memorización, esto a través de la retroalimentación de las salidas hacia las propias neuronas (bucles con sí mismas).

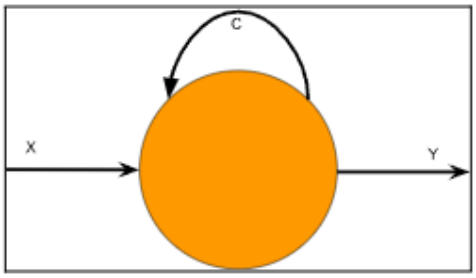


Figura 13: Neurona con recursión.

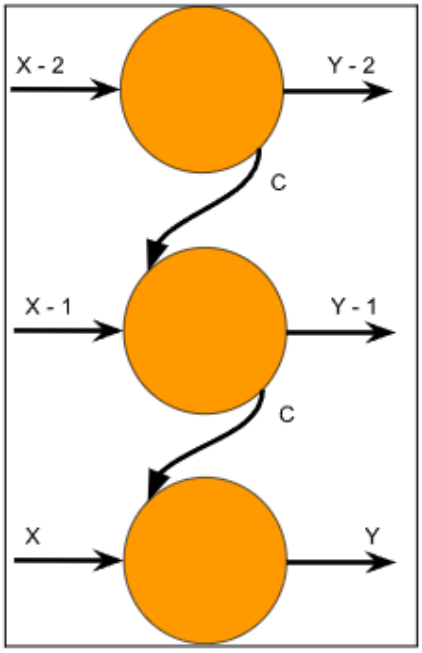


Figura 14: Desglose de la ejecución de una neurona en el tiempo.

3.3.1 Long-Short Term Memory (LSTM)

"Las redes neuronales de memoria de corto y largo plazo (LSTM) son un tipo particular de RNN que atacan el problema de las RNN asociado a la memoria de corto plazo: el desvanecimiento o decaimiento del gradiente, y su explosión" [15].

La clave de las redes LSTM estaba basada en el uso del 'estado de celda', este puede observarse en la Figura 14 como la entrada C(t-1), el estado varía en función de distintas entradas que permiten recordar nueva información, dejar de recordarla o mantenerla sin cambios.

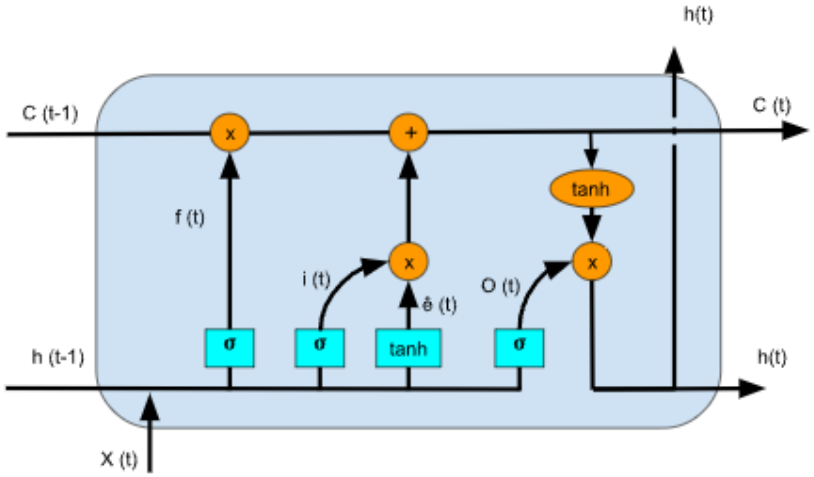


Figura 15: Composición de una neurona LSTM.

Los componentes **σ** indican una capa Sigmoidea, los componentes X y + representan operaciones de producto y suma respectivamente y por último TANH es una capa que aplica la función Tangente Hiperbólica.

Las entradas a la celda son C(t-1) la cual representa el estado de celda en un tiempo previo, h(t-1) indica el estado oculto o salida previa y X(t) son las entradas a la red.

Las funciones f(t), i(t). O(t), ê(t) son descritas a continuación:

f(t): es el resultado de una capa que aplica la función sigmoide a los datos de entrada X(t) y los valores del estado oculto h(t-1), al realizarse el producto con el estado de celda C(t-1) se están evaluando qué datos son los que se olvidaran, esto es conocido como "Forget Gate Layer".

ê(t): aquí h(t-1) y X(t) son las entradas a una capa que aplica la función TANH, esta se encarga de evaluar cuales son los datos candidatos a memorizar.

i(t): esta función es la misma que f(t), solamente que al aplicarse el producto de su salida con la salida de ê(t) se deciden realmente cuales son los valores a memorizar y cuanto es el grado de actualización. Luego estos se adicionan con los valores de f(t) * C(t-1). Esta es conocida como "Input Gate Layer".

O(t): por último se debe obtener la salida de la celda, para ello se obtiene el resultado de una capa sigmoidea, como se realizó en f(t) e i(t), este resultado será utilizado como entrada para el cálculo del producto con el nuevo estado de celda (obtenido en la descripción de i(t)) que atraviesa a su vez por el cálculo de una capa TANH. Conocida como "Output Gate Layer".

3.3.2 Gated Recurrent Units (GRU)

Otro tipo de RNN que se encarga de paliar el problema del desvanecimiento/explosión del gradiente son las Gated Recurrent Units o GRU.

La composición de una celda GRU es visible en la Figura 15, y como es posible observar, posee una gran similitud a las celdas LSTM, la diferencia principal es la desaparición del 'estado de celda'.

Las funciones que permiten adoptar nueva información y/o olvidar información actual están dadas por las funciones R(t) y Z(t). R(t) es conocida como "Reset Gate" y es la encargada de indicar qué información será la siguiente en olvidarse. Mientras que por otro lado Z(t) indica aquella nueva información a recordar.

Por último h~(t) representa a los valores candidatos al estado oculto h(t) los cuales son parte de la entrada para el cálculo del producto con los valores Z(t).

La función (1 -) representa un vector de 1's, esta posee como entrada a Z(t) cuyos valores son obtenidos a través de una función Sigmoidea, por lo tanto caen en el intervalo [0, 1] (sección 2.2.2), por lo que a través de la función (1 -) se obtienen los valores complementarios de Z(t).

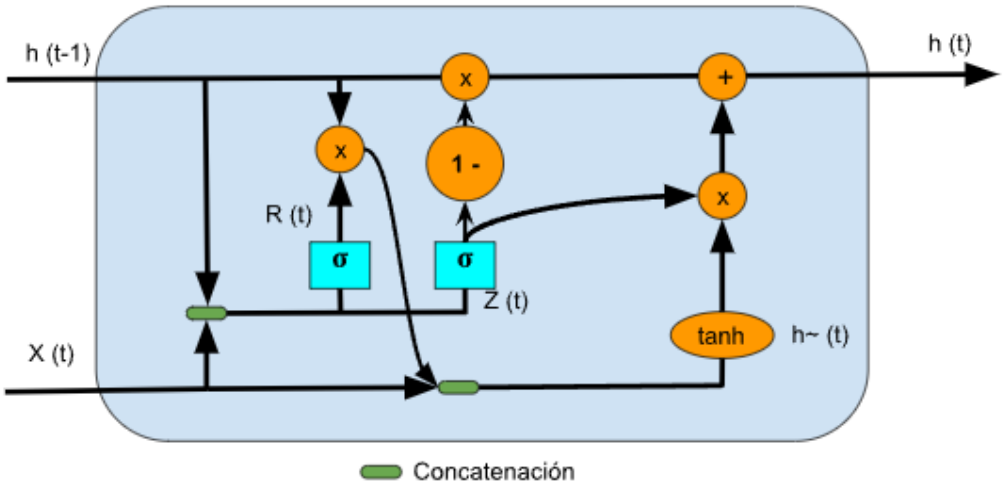


Figura 16: Composición de una neurona GRU.

3.4 Visión Artificial (Computer Vision)

La visión por computadora se puede interpretar de distintas maneras, dependiendo el objetivo con el cual se usa. Se puede utilizar para el Procesamiento Digital de Imágenes, para el Análisis de Imágenes, Reconocimiento de Patrones o Computación Gráfica entre otras.

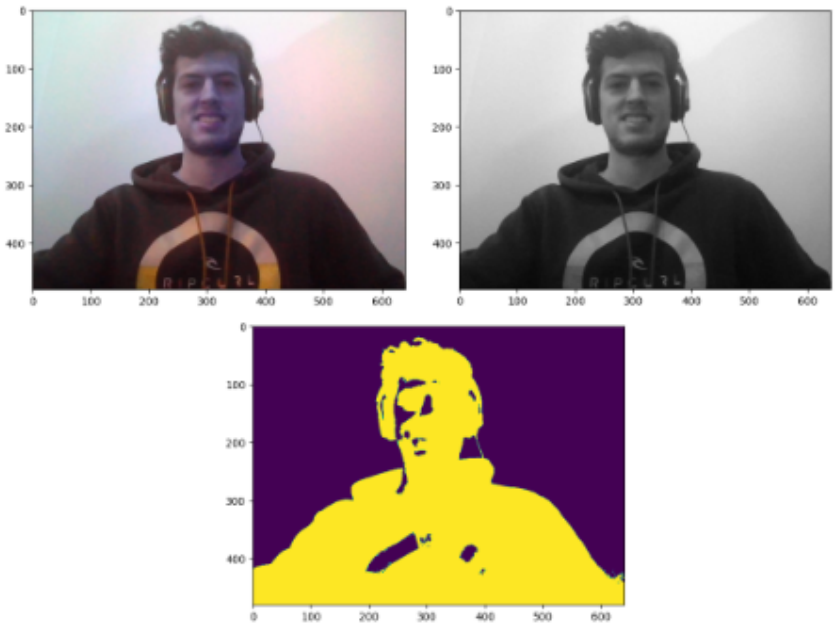


Figura 17: Imagen a color en escala BGR, imagen procesada a escala de grises, imagen segmentada resaltando el sujeto del fondo.

La visión por ordenador es un campo del ML que permite a las máquinas ver, identificar y procesar imágenes como los humanos.

Cabe aclarar que no es lo mismo hablar de Procesamiento de Imágenes y Visión Computacional, la primera trata sobre la mejora de una imagen para ser interpretada por personas, mientras que la segunda busca la interpretación de las imágenes por la misma computadora [18].

Como se describe en [18], la visión por computadora puede dividirse en 3 etapas no necesariamente secuenciales:

a. Procesamiento de Bajo Nivel: se extraen propiedades como orillas, profundidad, texturas, color, etc.
b. Procesamiento de Nivel Medio: se agrupan las propiedades del nivel inferior para obtener contornos y regiones los cuales permiten la segmentación.
c. Procesamiento de Alto Nivel: con la información obtenida de los niveles previos y conocimiento de un dominio específico se puede realizar la interpretación de los entes obtenidos.

El objetivo final es obtener información valiosa de una imagen o conjunto de imágenes, las cuales al fin y al cabo no son más que un conjunto de píxeles.

Las imágenes pueden ser transformadas en otras imágenes haciendo uso de distintas funciones de transformación, estas pueden implicar un cambio de colores o la segmentación de objetos como puede observarse en la Figura 16, se pueden realizar mejoras intentando reducir el ruido en las mismas, incluso se pueden llevar a cabo funciones de redimensión/escalado o rotación sobre las mismas.

3.5 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales son una variación de las Redes Neuronales Artificiales donde se hace uso de capas ocultas específicas como capas de convolución,

submuestreo (subsampling en Inglés) o pooling, surgen principalmente para el tratado/clasificación de imágenes y reconocimiento de objetos en estas.

3.5.1 Capas Convolucionales

Estas capas ocultas se utilizan en determinado orden y con distintos objetivos. Colocar capas convolucionales en la primera posición de las capas ocultas servirá para realzar detalles básicos como líneas o bordes que permiten distinguir objetos dentro de una imagen.

Esta capa de convolución aplica una función cuya salida S(t) representa un mapa de características o feature maps en inglés, esa función se basa en operar la entrada X(t) (grupo de píxeles cercanos de una imagen) con respecto a un kernel, filtro o matriz específica W(t).

$$S(t) = X(t) * W(t)$$

Por cada capa de convolución, se pueden aplicar varios filtros o kernels, el resultado de varios filtros puede implicar una salida de gran tamaño para la capa de convolución. Por ejemplo, si se tiene una imagen de 28*28 pixeles de alto y ancho, se tendrán 784 neuronas, si se aplicaran 32 filtros o kernels, la salida tendría un tamaño de 25.088 (784*32), para reducir el tamaño de esta salida para las capas de convolución, estas son seguidas por capas de sampleo o pooling donde se reduce el tamaño de esta salida, tomando subconjuntos de la nueva imagen mediante una función manteniendo las características importantes (Figura 18).

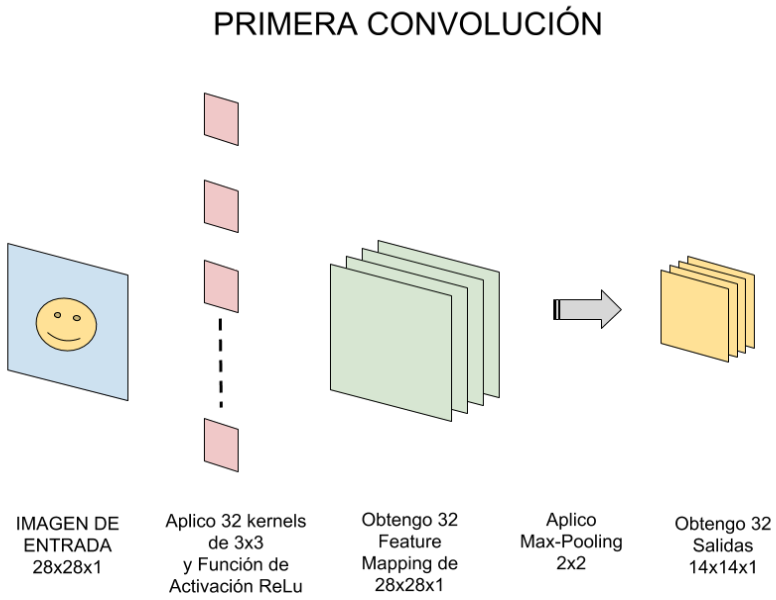


Figura 18: Convolución realizada sobre una imagen (Autor: Juan Barrios. Redes Neuronales Convolucionales).

3.5.2 Capas de Pooling

Las capas de pooling se utilizan para disminuir la cantidad de datos obtenidos como salida de la función nombrada anteriormente S(t) antes de realizar una nueva convolución. Existen diversas técnicas para llevar a cabo el pooling, algunas de ellas son Max-Pooling y Average-Pooling. Para llevar a cabo esa disminucion de datos se procesa la salida S(t) tomando regiones de dimensión NxN, el resultado de procesar cada dimensión utilizando por ejemplo con Max-Pooling, será el mayor valor de cada región, de esta manera, cada región de NxN fue reducida a 1x1 como muestra la Figura 19.

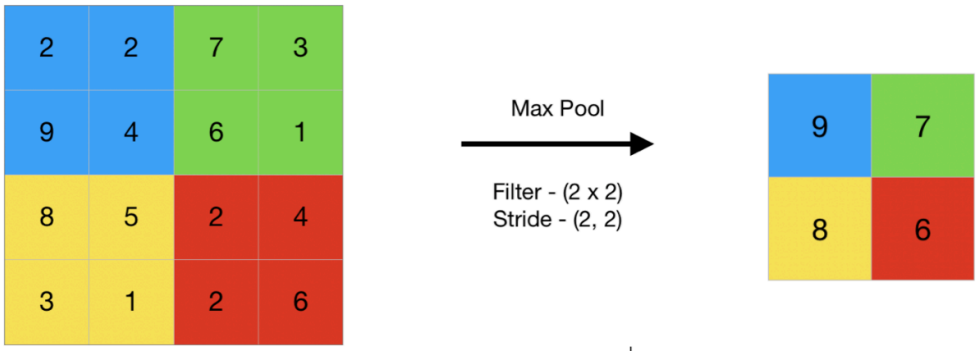


Figura 19: Operación de Pooling sobre una región determinada. (Autor: geeksforgeeks.org)

3.6 Árboles de Decisión

Los árboles de decisión son un algoritmo de aprendizaje supervisado, que se utiliza tanto para tareas de clasificación como de regresión. Tienen una estructura de árbol jerárquico, que consta de un nodo raíz, ramas, nodos internos y nodos hoja. Los nodos internos representan distintas decisiones mientras que los nodos finales indican la predicción final. El nodo raíz o inicial por el que comienzan analiza la variable mejor definida por el modelo contrastando esta observación con lógica binaria a fin de intentar diferenciar la variable analizada con otros valores que mejor se adapten, separándose en ramificaciones y repitiendo la operación lógica en tantos nodos internos como profundidad se haya especificado hasta llegar a predecir el valor más acorde a las respuestas que se estén buscando.

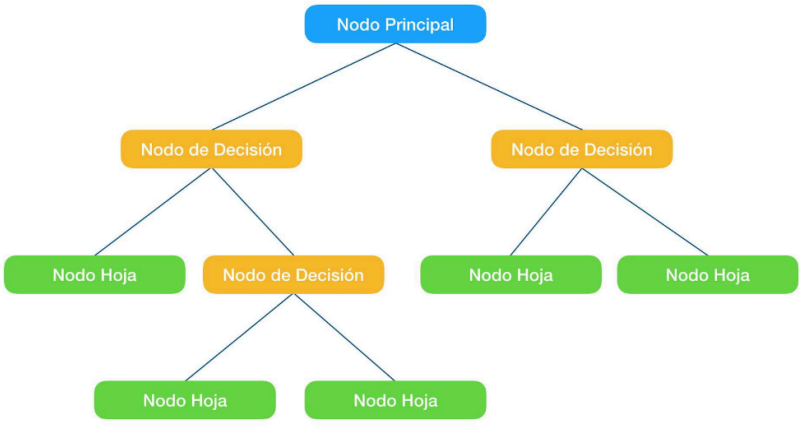


Figura 21: Árbol de decisión. (Autor: Vicente Rodriguez. Decision Trees in Python)

3.7 Sistema Web

Un sistema web es una combinación de recursos, pudiendo estar dispersos geográficamente, los cuales trabajan en conjunto con el fin de lograr un objetivo específico y para el cual fueron construidos. También se les conoce con el nombre de aplicaciones web. Estos sistemas no se encuentran instalados sobre un sistema operativo como sucede con las aplicaciones de escritorio. Se encuentran alojados en un servidor. Se pueden usar a través de cualquier navegador web sin importar el sistema operativo y/o dispositivo (computadora, smartphone, tablet, etc). Tampoco es necesario realizar una instalación de estas en el dispositivo.

3.7.1 Aplicación Web

Una aplicación web está compuesta por 3 partes básicas, el cliente, el servidor y un protocolo definido para la comunicación e intercambio de información entre cliente y servidor. El protocolo en aplicaciones web e internet es el protocolo HTTP el cual forma parte de los protocolos de comunicaciones sobre TCP/IP. El cliente está representado por un navegador web o web browser en inglés, en este navegador se ejecutará la interfaz con la que interactúa el usuario, formada por código HTML, estilos definidos con CSS y puede estar acompañada por cierta lógica definida en lenguajes de script, mayormente Javascript. El servidor por su parte se encuentra escuchando peticiones realizadas por parte de los clientes, este procesa las peticiones, realizando una acción predefinida y conocida por ambos extremos y por último entregando un resultado al cliente.

3.7.2 Arquitectura Cliente/Servidor

Las aplicaciones web son un tipo especial de aplicaciones cliente/servidor. En estas aplicaciones, un dispositivo situado en un extremo sirve de cliente, el cual hará de consumidor

sobre los servicios que, desde el otro extremo, otorga el servidor. El servidor se encargará de manejar todas las peticiones que los clientes realicen. Una ventaja de esta arquitectura es que el servidor puede replicarse y/o localizarse en un dispositivo que mejor se adapte a sus requerimientos.

Como propiedades de esta arquitectura, se tiene que nombrar la separación en capas que se logra:

- Lógica de Presentación (cliente web): se encarga de presentar la interfaz de usuario y manejar la interacción con el mismo.
- Lógica de Negocios (servidor web): esta capa recibe la información enviada desde la capa de Lógica de Presentación, realiza las acciones necesarias sobre los mismos e interactúa con la capa de Lógica de Datos. Actúa como un intermediario entre ambas.
- Lógica de Datos (base de datos): por último ésta capa recibe los datos y se encarga de su almacenamiento y persistencia.

La comunicación entre la Lógica de Presentación y la Lógica de Negocios se suele llevar a cabo a través de API's.

3.7.3 Interfaz de Programación de Aplicaciones (API)

Como se introdujo previamente, una API facilita la comunicación entre dos programas de software. Las API's definen la firma de las funciones disponibles, protocolos, como deben ser llamadas como así también la estructura de sus respuestas. La definición de API's permite que puedan ser consumidas por una amplia gama de aplicaciones, escritas en diferentes lenguajes y localizadas en distintas ubicaciones/servidores.

3.8 Lengua de Señas

"La Lengua de Señas Argentina o LSA es la lengua de las personas Sordas que conforman la comunidad Sorda argentina. Es una lengua natural, con una estructura

gramatical diferente a la del español. Esto significa que la LSA y el español son dos lenguas distintas. Existen muchas lenguas de señas en el mundo. La existencia de una o más lenguas en un mismo país o en más de un país depende de la historia de las comunidades Sordas de cada lugar." [18].

Cada seña que compone a una lengua está constituida por la combinación de 5 elementos:

- Configuración: indica la forma de colocar los dedos al momento de iniciar la seña. Las configuraciones de las manos pueden aparecer sobre una mano que queda inmóvil o que se mueve, sobre las 2 manos simétricas que se mueven o sobre las 2 manos asimétricas -la mano dominante actúa sobre la otra que no se mueve-.
- Orientación: posición de la palma de la mano, pudiendo ser esta hacia arriba/abajo, palmas enfrentadas, palmas una sobre la otra, juntas hacia fuera/dentro, brazos horizontales, verticales, oblicuos, hacia el cuerpo, etc.
- Ubicación: lugar sobre el cuerpo o el espacio en donde se hacen las señas.
- Movimiento: los movimientos de las manos, de los dedos, de las muñecas y de los brazos.
- Componentes no manuales: hace referencia a movimientos de hombros, cabeza, brazos, expresiones faciales o del rostro.

"La estructura básica entre el español y la Lengua de Señas Argentina es diferente. En vez de la estructura del español 'sujeto, verbo, objeto' (el perro corre la pelota), en la Lengua de Señas Argentina el verbo va al final (el perro a la pelota corre). Este orden es canónico, es decir, es el orden hacia el que la lengua tiende, aunque en el discurso se viola por efectos semánticos y pragmáticos" [21].

3.8.1 Sintaxis

Las señas formadas por estos componentes se unen para formar frases gestuales. El cambio sobre algún componente puede modificar la frase, por ejemplo, a través del cambio de una expresión facial se puede intensificar, cambiar el sentido de la seña o cambiarla en una pregunta, sucediendo lo mismo con el cambio de dirección del movimiento de un verbo que muestra quien hace y quien recibe [22].

La naturaleza de la sintaxis de la Lengua de Señas Argentina (LSA) hace que la ubicación de las señas en el espacio sea más importante que el orden de las palabras en la línea del tiempo.

3.8.2 El Espacio

El espacio del señante abarca el espacio definido delante de él, verticalmente entre la cabeza y la cintura y horizontalmente entre los brazos extendidos hasta el codo. El espacio también sirve para expresar el tiempo, en la LSA existe una línea del tiempo imaginaria a la altura del hombro donde se sitúan las señas que hacen referencia al tiempo.

3.8.3 Consideraciones

Según se indica en [21] se deben tener en cuenta las siguientes consideraciones respecto a la LSA:

La LSA no se escribe, se glosa: Glosas es la forma convencional utilizada para escribir todas las lenguas de señas, ya que estas carecen de escritura implican la relación más próxima entre el significado de la seña y la o las palabras españolas que la representan.

- En referencia a una oración, el orden que más se observa en la LSA es:

  Tiempo + Sujeto (S) + Objeto (O) + Verbo (V)

- No existen artículos, tampoco nexos. Por ejemplo:

  Lengua Oral: “La casa es Linda” - LSA: “Casa Linda”

- El verbo se encuentra en infinitivo y se conjuga con la presencia del tiempo. Generalmente se encuentra al final de la oración. Cuando no se marca el tiempo la oración se considera en presente.
- Sujeto o persona: siempre están presentes en la oración, pueden estar después del tiempo o antes ya que esto es variable, lo que nunca puede ocurrir es no señalar el sujeto porque sino no existe.
- Pronombres posesivos: siempre se señan después de la persona o el sujeto.

  Lengua Oral: “Mi perro es lindo” - LSA: “Perro mio lindo”
- Negación o interrogación: por lo general se ubica al final del mensaje o puede estar incorporada por componentes gestuales no manuales.

“Las señas pueden ser realizadas con una sola mano, llamadas por ello unimanuales o con las dos manos denominadas bimanuales. La mano más productiva es conocida como mano activa mientras que la mano que menos produce constituye para el señante, su mano pasiva” [24].

4. Metodología

4.1 Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux (software libre y multiplataforma). Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C#, Java, Python, PHP, Go, .NET). Tiene una buena integración con Git, cuenta con soporte para depuración de código y al disponer de un sinnúmero de extensiones básicamente da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

4.2 Python

Python es un lenguaje de programación interpretado, orientado a objetos, de alto nivel y semántica dinámica. Sus estructuras de datos incorporadas de alto nivel, combinadas con la tipificación dinámica y la vinculación dinámica, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o glue para conectar componentes existentes entre sí. Su sintaxis, sencilla y fácil de aprender, favorece la legibilidad y, por tanto, reduce el coste de mantenimiento de los programas. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario de forma gratuita para las principales plataformas, y pueden distribuirse libremente.

4.2.1 Paquetes

4.2.1.1 XGBoost.

XGBoost proviene del Inglés “Extreme Gradient Boosting” o “Refuerzo extremo del gradiente”, es una herramienta desarrollada para entrenar mediante el aprendizaje supervisado árboles de decisión. Cada árbol que se genere aprenderá de sus predecesores e intentará disminuir el error incurrido en ellos.

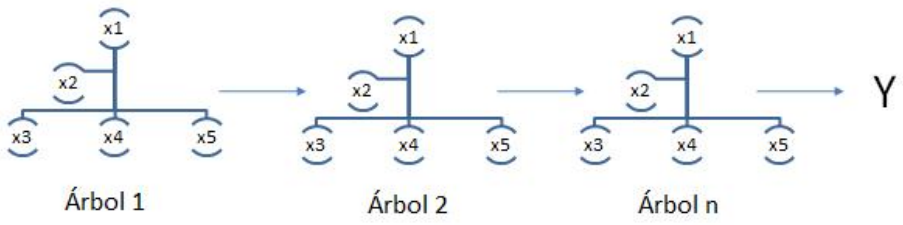


Figura 20: Árboles generados por XGBoost durante el entrenamiento [19].

En [19] se describe el funcionamiento del algoritmo XGBoost:

"Se obtiene un árbol inicial F0 para predecir la variable objetivo 'y', el resultado se asocia con un residual (y - F0).

Se obtiene un nuevo árbol h1 que ajusta el error del paso previo.

Los resultados de F_0 y h_1 se combinan para obtener el árbol F_1, donde el error cuadrático medio de F_1 será menor que el de F_0:

$$F_1(x) = F_0(x) + h_1(x)$$

Este proceso se sigue iterativamente hasta que el error es minimizado lo más posible de la siguiente forma:

$$F_m(x) = F_{m-1}(x) + h_m(x)$$"

4.2.1.2 MediaPipe.

MediaPipe es un Framework que hace uso de técnicas de Machine Learning enfocadas a la detección de rostros, manos, posición corporal, distintas acciones como puede ser el tracking de ojos (iris), detección y/o segmentación de objetos con respecto al fondo y más soluciones con distinto enfoque.

Dentro de este conjunto de soluciones se procede a detallar la solución "Holistic landmarks", la cual permite la detección completa del cuerpo mediante el uso y combinación de soluciones previas para extraer puntos o *landmarks* correspondientes al rostro, manos y posición corporal. Las soluciones previas que integra este modelo son "human pose", "face landmarks" y "hand tracking" los cuales están optimizados cada uno para su dominio, por lo cual la entrada de uno de ellos puede no ser adecuada para otro. Lo que hace "Holistic landmarks" es definir un pipeline con etapas múltiples para tratar las distintas regiones de una imagen con sus resoluciones correspondientes.

Un detalle de esta solución es que utiliza un enfoque de seguimiento basado en la premisa de que el objeto no se mueve significativamente entre fotogramas y usa una estimación del frame previo como guía de la región del objeto en el frame actual.

El resultado de este modelo son un total de 543 coordenadas o landmarks que describen la ubicación de la persona en una imagen, encontrando estos puntos normalizados dentro del intervalo [0, 1] relativo al ancho y alto de la imagen en cuestión.

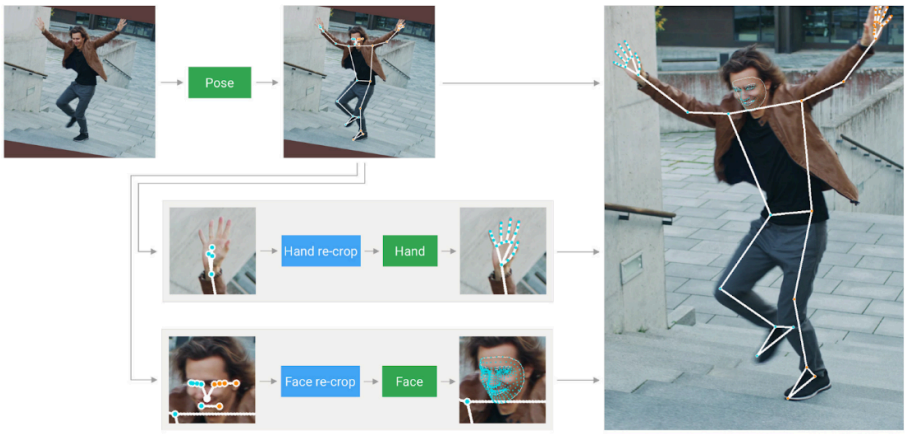


Figura 17: Pipeline realizado por MediaPipe al procesar una imagen con la solución 'Holistic'. (Google Research, MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device)

El funcionamiento base de todas estas soluciones utiliza Redes Neuronales Convolucionales o CNN, las cuales a través de distintas convoluciones sobre la imagen de entrada se logra discriminar los distintos elementos en las imágenes y así inferir su posición.

4.2.1.3 Tensorflow.

Para el desarrollo de la inteligencia artificial y el aprendizaje automático, TensorFlow juega un papel muy importante. Se trata de una librería de código libre para Machine Learning (ML). Fue desarrollado por Google para satisfacer las necesidades a partir de redes neuronales artificiales. TensorFlow te permite construir y entrenar redes neuronales para detectar patrones y razonamientos usados por los humanos. Además de trabajar con redes

neuronales, TensorFlow es multiplataforma. Trabaja con GPUs y CPUs e incluso con las unidades de procesamiento de tensores (TPUs).

Facilita la creación e implementación de modelos de aprendizaje automático. Lo que buscamos hoy en día es automatizar cientos de procesos y TensorFlow nos permite llegar a esta automatización. Nos permite crear y entrenar modelos de Aprendizaje Automático con facilidad mediante APIs intuitivas.

4.2.1.4 Django.

Django es un framework de desarrollo el cual permite organizar y escribir código de manera más eficiente y reducir significativamente el tiempo de desarrollo web.

Mediante el uso de Django es posible construir aplicaciones web basadas en arquitecturas MVC (Modelo-Vista-Controlador) aunque en Django este término es reemplazado por MVT o Modelo-Vista-Template.

No sólo es útil para la construcción de aplicaciones web completas sino que permite la construcción de API 's sin necesidad de implementar un frontend.

4.3 Google Colab

Google Colab, también conocido como "Colaboratory", permite programar y ejecutar Python en un navegador con las siguientes ventajas:

- No requiere configuración
- Acceso a GPUs sin coste adicional
- Permite compartir contenido fácilmente

Colab puede facilitar el trabajo, ya sea a estudiantes, científicos de datos o investigadores de IA. No se necesita instalar, ni tiempo de ejecución ni actualizar el hardware del ordenador para cumplir los requisitos de carga de trabajo intensiva de CPU/GPU de Python. Además, Colab ofrece acceso gratuito a infraestructuras informáticas como almacenamiento, memoria,

capacidad de procesamiento, unidades de procesamiento gráfico (GPU) y unidades de procesamiento tensorial (TPU).

4.4 Angular

Angular es un Framework de JavaScript de código abierto escrito en TypeScript. Su objetivo principal es desarrollar aplicaciones de una sola página. Google se encarga del mantenimiento y constantes actualizaciones de mejoras para este framework. Es un framework basado en componentes para crear aplicaciones web escalables. Una colección de bibliotecas bien integradas que cubren una amplia variedad de características, que incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más. Un conjunto de herramientas para desarrolladores que permiten desarrollar, compilar, probar y actualizar el código fuente de la aplicación. Posee un conjunto de herramientas y módulos que se pueden reutilizar para diferentes proyectos. Facilitando en varios aspectos el desarrollo, mejorando el tiempo, esfuerzo, organización.

4.4.1 Paquetes

4.4.1.1 MediaPipe para Javascript.

Este framework fue introducido en la sección 4.2.2, en este apartado se utiliza la versión de MediaPipe para Javascript, la cual es utilizada para obtener los puntos corporales en la aplicación web en tiempo real. En específico se utiliza el siguiente paquete: "@mediapipe/holistic": "0.5.1635989137"

4.4.1.2 Tensorflow JS.

Similar a lo que ocurre con MediaPipe, también se hace uso de la versión para Javascript de Tensorflow, lo que nos permite hacer uso de este paquete de aprendizaje automático en la web. La versión utilizada es: "@tensorflow/tfjs": "4.1.0".

4.4.1.3 Typescript.

TypeScript (TS) es un lenguaje de programación construido a un nivel superior de JavaScript (JS). Esto quiere decir que TypeScript dota al lenguaje de varias características adicionales que hacen que se pueda escribir código con menos errores, más sencillo, coherente y fácil de probar, en definitiva, más limpio y sólido. Añade más funcionalidades, como tipado fuerte, anotaciones o módulos que en JS no se encuentran disponibles.

Fue creado por Microsoft en 2012 y, desde entonces, su adopción no ha hecho más que crecer. Especialmente, desde que Google decidió adoptarlo como lenguaje por defecto para desarrollar con Angular.

4.4.1.4 HTML

El Lenguaje de Marcado de Hipertexto (HTML) es el código que se utiliza para estructurar y desplegar una página web y sus contenidos.

HTML no es un lenguaje de programación; es un lenguaje de marcado que define la estructura del contenido en una página web. HTML consiste en una serie de elementos que se utilizan para encerrar diferentes partes del contenido, para que se vean o comporten de una determinada manera. Las etiquetas de encierre pueden hacer de una palabra o una imagen un hipervínculo a otro sitio, se pueden cambiar palabras en cursiva, agrandar o achicar la letra, etc.

4.4.1.5 CSS.

Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. (incluyendo varios lenguajes basados en XML como SVG, MathML o XHTML). CSS describe cómo debe ser renderizado el elemento estructurado en la pantalla. CSS es uno de los lenguajes base de la Open Web y posee una especificación estandarizada por parte del W3C.

CSS es utilizado para diseñar y dar estilo a las páginas web, por ejemplo, alterando la fuente, color, tamaño y espaciado del contenido, dividirlo en múltiples columnas o agregar animaciones y otras características decorativas.

4.5 Figma

Para la realización de los prototipos se usó la herramienta Figma, la cual es un editor de gráficos vectorial y una herramienta de generación de prototipos de acceso gratuito, principalmente basada en la web aunque esta posee también su versión para escritorio.

Figma es una plataforma de edición gráfica y diseño de interfaces. Además, es una plataforma online y colaborativa. Con Figma se puede hacer un poco de todo a nivel de diseño gráfico, desde diseñar páginas web e interfaces gráficas de aplicaciones, o crear publicaciones para redes sociales, hasta la posibilidad de poder crear presentaciones. Por este motivo, es una de las herramientas más valoradas por empresas y estudios de diseño gráfico.

4.6 Base de Datos de Lengua de Señas

Para el entrenamiento de las Redes Neuronales es necesario contar con un Set de Datos o Base de Datos con el cual llevar a cabo ese entrenamiento. Este Set de Datos puede construirse a medida para el problema en particular o también, en caso de ser posible, usar un Set de Datos ya construido y que pueda ser aplicado al contexto del problema.

En el presente proyecto de investigación se hace uso del Set de Datos “LSA64” para llevar a cabo el entrenamiento, este Set de Datos fue obtenido como parte de los resultados de la Tesis Doctoral de Franco Ronchetti en [25] y se permite la utilización del mismo en [26].

El Set de Datos cuenta con un conjunto de 64 señas donde cada seña fue interpretada por 10 personas las cuales, a su vez, grabaron 5 videos para cada seña, obteniendo un total de 3200 videos, 50 videos por cada seña.

La particularidad de los videos contenidos en este set es que los interpretes utilizan un guante de color diferente por mano (rosa fuerte en la mano derecha y verde claro en la mano izquierda) y la mano dominante en estos es la mano derecha.

Figura 22: Frame de video perteneciente al set de datos "LSA64", seña "Burla"- 5ta persona - 1er video.

En [25] se indica lo siguiente:

"La base de datos fue construida en dos sets de grabación distintos. En el primero fueron grabadas 23 señas con una sola mano y se utilizó luz natural en un entorno abierto. En el segundo set, se agregaron 41 señas más (22 con dos manos, y 19 con una mano) y se utilizó luz artificial en un entorno semi-cerrado….La base de datos es públicamente accesible bajo licencia Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International sólo para uso académico. Se encuentra almacenada en https://midusi.github.io/lsa64/index.html.".

Otra característica de los videos es que estos fueron capturados en calidad FullHD o 1920x1080 y a 60 fotogramas por segundo (FPS).

A continuación, se listan las señas disponibles en el set de datos:

Seña	Cantidad De Manos	Número	Seña	Cantidad de Manos	Número
Opaco	1	1	Hambre	1	33
Rojo	1	2	Mapa	2	34

Seña	Cantidad De Manos	Número	Seña	Cantidad de Manos	Número
Verde	1	3	Moneda	2	35
Amarillo	1	4	Música	2	36
Brillante	1	5	Nave Espacial	1	37
Celeste	1	6	Ninguno	1	38
Colores	1	7	Nombre	1	39
Rosa	1	8	Paciencia	1	40
Mujer	1	9	Perfume	1	41
Enemigo	1	10	Sordo	1	42
Hijo	1	11	Trampa	2	43
Hombre	1	12	Arroz	2	44
Lejos	1	13	Asado	2	45
Cajón	1	14	Caramelo	1	46
Nacer	1	15	Chicle	1	47
Aprender	1	16	Fideos	2	48
Llamar	1	17	Yogurt	2	49
Espumadera	1	18	Aceptar	2	50
Amargo	1	19	Agradecer	2	51
Dulce	1	20	Apagar	1	52
Leche	1	21	Aparecer	2	53
Agua	1	22	Aterrizar	2	54
Comida	1	23	Atrapar	2	55
Argentina	1	24	Ayudar	2	56
Uruguay	1	25	Bailar	2	57
País	1	26	Bañarse	2	58
Apellido	1	27	Comprar	1	59

Seña	Cantidad De Manos	Número	Seña	Cantidad de Manos	Número
Donde	1	28	Copiar	2	60
Burla	2	29	Correr	2	61
Cumpleaños	1	30	Darse Cuenta	1	62
Desayuno	2	31	Dar	2	63
Foto	2	32	Encontrar	1	64

Tabla 1: Conjunto de señas del Set de Datos: “LSA64”

5. Antecedentes

La interpretación de la Lengua de Señas es un tema que se ha estado tratando durante varios años en las ciencias computacionales. Se pueden encontrar trabajos datados del año 1977 en [27] donde se desarrolló una mano mecánica encargada de establecer distintas configuraciones de la misma, su funcionamiento en ese entonces estaba basado en recibir como entrada texto ingresado por teclado y traducir ese texto a distintos comandos que son visualizados como movimientos mecánicos que lleva a cabo la mano para formar la composición adecuada al alfabeto manual de una mano. El desarrollo de trabajos similares basados en el uso de robots continuó durante los años 80’s y 90’s, uno de los trabajos reconocidos es RALPH [28] el cual es la 4ta generación de una mano mecánica con mejoras en los sistemas mecánicos y tamaños. En ese entonces estas manos mecánicas permitían su uso por parte de personas ciegas-sordas las cuales a través del tacto podían inferir las letras y poder formar las palabras a comunicar.

Mejoras en la tecnología permitieron el cambio de las manos mecánicas al desarrollo de guantes que permiten traducir la Lengua de Señas a lenguaje oral, trabajos actuales como [29] o [30] hacen uso de hardware específico para analizar el movimiento que realizan las

articulaciones de las mano y en base a las mediciones obtenidas poder dilucidar la seña/letra realizada.

La diferencia a destacar entre estos 2 enfoques es la dirección de la comunicación, ya que con el uso de guantes la persona sorda puede darse a entender, pero no en el sentido contrario. Mientras que con la mano mecánica es la persona sorda-ciega quien interpreta el mensaje mediante el contacto/visualización de la mano mecánica al moverse.

Los trabajos que se describen a continuación no utilizan artefactos externos con los cuales se pueda obtener información alguna del entorno/señante y/o características especiales como podría ser mediante el uso de guantes electrónicos o el uso de tecnología Kinect.

Dentro del área del aprendizaje automático pueden encontrarse una amplia gama de trabajos dedicados a la interpretación y/o traducción de distintas Lenguas de Señas.

Con lo que respecta al uso del del Set de Datos “LSA64” pueden encontrarse trabajos realizados como [31] donde se realiza la construcción de modelos de distintas arquitecturas haciendo uso de Redes Neuronales Convolucionales (CNN), de Redes Neuronales Recurrentes y una combinación de ambas, e incluso se llevan a cabo ciertos procesamientos extra sobre los datos en búsqueda de mejoras en el aprendizaje (variación en longitudes de videos, tamaño de frames, etc).

En [32] similar al trabajo previo aquí también se utiliza una combinación de CNN y RNN, donde las capas que realizan convoluciones sobre los frames extraen características espaciales de las manos gracias a los colores especiales de los guantes que usan los señantes (segmentación por color). Como resultado el modelo obtenido alcanzó un accuracy aproximado al 94%.

En el trabajo [33] se utiliza una red pre-entrenada o modelo para obtener información correspondiente a puntos corporales del cuerpo, manteniendo el uso de RNN y añadiendo histogramas de sistemas dinámicos lineales (LDS). La limitación que presenta este trabajo es

que solo se tiene en cuenta la mano derecha del señante, descartando aquellas señas que hacen uso de ambas manos.

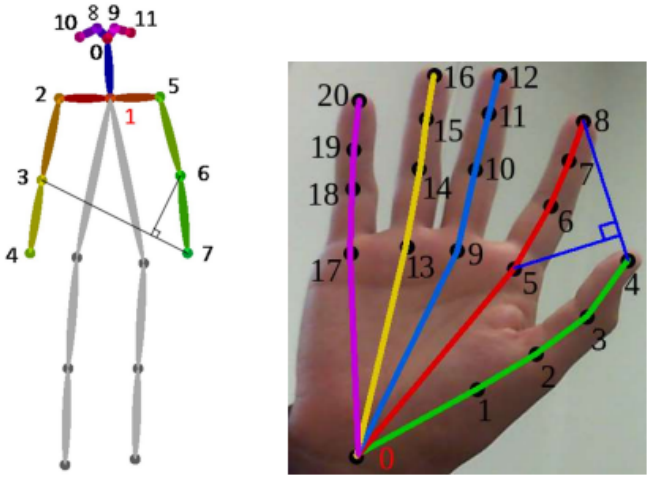


Figura 23: Figura extraída de [33]

Dejando de lado el uso del Set de Datos argentino "LSA64" se pueden encontrar trabajos como [34] donde se desarrolla un Set de Datos compuesto de 14.672 videos perteneciente a la Lengua de Señas Coreana. El mismo también hace uso de la extracción de puntos corporales como sucede en [33] con el uso de la librería OpenPose, lo que destaca en comparación con los trabajos nombrados anteriormente es el uso de una Red Codificadora-Decodificadora basada en la atención (Attention-based Encoder-Decoder Network)

Figura 24: Imágenes extraídas de [34] donde se muestran los puntos corporales de un frame.

En [35] se desarrolló un Set de Datos compuesto de 21.083 vídeos realizados por un total de 119 señantes y cada seña fue realizada por al menos 3 personas diferentes. Otra característica de los videos es que cuentan con una amplia variedad de fondos (colores), diferentes condiciones de iluminación y señantes de apariencia diferente con el objetivo de lograr una generalización o independencia del sistema por sobre la persona.

Aquí se experimentó con la creación de distintos modelos, el primero de ellos y como sucedía en [32] se utiliza una combinación de Redes Convolucionales y Recurrentes, siendo la red convolucional un modelo pre-existente y re-entrenado conocido como VGG (Visual Geometry Group). Para el entrenamiento se realizó una selección aleatoria de máximo 50 frames consecutivos.

El siguiente enfoque tomado fue la estimación de la posición a través del uso de puntos corporales, similar a [33] y [34], se destaca que a pesar de que se realiza la extracción de puntos corporales, no se están explorando las dependencias espaciales entre los mismos. La extracción de puntos también fue realizada con la librería OpenPose y estos actúan de entrada a una RNN compuesta por 2 capas GRU. Para el entrenamiento se utilizaron 50 frames aleatorios consecutivos aunque en el proceso de testing o testeo se utilizaron todos los frames de los videos.

6. Trabajo Desarrollado

Se plantea la combinación de distintos enfoques, vistos en el capítulo previo, para la generación de un modelo capaz de interpretar la Lengua de Señas Argentina y en paso posterior integrar el mismo en una Aplicación Web que permita el acceso y uso por parte de cualquier persona que pueda acceder a un dispositivo móvil o computadora.

Se realiza la construcción y evaluación de distintos modelos de Redes Neuronales, con el objetivo de contrastar y encontrar, de ser posible, un modelo que presente buenas características (accuracy elevado).

Seguido a la construcción y obtención de un modelo que presente características alentadoras, se procede a exportar el modelo con el fin de que este pueda ser importado y usado en aplicaciones Web.

Una vez obtenido este modelo exportado, se presentan los prototipos de interfaces de Aplicación Web y el servicio de backend donde residirá el modelo y se realizará la interpretación de la información en tiempo real.

6.1 Preprocesamiento

El primer paso, previo a la construcción y entrenamiento de los modelos, consistió en obtener los datos sobre los cuales el modelo basa su aprendizaje. Del conjunto principal de 64 señas se conformó un subconjunto de este, compuesto por las siguientes señas:

Seña	Cantidad De Manos	Número	Seña	Cantidad de Manos	Número
Brillante	1	5	Uruguay	1	25
Mujer	1	9	País	1	26
Hijo	1	11	Donde	1	28
Hombre	1	12	Nave Espacial	1	37
Lejos	1	13	Ninguno	1	38
Nacer	1	15	Nombre	1	39
Aprender	1	16	Perfume	1	41
Espumadera	1	18	Sordo	1	42
Amargo	1	19	Comprar	1	59
Leche	1	21	Encontrar	1	64
Comida	1	23			

Tabla 2: Subconjunto de señas

Este subconjunto de palabras fue elegido con el objetivo de construir frases con sentido lógico, algunas frases que se pretende poder formar con el sistema final son las siguientes:

- “El hombre nació sordo.”
    - LSA: “Hombre sordo nacer.”
- “La mujer compra comida amarga.”.
    - LSA: “Mujer comida amarga comprar.”
- “El hijo encontró la espumadera brillante.”.
    - LSA: “Hijo espumadera brillante encontrar.”
- “Uruguay es un país lejano.”
    - LSA: “País Uruguay lejos.”
- “El hombre encuentra la nave espacial.”
    - LSA: ”Hombre nave espacial encontrar.”
- “La mujer aprende ningún nombre.“
    - LSA: “Mujer nombre ninguno aprender.”
- “¿Dónde compran el perfume la mujer y el hombre?“
    - LSA: (Conjunción) “Mujer, hombre perfume comprar, dónde?”
- “La mujer, el hombre y el hijo compran leche.”.
    - LSA: (Enumeración) “Mujer, hombre, hijo leche comprar.”

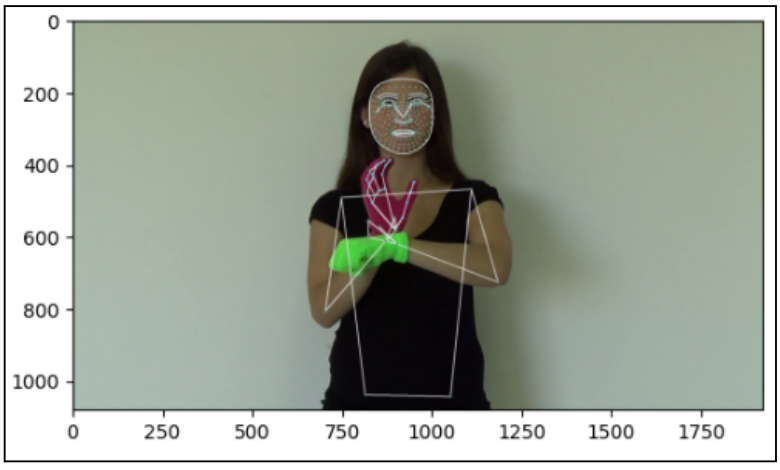
Los frames de los videos no fueron utilizados de manera directa para realizar el entrenamiento de las Redes Neuronales, estos atravesaron un proceso mediante el cual se obtuvieron coordenadas de puntos específicos del cuerpo correspondientes al cuerpo, cara y manos.

El método utilizado para obtener información de los videos se basa en el procesamiento y obtención de “puntos corporales” o “skeletal data” de la persona presente en el video a través de cada frame o imagen.

Para la obtención de estos puntos corporales se utilizó un framework de Machine Learning llamado “MediaPipe”.

Dentro de este framework se encuentran distintos modelos pre-entrenados y listos para su uso de los cuales se tomó el llamado “Holistic Tracking”, la justificación del modelo seleccionado está basada en que permite obtener una estimación de puntos corporales correspondientes a ambas manos, la cara y la posición corporal con un único modelo.

En la Figura 25, se muestran 2 imágenes procesadas con MediaPipe, en la primera se observa una imagen vista previamente (Figura 22) a la que se le añaden los puntos corporales detectados, y en la segunda se observa un gráfico en 3D de los puntos corporales correspondientes solamente a la posición corporal o esqueleto (sin incluir cara o manos).



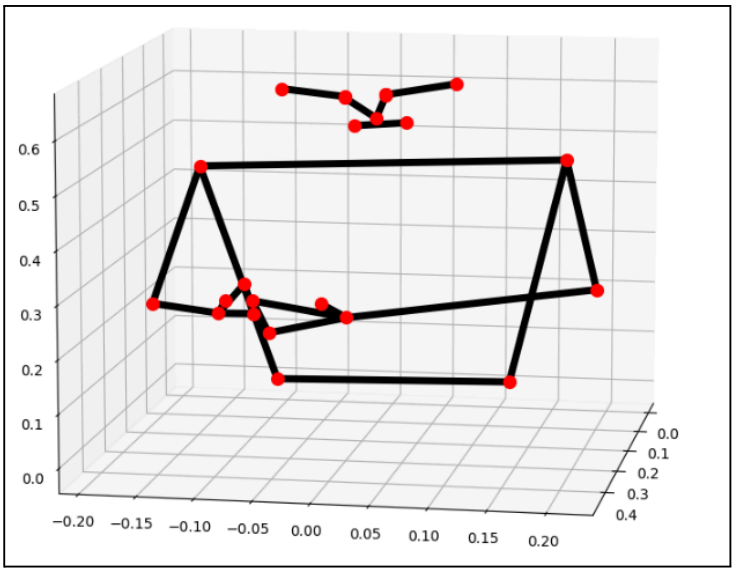


Figura 25: Frame procesado con MediaPipe (arriba) y datos obtenidos graficados en 3 dimensiones (abajo).

Por cada frame entregado al modelo se obtuvo como resultado un objeto con distintas propiedades, de las cuales se hizo foco en los siguientes componentes:

1. left_hand_landmark: puntos de mano izquierda.
2. right_hand_landmarks: puntos de mano derecha.
3. pose_landmarks: puntos correspondientes a la pose corporal o esqueleto.
4. face_landmarks: puntos faciales.

Mediante el uso de este modelo se obtiene una lista de coordenadas para cada punto detectado, y dependiendo el componente a tener en cuenta, es decir, cara, manos o pose, el tamaño de esta lista variará. Relacionando con los componentes ya nombrados se tiene:

- Puntos faciales o Cara (face_landmarks): Se obtiene una lista de 468 puntos faciales, donde cada punto indica una coordenada dada por los puntos (x, y, z).
- Pose o Esqueleto (pose_landmarks): retorna una lista de 33 puntos corporales, cada uno formado también por coordenadas (x, y, z) y se añade otro valor llamado Visibilidad.

- Mano izquierda/Mano derecha (left/right_hand_landmarks): se cuenta con una lista de 21 puntos por cada mano y cada punto queda representado por coordenadas (x, y, z).

En la documentación del modelo elegido de MediaPipe se aclara que los puntos obtenidos ya se encuentran normalizados respecto al tamaño de la imagen, es decir, se obtienen coordenadas (x, y) dentro del intervalo [0.0, 1.0], donde este intervalo es relativo al alto y ancho de la imagen. Los puntos correspondientes al eje Z (valor que indica la profundidad) no son tenidos en cuenta ya que el modelo, como indica su documentación, no tiene el conocimiento/entrenamiento suficiente para realizar correctas predicciones sobre este valor.

Si el modelo no detecta una mano en la imagen a procesar, no retorna valor alguno, en estos casos se constituye una lista de ceros de longitud fija dependiente del componente.

El resultado de procesar una imagen con este modelo son 4 listas de 468, 33, 21 y 21 puntos o coordenadas correspondientemente. Si se concatenan estas 4 listas en una sola, se obtiene una lista de longitud (468 + 33 + 21 + 21) * 2 = 1086, el factor 2 se añade ya que por cada punto o coordenada tenemos en cuenta los valores para los ejes X e Y

Para la obtención de los puntos corporales fue necesario procesar los 1050 videos con el framework mencionado y luego almacenar los resultados. Estos fueron almacenados en un DataFrame como datos tabulares en lugar de almacenar la información de cada video por separado, ya que de esta última forma se incurrirá en información excesiva para las cabeceras de los archivos. Se realizó una prueba con respecto al almacenamiento de la información, primero almacenando el resultado de cada video en un archivo .npy (numpy), en total la información resultante ocupaba 1.77GB, mientras que almacenando los resultados en un DataFrame el espacio utilizado disminuyó a 51.7MB para el muestreo con mayor cantidad de información (sección 6.2).

El DataFrame resultante posee la siguiente estructura:

- Cada frame de un video representa una fila en la tabla, fila a la cual se concatena, como última columna, la seña a la cual pertenece.
- El orden de las columnas está dado de la siguiente manera:
    - coordenadas faciales, coordenadas corporales, coordenadas mano izquierda y coordenadas mano derecha.
- Cada columna representa un punto de coordenada, de manera que una fila quedaría representada como:

  [face_0_x, face_0_y, face_1_x,..., right_hand_41_x, right_hand_41_y].

Al contar con un set de datos con un amplio repertorio de videos, se realizó una evaluación de estos para obtener una lista de valores máximos y mínimos sobre la cantidad de frames que posee cada video, esto con el objetivo de poder establecer un estándar de longitud para la lista de datos a almacenar. También se realizó el cálculo del promedio de frames sobre todo el conjunto de datos.

Los resultados de la Figura 26 informan el número máximo y mínimo de frames sobre todos los vídeos analizados:

```
Maximo numero de frames 201.0. Seña: 40
Minimo numero de frames 14. Seña: 14
Promedio: 81.0.
```

Figura 26: Videos con mayor y menor cantidad de frames en el set de datos.

El video que cuenta con el máximo número de frames fue “041_006_005.mp4” con 201 frames de longitud correspondiente a la seña “Perfume”. Este valor se tuvo en cuenta para evitar recortar información en un primer procesamiento de los videos, además, se utilizó para almacenar el resultado de los videos procesados con una misma cantidad de frames, aunque los frames añadidos no fueron utilizados.

El DataFrame resultante puede entenderse como una matriz de 1087 columnas y la concatenación de 201 filas por video, donde cada fila representa un solo frame. Los nombres de las columnas se definieron de la siguiente manera:

Los primeros datos que se colocaron en la lista de puntos son los correspondientes a la cara, estas columnas se nombraron: “face_pZ_x” y “face_pZ_y”, dónde el valor Z se encuentra definido en {Z: 0 <= Z < 468}.

Luego las columnas siguientes corresponden a la posición corporal: “pose_pZ_x” y “pose_pZ_y” con {Z: 0 <= Z < 33}.

A continuación le siguen las columnas respectivas a la mano izquierda y luego derecha: “left_hand_pZ_x”, “left_hand_pZ_x”, “right_hand_pZ_x” y “right_hand_pZ_x”, {Z: 0 <= Z < 21}.

Por último se añade una columna que almacenará el nombre de la seña a la que pertenece el frame, en este caso la columna fue llamada “sign”.

Una fila de esta tabla puede observarse en la Figura 27:

right_hand_p17_y	right_hand_p18_x	right_hand_p18_y	right_hand_p19_x	right_hand_p19_y	right_hand_p20_x	right_hand_p20_y	sign
3.0	3.0	3.0	3.0	3.0	3.0	3.0	hijo

Figura 27: Columnas del set de datos.

La cantidad resultante de columnas en el DataFrame es el valor ya nombrado de (1086 + 1), siendo este + 1 la columna que representa la palabra a la que pertenece cada fila. Por otro lado, como los videos presentan una longitud variable de frames, se toma el valor máximo previamente hallado (201), de manera que a los videos que no alcancen esta cantidad de frames se les concatenan frames (Padding) con valores fuera del intervalo que puede retornar Media Pipe, el valor 3.0 fue seleccionado para completar estos frames ya que este se encuentra fuera del intervalo. De esta manera se evita recortar información en esta primera instancia.

En pasos posteriores, al no haber recortado información, es posible saltear frames, intercalar, o realizar otras posibles acciones sobre estos videos ya procesados.

Una característica de este set de datos es que cuenta con los videos originales recortados, de manera que los videos comienzan y terminan con frames pertenecientes al movimiento de la seña y no cuenta con frames estáticos o sin movimiento alguno, es por esto que se toma el valor máximo como longitud para la lista de datos.

El resultado del procesamiento de los videos es almacenado en formato Parquet (Apache Parquet) con el uso de la librería Pandas.

"Apache Parquet es un formato de archivo de datos de código abierto orientado a columnas, diseñado para el almacenamiento y la recuperación eficiente de datos. Proporciona esquemas eficientes de compresión y codificación de datos con un rendimiento mejorado para manejar datos complejos en masa" [36].

6.2 Muestreo o Sampling

Una vez obtenidos los datos numéricos a través del pre-procesamiento de los videos, se procedió a tomar un subconjunto de frames de manera aleatoria mediante distintos enfoques con el fin de acotar la cantidad de frames que se utilizaron para el entrenamiento.

El video más corto, en lo que respecta a la cantidad de frames, posee un valor de 14, este dato fue tomado como referencia para realizar los siguientes muestreos, por ejemplo, sería posible tomar 10 frames, con el objetivo de luego poder descartar aleatoriamente ciertos frames.

A continuación se detallan 3 estrategias para acotar los videos a 10 y 30 frames (X):

- Muestreo Random: Se toma para cada video una cantidad de X frames aleatoriamente, siendo estos frames tomados del conjunto de frames/imágenes que componen cada video.

- Muestreo Random con Porcentajes: En este caso se separa cada video en 3 secciones: inicial, intermedia y final. La sección inicial está compuesta por frames aleatorios provenientes del primer 20% del video, luego el siguiente 60% de frames conforma la sección intermedia y el 20% restante representa la sección final. De cada sección se tomará la cantidad necesaria de frames de manera de poder completar el primer 20% de X, luego 60% y 20% restante. Por ejemplo: para un muestreo de 10 frames, se tomarán 2 frames de la sección inicial, 6 frames de la sección intermedia y por último 2 frames de la sección final.
- Muestreo Random con Poda: El tercer muestreo realiza una poda de frames del 15% de frames los iniciales y finales, es decir, una poda del 30%, exceptuando aquellos videos donde al quitar ese 30% la cantidad de frames restantes no es suficiente para completar la cantidad de X frames necesarios.

Estas 3 estrategias se utilizaron para armar sets de datos acotados a 10 y 30 frames, además, se armaron sets de datos con y sin datos faciales con el objetivo de analizar la performance ante la presencia de estos datos. Como salida de este paso, se obtuvieron 12 sets de datos, resultantes de la combinación de los distintos muestreos realizados para 10 o 30 frames, con y sin datos faciales.

Para los casos en los que se realizó muestreo con 30 frames y el video no contaba con la cantidad mínima para realizarlo, se permitió la repetición de frames aleatorios para alcanzar la cifra objetivo, siempre respetando el orden de aparición de los frames en todos los muestreos.

La diferencia con respecto al set de datos original es que en los sets resultantes cada fila representa un video y no un frame como sucedía anteriormente, de manera que cada fila queda compuesta de la siguiente manera:

[fr_0_face_0_x, fr_0_face_0_y, fr_1_face_1_x, ..., fr_9_right_hand_41_x, …] (fr = frame).

Durante la construcción de estos nuevos sets se disminuyó en parte la información almacenada, en este caso con respecto a los datos faciales y de la posición. Como se comentó en la sección 6.1, la información que se obtiene relacionada a la cara son 468 puntos faciales de los cuales no todos representan información valiosa, por lo que se acotó la información relacionada a la cara, pasando de un conjunto de 468 puntos de coordenadas a 68 puntos de coordenadas. Los puntos seleccionados corresponden a las cejas y los labios, los cuales son componentes que presentan movimientos claros a diferencia de las mejillas o mentón.

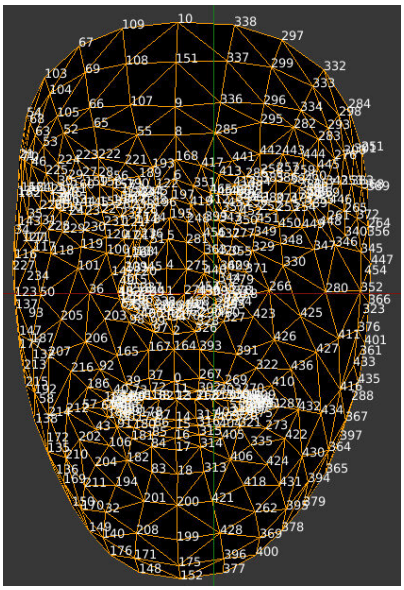


Figura 28: Puntos detectados por MediaPipe para la cara (Autor: Analytics Vidhya).

Otra disminución en la información tomada con respecto a los resultados obtenidos por MediaPipe se realizó sobre puntos de la posición, la lista de 33 puntos original se limitó a 23 de estos, ya que los puntos de la cintura hacia abajo no son utilizados en los videos como puede observarse en la Figura 29.

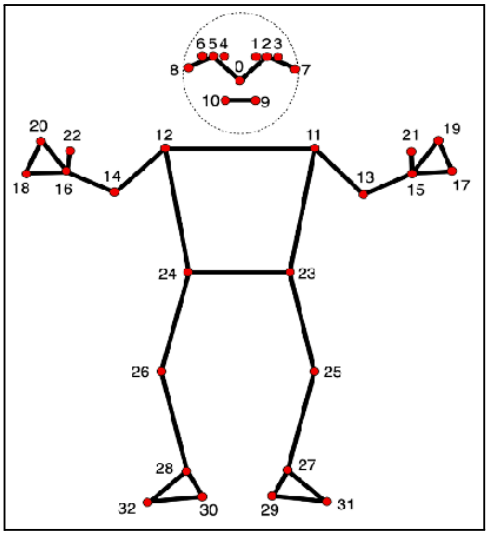


Figura 29: Puntos detectados por MediaPipe para la posición corporal (Autor: Google Developers).

6.3 Limpieza de Datos

La limpieza de datos consiste en quitar las columnas o conjuntos de datos compuestos por valores nulos o que poseen valores no válidos para el entorno del sistema. Es un proceso esencial para preparar los datos sin procesar para las aplicaciones de ML.

Es posible que los datos sin procesar contengan numerosos errores, que pueden afectar a la precisión de los modelos de ML y dar lugar a predicciones incorrectas.

Algunos pasos claves para la limpieza de datos son modificar y eliminar campos de datos incompletos e incorrectos, identificar y eliminar información duplicada y datos sin relación, y corregir el formato, los valores faltantes y los errores de ortografía.

Los tipos de limpieza utilizados son;

- Limpieza por Correlación: Se mide la dependencia entre las distintas características o *features* del set de datos. Se establece un umbral máximo, por ejemplo 0.7, aquellas características que superen este valor se eliminan al estar altamente relacionadas.
- Limpieza por Varianza: Elimina todas las características de baja varianza. Se establece un umbral de valor mínimo, por ejemplo 0.1, las características con

una varianza del conjunto de entrenamiento inferior a este umbral se eliminarán.

También es posible realizar limpieza mediante “Outliers” para ignorar aquellas observaciones que poseen valores atípicos o fuera de la distribución esperada para determinada característica, aunque en este trabajo no fue utilizado ya que los valores se encuentran normalizados dentro del intervalo [0, 1] por MediaPipe.

Luego de haber realizado la limpieza de datos sobre los distintos sets de datos se obtuvieron sets equivalentes pero con una menor cantidad de columnas, habiéndo eliminado de manera automática aquellas que tenían una alta correlación o baja varianza.

6.4 Entrenamiento y Optimización de Modelos

En el siguiente paso se realizó la construcción, entrenamiento y optimización de los modelos sobre los distintos sets de datos obtenidos en el paso previo. Como primer modelo se utilizaron árboles de decisión a través del uso de XGBoost, luego mediante el uso de Redes Neuronales Recurrentes se construyó un modelo con capas LSTM y otro modelo con capas GRU.

Luego de haber construido estos modelos y realizado el entrenamiento sobre un porcentaje de cada set de datos (partición de entrenamiento/train), se realizó una optimización de los hiperparámetros, con el fin de mejorar la performance de cada modelo. En todos los modelos entrenados se analizaron las siguientes métricas: precisión, recall, F1, F05, F2, roc_auc, accuracy.

La columna categórica representativa de la seña a la que pertenece cada fila de los sets de datos fue transformada por un equivalente numérico (Label Encoding) antes de realizar la separación en subconjuntos.

Todos los sets de datos fueron separados en 3 subconjuntos, un subconjunto de entrenamiento (train data) con el 80% de datos, con la partición compuesta del 20% restante

se armó un conjunto de testeo (test data) con el 80% de esta partición, luego el conjunto de validación (validation data) con el 20% respectivamente.

6.4.1 Modelo con XGBoost

El primer modelo desarrollado se basó en árboles de decisión con la particularidad de que este utiliza un algoritmo del descenso del gradiente para su optimización, de allí obtiene su nombre en inglés XGBoost (Xtreme Gradient Boosting).

Este modelo recibe como entrada las distintas filas correspondientes de cada set de datos y subconjunto con el que se trabaje, es decir, la estructura con la que se conformaron los distintos sets de datos puede usarse como entrada sin necesidad de realizar modificación alguna sobre estos, con excepción del paso “Label Encoding” nombrado previamente.

Debido a que el conjunto de datos final obtenido para realizar el entrenamiento no es de gran tamaño (1050 filas), el tiempo empleado en completar el entrenamiento es veloz, este varía entre 5 a 10 minutos dependiendo de si se incluyen datos faciales y 10/30 frames. Dentro de este tiempo de entrenamiento se realizó una validación cruzada sobre el conjunto de entrenamiento dividiendo el conjunto en 6 grupos o lotes. El equipo utilizado y en el cual se trabajó posee los siguientes componentes: Intel I5 8va generación, GPU GTX 1050 2GB, aunque también se trabajó paralelamente sobre Google Colab el cual permite el uso de manera gratuita de una CPU o T4 GPU, 12.7 GB de memoria RAM, 15GB de GPU-RAM y 78.2 GB de almacenamiento en disco.

En la Tabla 3 se presenta la desviación estándar o ‘std’ que se obtuvo para algunos set de datos con y sin datos faciales y 10/30 frames, evaluado sobre el modelo de XGBoost:

Métrica	**03 Criterio Random 30 frames sin cara**	**04 Criterio Random 30 Frames con cara**	**05 Criterio Proporción 10 Frames sin cara**	**07 Criterio Proporción 30 frames sin cara**
Precisión	0.02	0.03	0.02	0.02
Recall	0.03	0.03	0.02	0.03
F1	0.03	0.03	0.02	0.03

F05	0.03	0.03	0.02	0.03
F2	0.03	0.03	0.02	0.03
Accuracy	0.03	0.03	0.02	0.03

Tabla 3: Desviación estándar sobre sets de datos 03, 04, 05 y 07 sobre métricas precisión, recall, F1, F05, F2 y accuracy.

Los resultados observados en la tabla representan, en este caso, la desviación estándar de cada métrica sobre los resultados obtenidos por la función de validación cruzada o cross_validate. Un valor bajo como lo es 0.03/0.02 indica una variabilidad baja entre los diferentes lotes que fueron conformados por la validación cruzada. El resultado bajo de 'std' indica que los valores tienden a estar cerca de la media o que se encuentran menos dispersos, a su vez, indica que el modelo es menos susceptible a variaciones en los datos.

6.4.2 Modelos con LSTM/GRU

El segundo y tercer modelo desarrollado están compuestos por capas pertenecientes a Redes Neuronales Recurrentes siendo estas capas LSTM para el segundo y capas GRU para el tercero respectivamente. Los modelos fueron construidos de manera secuencial, de forma que las capas se añaden una luego de otra.

Dentro de las arquitecturas de Redes Neuronales no existe un instructivo que indique cómo se deben colocar estas capas, en qué orden y/o cantidad de neuronas por capa, es por esto que se definieron los modelos de la siguiente manera:

El modelo LSTM (Figura 30)está compuesto por 2 capas LSTM, ambas capas compuestas por 128 unidades y una función de activación "tanh", esto debido a que el framework utilizado para construir estos modelos (Tensorflow) permite el uso de GPU al utilizar específicamente la función "tanh". Estas primeras capas se encuentran seguidas de 3 capas Densas o Fully-connected, estas son capas donde cada neurona recibe como entrada todas las salidas de las neuronas de la capa previa. La cantidad de unidades para estas últimas

capas son de 64, 32 y 21 respectivamente, siendo el valor 21 el que representa a la cantidad de clases o señas que el modelo es capaz de interpretar y que se usan como salida u output del modelo.

```
model = Sequential()
model.add(LSTM(128, input_shape=(10,130), return_sequences=True, activation=activation_))
model.add(LSTM(neurons_input, return_sequences=False, activation=activation_))
model.add(Dense(64, activation=activation_))
model.add(Dense(32, activation=activation_))
model.add(Dense(np.array(le.classes_).shape[0], activation='softmax'))

optimizer = Adam(learning_rate=learning_rate_)
```

Figura 30: Arquitectura LSTM.

Por otro lado, el modelo GRU (Figura 31) cambia sus primeras 2 capas por las correspondientes capas GRU.

```
model = Sequential()
model.add(GRU(128, input_shape=(10,130), return_sequences=True, activation=activation_))
model.add(GRU(neurons_input, return_sequences=False, activation=activation_))
model.add(Dense(64, activation=activation_))
model.add(Dense(32, activation=activation_))
model.add(Dense(np.array(le.classes_).shape[0], activation='softmax'))

optimizer = Adam(learning_rate=learning_rate_)
```

Figura 31: Arquitectura GRU.

Para ambos modelos (LSTM/GRU) se realizó una transformación sobre las filas de los sets de datos para entregar al modelo la estructura correspondiente que necesita como entrada. La transformación realizada agrupa para cada fila, los datos pertenecientes a cada frame en una lista o arreglo y luego, a su vez, se agrupan estos frames dentro de una lista que representa a esta fila o video. Por último se agruparon todas las filas en una sola lista que sirve de entrada al modelo. Esta transformación se realizó con el objetivo de transformar los datos tabulares en series temporales.

Ambos modelos pueden visualizarse de manera gráfica en la Figura 32:

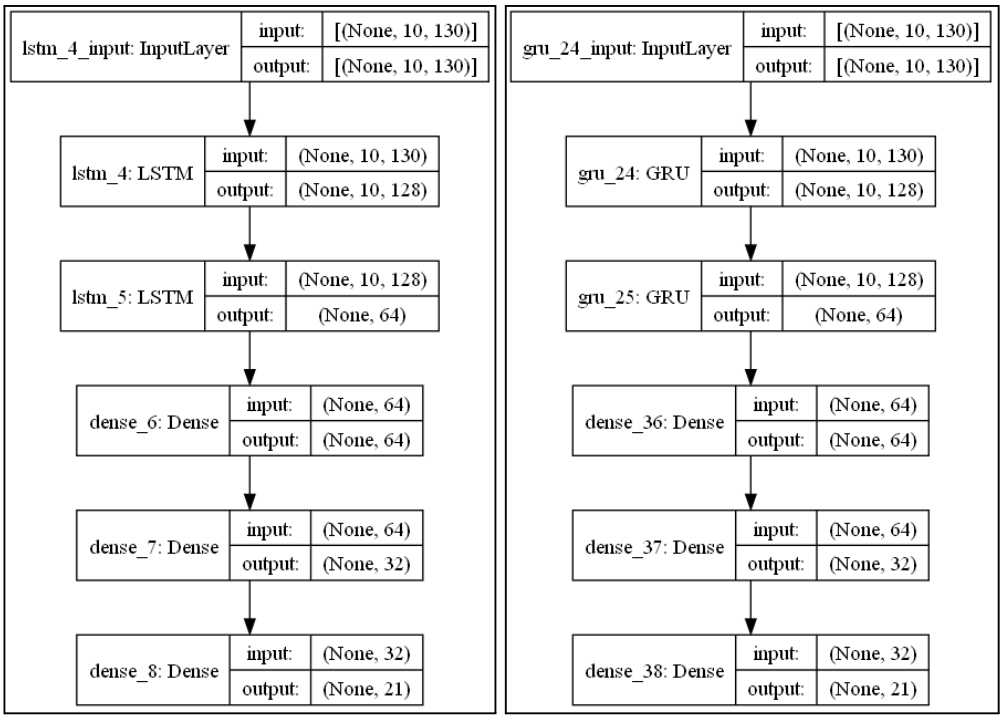


Figura 32: Estructura gráfica de red neuronal con capas LSTM (izquierda) y capas GRU (derecha)

6.4.3 Optimización

Finalizado el entrenamiento y evaluación de métricas sobre el modelo base, se realizó una optimización de hiperparámetros sobre los modelos con el fin de elevar, de ser posible, el rendimiento de los modelos. Los hiperparámetros evaluados para el modelo de XGBoost son los siguientes**:**

- **learning_rate**: Tasa de aprendizaje que controla la contribución de cada árbol. Un valor más bajo puede hacer que el modelo sea más preciso, pero también puede requerir más árboles y tiempo de entrenamiento.
- **n_estimators**: Número de árboles en el modelo. Un valor más alto puede mejorar el rendimiento, pero también aumentará el tiempo de entrenamiento.
- **max_depth**: Profundidad máxima de cada árbol. Controla la complejidad del modelo. Un valor más alto puede capturar relaciones más complejas, pero también puede provocar sobreajuste.

- **min_child_weight**: Peso mínimo necesario para crear un nuevo nodo en el árbol. Controla la regularización y ayuda a evitar divisiones insignificantes. Un valor más alto puede evitar el sobreajuste.
- **subsample**: Proporción de muestras utilizadas para entrenar cada árbol. Un valor menor reduce la varianza y puede evitar el sobreajuste.
- **colsample_bytree**: Proporción de características utilizadas para entrenar cada árbol. Un valor menor reduce la varianza y puede evitar el sobreajuste.
- **reg_alpha**: Término de regularización L1 (Lasso) aplicado a los pesos de los nodos del árbol. Un valor más alto impone una mayor regularización.
- **reg_lambda**: Término de regularización L2 (Ridge) aplicado a los pesos de los nodos del árbol. Un valor más alto impone una mayor regularización.

Los hiperparámetros evaluados para redes LSTM/GRU:

- **neurons_input**: Establece el número de neuronas o unidades que estarán presente en las capas donde se define este valor como variable.
- **learning_rate:** Define la rapidez con la que la red actualiza sus parámetros. Establecer una tasa de aprendizaje alta acelera el aprendizaje, pero el modelo puede no converger. Por el contrario, una tasa baja ralentiza drásticamente el aprendizaje, ya que los pasos hacia el mínimo de la función de costo serán minúsculos, pero permitirá que el modelo converja sin problemas.

En Tabla 4 y Tabla 5 pueden observarse los resultados obtenidos para las distintas métricas luego de haberse realizado el entrenamiento y posterior optimización de los modelos mencionados. Estos resultados se obtuvieron al evaluar el subconjunto de test para cada set. De los 12 sets de datos construidos con las distintas técnicas de muestreo, se tomaron los que presentaron mejores resultados y son estos los analizados en las tablas indicadas, interpretando como mejor resultado a los modelos que presentan un valor elevado para la

métrica F1. Se eligió esta métrica ya que su valor se basa en los resultados de las métricas Precisión y Recall como se habló en la sección “3.2.10.4”, de manera que con un solo valor podemos comparar los distintos modelos.

Set de Datos	*métrica/modelo*	*Precisión*	*Recall*	*F1*
03 Criterio Random 30 frames sin cara	*XGB*	*0.92*	*0.91*	*0.91*
	LSTM	*0.93*	*0.92*	*0.92*
	GRU	*0.92*	*0.90*	*0.90*
05 Criterio Proporción 10 frames sin cara	*XGB*	*0.93*	*0.91*	*0.91*
	LSTM	*0.90*	*0.89*	*0.89*
	GRU	*0.91*	*0.90*	*0.89*
08 Criterio Proporción 30 con cara	*XGB*	*0.95*	*0.93*	*0.93*
	LSTM	*0.92*	*0.90*	*0.93*
	GRU	*0.91*	*0.90*	*0.90*

Tabla 4: Resultados para subconjuntos de testing luego de entrenamiento.

Set de Datos	*Modelo\Métrica*	*Precisión*	*Recall*	*F1*
03 Criterio Random 30 frames sin cara	*XGB*	*0.92*	*0.91*	*0.91*
	LSTM	*0.93*	*0.92*	*0.92*
	GRU	*0.92*	*0.91*	*0.91*
05 Criterio Proporción 10 frames sin cara	*XGB*	*0.91*	*0.90*	*0.90*
	LSTM	*0.90*	*0.89*	*0.89*
	GRU	*0.93*	*0.92*	*0.92*
08 Criterio Proporción 30 con cara	*XGB*	*0.92*	*0.91*	*0.91*
	LSTM	*0.92*	*0.90*	*0.90*
	GRU	*0.84*	*0.82*	*0.82*

Tabla 5: Resultados para subconjuntos de testing luego de optimización de hiperparámetros.

Sumado a los resultados obtenidos sobre los subconjuntos de testing, en la Tabla 6 se presentan los resultados obtenidos luego de optimizar para los subconjuntos de validación.

Set de Datos	*Modelo\Métrica*	*Precisión*	*Recall*	*F1*
03 Criterio Random 30 frames sin cara	*XGB*	*0.95*	*0.94*	*0.94*
	LSTM	*0.93*	*0.92*	*0.92*
	GRU	*0.94*	*0.94*	*0.94*
05 Criterio Proporción 10 frames sin cara	*XGB*	*0.93*	*0.92*	*0.92*
	LSTM	*0.95*	*0.94*	*0.94*
	GRU	*0.95*	*0.94*	*0.94*
08 Criterio Proporción 30 con cara	*XGB*	*0.92*	*0.92*	*0.92*
	LSTM	*0.94*	*0.93*	*0.93*
	GRU	*0.89*	*0.87*	*0.87*

Tabla 6: *Resultados para subconjuntos de validación luego de optimización de hiperparámetros.*

En base a los resultados obtenidos luego de optimizar y teniendo en cuenta la métrica F1 analizada en la Tabla 5, de los 3 sets de datos seleccionados se observó que presentan resultados que varían entre 0.82 - 0.92. Para la elección de la técnica de muestreo a utilizar en el sistema web, al presentar los resultados de la Tabla 5 valores similares, se realizó una comparación de los mismos como puede observarse en la Tabla 7 y un promedio de la métrica F1 en base a esta tabla y los cuales pueden observarse en la Tabla 8. Se realizó esta comparación y cálculo de promedios de los 3 modelos (Tabla 8) para cada set con el objetivo de seleccionar el set de datos que entregó mejores resultados para todos los modelos.

Set de Datos	*03 Criterio Random 30 frames sin cara*	*05 Criterio Proporción 10 frames sin cara*	*08 Criterio Proporción 30 con cara*
XGB	*0.91*	*0.90*	*0.91*
LSTM	*0.92*	*0.89*	*0.90*

GRU	*0.91*	*0.92*	*0.89*

Tabla 7: Comparación de métrica F1 para sets de datos 03, 05 y 08 basados en Tabla 5.

Set de Datos	*Promedio F1*
03 Criterio Random 30 frames sin cara	*0.913*
05 Criterio Proporción 10 frames sin cara	*0.903*
08 Criterio Proporción 30 con cara	*0.876*

Tabla 8: Promedio métrica F1 para sets de datos 03, 05 y 08 basados en Tabla 7.

Como puede observarse en la Tabla 7, al comparar los resultados de los modelos sobre los distintos sets de datos, se observó que los resultados predominantes o de mayor valor en la mayoría de los casos (2/3) se encontraron para el set de datos “03 Criterio Random 30 frames sin cara” en base al análisis de la métrica F1. En la Tabla 8 se observan los promedios obtenidos para cada set de datos llegando a la misma conclusión sobre el set de datos “03” por lo que se implementará un muestreo Random con 30 frames en el sistema en tiempo real para entregar datos al modelo.

Seleccionada la estrategia de muestreo, se procedió a seleccionar el modelo de entre XGBoost, LSTM y GRU para integrar en el sistema web. De entre los valores de XGboost (0.94), LSTM (0.92) y GRU (0.94) (Tabla 6), valores similares, se eligió el modelo de XGBoost ya que no es necesario realizar transformación alguna sobre los datos que se entregan al modelo, teniendo en cuenta que estos modelos se utilizarán para interpretar señas en tiempo real, priorizando los tiempos de respuesta.

A continuación, en las Figuras 33, 34 y 35 pueden observarse las matrices de confusión luego del proceso de optimización sobre el set de datos “03 Criterio Random 30 frames sin cara” sobre los conjuntos de validación.

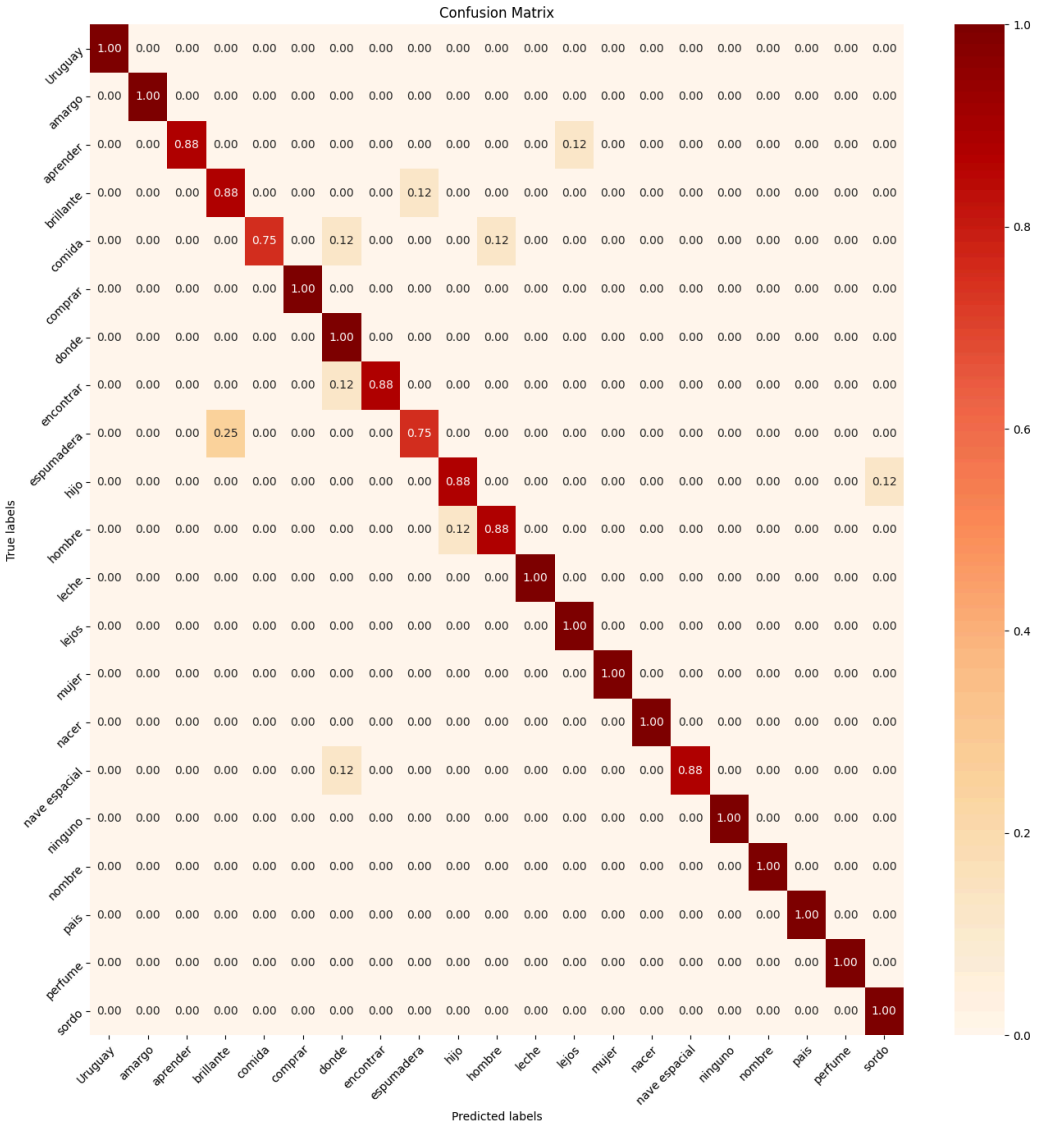


Figura 33: Matriz de confusión sobre conjunto de validación para set de datos 03 luego de optimización XGBoost.

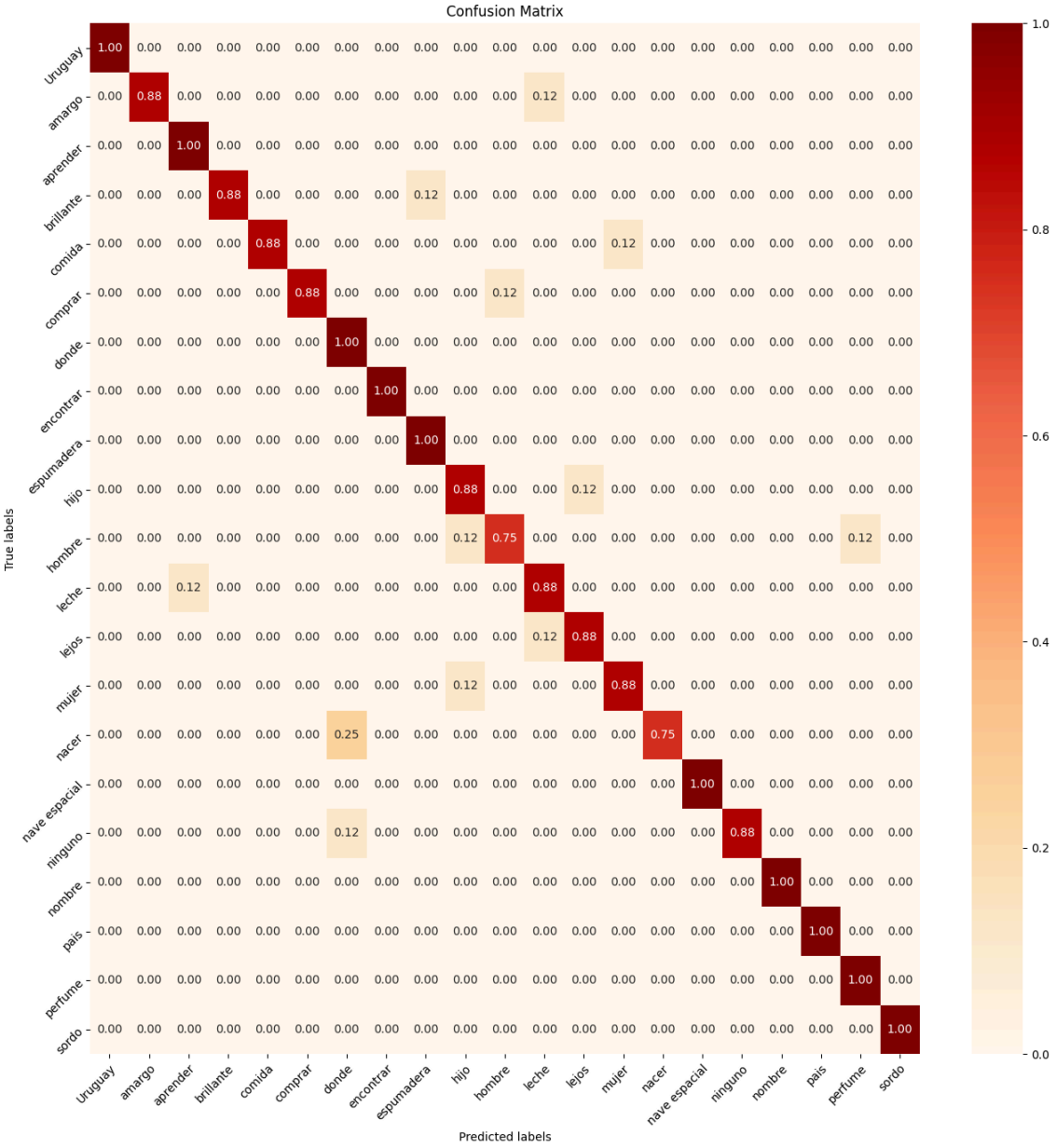


Figura 34: Matriz de confusión sobre conjunto de validación para set de datos 03 luego de optimización LSTM.

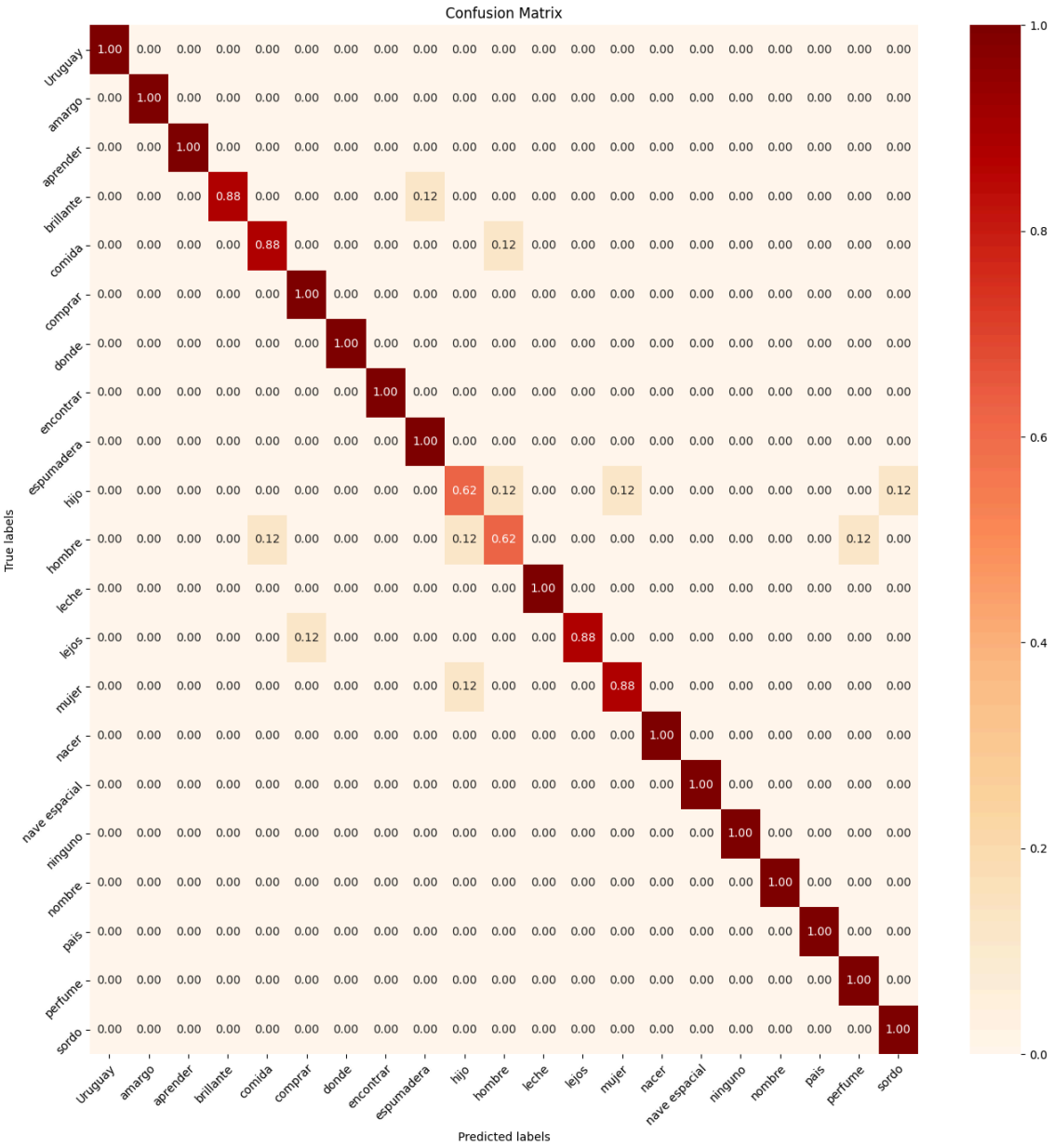


Figura 35: Matriz de confusión sobre conjunto de validación para set de datos 03 luego de optimización GRU.

6.4.4 Conclusión

De los 3 modelos analizados sobre los distintos sets de datos, se eligió para continuar con la construcción del sistema web el modelo de XGBoost entrenado con datos obtenidos de la estrategia de muestreo Random con 30 frames: “03 Criterio Random 30 frames sin cara”. Su elección está basada en el rendimiento obtenido (0.94 de métrica F1), en su velocidad de

respuesta (0.08ms) y en la facilidad de entregar datos al modelo sin realizar transformación alguna sobre la estructura de los datos que se entregan al mismo. El modelo elegido fue almacenado con una función (save_model) propia del paquete que proporciona XGBoost y el cual facilita su posterior carga (load_model) y uso.

6.5 Sistema Web

El actual proyecto de investigación tiene por objetivo general: “Desarrollar un sistema web capaz de interpretar una secuencia de imágenes que contienen a personas realizando movimientos pertenecientes a la Lengua de Señas Argentina”. Partiendo de este objetivo general y al haberlo desglosado en una lista de objetivos específicos, se nombran los siguientes objetivos que guían la actual etapa:

- Prototipar e implementar la interfaz gráfica de un sistema web.
- Integrar el modelo de red neuronal en el sistema web.

6.5.1 Diagrama de Secuencia

En la Figura 36 se presenta un diagrama de secuencia regido por la siguiente descripción de lo que se pretende realizará el sistema:

“El sistema deberá conectarse a la cámara del dispositivo donde se ejecute. Una vez que el sistema tiene acceso a la cámara, procederá a captar frames y realizará la detección de la persona en la imagen. El usuario será capaz de:

- Seleccionar el valor de umbral sobre el cual el resultado de una interpretación se toma como válida.
- Cambiar la forma de capturar imágenes de la cámara (modo selfie o no).
- Seleccionar un video para su interpretación.

Una vez que se tiene la cantidad necesaria de frames para realizar la interpretación de una seña, se enviará esta lista al endpoint de backend encargado de procesar esa información.

El backend recibirá esta lista y según el modelo elegido en frontend, se realizará la interpretación. El resultado será devuelto al frontend/navegador donde, en caso de haber interpretado un seña, se mostrará la seña interepretada.

Para una correcta interpretación, la persona deberá situarse al centro de la imagen, a una distancia aproximada de 2 metros frente a la cámara.”

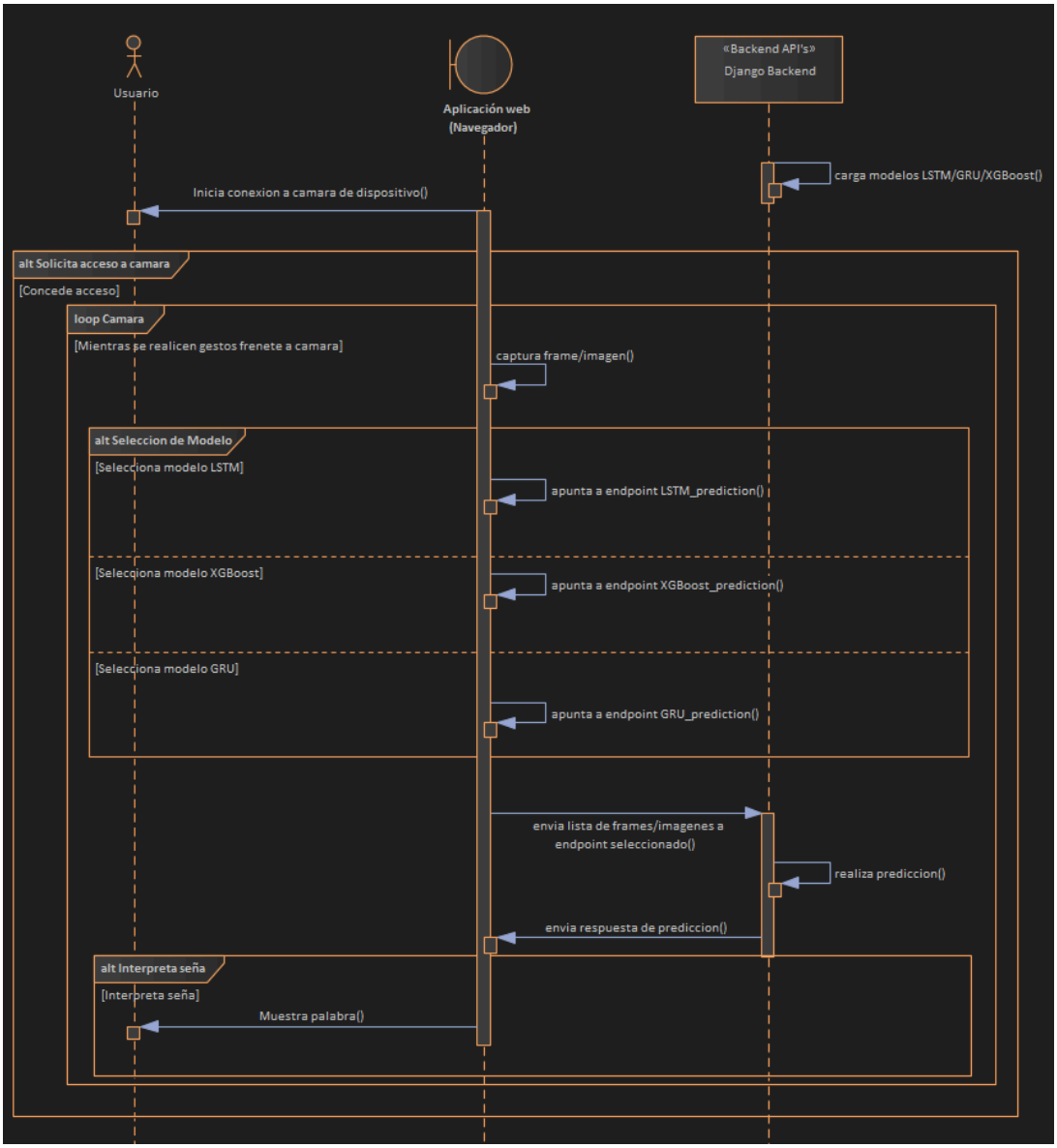


Figura 36: Diagrama de Secuencia.

6.5.2 Prototipo de Interfaz de Usuario.

Al momento de comenzar el prototipado de las interfaces se definió la cantidad de pantallas a visualizar, qué información es necesaria mostrar en cada pantalla y su funcionalidad.

El sistema web desarrollado cuenta con las siguientes pantallas definidas:

1. Pantalla principal o landing page: esta pantalla tiene por objetivo recibir al usuario e introducirlo al funcionamiento del sistema.
2. Pantalla de interpretación: en esta pantalla el usuario podrá ver lo que se encuentre capturando su cámara y las interpretaciones (resultados) realizadas por el modelo elegido luego de haber capturado un conjunto de frames. El usuario podrá seleccionar distintas opciones como el modelo, el umbral que utilizará el modelo para tomar una interpretación como correcta, y la opción para marcar si es zurdo o no.
3. Pantalla detalles técnicos: por último se añade una pantalla con los detalles técnicos que posee el sistema, como lo es la tecnología utilizada en el desarrollo del mismo.

Estas pantallas han sido diseñadas teniendo en cuenta su uso en dispositivos móviles como computadoras de escritorio, es por esto que se las denomina “Responsive”, ya que se adaptan visualmente al alto y ancho de la pantalla donde se utilice.

En las Figuras 37, 38 y 39 se presentan los prototipos construidos en Figma.

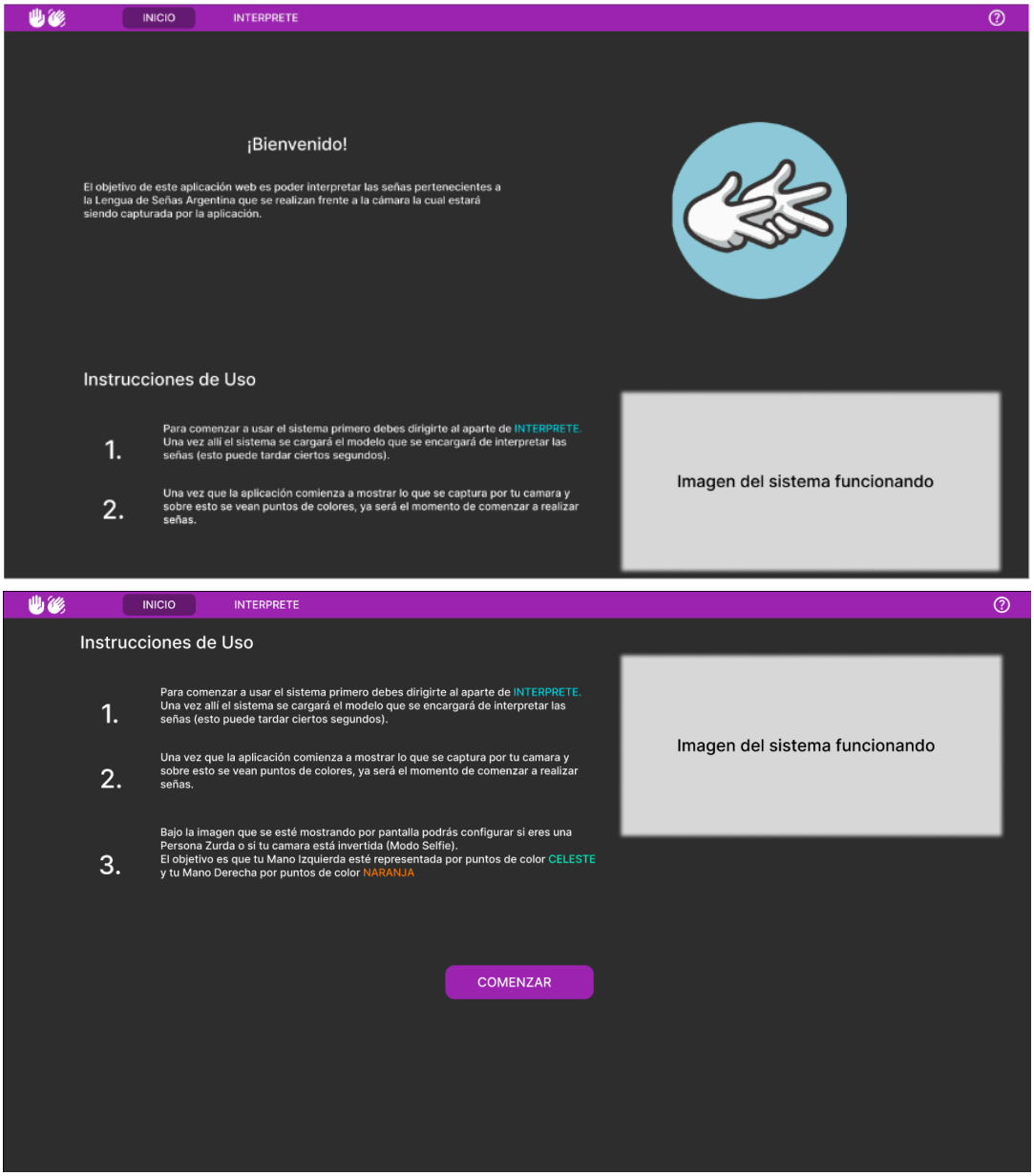


Figura 37:Pantalla de Presentación o Landing Page.

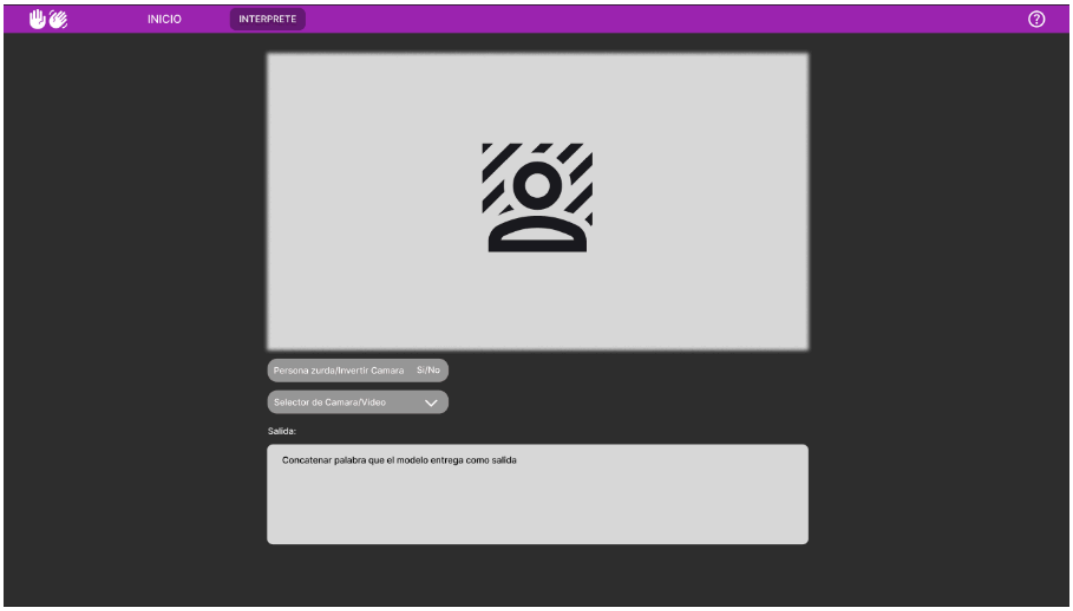


Figura 38: Pantalla de Captura de Imagen e Interpretación de señas.

Figura 39: Pantalla de Tecnología usada.

Como iconos para el sistema, a pesar de que es una cantidad ínfima la utilizada, se tomaron los iconos gratuitos de Google Material Symbols. Por otro lado, las imágenes utilizadas en el sistema fueron obtenidas de Google.

Una vez realizados los prototipos, se procedió a maquetar e implementarlos con tecnologías web.

Estas pantallas y el sistema web, se realizaron con el Framework Angular, la elección de este Framework por sobre otros está basada en experiencia previa de trabajo sobre el mismo. Angular nos permitirá crear una SPA o Single Page Application, es decir, una aplicación de una sola página donde el contenido variará dinámicamente según la interacción del usuario sin tener que recargar la página por cada interacción realizada.

6.5.3 Integración del modelo al sistema

Siguiente objetivo: integrar las librerías y modelos de RN que permiten al sistema realizar la captura e interpretación de la secuencia de imágenes.

Para la obtención de los puntos corporales en la web, se utilizó la versión para Javascript de “MediaPipe” en el frontend. La lógica utilizada en frontend para la recolección y envío de información es la siguiente: cada frame capturado por la webcam o cámara de dispositivo es enviado a MediaPipe para que éste detecte y retorne los puntos corporales presentes en dicho frame, los resultados son almacenados en una lista que hará la función de ventana deslizante en los casos que el resultado de la interpretación no supere el umbral, descartando el frame más antiguo, mientras que en caso de un resultado favorable (mayor o igual al umbral indicado) esta lista es vaciada o reiniciada para no tener en cuenta los datos de una seña ya interpretada. Esta lista se completa con la información de 30 frames, que es la cantidad que esperan como entrada los modelos elegidos. Una vez el frontend recolecta la cantidad de información necesaria, envía esta información a una API de backend donde residen los modelos y se realiza la interpretación.

El backend fue construido con Python y Django, la elección de esta tecnología se basa en experiencia previa de uso y también en la facilidad de instalar y utilizar los mismos paquetes con los que se construyeron los modelos ya que se utiliza el mismo lenguaje.

Este backend se encarga de cargar el/los modelos necesarios para realizar la interpretación y define un endpoint que puede ser consumido desde el frontend.

6.5.4 Arquitectura del sistema

La arquitectura del sistema desarrollado está compuesta por 2 niveles, siendo uno de ellos el frontend o la parte visual del sistema y con la que interactúa de manera directa el usuario; por otro lado se tiene al backend, el encargado de manejar las peticiones que realiza el frontend. En la Figura 40 puede observarse la arquitectura del sistema. Al no ser necesario almacenar información o persistir datos no se realiza la conexión a una base de datos.

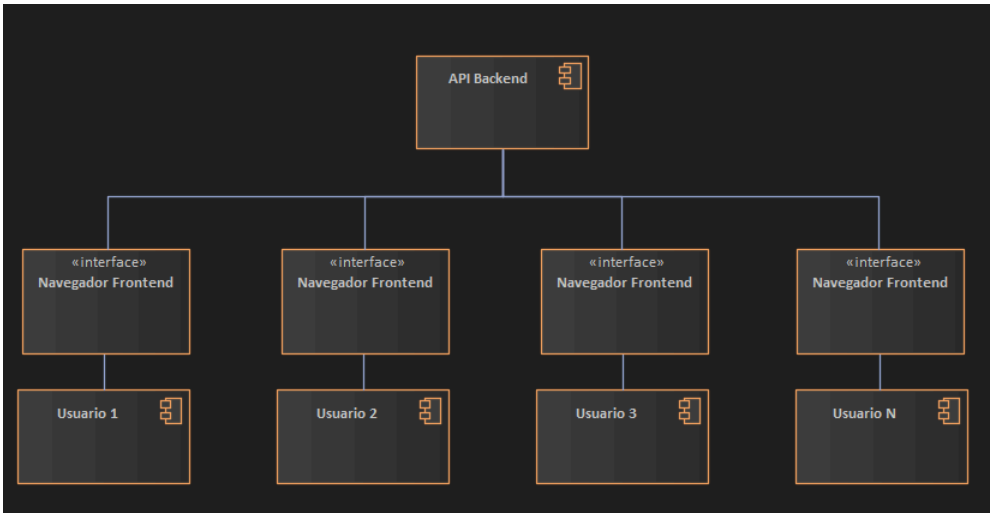


Figura 40: Arquitectura del sistema web.

7. Conclusión

Se desarrolló un sistema web intérprete de la Lengua de Señas Argentina que permite interpretar un conjunto de señas en tiempo real desde cualquier dispositivo con conexión a internet y una cámara o webcam.

Para la construcción de este sistema fue necesario analizar distintas opciones de modelos de Machine Learning y elección de alternativas específicas adecuadas al problema en investigación. Como modelos que realizan la interpretación se propuso: un enfoque orientado hacia árboles de decisión con XGBoost y dos enfoques de Redes Neuronales Recurrentes mediante el uso de capas LSTM y GRU, orientando el problema como un análisis de series temporales. Para poder realizar el entrenamiento de estos modelos se utilizó un set de datos

preexistente conocido como “LSA 64” y el cual cuenta con un amplio conjunto de videos. Sobre estos videos se realizó un preprocesamiento del cual se obtuvo como resultado un conjunto de puntos corporales por cada frame de cada video seleccionado. Una vez obtenido el equivalente numérico de cada video, se acotó la cantidad de datos a tomar para el entrenamiento. Se definieron 3 estrategias de muestreo de datos que permitieron realizar un muestreo y reducción de los datos. Una vez obtenidos estos subconjuntos del set de datos original se realizó una limpieza de los mismos por correlación y por varianza. Luego se procedió a entrenar los modelos sobre cada nuevo conjunto de datos.

Finalizado el entrenamiento se realizó un paso de optimización de hiperparámetros con el objetivo de elevar el rendimiento de cada modelo.

Set de Datos	*Modelo\Métrica*	*Precisión*	*Recall*	*F1*
Resultados sobre subconjuntos de Testing				
03 Criterio Random 30 frames sin cara	*XGB*	*0.92*	*0.91*	*0.91*
	LSTM	*0.93*	*0.92*	*0.92*
	GRU	*0.92*	*0.91*	*0.91*
05 Criterio Proporción 10 frames sin cara	*XGB*	*0.91*	*0.90*	*0.90*
	LSTM	*0.90*	*0.89*	*0.89*
	GRU	*0.93*	*0.92*	*0.92*
08 Criterio Proporción 30 con cara	*XGB*	*0.92*	*0.91*	*0.91*
	LSTM	*0.92*	*0.90*	*0.90*
	GRU	*0.84*	*0.82*	*0.82*
Resultados sobre subconjuntos de Validación				
Set de Datos	*Modelo\Métrica*	*Precisión*	*Recall*	*F1*
03 Criterio Random 30 frames sin cara	*XGB*	*0.95*	*0.94*	*0.94*
	LSTM	*0.93*	*0.92*	*0.92*
	GRU	*0.94*	*0.94*	*0.94*

Set de Datos	*Modelo\Métrica*	*Precisión*	*Recall*	*F1*
05 Criterio Proporción 10 frames sin cara	*XGB*	*0.93*	*0.92*	*0.92*
	LSTM	*0.95*	*0.94*	*0.94*
	GRU	*0.95*	*0.94*	*0.94*
08 Criterio Proporción 30 con cara	*XGB*	*0.92*	*0.92*	*0.92*
	LSTM	*0.94*	*0.93*	*0.93*
	GRU	*0.89*	*0.87*	*0.87*

Tabla 9: Resultados luego de optimización de hiperparámetros.

Finalizado el entrenamiento y optimización se eligió sobre todos los modelos, el modelo con mejor performance, resultando elegido el modelo construido con XGBoost para el set de datos “03 Criterio Random 30 frames sin cara” que utilizó la estrategia de muestreo random con 30 frames para su construcción. El modelo seleccionado pasó a integrarse en el sistema web.

El paso siguiente consistió en la definición del sistema web, diferenciando los componentes de frontend y backend. En el lado del frontend se diseñaron pantallas responsive con las que el usuario puede interactuar, mientras que, en el backend se definieron los distintos endpoints que el frontend consumirá para procesar los datos recolectados por el mismo. Además, en backend, se integró en el endpoint correspondiente el uso del modelo seleccionado.

7.1 Trabajo Futuro

Como resultado de la presente investigación se proponen ciertas mejoras que pueden realizarse sobre el sistema obtenido y/o sobre los modelos integrados en el sistema.

Es posible ampliar la cantidad de señas utilizadas para el entrenamiento, permitiendo aumentar las señas que el sistema es capaz de reconocer.

El diseño en capas del sistema, permite diseñar nuevos modelos e integrarlos en el mismo backend en un nuevo endpoint sin mayores complicaciones, estableciendo su correspondiente consumo en el frontend.

Como mejora o aumento de la capacidad del sistema, se puede añadir un apartado capaz de capturar y almacenar grabaciones de nuevas señas en una base de datos para la construcción o ampliación de un nuevo set de datos. Estas grabaciones requieren de gente especializada en la Lengua de Señas para analizar la correctitud de los videos que se almacenan.

Otra posible mejora es la integración de procesamiento de lenguaje natural o NLP por sus siglas en inglés Natural Language Processing, para ir desde una frase armada en Lengua de Señas hacia una oración legible ya que, como se mencionó anteriormente, las frases que una persona sorda arma poseen una estructura diferente a la que usamos cotidianamente personas no sordas.

Se pueden construir modelos capaces de detectar los tiempos verbales en la frase al momento de interpretar una seña.

Por último se puede incrementar la robustez de los modelos añadiendo nueva información como datos espaciales además de los temporales que tienen en cuenta los modelos actuales.

8. Referencias

[1] Moryossef Amit, Goldberg Yoav: Sign Language Procesing. Recuperado el 01 de Agosto del 2023 de https://research.sign.mt/

[2] Cámara de Diputados de la Provincia de San Juan. Ley Provincial N° 7412 (2003).

[3] Cámara de Diputados de la Provincia de San Juan. Ley Provincial N° 761-S (2014).

[4] Web: Confederación Argentina de Sordos. Recuperado el 19 de Febrero del 2024 de https://cas.org.ar/prensa/preguntas-frecuentes/.

[5] Confederación Argentina de Sordos. Status Lingüístico de las Lenguas de Señas en el mundo. (s.f.). Recuperado el 31 de Agosto del 2023 de https://inalsa.cas.org.ar/nuestra-lsa/la-ls-en-el-mundo/

[6] El Naqa, I., Murphy, M.J.: What Is Machine Learning?. En: El Naqa, I., Li, R., Murphy, M. (eds) Machine Learning in Radiation Oncology. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18305-3_1

[7] Ethem Alpaydin: Introduction to Machine Learning. 4ta edición, The MIT Press (págs. 3-4) (2020)

[8] Zachary C. Lipton & John Berkowitz: A Critical Review of Recurrent Neural Networks for Sequence Learning. (pág. 1) (2015).

[9] Simeone, O.: A Very Brief Introduction to Machine Learning With Applications to Communication Systems. IEEE Transactions on Cognitive Communications and Networking, 4, 648-664. (págs. 1-3) (2018).

[10] Fernando Izaurieta y Carlos Saavedra . Redes Neuronales Artificiales. Departamento de Física, Universidad de Concepción, Concepción, Chile (pág. 1) (s.f.).

[11] Pedro I. Viñuela, Inés M. G. León: Redes de Neuronas Artificiales. Un Enfoque Práctico. (págs. 10-15, 26-54) (2004).

[12] Luthfi Ramadhan: Neural Network: The Dead Neuron. The biggest drawback of ReLU activation function. (2021). Recuperado el 31 de Agosto del 2023 de https://towardsdatascience.com/neural-network-the-dead-neuron-eaa92e575748

[13] Jason Brownlee: A Gentle Introduction to Cross-Entropy for Machine Learning. Sitio Web. (2019). Recuperado el 31 de Agosto del 2023 de https://machinelearningmastery.com/cross-entropy-for-machine-learning/

[14] J.A. Martínez Pérez y P.S. Pérez Martin: La curva ROC (26 de abril de 2022). Recuperado el 20 de Febrero del 2024 de https://doi.org/10.1016/j.semerg.2022.101821

[15] Carlos Arana: Redes Neuronales Recurrentes: Análisis de los Modelos Especializados en Datos Secuenciales, CEMA Working Papers: Serie Documentos de Trabajo. 797, Universidad del CEMA (2021).

[16] Pedro Larrañaga, Iñaki Inza, Abdelmalik Moujahid. Tema 8: Redes Neuronales. Departamento de Ciencias de la Computación e Inteligencia Artificial. Universidad del País Vasco–Euskal Herriko Unibertsitatea. (s.f.).

[17] Andrés Mañas Mañas: Notas sobre pronóstico del flujo de tráfico en la ciudad de Madrid.. Métodos basados en Deep Learning. (2019). Recuperado el 01 de Septiembre del 2023 de https://bookdown.org/amanas/traficomadrid/resumen.html

[18] L. Enrique Sucar, Giovani Gómez: Vision Computacional. (2013). Recuperado el 01 de Septiembre del 2023 de https://ccc.inaoep.mx/~esucar/Libros/vision-sucar-gomez.pdf

[19] Espinoza, J.: Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. Ingeniería Investigación y Tecnología volumen XXI, 3 (2010).

[20] Confederación Argentina de Sordomudos - CAS: Señario de términos y expresiones en Lengua de Señas Argentina. 1a edición bilingüe, Ciudad Autónoma de Buenos Aires (2019).

[21] María Ignacia Massone. Lenguas de señas: “cada comunidad desarrolló la propia por necesidad”. CICLO DE ENTREVISTAS CONICET (2012).

[22] Manual lenguaje de signos, Argentina. (s.f.). Recuperado el 04 de Septiembre del 2023 de https://issuu.com/macjason/docs/lengua.de.sinos.argentina.

[23] Asociación Tucumana de Sordos: Consideraciones Gramaticales de la Lengua de Señas Argentina (2013). Recuperado el 12 de Septiembre del 203 de https://iesmc-tuc.infd.edu.ar/sitio/wp-content/uploads/2020/04/LENGUA-DE-SE%C3%91AS-II-Material-de-apoyo.doc.

[24] Viviana Burad: La glosa: Un sistema de notación para la lengua de señas. Mendoza (2011). Recuperado el 04 de Septiembre del 2023 de https://cultura-sorda.org/la-glosa-un-sistema-de-notacion-para-la-lengua-de-senas/.

[25] Ronchetti, Franco: Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas. Tesis Doctoral, Facultad de Informática (2017).

[26] Ronchetti, Franco and Quiroga, Facundo and Estrebou, Cesar and Lanzarini, Laura and Rosete, AlejandroLSA64: A Dataset for Argentinian Sign Language. Recuperado el 01 de Agosto del 2023 de http://facundoq.github.io/datasets/lsa64/

[27] David L. Jaffe: Evolution of mechanical fingerspelling hands for people who are deaf-blind. MSE, Department of Veterans Affairs Medical Center, Rehabilitation Research and Development Center, Palo Alto, CA 94304. Development Vol. 31 No. 3, August 1994. Pages 236–244.

[28] David L. Jaffe: RALPH: a fourth generation fingerspelling hand. MS. Project Reports (1994).

[29] Zhou, Z., Chen, K., Li, X. et al. Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays. Nat Electron 3, pp 571–578 (2020).

[30] Korzeniewska E, Kania M, Zawiślak R. Textronic Glove Translating Polish Sign Language. Sensors. (2022);

[31] Ivan Mindlin: Reconocimiento de Lengua de Señas con Redes Neuronales Recurrentes. Universidad Nacional de La Plata (2021).

[32] Sai B. Padigala, Gogineni H. Madhav, Saranu K. Kumar, Dr. Narayanamoorthy M: VIDEO BASED SIGN LANGUAGE RECOGNITION USING CNN-LSTM. IRJET, pp 1658-1663 (2022)

[33] Dimitrios Konstantinidis, Kosmas Dimitropoulos and Petros Daras: Sign Language Recognition based on Hand and Body Skeletal Data. ITI-CERTH, 6th km Harilaou-Thermi, 57001, Thessaloniki, Greece (2018).

[34] Ko, S.-K., Kim, C. J., Jung, H., & Cho, C.: Neural Sign Language Translation Based on Human Keypoint Estimation. MDPI, pp 2683 (2019).

[35] DONGXU LI, Cristian Rodriguez, Xin Yu, HONGDONG LI:Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison; WACV, pp 1459-1469 (2020).

[36] Documentación Apache Parquet. Recuperado el 01 de Septiembre del 2023 de https://parquet.apache.org/.