ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UN INTÉRPRETE DE 20 FRASES DE LENGUA DE SEÑAS ECUATORIANA A VOZ EN TIEMPO REAL USANDO REDES NEURONALES RECURRENTES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

JUAN DIEGO GUEVARA SANANDRÉS

juan.guevara01@epn.edu.ec

DIRECTOR: Dr. MARCO E. BENALCÁZAR

marco.benalcazar@epn.edu.ec

CODIRECTOR: Msc. CARLOS E. ANCHUNDIA

carlos.anchundia@epn.edu.ec

Quito, 18 de julio de 2022

DECLARACIÓN

Yo, Juan Diego Guevara Sanandrés, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente de este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

JUAN DIEGO GUEVARA SANANDRES

Digitally signed by JUAN DIEGO GUEVARA SANANDRES
Date: 2023.02.09 19:52:05 -05'00'

Juan Diego Guevara Sanandrés

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Juan Diego Guevara Sanandrés, bajo mi supervisión

Firmado electrónicamente por:
MARCO ENRIQUE
BENALCAZAR PALACIOS

Marco E. Benalcázar
DIRECTOR

Firmado electrónicamente por:
CARLOS EDUARDO
ANCHUNDIA VALENCIA

Carlos E. Anchundia
CO-DIRECTOR

AGRADECIMIENTOS

A mi madre por haberme dado la vida y estar siempre a mi lado

A mi padre por mostrarme la fortaleza que hay que tener en los momentos difíciles.

A mi hermana por enseñarme que los sentimientos son el motor vital de todo ser humano.

A mis amigos que me dieron la oportunidad de conocer nuevas perspectivas del mundo.

A mis tutores Marco y Carlos que decidieron tomar este reto junto a mí y me mostraron que a veces hay que tropezar para seguir adelante.

Al Instituto CRE-SER que fue una pieza clave en la elaboración de este proyecto.

DEDICATORIA

A todas las personas que he tenido la dicha de conocer y de las cuales he aprendido algo valioso.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En la actualidad las personas sordas signantes tienen varios problemas para realizar sus actividades diarias con normalidad. La independencia y accesibilidad de estas personas se ve afectada ya que por lo general necesitan estar acompañados de una persona de habla que les ayude en su día a día. La problemática en cuestión tiene efectos negativos en la calidad de vida de las personas sordas signantes.

El presente proyecto busca desarrollar un intérprete en tiempo real de 20 frases de la lengua de señas ecuatoriana a voz mediante el uso de inteligencia artificial para facilitar la comunicación entre personas sordas y oyentes. Dicho intérprete tendrá una implementación simple de modo que este pueda ser portable y en un futuro convertirlo en una aplicación móvil.

Para el proceso de interpretación en este trabajó se utilizó una combinación de una red neuronal convolucional y una red neuronal recurrente (CNN-LSTM). La primera extrae las características de una imagen de entrada que será la región de interés obtenida por la librería MediaPipe de Google. La segunda utiliza el vector de características como entrada para analizar y memorizar las secuencias de imágenes en los videos de entrenamiento para predecir la frase signada.

Palabras Claves: Región de interés, Redes neuronales convolucionales, Redes neuronales recurrentes, Lengua de señas, Inteligencia artificial

ABSTRACT

Nowadays, deaf signers have several problems to carry out their daily activities normally. The independence and accessibility of these people is affected as they usually need to be accompanied by a speaking person to help them in their daily life. This problem has negative effects on the quality of life of deaf signers.

This project seeks to develop a real-time interpreter of 20 sentences of Ecuadorian sign language into speech using artificial intelligence to facilitate communication between deaf and hearing people. This interpreter will have a simple implementation so that it can be portable and, in the future, turn it into a mobile application.

For the interpretation process in this work, a combination of a convolutional neural network and a recurrent neural network (CNN-LSTM) was used. The first one extracts the features from an input image that will be the region of interest obtained by Google's MediaPipe library. The second uses the feature vector as input to analyze and memorize the image sequences in the training videos to predict the signed sentence.

Keywords: Region of Interest, Convolutional Neural Networks, Recurrent Neural Networks, Sign Language, Artificial Intelligence.

1. INTRODUCCIÓN

1.1. Descripción del problema

En el año 2020 la Organización Mundial de la Salud determinó que más del 5% de la población mundial (430 millones de personas) padece una pérdida de audición discapacitante y requiere rehabilitación (432 millones de adultos y 34 millones de niños). Se calcula que en 2050 esa cifra superará los 700 millones (una de cada diez personas) [1]. En base a los datos del Consejo Nacional para la Igualdad de Discapacidades, en Ecuador existen 66 903 personas sordas [2], de las cuáles la mayoría son usuarias de la lengua de señas ecuatoriana (LSEC).

En la actualidad las personas sordas signantes tienen varios problemas para realizar sus actividades diarias con normalidad como:

- Llevar a cabo trámites en bancos, instituciones públicas o centros de salud.
- Recibir un servicio de calidad en restaurantes, bares o tiendas.

La independencia y accesibilidad de estas personas se ve afectada ya que por lo general necesitan estar acompañados de una persona de habla que les ayude en su día a día. La problemática en cuestión tiene efectos negativos en la calidad de vida de las personas sordas signantes que hoy en día se sienten aislados y presentan dificultades al realizar sus actividades diarias. Es importante aportar a la inclusión y mejorar el estilo de vida de este grupo social a través de soluciones efectivas suportadas en nuevas tecnologías. En el presente proyecto se propone desarrollar un intérprete en tiempo real de 20 frases de la lengua de señas ecuatoriana a voz mediante redes neuronales recurrentes. Dicho intérprete estará basado en el modelo *bidirectional spatial-temporal* LSTM *fussion attention network* (Bi-ST-LSTM-A) [3].

1.2. Objetivo general

Desarrollo de un intérprete de 20 frases de lengua de señas ecuatoriana a voz en tiempo real usando redes neuronales recurrentes.

1.3. Objetivos específicos

- Construir un conjunto de datos para las 20 frases en video de la LSEC completo en términos de volumen, variedad, veracidad y valor para obtener mejores resultados en el entrenamiento del intérprete.
- Analizar el desempeño del intérprete de 20 frases de la LSEC a voz en términos de exactitud de reconocimiento y tiempo de procesamiento.
- Evaluar el funcionamiento de la aplicación en modo online (usando un sujeto real)

1.4. Justificación teórica

La interpretación de frases en lengua de señas es compleja. Una frase es un conjunto de palabras (en este caso señas) que tienen sentido. El sentido está ligado con la sintaxis y la semántica, por lo que no solo se trata de traducir señas de forma aislada, sino en conjunto. El sistema debe ser capaz de memorizar lo que se ha expresado para evaluar si la siguiente seña tiene sentido, además de darle la semántica correcta en tiempo real [4].

Este tipo de mecanismos puede ser implementados con Redes Neuronales Recurrentes (RNN) ya que son una clase de redes para analizar datos de series temporales permitiendo tratar la dimensión de “tiempo”. Dado a que es necesario que la RNN recuerde la secuencia de datos de entradas a lo largo del tiempo, es importante el uso de *Long-Short Term Memory* (LSTM) [5] y bidireccionalidad [6].

El conjunto de datos es parte importante para entrenar una red neuronal. Utilizar una gran cantidad de datos puede mejorar la precisión de un modelo; sin embargo, se debe extraer las características de los datos para reducir el costo computacional al procesarlos dentro de una RNN. Una *Convolutional Neural Network* (CNN) [7] es ideal para extraer las características de los datos de entrada.

En la Figura 1 se puede apreciar el procedimiento de forma general del intérprete propuesto. Los gestos realizados por una persona son captados por una cámara. El video es dividido en imágenes o *frames*. De cada *frame* se creará una figura a partir de la

posición de la cara y manos para crear una noción espacial de cada gesto (Figura 2), además se tomará la posición de las manos. Las 2 imágenes alimentan a una Red Neuronal Convolucional (CNN) que realiza la extracción de características. La información obtenida alimenta una Red Neuronal Recurrente (RNN) llamada *spatial-temporal long short term memory* (ST-LSTM) que tiene un proceso de codificación y decodificación en el estado oculto que lo vuelve bidireccional para obtener como salida una palabra y así sucesivamente hasta completar la frase con la semántica correcta. La frase será mostrada en texto y audio.

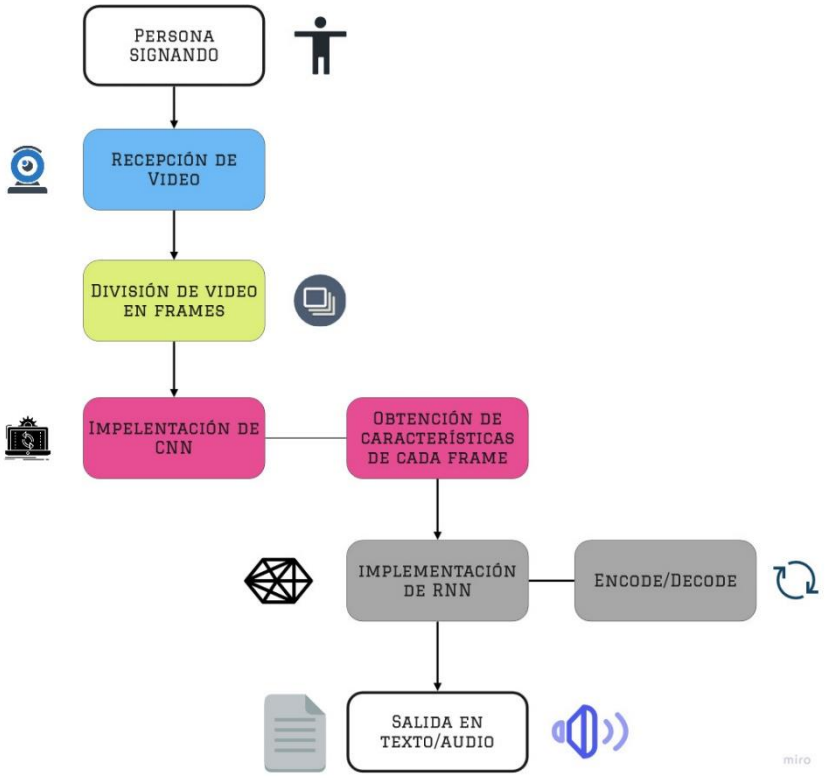


Figura 1. Proceso de interpretación LSEC

Fuente: elaboración propia

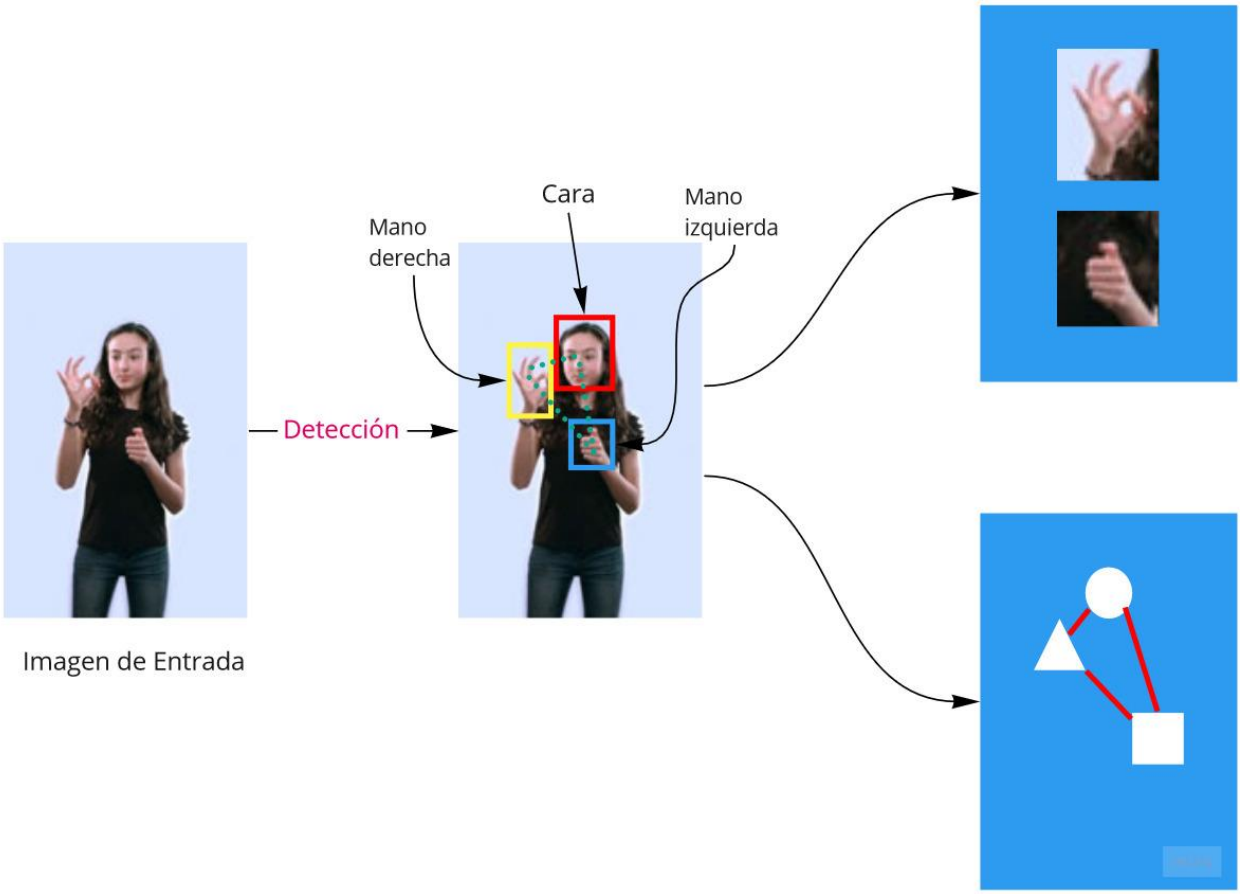


Figura 2. Detección de región de interés

Fuente: elaboración propia

1.5. Justificación metodológica

El desarrollo de un intérprete de LSEC a voz es más que solo la implementación de un *software*. El éxito del proyecto depende de aplicar una metodología que establezca tiempo, recursos, personas y estrategias para alcanzar los objetivos propuestos.

La metodología que se propone es *Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance CRISP ML (Q)* [8]. Esta metodología nace a partir de la necesidad de implementarlo en proyectos de aprendizaje automático (*machine learning*), ya que su ciclo de vida presenta variantes con respecto a los *softwares* tradicionales.

En este proyecto se propone un intérprete que será implementado con *Deep Learning*. En este contexto CRISP-ML (Q) es idóneo. La primera fase (*Business and Data*

Understanding) nos ayudará a establecer de forma clara el alcance, criterios de aceptación y factibilidad del proyecto para con esto determinar la calidad y volumen del conjunto de datos a recolectar. La segunda fase (*Data Preparation*) servirá para preparar de forma adecuada nuestro conjunto de datos que alimentará nuestro modelo. La tercera fase (*Modeling*) es la implementación del *software*. La cuarta fase (*Evaluation*) será la parte más importante en donde se evaluará la eficacia del intérprete, se realizan cambios pequeños y se prueba nuevamente, en caso de no cumplir los criterios de aceptación el ciclo se repite volviendo a la fase 1, hasta obtener mejores resultados del modelo. En cada fase se realiza un aseguramiento de la calidad para la mitigación de riesgos en fases posteriores. Si bien CRISP-ML (Q) tiene más fases (*Deployment*, *Monitoring and Maintenance*) que forman parte importante para el cumplimiento de la metodología, se considera que no es necesario aplicarlas en este proyecto.

1.6. Justificación práctica

En la actualidad la comunidad con discapacidad auditiva no tiene los medios para llevar a cabo una comunicación clara con una persona que usa el lenguaje verbal. Esta brecha repercute en un nivel de accesibilidad muy bajo a diferentes servicios o productos, dificultando la realización de actividades cotidianas. El desarrollo de un sistema que interprete lengua de señas a lenguaje verbal representa una gran ayuda para esos grupos sociales que no han podido aprovechar en su totalidad el potencial de las nuevas tecnologías.

Este proyecto, a diferencia de otros proyectos enfocados a interpretación de palabras o números, pretende ofrecer una herramienta más completa a nivel gramático. Este proyecto propone un desarrollo enfocado al LSEC, debido a que la lengua de señas no es universal. Para ello, se busca entrenar un intérprete en base a palabras y frases cotidianas usadas por las personas sordas signantes para una mayor utilidad de la herramienta. Gracias a la aplicación de la metodología CRISP-ML(Q) el *software* tendrá una implementación simple de modo que este pueda ser portable y en un futuro convertirlo en una aplicación móvil.

2. MARCO TEÓRICO

Para poder entender el desarrollo de este proyecto, es necesario comprender elementos de la problemática y solución que se abordará. Los términos de más implicancia son: lengua de señas, visión artificial, redes neuronales convolucionales y recurrentes. A continuación, se describen los aspectos fundamentales.

2.1. Conceptos lingüísticos

2.1.1. Lengua de señas

Las lenguas de señas son lenguas naturales no verbales usadas por personas sordas o con discapacidad auditiva [9]; a quienes nos referiremos como 'sordo/a' partir de este punto. La lengua de señas es un sistema de comunicación compuesto por el movimiento de las manos y brazos, la expresión facial y la postura del cuerpo [10], [11] por lo que su percepción es netamente visual. Sin embargo, las lenguas de señas no son universales por lo que difiere entre países y hasta regiones. Por ejemplo, la lengua de señas ecuatoriana (LSEC) se compone de aproximadamente un 30% de ASL (*American Sign Language*), un 20% de LSE (Lengua de señas española) y un 50% de señas originales de Ecuador [12]. Esto da como resultado que cada lengua de señas tenga una lingüística propia, como cualquier otra lengua verbal.

William Stokoe [13] en su análisis lingüístico determinó que las lenguas de señas están estructuradas en niveles: querémico, morfológico, sintáctico y semántico. El nivel querémico es la estructura interna de la seña, es decir, el componente mínimo de una frase [14]. El nivel morfológico es el significado de la seña y la combinación única de sus partes producidas simultáneamente. El nivel sintáctico es la estructura de una oración, en la lengua de señas los elementos (sujeto, verbo, objeto) no tienen un orden definido [15]. El nivel semántico relaciona los significados de las señas y sus combinaciones [14].

2.2. Conceptos informáticos

2.2.1. Visión artificial

La visión artificial (VA) es el conjunto de algoritmos que otorgan la capacidad de ver a un computador. En general, el acto de "ver" es el proceso por el cual el sistema visual humano procesa e interpreta el mundo real a través de imágenes. El computador puede

simular ciertos procesos del sistema visual humano mediante la obtención, caracterización e interpretación de imágenes digitales [16].

Imagen digital

Una imagen es una representación visual de un objeto, una persona o una escena. Una imagen digital es una función bidimensional $f(x, y)$ que es una proyección de una escena tridimensional en un plano de proyección bidimensional, donde (x, y) representa la ubicación del elemento de la imagen o píxel y contiene el valor de la intensidad. Cuando los valores de (x, y) y la intensidad son discretos, se dice que la imagen es una imagen digital. Matemáticamente, una imagen digital es una representación matricial de una imagen bidimensional que utiliza un número finito de elementos de celdas de puntos, normalmente denominados píxeles (px). Cada píxel está representado por valores numéricos: en el caso de las imágenes en escala de grises, basta con un único valor que representa la intensidad del píxel (normalmente en un rango [0, 255]); en el caso de las imágenes en color, se almacenan tres valores (que representan la cantidad de rojo (R), verde (G) y azul (B)) [17]. La imagen digital es el elemento que se usará como dato de entrenamiento para la red neuronal, pero, antes debe pasar por un proceso que se detalla a continuación.

Captura

La captura es el proceso donde se obtiene una imagen digital a través de un dispositivo como una cámara digital, video cámara, escáner, telescopio, satélite, etc. [18]. Los datos de este trabajo son grabados mediante una cámara web.

Preprocesamiento de imágenes

En el computador, la imagen capturada se representa en forma de matriz numérica, con información en cada celda. El preprocesamiento de imágenes es una teoría matemática que se ocupa de la transformación y el análisis de las imágenes. Esta transformación y análisis es el resultado de la implementación de varias operaciones matemáticas. El preprocesamiento incluye técnicas tales como la reducción del ruido, mejoramiento del contraste, nitidez de la imagen, realce de ciertos detalles o características de la imagen,

restauración de la imagen [17]–[19]. Estas técnicas sirven en ciertos casos para obtener una información específica de la imagen conocida como, región de interés.

Segmentación

La segmentación es el proceso que divide una imagen en diferentes objetos o regiones, de forma que cada una de ellas sea homogénea según nuestro interés de estudio. Este proceso se repite una y otra vez hasta que se haya estabilizado la detección de objetos o regiones de interés[20]. Con la segmentación es posible detectar varias ROI, en este tipo de situaciones se debe diferenciar entre cada ROI.

Descripción

La descripción es el proceso que obtiene características relevantes y convenientes para diferenciar un tipo de objeto o región de otro. Estas características pueden ser externas, cuando el objetivo principal son las características de la forma; o internas cuando el objetivo principal son las propiedades regionales, como el color y la textura [20].

Reconocimiento e Interpretación

El reconocimiento es el proceso que clasifica en categorías los objetos presentes en la imagen utilizando los descriptores del proceso anterior. Los objetos detectados que presenten descriptores semejantes se agrupan automáticamente en una misma clase o categoría. Para esto se utilizan varias técnicas de las cuales nos centraremos en las redes neuronales artificiales [18]. La red neuronal será capaz de interpretar los resultados y en consecuencia tomar decisiones.

2.2.2. Redes neuronales artificiales

Una red neuronal artificial, ANN (*Artificial Neural Networks*) es un modelo matemático que intenta simular la estructura y funcionalidades de las redes neuronales biológicas [21]. Una ANN consta de una capa de entrada, donde se reciben las entradas del problema, una o varias capas ocultas, donde la relación entre las entradas y las salidas se determina y representa mediante pesos sinápticos (w), y una capa de salida que emite las salidas del problema [22], véase Figura 3.

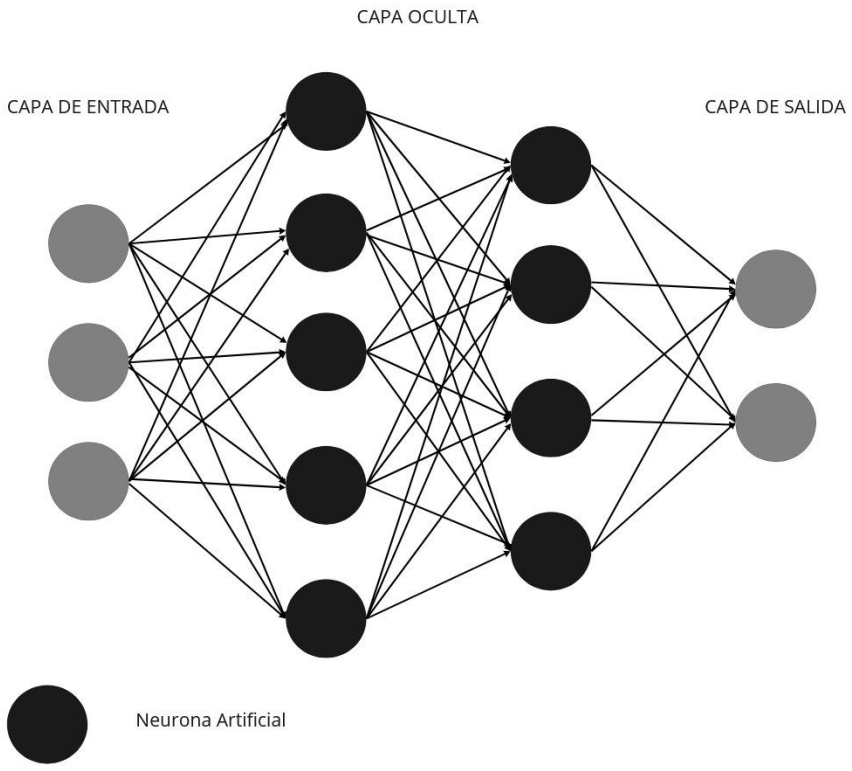


Figura 3. Arquitectura de una red neuronal feedforward

Fuente: elaboración propia

En la capa oculta, las ANNs contiene un número específico de componentes llamados neuronas artificiales. La Figura 4. Estructura de una neurona artificial muestra la estructura de una neurona artificial, donde $X\{x_1,\ x_2,\ x_3, \ldots,\ x_n\}$ representa los datos de entrada asociados a un peso $W\{w_1,\ w_2,\ w_3, \ldots,\ w_n\}$. A la sumatoria de las entradas multiplicadas por sus pesos se le añade un sesgo b, el resultado pasa por una función de activación para obtener la salida Y [21].

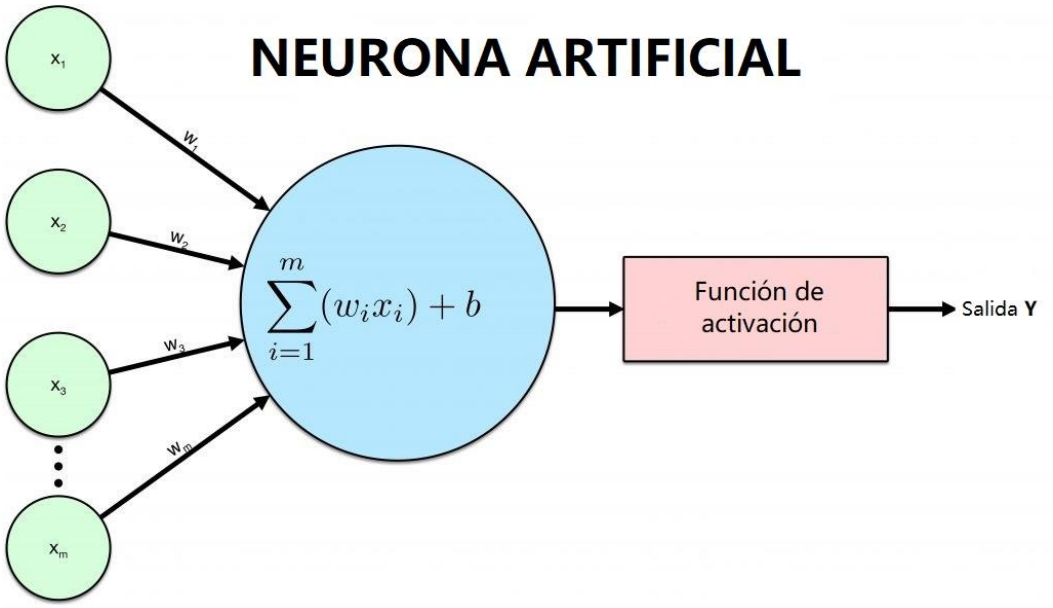


Figura 4. Estructura de una neurona artificial

Fuente: [23]

La función de activación es una función no lineal que limita la amplitud de salida de una neurona a un valor finito. Existen varias funciones de activación (Figura 5) y su aplicación varia en base al problema que se desea resolver [22].

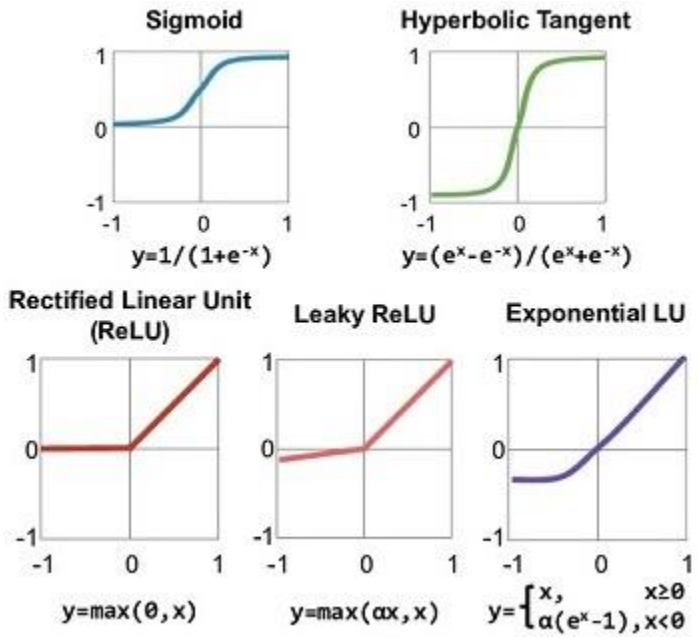


Figura 5. Funciones de activación

Fuente: [24]

Aprendizaje

Para entrenar una red neuronal es necesario establecer un tipo de aprendizaje. El aprendizaje supervisado o aprendizaje asociativo entrena la red proporcionándole patrones de entrada y salida coincidentes, es decir, para cada dato de entrada existe un dato de salida [22]. El proceso del aprendizaje supervisado es el siguiente. Primero las entradas son recibidas en la capa de entrada. Segundo, con los datos de entrada se realizan las operaciones en las neuronas de las capas ocultas. Tercero, aplicamos un algoritmo de *back propagation* (BP). En el algoritmo BP, las salidas de las capas ocultas se propagan a la capa de salida donde se calcula la salida. Esta salida se compara con la salida deseada para la entrada dada. Se calcula el error en función de esta diferencia, el error se propaga desde la capa de salida a la capa oculta y desde la capa oculta a la capa de entrada. A medida que el flujo retrocede, cambia el peso entre las neuronas. Este ciclo de avanzar desde la entrada y la salida, y de la salida a la entrada, se denomina época. La red pasa por muchas de estas épocas hasta que el error (diferencia entre la salida real y la salida deseada) esté dentro de cierta tolerancia. Ahora se dice que la red está entrenada. Este proceso de entrenamiento establece los pesos entre todas las neuronas en todas las capas. Los pesos así obtenidos de una red entrenada se utilizan para calcular la respuesta de la red a datos desconocidos [25].

Otro aspecto importante para el aprendizaje de una red neuronal es el conjunto de datos. Estos datos deben cumplir con ciertos estándares de calidad para no ocasionar un *overfitting* de la red neuronal. El *overfitting* es una dificultad que se presenta en el aprendizaje de una red neuronal cuando un conjunto de datos tiene demasiados detalles o ruido [26]. Existen varias técnicas para resolver esta problemática; este proyecto se centrará en 2: *Batch Normalization* y *Dropout*. *Batch Normalization* normaliza el valor del promedio y la varianza de los datos de entrada [27], mientras que *Dropout* descarta un porcentaje de neuronas asignándoles el valor 0 [28].

2.2.3. Redes neuronales convolucionales

Las *Convolutional Neuronal Netwoks* (CNN), también conocidas como Redes neuronales convolucionales son un tipo de redes neuronales artificiales (*feedforward*) que se utilizan principalmente en el campo del reconocimiento de patrones dentro de las imágenes. Las

CNN son muy eficientes en el procesamiento de imágenes, ya que realizan diversas operaciones matemáticas para extraer características [29], [30]. Las CNN se componen normalmente de tres tipos de capas (o bloques de construcción): *convolution*, *pooling*, y *fully connected*. Las 2 primeras, las capas de *convolution* y de *pooling*, realizan la extracción de características, mientras que la tercera, *fully connected*, convierte las características extraídas en un vector [30].

Capa *convolution*

La capa *convolution* es un componente fundamental en la arquitectura de una CNN debido a su capacidad para realizar extracción de características en las imágenes. Esta capa se configura con una serie de hiperparámetros (Tabla 1) para realizar una operación llamada convolución.

Convolución

La convolución es una operación, donde un pequeño arreglo de números, llamado filtro, se aplica sobre una matriz de entrada (imagen digital). El filtro de convolución se superpone sobre la imagen de entrada, se calcula el producto entre los números en la misma ubicación del filtro y la matriz, se obtiene un único número sumando estos productos. El resultado pasa a través de una función de activación, se obtiene la salida Y_0. A continuación, el filtro se desplaza en función del valor del *stride* (por defecto 1) ya sea a la derecha o a la izquierda. Repetimos la convolución para cada ubicación posible de la matriz hasta que hayamos movido el filtro a la esquina inferior derecha de la imagen de entrada, como se muestra en la Figura 6 [30] [31].

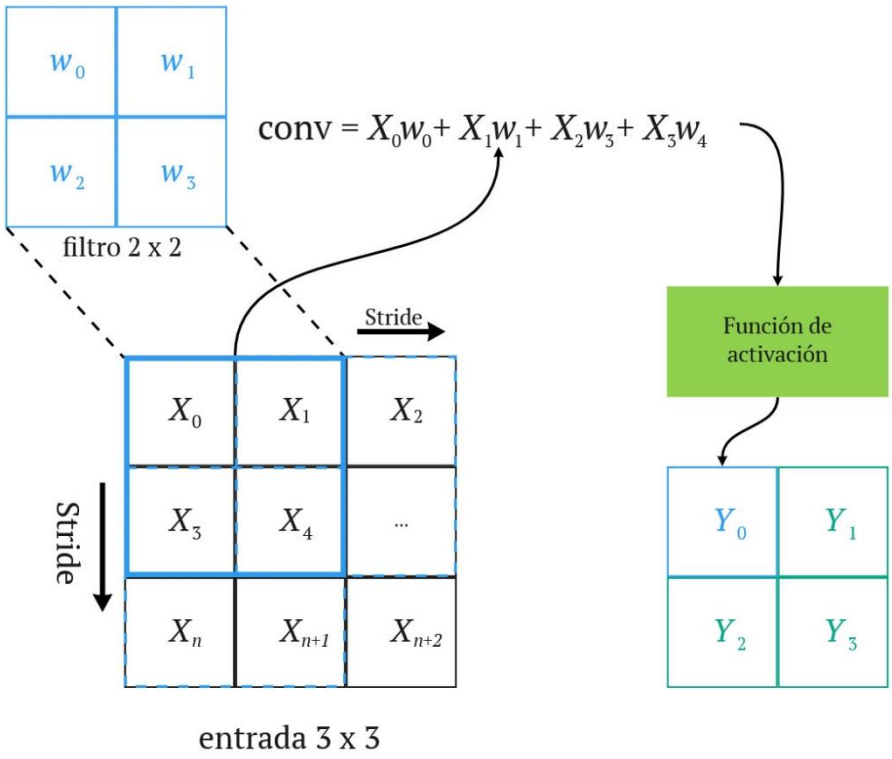


Figura 6. Proceso de convolución

Fuente: elaboración propia

Tabla 1. Hiperparámetros de una CNN

Hiperparámetro	Función
Padding	Amplia las dimensiones de la imagen de entrada. Los valores especifican la cantidad de filas y columnas que se añaden a la matriz de la imagen, así como el valor de cada celda creada.
Stride	Determina la cantidad de saltos que da el filtro sobre la matriz de entrada luego de cada convolución. En la Figura 7 el valor del stride es 1.
Filtro	Realiza operaciones de convolución sobre la matriz de entrada.

Capa *pooling*

La capa *pooling* está compuesta de operaciones que se aplican sobre la matriz de salida de la capa anterior para reducir el número de parámetros y la complejidad computacional del modelo. La operación más usada es *max pooling,* la cual extrae secciones de la matriz de entrada, devuelve el máximo valor de cada sección, y descarta los demás valores. Esta capa se configura con los mismos hiperparámetros que en la capa *convolution*, con la diferencia de que el filtro tiene dimensión, pero no contiene neuronas con pesos, véase la Figura 7.

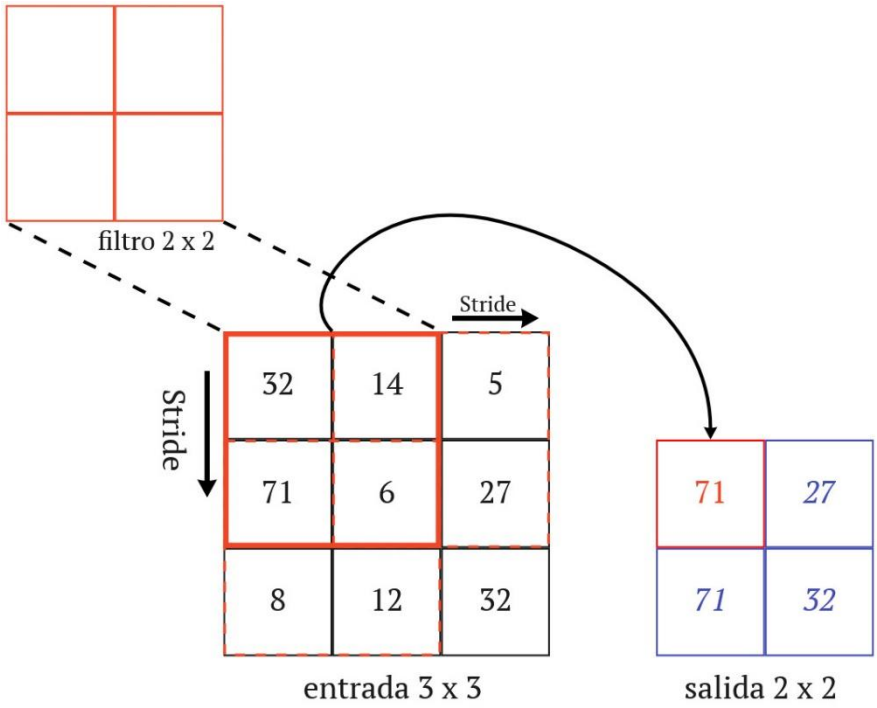


Figura 7. Proceso max pooling

Fuente: elaboración propia

Capa *fully conected*

En la capa *fully conected* la matriz que se obtiene como salida de una capa *convolution* o *pooling* normalmente es aplanada en un vector (arreglo de números unidimensional). Este vector se convierte en la entrada de una o más capas totalmente conectadas, también conocidas como capas *dense*, en las que cada entrada está conectada a cada salida mediante un peso [30].

2.2.4. Redes neuronales recurrentes

Las redes neuronales recurrentes (RNN) son un tipo de arquitectura de red neuronal que se utiliza principalmente para detectar patrones en una secuencia de datos. Dichos datos pueden ser la escritura, el genoma, el texto o en nuestro caso videos (secuencia de imágenes). Lo que diferencia a las redes neuronales recurrentes de las redes neuronales *feedforward*, es la forma en que la información pasa a través de la red. Mientras que las redes *feedforward* pasan la información a través de la red sin ciclos, la RNN tiene ciclos y transmite la información de vuelta a sí misma (Figura 8) [32].

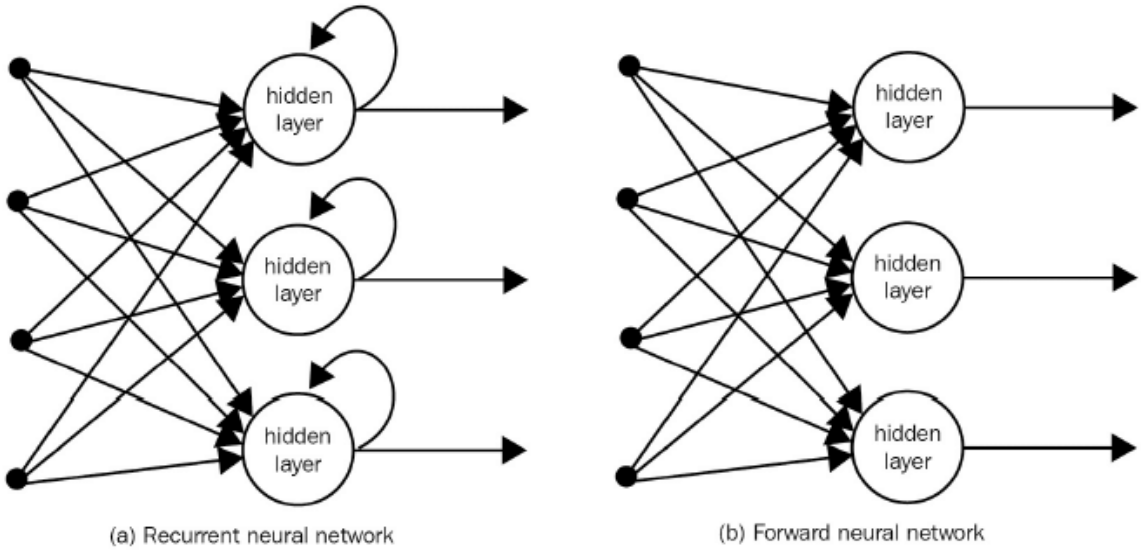


Figura 8. Comparación de arquitecturas de redes neuronales

Fuente: [33]

Arquitectura

Las redes neuronales recurrentes se clasifican en 4 tipos de arquitectura. Pueden tomar un único valor y devolver un único valor, tomar una secuencia de valores y devolver un único valor; tomar un valor y devolver una secuencia de valores; o tomar una secuencia de valores y devolver una secuencia de valores [34].

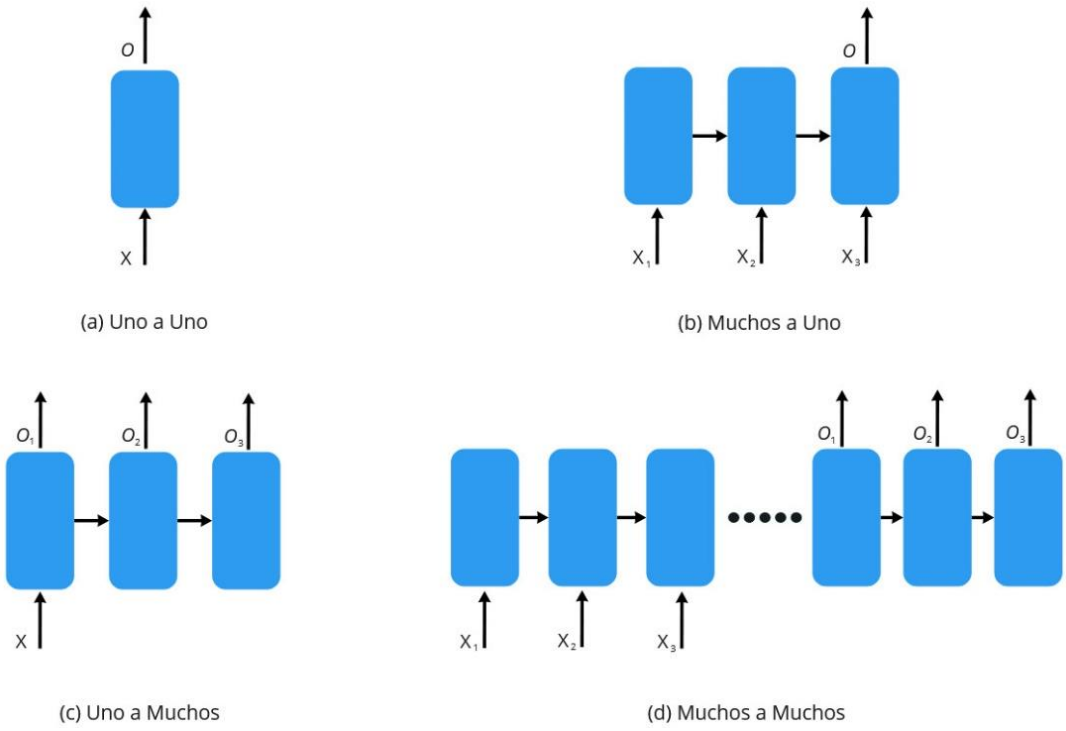


Figura 9. Tipos de arquitectura de las RNNs

Fuente: elaboración propia

Long-Short Term Memory

Las redes neuronales recurrentes *long-short term memory* (LSTM) son un modelo de RNN capaz de recordar secuencias de datos a largo plazo. El modelo LSTM introduce un tipo intermedio de almacenamiento a través de la célula de memoria. Una célula de memoria es una unidad que controla las entradas y salidas a través de compuertas (*gates)*. Estas *gates* controlan el flujo de información a las neuronas ocultas y preservan las características extraídas de los pasos de tiempo anteriores $t - 1$ [35], [36].

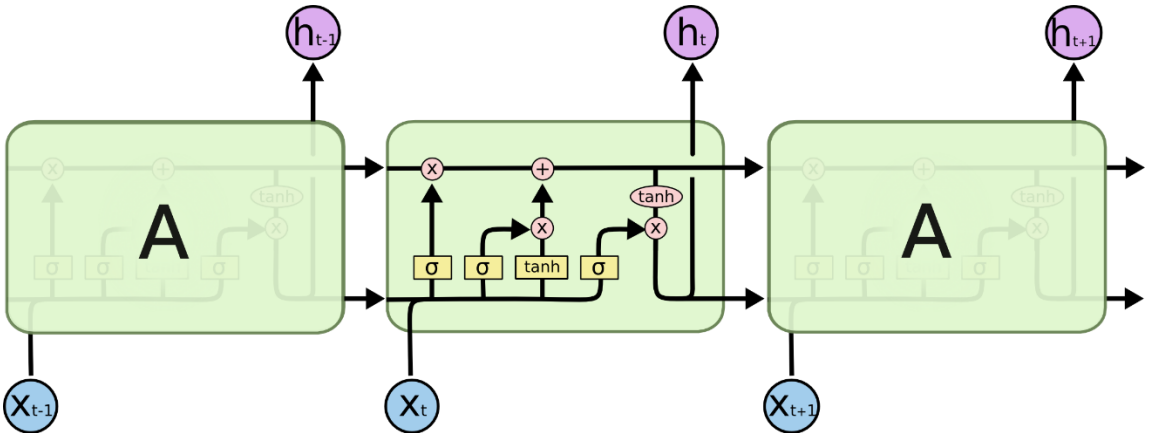


Figura 10. Módulo de una LSTM

Fuente: [37]

La estructura de una LSTM se presenta en la Figura 10, donde cada línea representa un vector que conecta la salida del instante $t-1$ con la entrada del instante t, cada circulo rosa representa una operación entre vectores y cada caja amarilla una capa de la red neuronal donde la información pasa por una función de activación [38]. Los elementos de una LSTM se describen a continuación.

Estado de celda: Es el vector que atraviesa la parte superior del diagrama (Figura 11). El estado de celda es como una cinta transportadora. La información fluye sin cambios, es decir, con solo algunas interacciones lineales. La falta de alteración en la información es lo que permite a la red recordar con más facilidad.

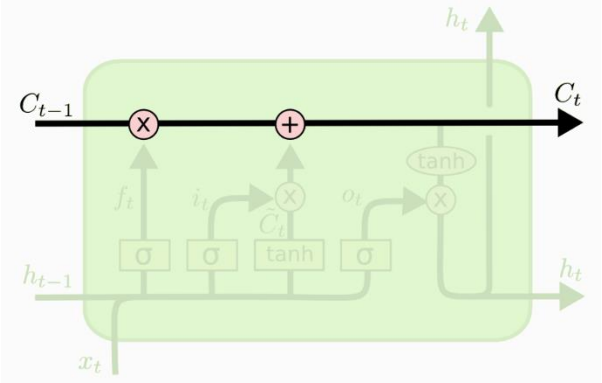
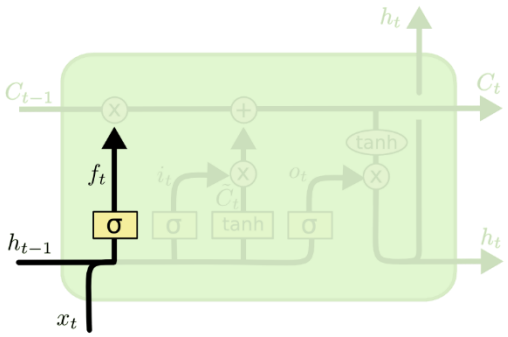


Figura 11. Cell state de una LSTM

Fuente: [37]

***Forget gate*:** Esta compuerta decide qué información se mantiene o se desecha. Como entrada recibe la información h_{t-1} (salida de la iteración anterior) y x_t (entrada de la presente iteración t). Posteriormente, esta información pasa por una función sigmoide que produce un número entre 0 y 1, siendo 0 la información irrelevante.

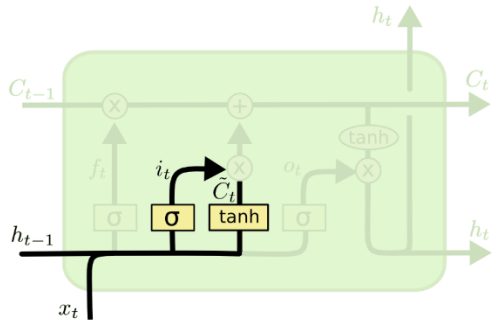


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figura 12. Forget gate

Fuente: [37]

***Input gate*:** Esta compuerta decide que nueva información se va a almacenar en el estado de celda actual. Primero la función sigmoidea decide que valores serán actualizados, mientras que la función tanh (tangente hiperbólica) crea un vector $\tilde{C}_t$ que contiene los candidatos a ser añadidos al estado de celda [38].

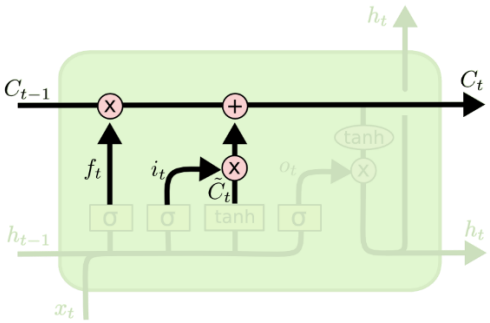


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figura 13. Input gate

Fuente: [37]

Con la información obtenida se actualiza el estado de la celda anterior, C_{t-1}, al estado actual C_t, eliminando los datos irrelevantes ya mencionados. Para ello, se multiplica el estado anterior por f_t y se añade el producto entre los valores del *input gate* ($i_t * \tilde{C}_t$).

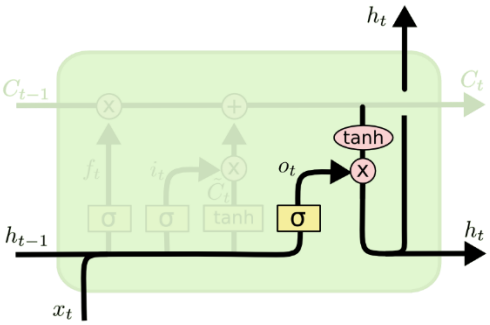


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figura 14. Actualización de estado de celda

Fuente: [37]

Output gate: Esta compuerta realiza las operaciones finales para obtener la salida de la célula de memoria, que será la información de entrada en la iteración $t + 1$.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figura 15. Output gate

Fuente: [37]

Tabla 2. Hiperparámetros de una LSTM

Parámetros	Descripción
Hidden_units	Número de neuronas en la capa oculta de la LSTM.
Timesteps	Pasos temporales en los que se dividirán las secuencias de aprendizaje.
Input_dim	Dimensión de la información de entrada.

2.3. Conceptos metodológicos

2.3.1. Metodología CRISP-ML(Q)

CRISP-ML(Q) es una metodología centrada en el desarrollo de aplicaciones de *machine learning* (ML). A diferencia de otras metodologías [39], CRISP-ML(Q) propone el estudio de una problemática y la información necesaria para resolverla. Además, aplica un componente de aseguramiento de la calidad en cada una de las fases para mitigar el riesgo de errores en etapas finales [40].

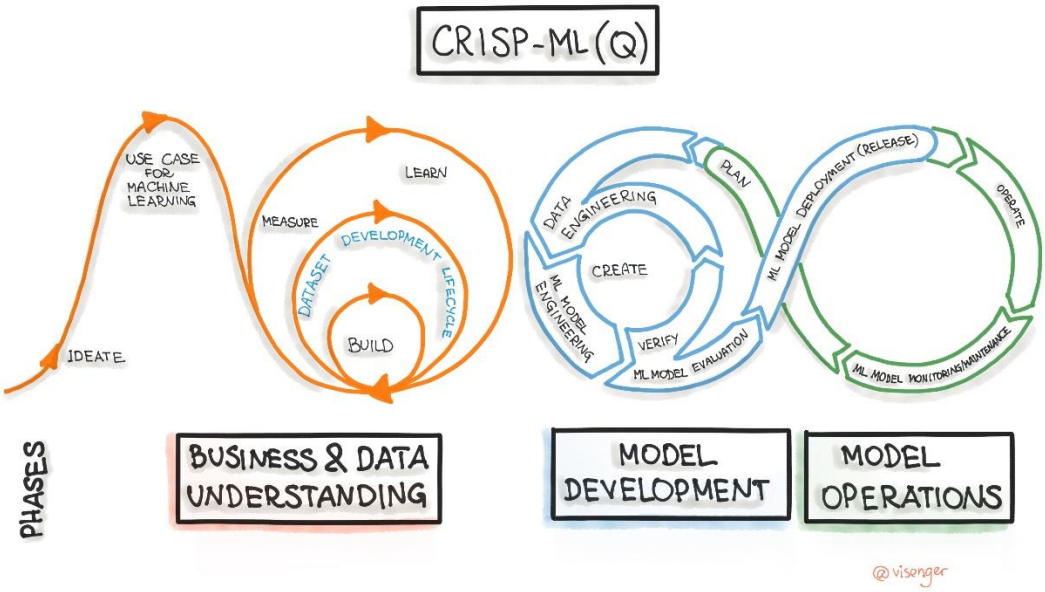


Figura 16. Proceso de CRISP ML(Q)

Fuente: [41]

CRISP-ML(Q) está compuesta por 6 fases:

Comprensión del negocio y datos: El objetivo de esta primera fase es asegurar la viabilidad del proyecto. Se realizan tareas como: identificar el alcance de la aplicación ML, definir los criterios de éxito, recopilar los datos y verificar la calidad de estos.

Preparación de los datos: La segunda fase tiene como objetivo preparar los datos para la siguiente fase (*modeling*). Durante esta fase se realizan tareas *como:* selección de datos, limpieza de datos, ingeniería de características y estandarización de datos.

Modelado: En esta fase se realiza la construcción del modelo ML. Este proceso se lleva a cabo mediante la selección, implementación y entrenamiento de un modelo. Al final se debe aplicar métricas de evaluación para determinar si el modelo en cuestión cumple con las restricciones y requerimientos especificados en la primera fase. Es necesario documentar la información (algoritmos, hiperparámetros, precisión de entrenamiento y validación) de los modelos entrenados.

Evaluación: Durante esta fase se realiza una validación del rendimiento del modelo. Se utiliza un conjunto de datos que no han sido utilizados antes. Además, el modelo es evaluado con datos de entrada ruidosos o incorrectos. Todos los resultados de la fase de evaluación son documentados.

Despliegue: Después de tener éxito en la fase de evaluación en el ciclo de vida de desarrollo de ML, el modelo de ML se gradúa para implementarse en el entorno de (pre)producción. La implementación denota un proceso de integración del modelo ML en un sistema de software.

Monitoreo y mantenimiento: El objetivo de esta fase es mantener un proceso de monitoreo y mantenimiento constante. Normalmente el rendimiento de un modelo ML se ve afectado cuando este es puesto en producción y empieza a operar con datos no vistos.

2.3.2. Machine learning canvas

El machine learning canvas es una herramienta estructurada que permiter confirmar la viabilidad de un proyecto de ML. Además, se utiliza para definir los elementos importantes durante todo el ciclo de vida del proyecto ML.

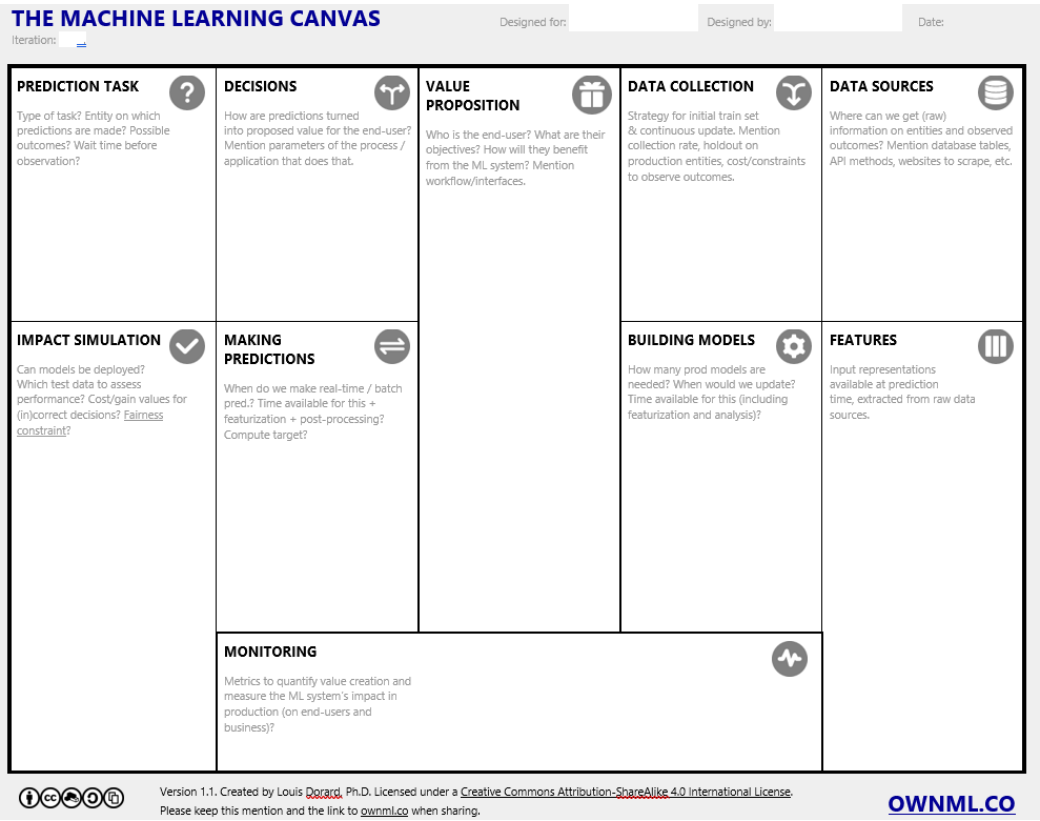


Figura 17. Machine Learning Canvas

Fuente: [42]

En la *Figura 17* se puede apreciar información general sobre las diferentes etapas en el proceso de construcción del modelo ML y la Tabla 3 muestra la descripción de estas. El *machine learning* canvas es una herramienta que está presente durante todo el proceso metodológico.

Tabla 3. Elementos de Machine Learning Canvas

Etapa	Descripción
Value Proposition	Se define los beneficios o aportes de valor del producto ML y los usuarios finales.
Data Sources	Explica las fuentes que se utilizarán para la construcción del conjunto de datos.

Data Collection	Son las estrategias que se implementarán para la construcción del conjunto de datos.
Features	Se especifica los tipos de datos que serán usados como entradas del modelo ML.
Prediction Task	Es el proceso de predicción del modelo ML. Se especifica un tipo de dato de entrada y la salida esperada.
Decisions	Es una descripción de las decisiones que puede tomar la aplicación ML en base a los resultados obtenidos por cada entrada de datos.
Building Models	Se detallan las características de los modelos de redes neuronales que se van a utilizar.
Impact Simulation	Son las pruebas en ambientes reales para evaluar el rendimiento del modelo ML construido.
Making Predictions	Se detallan elementos dentro de la predicción del modelo como: tiempo de respuesta, limitaciones en hardware o software.
Monitoring	Son los métodos y métricas para evaluar la aplicación después del primer lanzamiento.

2.4. Herramientas y librerías de desarrollo

En esta sección se detallan las herramientas y librerías más relevantes que se utilizarán para la preparación de los datos del intérprete de LSEC, como también para la codificación de los modelos inteligentes.

2.4.1. Herramientas

Python

Python es un lenguaje de programación de alto nivel que permite a los programadores expresar conceptos en menos líneas de código de lo que sería posible en otros lenguajes. Python es flexible y de codificación sencilla. Este lenguaje puede soportar diferentes estilos de programación, incluyendo el estructural y el orientado a objetos [43]. Python es el lenguaje de programación que se utilizará para desarrollar el intérprete propuesto.

Anaconda

Anaconda es un software gratuito que te proporciona un conjunto de herramientas adaptadas a la investigación y la ciencia. Anaconda brinda acceso diferentes entornos que te permiten codificar en Python. Estos entornos, también conocidos como entornos de desarrollo integrado (IDE), son plataformas o aplicaciones que facilitan enormemente el desarrollo de código [44].

Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que se puede utilizar para crear y compartir documentos que contengan código en vivo, ecuaciones, visualizaciones y texto. Jupyter Notebook es el espacio de trabajo estándar para la mayoría de los científicos de datos de Python [45].

CSV

Un archivo CSV (valores separados por comas) es un tipo especial de archivo que puede crear o editar en Excel. En lugar de almacenar la información en columnas, los archivos CSV almacenan datos separados por comas. Cuando el texto y los números se guardan en un archivo CSV, es fácil moverlos de un programa a otro [46].

JSON

JavaScript Object Notation (JSON), es un formato ligero para almacenar y transportar datos. JSON se utiliza a menudo cuando se envían datos desde un servidor a una página web [47].

2.4.2. Librerías

OpenCV

OpenCV es una librería de procesamiento de imágenes que está diseñada para una eficiencia computacional fuertemente enfocada a aplicaciones en tiempo real. OpenCV requiere que las imágenes estén en RGB o en escala de grises para poder mostrarlas o guardarlas [48].

MediaPipe Hands

MediaPipe Hands es una solución de detección y seguimiento de manos y dedos de alta fidelidad. Emplea *machine learning* (ML) para inferir 21 puntos de referencia (Figura 18) de una mano a partir de un solo fotograma (imagen digital). MediaPipe Hands está compuesta por 2 modelos. Un modelo de detección de la palma de la mano que opera sobre la imagen completa y devuelve un cuadro delimitador. Un modelo de referencia de la mano que opera en la región de la imagen recortada definida por el detector de la palma y devuelve los puntos de referencia de la mano en un plano 3D (coordenadas (x, y, z)) [49], [50].

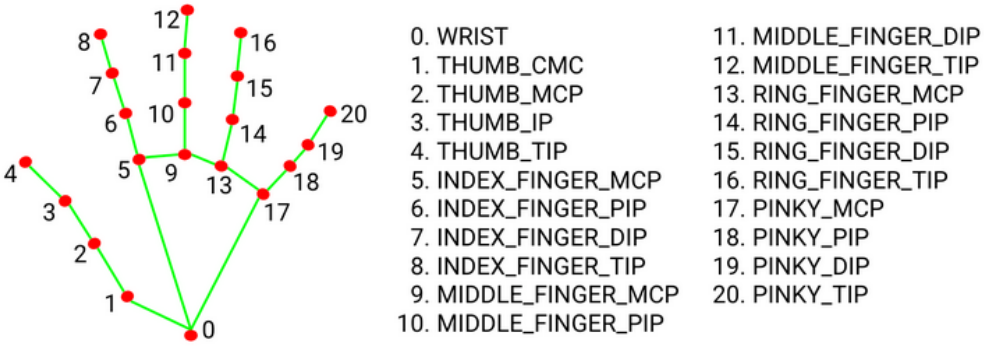


Figura 18. Puntos de referencia de la mano

Fuente: [49], [50]

Tensorflow

Tesorflow es una librería de código abierto creada por Google para el desarrollo de aplicaciones de ML. Se utiliza tanto comercialmente como por los desarrolladores para construir e implementar modelos de ML [51].

Keras

Keras sirve como API de alto nivel de TensorFlow para construir y entrenar aplicaciones de *Deep learning*. Keras es una interfaz de usuario, que se ocupa de las capas, los modelos, los optimizadores, las funciones de pérdida, las métricas y mucho más [52].

Scikit-learn

Scikit-learn es una librería de ML de código abierto que soporta el aprendizaje supervisado y no supervisado. También proporciona varias herramientas para el ajuste de modelos, el preprocesamiento de datos, la selección de modelos, la evaluación de modelos y muchas otras utilidades [53].

NumPy

NumPy es una librería de Python para desarrollar computación científica. NumPy contiene una variedad de operaciones (matemáticas, lógicas, ordenamiento, manipulación de dimensión, estadísticas) que se pueden aplicar sobre estructuras de datos para obtener la información deseada [54].

Matplotlib

Matplotlib es una librería completa para crear, personalizar, exportar visualizaciones en Python. Estas visualizaciones pueden ser de 3 tipos: estáticas, animadas e interactivas [55].

3. METODOLOGÍA

En este capítulo se abordarán las fases en el desarrollo del intérprete de LSEC en base a los conceptos presentados en el capítulo anterior. La primera fase describe la viabilidad del proyecto, seguido de la definición y recolección de los datos. Posteriormente, se prepara el conjunto de datos utilizando técnicas de VA. Luego, se define e implementa la arquitectura del modelo de red neuronal. Se verifica que el modelo cumpla con los criterios de éxito. Finalmente se evalúa el prototipo en un ambiente real.

3.1. Comprensión del negocio y datos

La definición de los datos, el alcance y los criterios de éxito son el punto de partida en el desarrollo de las aplicaciones ML en base a la metodología CRISP ML(Q).

3.1.1. Machine learning canvas

Se elaboró el Machine Learning Canvas para determinar la viabilidad, los criterios de éxito y alcance del proyecto propuesto, desarrollar un intérprete de LSEC.

<table>
<tr>
<td>PREDICTION TASK
Interpretar frases de la LSEC a voz.
Entrada: Video de una frase signada en LSEC.
Salida: Frase en español</td>
<td>DECISIONS
Se carga un video.
Se detecta la posición de la o las manos y se obtiene 1 o 2 imágenes.
Cada imagen de todo el video se clasifica. Se predice que frase se está signando
Reproducir la frase interpretada a voz.
Proveer una serie de respuestas a la frase</td>
<td rowspan="2">VALUE PROPOSITION
Proporcionar una herramienta para facilitar la comunicación entre las personas sordas y oyentes dentro del contexto de la Lengua de Señas Ecuatoriana (LSEC)
Interpretar frases en Lengua de señas ecuatoriana (LSEC) a español.</td>
<td>DATA COLLECTION
Lanzar una encuesta para encontrar las frases cotidianas de la LSEC.
Definir una lista de 20 frases cotidianas.
Recolectar por video, personas signando las 20 frases cotidianas.
Recolectar 400 videos.</td>
<td>DATA SOURCES
Institutos Tecnológico CRE-SER
Comunidad sorda del Ecuador</td>
</tr>
<tr>
<td>IMPACT SIMULATION
Utilizar el conjunto de datos de prueba para evaluar el modelo en términos de precisión y tiempo de procesamiento
Validar que el modelo tenga más de un 70% de precisión
Validar que el tiempo de procesamiento sea menor que 300 ms</td>
<td>MAKING PREDICTIONS
Una persona no oyente necesita comunicarse con alguien.
Graba un video con su celular signando en LSEC.
Se predice la frase signada.
Tiempo estimado para realizar la interpretación: 5-10 segundos
Limitaciones: calidad de video, visualización de las manos.</td>
<td>BUILDING MODELS
Es necesario un modelo de red neuronal: CNN + RNN
El modelo será actualizado hasta tener una precisión por encima del 60% en un ambiente real
Actualización del modelo: añadir una frase a la vez
Se creará el modelo añadiendo una frase a la vez.</td>
<td>FEATURES
400 videos de frases cotidianas en LSEC.
Posición de la o las manos del signante durante el video.
Imagen de la mano o manos del signante</td>
</tr>
<tr>
<td></td>
<td>MONITORING</td>
<td>Reducir el tiempo que tarda una persona sorda en realizar un trámite en entidades públicas del gobierno</td>
<td>Establecer métricas de precisión del interprete en ambientes reales.</td>
<td></td>
</tr>
</table>

Figura 19. Machine Learning Canvas basado en el intérprete de LSEC

Fuente: elaboración propia

La información que se muestra en la Figura 11 corresponde a todos los elementos involucrados en el proyecto, desde especificar las fuentes donde se van a obtener los datos hasta las técnicas que se utilizarán para evaluar el modelo construido. Primero se definió la propuesta de valor (Value Proposition). La propuesta de valor nace de la necesidad de crear herramientas para facilitar la comunicación entre personas sordas y oyentes (sección 1.1). En base a los estudios realizados en a la sección 2.1.1, en el Ecuador no existe un software (producto) capaz de interpretar frases en LSEC.

3.1.2. Definición del conjunto de datos

El primer paso para el desarrollo del intérprete de LSEC fue definir y recolectar los datos. Se contactó a diferentes colectivos de la cultura sorda, con el fin de pedir apoyo en esta etapa. El Instituto CRE-SER prepara intérpretes profesionales en lengua de señas y accedió a colaborar con sus estudiantes para realizar las actividades de definición y recolección de los datos.

Figura 20. Logo del Instituto CRE-SER

Fuente: [56]

Se diseñó una encuesta en Google *forms*[1] para definir 20 frases cotidianas de la LSEC. Cada estudiante del Instituto CRE-SER realizó la encuesta a 2 personas sordas. La encuesta se compone de 2 enunciados:

1. Seleccione las 8 frases que más use en su día a día

2. Adicional a las frases anteriores. ¿Qué otras frases usas en tú vida diaria? Escribe 5

[1] https://docs.google.com/forms/d/1UFkjtyGFnpHOZ7J5uc7JoMZ44WB7wJ9-08VXWC-Z-oc/edit

Para el primer enunciado se utilizó una lista de 15 frases en LSEC. La lista se recolectó del diccionario [57], el cual tiene una sección llamada “frases comunes”. En esta primera parte de la encuesta el usuario solo puede escoger 8 de las 15 opciones. En el segundo enunciado se espera una respuesta del usuario más amplia y subjetiva.

La idea principal era seleccionar 10 frases del primer enunciado y 10 frases del segundo enunciado. Sin embargo, en el segundo enunciado algunos resultados no contenían las respuestas esperadas debido a la libertad que tuvieron los usuarios para ingresar una respuesta. En algunos casos las respuestas fueron palabras o nombres. En otros casos se repitieron frases que estaban en el enunciado anterior. Por lo que, se decidió agrupar los resultados de los enunciados 1 y 2 en función de la frecuencia de aparición de cada una de las frases. De esta manera se definió la lista 20 frases (Figura 21).

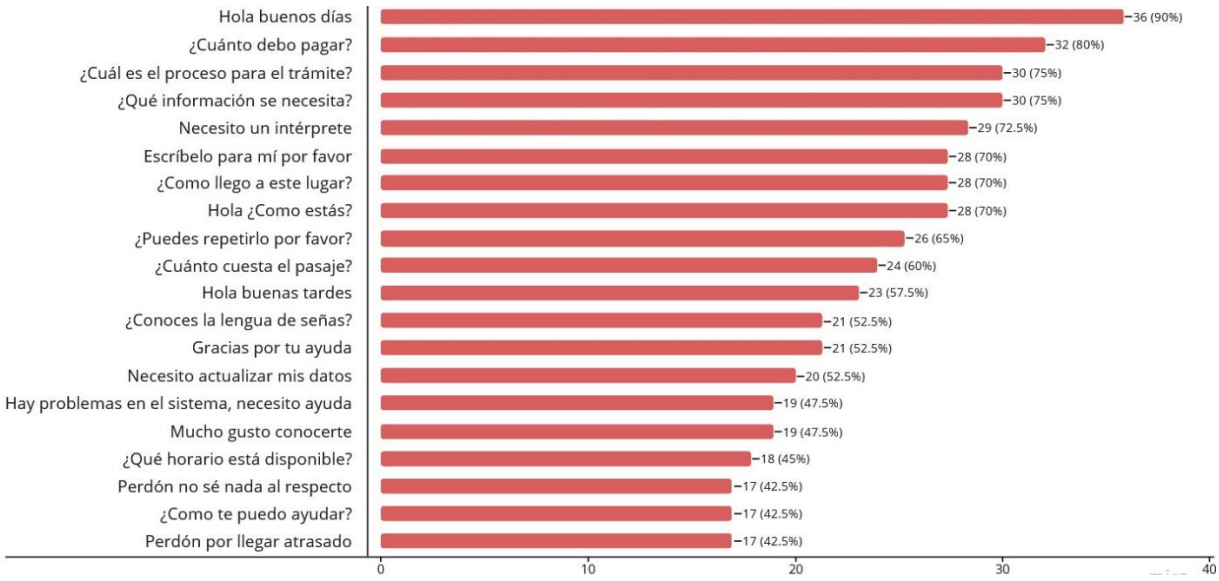


Figura 21. Frecuencia de aparición de frases

Fuente: elaboración propia

En la lengua de señas, una seña se puede interpretar como 1 o 2 palabra/s. En ciertos casos solo hacen falta 2 señas para construir toda una frase. Se determinó que, dentro de las 20 frases existe un vocabulario de 37 palabras. La Figura 22 contiene una representación del vocabulario de señas que se puede encontrar en el listado de frases.

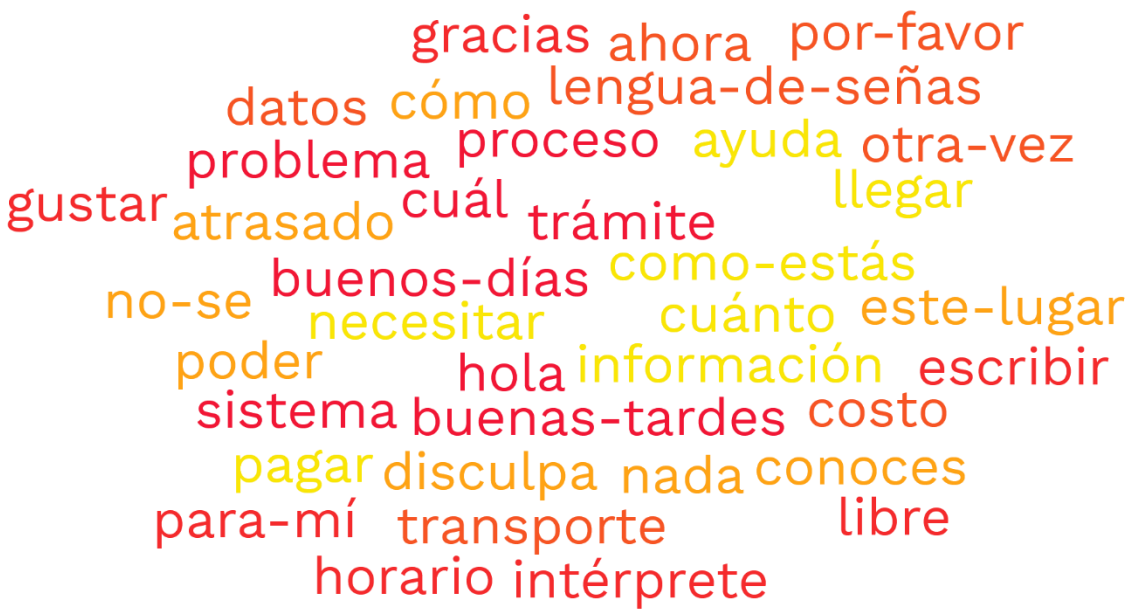


Figura 22. Vocabulario dentro de las frases en LSEC.

Fuente: elaboración propia

3.1.3. Recolección de datos

La única fuente de datos sobre la LSEC [57] contiene pocos videos y frases para entrenar una red neuronal. Por lo que fue necesario crear un conjunto de datos desde 0. Luego de definir las 20 frases en LSEC, se organizó la logística para la grabación de los videos (fecha, lugar, recursos). Debido a la pandemia del COVID-19 fue imposible realizar las grabaciones en las instalaciones del Instituto. Se elaboró una lista de lineamientos[2] que 20 estudiantes del instituto tuvieron que seguir para grabar los videos correspondientes a 20 frases en LSEC. Cada estudiante tuvo que subir a una carpeta en Google Drive[3] 20 videos, es decir, 1 video por frase. En cada video el estudiante debía signar la frase en LSEC 5 veces y cumpliendo con los lineamientos establecidos. También, se estableció una rúbrica de evaluación para control de calidad antes de pasar a la siguiente fase.

[2] https://drive.google.com/drive/folders/19iHaI7SrOgKy61Xjxk8y7pm5qzKdhFnj?usp=sharing
[3] https://drive.google.com/drive/folders/1FZRMGNl5wOiLupg9EBQ0LiqaIOkAwdRR?usp=sharing

3.1.4. Retrospectiva de las iteraciones

Se realizaron 3 iteraciones en las correcciones de los videos[4], lo cual representó un retraso en el tiempo establecido para la finalización de esta etapa. La falta de control que se tuvo sobre las grabaciones de los videos ocasionó diversos problemas como: frases mal signadas, mal posicionamiento del signante, videos corruptos, resolución y formato de video incorrecto.

El problema más determinante fue el regionalismo. En el Ecuador existen ciertas diferencias entre la LSEC usada en costa y sierra. En la Figura 23 se muestra un ejemplo claro de la problemática en cuestión. En la frase "¿Cuánto cuesta el pasaje?" la seña "pasaje" se lleva a cabo de manera distinta por los signantes. Se tomó la decisión de utilizar las señas de la región sierra como un estándar para que no existiera variabilidad en los datos.

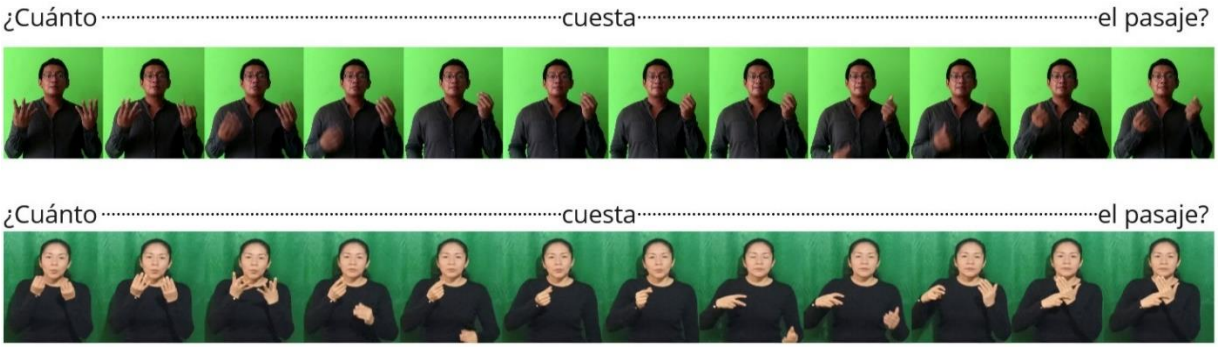


Figura 23. Regionalismo en la LSEC

Fuente: elaboración propia

Luego de establecer las correcciones necesarias, 395 videos pasaron todos los filtros de calidad. En base al ML canvas (Figura 19) el objetivo fue recolectar 400 videos, sin embargo, se excluyeron 5 porque no cumplían con los lineamientos, además nunca hubo una corrección por parte de los estudiantes en las fechas establecidas.

[4] https://drive.google.com/drive/folders/19iHaI7SrOgKy61Xjxk8y7pm5qzKdhFnj?usp=sharing

3.2. Preparación de los datos

En esta etapa los videos recolectados en la sección anterior serán procesados con el fin de construir un conjunto de datos etiquetado, preciso y estandarizado que se utilizará para entrenar el modelo de red neuronal.

3.2.1. Selección de los datos

En la sección 3.1.3 se estableció que, el contenido de cada video es una persona signando 1 frase en LSEC 5 veces. Se utilizó una herramienta web[5] para dividir cada video en 5 distintos. Los resultados (Figura 24) muestran que la distribución de los videos por frase no es simétrica ya que en ciertos casos el signante repitió la frase más de 5 veces como en la frase “¿Cómo te puedo ayudar?” o menos de 5 veces como en la frase “Gracias por tu ayuda”.

[5] https://vidds.co/online-video-cutter/

Figura 24. Cantidad de videos por frase

Fuente: elaboración propia

3.2.2. División de los datos

La Figura 25 muestra la distribución del conjunto de datos. El 60% será usado como datos de entrenamiento. Otro 20% será usado como datos de validación para medir la precisión del modelo entrenado. El 20% restante (pruebas) no se utilizará o modificará hasta llegar a la etapa final del proyecto (Evaluación).

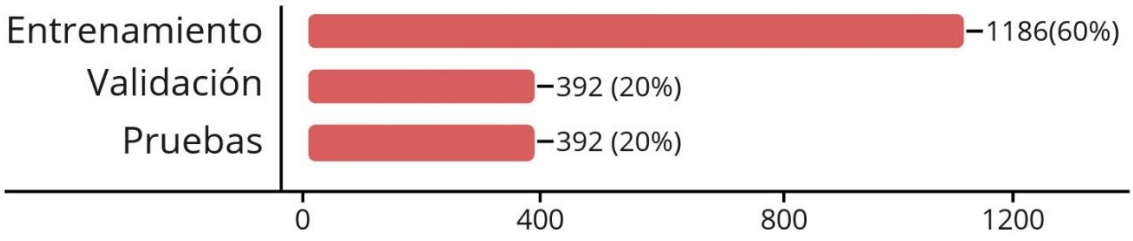


Figura 25. División del conjunto de datos

Fuente: elaboración propia

3.2.3. Limpieza y etiquetado de datos

Cada video del conjunto de datos está conformado por imágenes de 360x480 px. Sin embargo, no toda la información de dichas imágenes es relevante para este proyecto. Se estableció una región de interés (ROI) para reducir el ruido existente dentro de las imágenes y limpiar el conjunto de datos. La ROI son las manos del signante, más adelante se explicará el proceso de obtención de la ROI.

El conjunto de datos fue etiquetado en función de cada frase (Figura 26. Etiquetado de videos). El orden esta referenciado por el Anexo 7.1. La forma de clasificación de los videos está ligada a la arquitectura propuesta, ya que se va a clasificar datos de series temporales(videos).

Figura 26. Etiquetado de videos

Fuente: elaboración propia

3.2.4. Construcción del conjunto de datos

El primer paso en la construcción del conjunto de datos es definir un estándar de imágenes por video. Al tener muestras de 20 personas diferentes existió una variabilidad en la duración de cada video, por lo que no todos los videos contaban como el mismo número de imágenes. Se decidió construir 2 conjuntos de datos para obtener más resultados en el entrenamiento del modelo. Los formatos establecidos fueron 64 y 128 imágenes por video. En el caso de los videos que no llegasen a cumplir con el número máximo de imágenes (64 o 128) se añadieron valores aleatorios.

3.2.5. Retrospectiva de las iteraciones

Se llevaron a cabo 3 iteraciones en esta etapa. En las 2 primeras iteraciones, el etiquetado de los datos se llevó a cabo de manera incorrecta. En ambas iteraciones el conjunto de datos se clasificó en función del vocabulario existente dentro de las 20 frases. Durante la primera iteración se dividió los videos por imágenes, luego, a cada imagen en donde se detectó una seña se la etiquetó con un identificador (Anexo 7.2). En la segunda iteración se dividió los videos en función de la ROI (manos del signante), en cuanto al proceso de etiquetado, se dividió cada imagen por frase y luego se etiqueto cada imagen con un identificador (Anexo 7.3).

Los errores cometidos en el proceso de etiquetado no se consideraron un riesgo por lo que se avanzó a la siguiente etapa (Modelado). No mitigar este riesgo repercutió de manera negativa en el avance del proyecto. Las arquitecturas implementadas en la siguiente etapa mostraron resultados que no cumplían con los criterios de aceptación plasmados en el *Machine learning canvas.* La razón principal del problema fue que se rompió el componente temporal al etiquetar los datos a nivel de imagen. Fue necesario realizar una refactorización del proyecto. En la tercera iteración se solucionó dicho problema con lo mencionado en la sección 3.2.3.

3.3. Modelado

En la etapa de modelado se propone una arquitectura de red neuronal para resolver la problemática del proyecto, se entrena y calibra dicha arquitectura con el fin de obtener un modelo que cumpla con los criterios de aceptación. Todas las implementaciones de código serán registradas en un repositorio de GitHub[6].

3.3.1. Arquitectura de la red neuronal

La arquitectura que se utilizará es un híbrido CNN – LSTM, basado en [3], [58]. Se escogió está arquitectura ya que se pretende desarrollar una red neuronal simple que pueda escalar en el tiempo por medio de la metodología escogida. La Figura 27 muestra el proceso de la arquitectura propuesta. Primero se recibe un video de entrada, para cada imagen del video se extrae la región de interés.

[6] https://github.com/JDjuxx/Interprete_LSEC

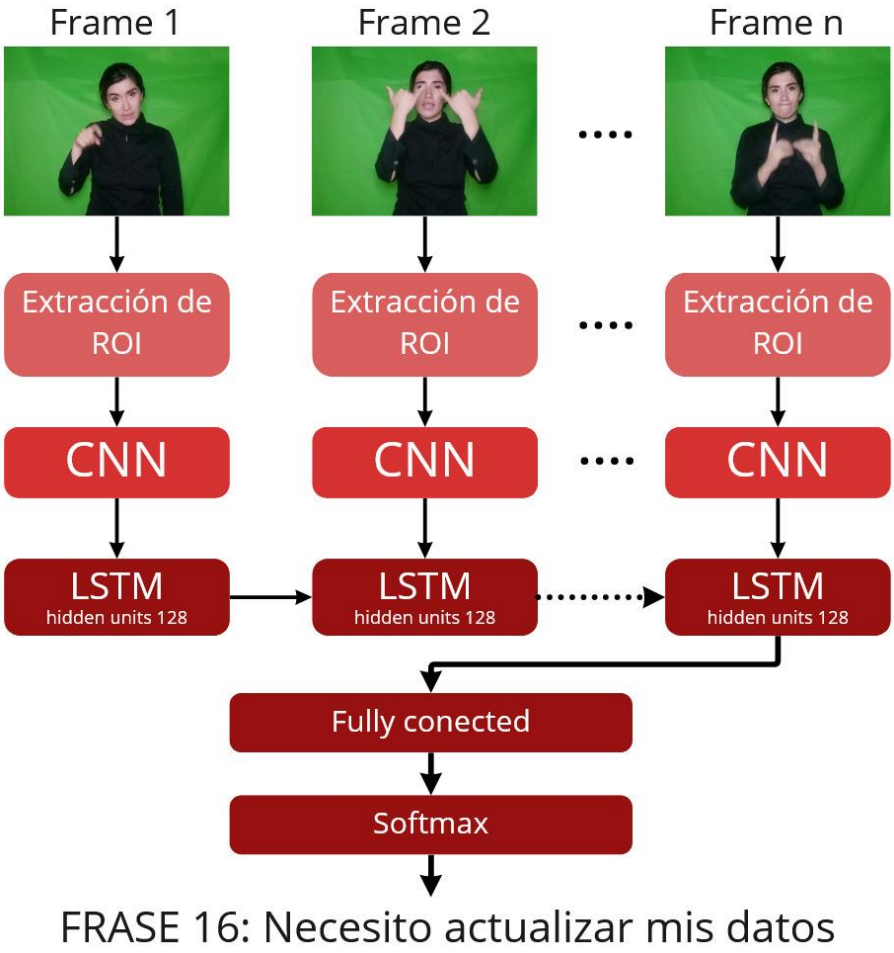


Figura 27. Arquitectura de red neuronal

Fuente: elaboración propia

Región de interés

Se utilizó la librería Mediapipe de Google para obtener los puntos de referencia de la región de las manos (Figura 18), se seleccionaron los mínimos y máximos valores en el eje (x, y) de la imagen. Estos valores permitieron crear un cuadro delimitador que contiene la mano detectada del signante. Con el cuadro delimitador se extrae una nueva imagen con la ROI. Posteriormente se redimensiona la imagen a un estándar 50x50 px, con el objetivo de que todos los datos tengan las mismas dimensiones. Finalmente, la imagen es convertida en un arreglo de 50x50x3, en donde 3 representa los canales RGB de la imagen. Se agregó un arreglo de las mismas dimensiones con valor 0 para rellenar el conjunto de datos en las situaciones en las cuales solo se detectó una mano del signante dentro de la imagen.

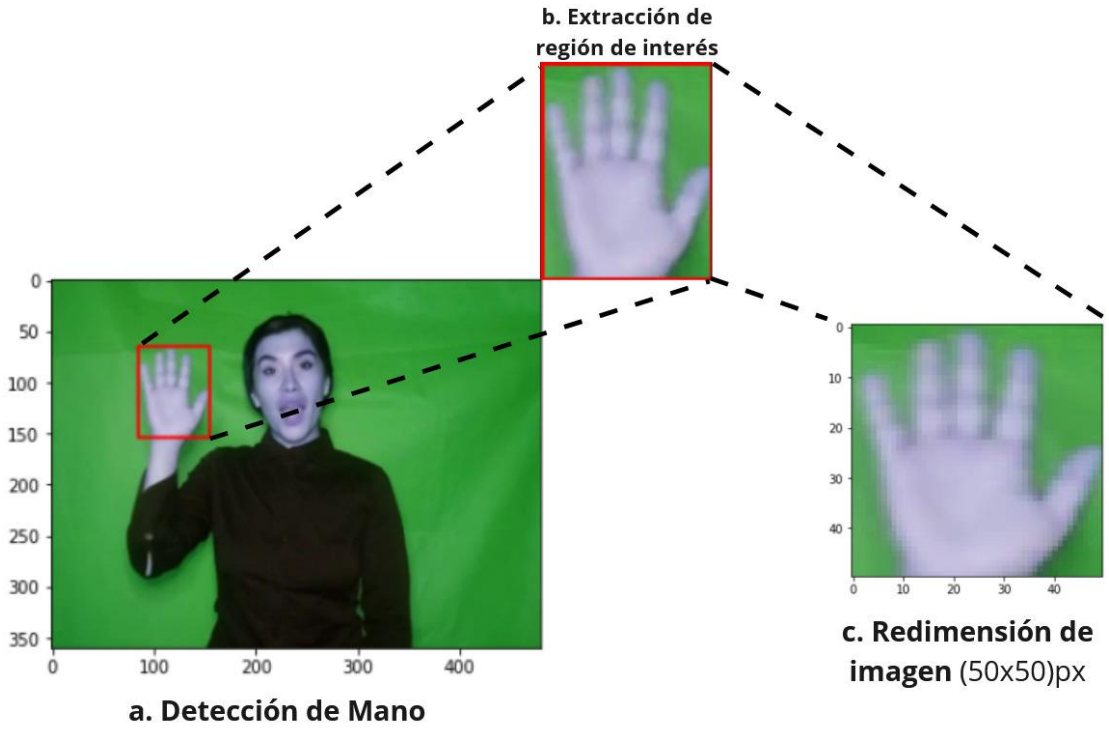


Figura 28. Proceso de obtención de región de interés

Fuente: elaboración propia

Red neuronal convolucional y recurrente

La red neuronal convolucional (Figura 29) se encarga de procesar la imagen y obtener un vector de características x, el vector es la entrada de la red neuronal recurrente (LSTM) para cada instante de tiempo t, donde t es el número de imágenes por video definido en la sección 3.2.4, finalmente el resultado pasa por una capa *fully conected* y *softmax* que clasifica la salida en una de las 20 frases en LSEC.

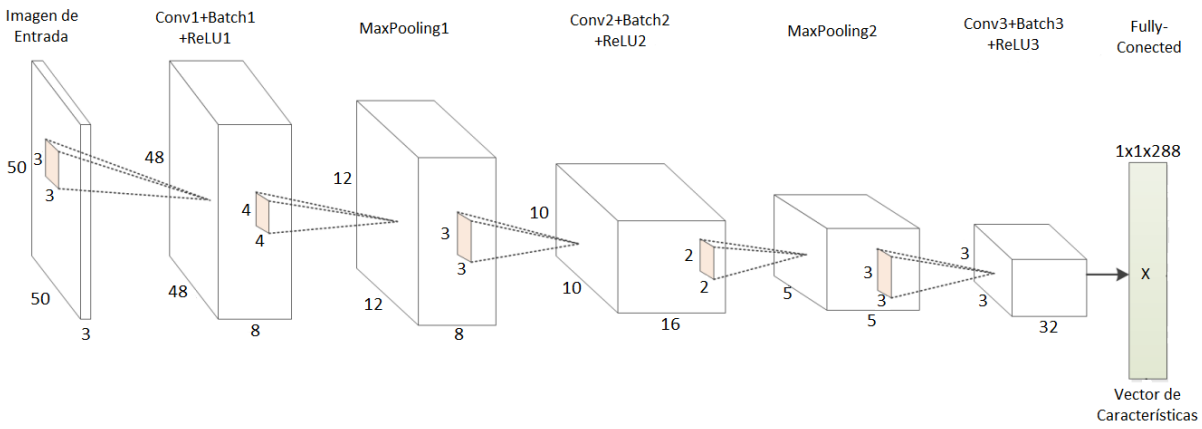


Figura 29. Red neuronal convolucional (CNN)

Fuente: elaboración propia

3.3.2. Entrenamiento

El objetivo de esta etapa es seleccionar el mejor modelo que cumpla con 2 criterios de aceptación. El primero, la precisión del modelo debe ser lo más cercana al 100%. El segundo, la diferencia entre la precisión del modelo en el conjunto de entrenamiento y validación no debe ser mayor a 5 puntos.

Se trabajó en base al componente iterativo de la metodología para seleccionar el mejor modelo posible. Durante cada entrenamiento del modelo los hiperparámetros fueron ajustados en función de los resultados del entrenamiento anterior. La información relacionada a cada entrenamiento ejecutado se encuentra registrada en un archivo CSV[7]. En la Figura 30 se puede apreciar los resultados del mejor modelo entrenado.

[7] https://drive.google.com/file/d/1rcEqAqR7f3Zc_a90Z1R5BJn4LAQrZFb6/view?usp=sharing

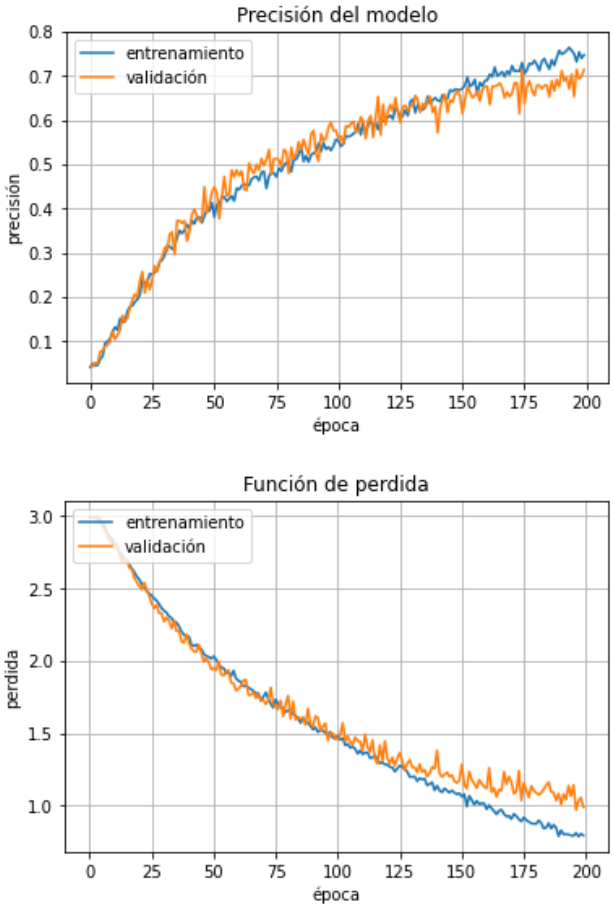


Figura 30. Resultados del modelo escogido

Fuente: elaboración propia

El modelo escogido posee un porcentaje de precisión en el conjunto de entrenamiento de 74.8%, mientras que, el porcentaje de precisión en el conjunto de validación es de 71.42%.

3.3.3. Otras arquitecturas

Se implementaron 2 arquitecturas diferentes con el fin de encontrar mejores resultados en cuanto a la precisión del modelo. En la Tabla 4 se puede apreciar que la implementación de dichas arquitecturas no presentó una mejoría en la precisión del modelo. En la arquitectura CNN-BiLSTM puede deberse a la baja cantidad de datos en el conjunto de entrenamiento y validación. En la arquitectura ResNet-LSTM la razón apunta a que el extractor de características pre-entrenado no responde bien a este tipo de datos.

Cabe la pena resaltar que no se realizaron pruebas exhaustivas sobre estas arquitecturas, como si se lo hizo con la propuesta en la sección 3.3.1, por lo que no se descarta que se puedan utilizar en un futuro.

Tabla 4. Comparación de resultados entre arquitecturas

Arquitectura	Precisión en entrenamiento	Precisión en validación
CNN-LSTM	74.8%	71.42%
CNN-BiLSTM	73.8%	68.34%
ResNet-LSTM	69.21%	65.5%

3.3.4. Retrospectiva de las iteraciones

En un principio se propuso implementar la arquitectura [3]. Sin embargo, debido a la complejidad del modelo y a la confusión en el etiquetado de imágenes (sección 3.2.5), se tomó la decisión de implementar una parte reducida de la arquitectura, con el fin de desarrollar una arquitectura sencilla que pueda aumentar su complejidad progresivamente con el uso de la metodología.

4. RESULTADOS

4.1. Evaluación

En esta última etapa se evaluará al modelo en un ambiente real, es decir, el modelo será puesto a prueba con datos que no ha visto anteriormente. Para esta tarea se utilizará el conjunto de datos de prueba (sección 3.2.2). El desempeño del modelo será medido con una matriz de confusión.

4.1.1. Resultados matrices de confusión

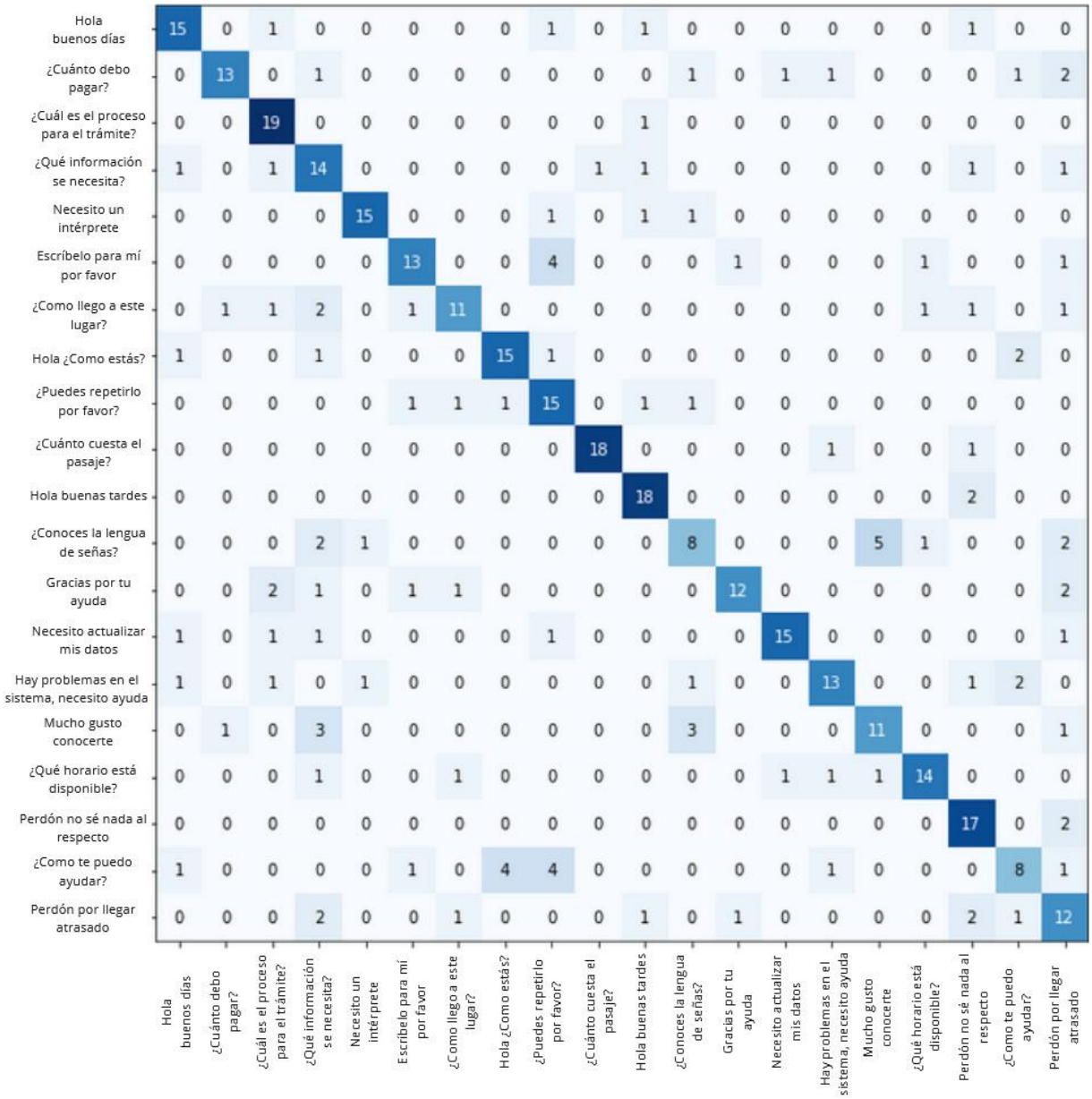


Figura 31. Matriz de confusión datos de prueba

Fuente: elaboración propia

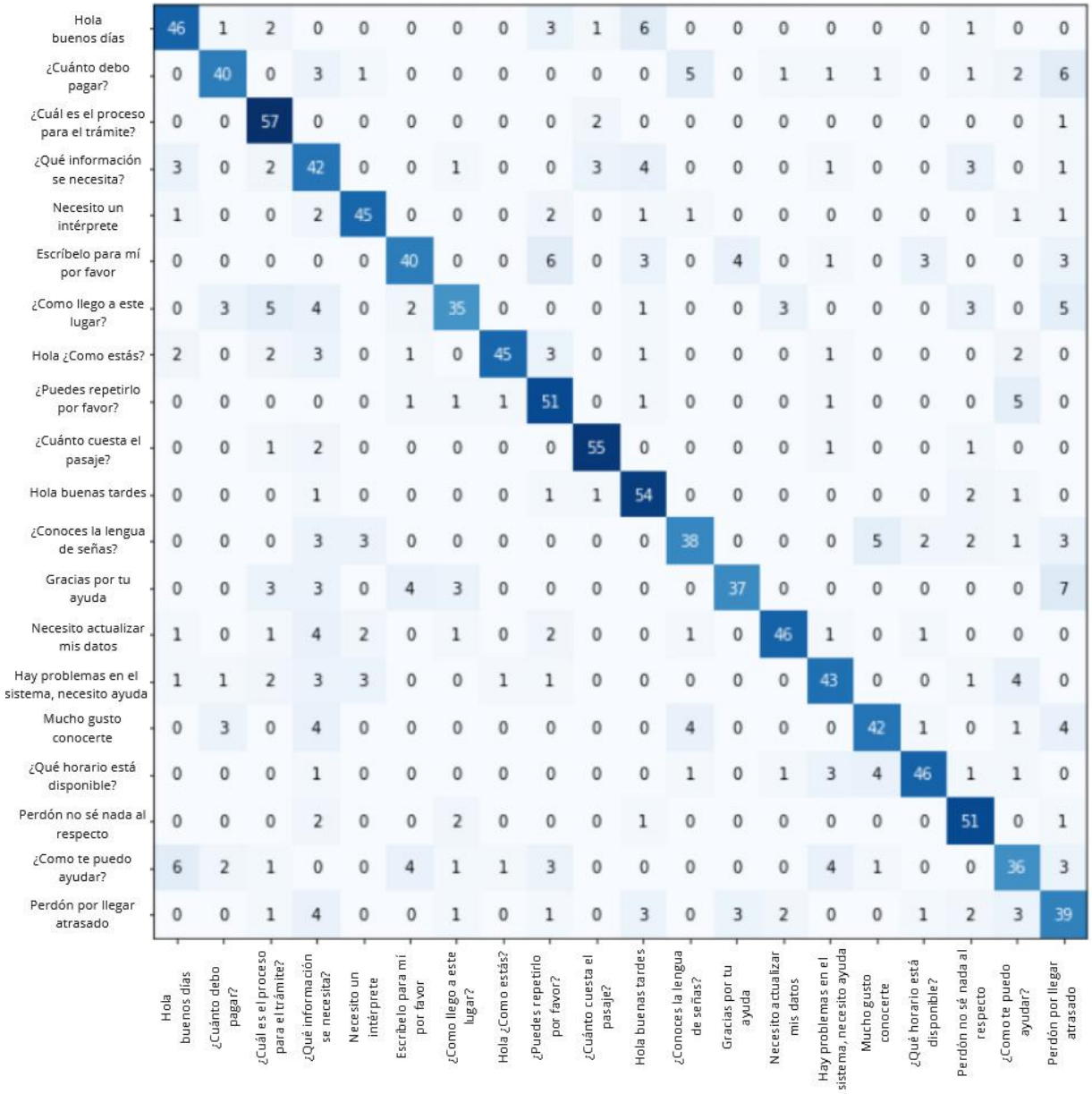


Figura 32. Matriz de confusión datos de entrenamiento

Fuente: elaboración propia

En una matriz de confusión la diagonal representa el número de verdaderos positivos (vp) que predijo el modelo. Se puede apreciar en la Figura 31 y Figura 32 que la frase que más acertó el modelo fue “¿Cuál es el proceso para el trámite?”, mientras que las frases que más fallaron fueron “¿Cómo llego a este lugar?” (Figura 32), “¿Cómo te puedo ayudar?” y “¿Conoces la lengua de señas?” (Figura 31). Se analizó que el fallo al predecir algunas frases está relacionado con el vocabulario existente dentro de la lista de las 20

frases en LSEC. Las señas como: “por favor”, “ayuda”, “¿Cómo?”, “hola” o “¿Cuánto?” se repiten en más de una frase. Por ejemplo, en la matriz de entrenamiento la frase “Hola buenos días” fue clasificada de forma errónea con la frase “Hola buenas tardes” en 6 ocasiones. También la frase “Escríbelo para mí por favor” fue confundida con la frase “¿Puedes repetirlo por favor?” en 6 oportunidades.

4.1.2. Desempeño del Modelo

Se evaluó el desempeño del modelo para los 392 videos del conjunto de datos de pruebas. Este cálculo se realizó mediante la Ecuación 1, en donde la exactitud es igual a la división entre los verdaderos positivos (vp) y verdaderos negativos (vn) con la suma de los positivos (p) y negativos (n).

$$Exactitud = \frac{vp + vn}{p + n}$$

Ecuación 1. Fórmula de exactitud en la matriz de confusión

Para los datos de prueba se obtuvo un porcentaje de exactitud igual a 70.6%, mientras que para los datos de entrenamiento el porcentaje fue de 74.8%. Al comparar los resultados obtenidos entre los datos de pruebas y los de entrenamiento se observa que no se presenta ningún problema de *overfitting* ya que la diferencia entre los porcentajes de exactitud de reconocimiento es del 4.2%.

El modelo también fue analizado en términos de tiempo de procesamiento. Mediante una librería de Python se calculó el tiempo promedio en el que la red procesaba en infería sobre cada imagen de los 1187 videos del conjunto de datos de entrenamiento. Se determinó que el tiempo de procesamiento promedio del intérprete es de 50ms por cada video.

4.1.3. Prototipo del intérprete de LSEC

La intención de este proyecto es crear un modelo inteligente como base para el desarrollo de un producto software. Por esta razón, en el alcance de este proyecto se decidió no llevar a cabo la etapa de Despliegue y Monitoreo. Sin embargo, se implementó un prototipo que recibe como entrada un video de una persona signando en LSEC, clasifica

el video en una de las 20 frases en LSEC, el resultado se muestra en texto y voz, véase Figura 33.

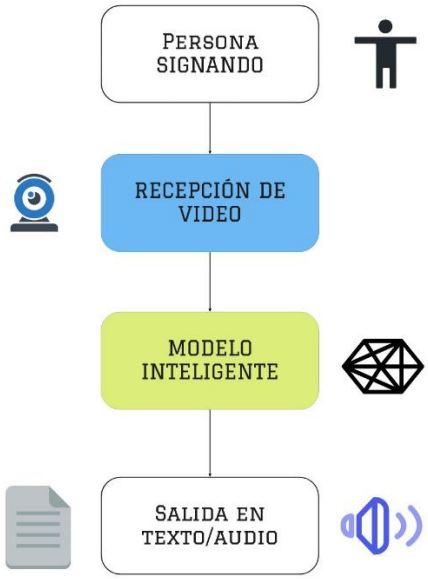


Figura 33. Prototipo de intérprete de LSEC

Fuente: elaboración propia

4.1.4. Retrospectiva de iteraciones

Se aplicó la matriz de confusión a las otras arquitecturas propuestas en la sección 3.3.3. Los resultados indican que no existe mucha variación entre las diferentes arquitecturas implementadas. El número de aciertos y fallos en la clasificación presentan valores similares al modelo escogido. Se seguirán realizando pruebas sobre estas arquitecturas.

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- La arquitectura propuesta (Figura 27) permitió desarrollar un intérprete capaz de: extraer la información de la región de interés (Figura 28), clasificar e interpretar 20 frases en LSEC a voz en tiempo real.

- Por medio de los lineamientos y rúbricas especificados en la sección 3.1.3, se logró construir un conjunto de datos, el cual está compuesto por 20 frases cotidianas en LSEC. El conjunto de datos resultó ser completo en términos de valor, variedad y veracidad. En cuanto a la característica de volumen, no cumplió con los criterios de éxito debido a la cantidad reducida de videos con los que se contó para el entrenamiento, validación y pruebas.

- El desempeño del intérprete fue analizado en términos de tiempo de procesamiento y exactitud. Con respecto al primero, el modelo es capaz de realizar inferencias en tiempo real, es decir en menos de 300 ms [4]. Por otro lado, la exactitud de clasificación del modelo en el conjunto de datos de prueba fue de 70.6%.

- El intérprete fue evaluado en un ambiente real, gracias al conjunto de datos de prueba. En base a los resultados se determinó que, el modelo está listo para avanzar a la siguiente etapa (Despliegue), sin embargo, esto no forman parte del alcance de este proyecto.

- La metodología CRISP ML(Q)[8] fue clave en el análisis de riesgos y en la detección de puntos críticos de mejora para la implementación de un prototipo de interprete de LSEC. Este prototipo representa una base sólida a futuro para el desarrollo de un producto de *software*.

5.2. Recomendaciones

- Los resultados de la arquitectura propuesta (Figura 29) fueron 74,8% de exactitud en el conjunto de datos de entrenamiento y 70.6% en pruebas. Si bien se implementó otro tipo de arquitecturas (sección 3.3.3), no se realizaron pruebas suficientes para descartarlas. Se recomienda realizar más pruebas sobre las otras arquitecturas propuestas como también implementar capas atención o arquitecturas *many to many* para mejorar el porcentaje de precisión del modelo.

- Durante la fase de recolección de datos, la mayoría de los estudiantes no cumplieron con los lineamientos establecidos para la grabación de los videos en la primera iteración. El corregir las fallas supuso tiempo de retraso en la finalización de esta fase. Se recomienda realizar la recolección de datos en un ambiente controlado, es decir, un espacio donde cada signante sea grabado por un mismo dispositivo y se pueda corregir al instante en caso de no acatar con los lineamientos.

- En la etapa de modelado se notó una carencia en el conjunto de datos. El volumen de videos con los que se entrenó al intérprete fue bajo en comparación con otros estudios [3], [9], [58]. Utilizar técnicas de aumento de datos para solventar esta falla puede resultar determinante para mejorar el rendimiento del modelo.

- El componente iterativo y de aseguramiento de la calidad de la metodología propuesta [8] permitió detectar de manera precisa las fallas cometidas durante cada iteración. Sin embargo, hubo riesgos que no se mitigaron a tiempo y repercutió en errores graves en etapas posteriores. Se recomienda ser más estrictos a la hora definir el impacto que tiene un riesgo sobre la culminación de cada etapa.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] World Health Organization, “Deafness and hearing loss,” *Deafness and hearing loss*, Apr. 01, 2021.

[2] Consejo Nacional para la Igualdad de Discapacidades, “Estadísticas de Discapacidad.” https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/ (accessed Apr. 27, 2022).

[3] Q. Xiao, X. Chang, X. Zhang, and X. Liu, “Multi-Information Spatial–Temporal LSTM Fusion Continuous Sign Language Neural Machine Translation,” *IEEE Access*, vol. 8, pp. 216718–216728, 2020, doi: 10.1109/ACCESS.2020.3039539.

[4] M. E. Benalcazar *et al.*, “Real-time hand gesture recognition using the Myo armband and muscle activity detection,” in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, Oct. 2017, pp. 1–6. doi: 10.1109/ETCM.2017.8247458.

[5] “Redes Neuronales Recurrentes - Jordi TORRES.AI,” Sep. 22, 2019. https://torres.ai/redes-neuronales-recurrentes/ (accessed Apr. 27, 2022).

[6] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997, doi: 10.1109/78.650093.

[7] “Deep Learning – Introducción práctica con Keras - Jordi TORRES.AI.” https://torres.ai/deep-learning-inteligencia-artificial-keras (accessed Apr. 27, 2022).

[8] S. Studer *et al.*, “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” Mar. 2020.

[9] K. Bantupalli and Y. Xie, “American Sign Language Recognition using Deep Learning and Computer Vision,” *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, pp. 4896–4899, Jan. 2019, doi: 10.1109/BIGDATA.2018.8622141.

[10] S. K. Liddell, *Grammar, gesture, and meaning in American Sign Language*. Cambridge University Press, 2003.

[11] T. Yuan *et al.*, “Large scale sign language interpretation,” *Proceedings - 14th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2019*, May 2019, doi: 10.1109/FG.2019.8756506.

[12] D. Eberle, E. Parks, S. Eberle, and J. Parks, “Sociolinguistic Survey Report of the Ecuadorian Deaf Community,” *SIL International*, pp. 4–14, 2012, Accessed: May 15, 2022. [Online]. Available: https://www.sil.org/resources/publications/entry/50228

[13] W. C. Stokoe and M. Marschark, “Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf,” *The Journal of Deaf Studies and Deaf Education*, vol. 10, no. 1, pp. 3–37, Jan. 2005, doi: 10.1093/DEAFED/ENI001.

[14] M. Cruz, “Gramática de la lengua de señas mexicana,” El Colegio de México, México, 2008. Accessed: May 11, 2022. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=3071717

[15] I. S. García, “Lenguaje de señas entre niños sordos de padres sordos y oyentes,” Universidad Mayor de San Marcos, Perú, 2002. Accessed: May 12, 2022. [Online]. Available: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/1229?show=full

[16] K. S. Fu, R. C. González, and C. S. G. Lee, *Robótica: control, detección, visión e inteligencia*. McGraw-Hill, 1988.

[17] V. Tyagi, “Understanding Digital Image Processing,” *Understanding Digital Image Processing*, Sep. 2018, doi: 10.1201/9781315123905.

[18] G. Santillán, I. Danilo, C. Sánchez, and V. Manuel, “La visión artificial y los campos de aplicación,” *Tierra Infinita*, vol. 1, no. 1, pp. 98–108, Dec. 2015, doi: 10.32645/26028131.76.

[19] S. Turkyilmaz, “Characterization of cracks in bridge beams with welded wire fabric reinforcement using image processing,” Texas Tech University, Texas, 2001. Accessed: May 23, 2022. [Online]. Available: https://ttu-ir.tdl.org/handle/2346/19000?show=full

[20] R. C. Gonzalez and R. E. Woods, *Digital Image Processing Third Edition*, 3rd ed. Pearson International Edition prepared by Pearson Education, 2002. Accessed: May 23, 2022. [Online]. Available: http://debracollege.dspaces.org/handle/123456789/428?mode=full

[21] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications*. InTech, 2011. Accessed: Jun. 02, 2022. [Online]. Available: https://books.google.com/books?hl=es&lr=&id=JuaODwAAQBAJ&oi=fnd&pg=PR11&dq=Artificial+neural+networks+methodological+advances+and+biomedical+applications&ots=XVkPXV8KTb&sig=yQkWKLG3MzMHb2X6_aeoWrqszeU

[22] A. Dongare, R. Kharde, and A. Kachare, “Introduction to artificial neural network,” *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 1, Jul. 2012.

[23] J. M. Alvarez, “BackPropagation archivos - Blog de Jose Mariano Alvarez,” 2019. http://blog.josemarianoalvarez.com/tag/backpropagation/ (accessed Jul. 11, 2022).

[24] I. Gavilán, “Catálogo de componentes de redes neuronales (II): funciones de activación,” May 20, 2018. https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-ii-funciones-de-activacion/ (accessed Jul. 11, 2022).

[25] H. Kukreja, N. Bharath, C. S. Siddesh, and S. Kuldeep, “An introduction to artificial neural network,” *Department of Electrical & Electronics Engineering, School of Engineering & Technology, Jain University*, vol. 1, no. 5, 2017, Accessed: Jun. 05, 2022. [Online]. Available: https://www.researchgate.net/profile/Kuldeep-Shiruru/publication/319903816_AN_INTRODUCTION_TO_ARTIFICIAL_NEURAL_NETWORK/links/59c0fe55458515af305c471a/AN-INTRODUCTION-TO-ARTIFICIAL-NEURAL-NETWORK.pdf

[26] X. Ying, “An Overview of Overfitting and its Solutions,” *Journal of Physics: Conference Series*, vol. 1168, p. 022022, Feb. 2019, doi: 10.1088/1742-6596/1168/2/022022.

[27] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, “Understanding batch normalization,” in *Advances in Neural Information Processing Systems*, 2018, vol. 2018-December.

[28] H. Gao, J. Pei, and H. Huang, “Demystifying dropout,” in *36th International Conference on Machine Learning, ICML 2019*, 2019, vol. 2019-June.

[29] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” Nov. 2015.

[30] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional neural networks: an overview and application in radiology,” *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018, doi: 10.1007/s13244-018-0639-9.

[31] W. Jianxin, *Introduction to Convolutional Neural Networks*. China: Nanjing University of Technology, 2017.

[32] R. M. Schmidt, “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,” Nov. 2019.

[33] J. H. Chan, “RNN and Vanishing/Exploding Gradients ,” Jun. 20, 2019. https://osfork.com/2019/06/23/rnn-and-vanishing-exploding-gradients/ (accessed Jul. 11, 2022).

[34] I. Mindlin, “Reconocimiento de Lengua de Señas con redes neuronales recurrentes,” Universidad Nacional de La Plata, Río de la Plata, 2021. Accessed: Jun. 16, 2022. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/129853

[35] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A Critical Review of Recurrent Neural Networks for Sequence Learning,” May 2015.

[36] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, “Recent Advances in Recurrent Neural Networks,” Dec. 2017, doi: 10.48550/arxiv.1801.01078.

[37] N. McCullum, “Long Short-Term Memory Networks (LSTMs).” https://nickmccullum.com/python-deep-learning/lstms-long-short-term-memory-networks/ (accessed Jul. 11, 2022).

[38] P. Granell, “Redes Neuronales Recurrentes: Una aplicación para los mercados bursátiles,” Universidad de Barcelona, Barcelona, 2018. Accessed: Jun. 16, 2022. [Online]. Available: http://diposit.ub.edu/dspace/handle/2445/124249

[39] C. Shearer, “The CRISP-DM model: the new blueprint for data mining,” *Journal of data warehousing*, vol. 5, pp. 13–22, 2000.

[40] S. Studer *et al.*, “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” Mar. 2021, Accessed: May 12, 2022. [Online]. Available: https://arxiv.org/pdf/2003.05155.pdf

[41] “CRISP-ML(Q).” https://ml-ops.org/content/crisp-ml (accessed Jul. 11, 2022).

[42] L. Dorard, “Machine Learning Canvas — OWNML,” 2015. https://www.ownml.co/machine-learning-canvas (accessed Jul. 11, 2022).

[43] K. R. Srinath, “Python-The Fastest Growing Programming Language,” *International Research Journal of Engineering and Technology*, vol. 4, no. 12, Dec. 2017, Accessed: Jun. 17, 2022. [Online]. Available: https://www.academia.edu/download/61651202/IRJET-V4I126620200101-109100-1nbuifw.pdf

[44] D. Rolon-Mérette, M. Ross, T. Rolon-Mérette, and K. Church, “Introduction to Anaconda and Python: Installation and setup,” *The Quantitative Methods for Psychology*, vol. 16, no. 5, pp. S3–S11, May 2020, doi: 10.20982/tqmp.16.5.S003.

[45] “Project Jupyter | Home.” https://jupyter.org/ (accessed Jun. 17, 2022).

[46] Microsoft, “Crear o editar archivos .csv para importarlos a Outlook.” https://support.microsoft.com/es-es/office/crear-o-editar-archivos-csv-para-importarlos-a-outlook-4518d70d-8fe9-46ad-94fa-1494247193c7 (accessed Jun. 17, 2022).

[47] W3schools, “What is JSON.” https://www.w3schools.com/whatis/whatis_json.asp (accessed Jun. 17, 2022).

[48] K. Mistry and A. Saluja, “An Introduction to OpenCV using Python with Ubuntu,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2016 IJSRCSEIT* |, vol. 5, no. 2, pp. 2456–3307, 2016, Accessed: Jun. 17, 2022. [Online]. Available: https://www.academia.edu/download/53206501/CSEIT16129.pdf

[49] Google, “MediaPipe Hands,” 2020. https://google.github.io/mediapipe/solutions/hands (accessed May 23, 2022).

[50] C. Lugaresi *et al.*, “MediaPipe: A Framework for Building Perception Pipelines,” Jun. 2019, doi: 10.48550/arxiv.1906.08172.

[51] S. Sharma, “The Ultimate Beginner’s Guide to TensorFlow,” *Towards Data Science*, Aug. 22, 2020. https://towardsdatascience.com/the-ultimate-beginners-guide-to-tensorflow-41d3b8e7d0c7 (accessed Jun. 17, 2022).

[52] fchollet, “Introduction to Keras for Researchers,” *Keras*, Feb. 10, 2020. https://keras.io/getting_started/intro_to_keras_for_researchers/ (accessed Jun. 17, 2022).

[53] scikit-learn developers, “Getting Started — scikit-learn 1.1.1 documentation.” https://scikit-learn.org/stable/getting_started.html (accessed Jun. 17, 2022).

[54] NumPy Developers, “NumPy documentation .” https://numpy.org/doc/stable/ (accessed Jun. 17, 2022).

[55] The Matplotlib Development team, “Matplotlib — Visualization with Python.” https://matplotlib.org/ (accessed Jun. 17, 2022).

[56] G. E. Ávila, “Instituto Tecnológico Superior CRE-SER,” 2018. http://creser.edu.ec/ (accessed Jul. 11, 2022).

[57] Consejo Nacional de Igualdad de Discapacidades, “Diccionario de Lengua de Señas Ecuatoriana Gabriel Román,” *http://www.plataformaconadis.gob.ec/~platafor/diccionario/*, 2014.

[58] T. Liu, W. Zhou, and H. Li, “Sign language recognition with long short-term memory,” *Proceedings - International Conference on Image Processing, ICIP*, vol. 2016-August, pp. 2871–2875, Aug. 2016, doi: 10.1109/ICIP.2016.7532884.

7. ANEXOS

7.1. Anexo I: Lista de frases en LSEC

Tabla 5. Lista de 20 frases cotidianas en LSEC

ID	Frase
Lsec_01	Hola buenos días.
Lsec_02	¿Cuánto debo pagar?
Lsec_03	¿Cuál es el proceso para el trámite?
Lsec_04	¿Qué información se necesita?
Lsec_05	Necesito un intérprete
Lsec_06	Escríbelo para mí por favor
Lsec_07	¿Como llego a este lugar?
Lsec_08	Hola ¿Como estás?
Lsec_09	¿Puedes repetirlo por favor?
Lsec_10	¿Cuánto cuesta el pasaje?
Lsec_11	Hola buenas tardes
Lsec_12	¿Conoces la lengua de señas?
Lsec_13	Gracias por tu ayuda
Lsec_14	Necesito actualizar mis datos

Lsec_15	Hay problemas en el sistema, necesito ayuda
Lsec_16	Mucho gusto conocerte
Lsec_17	¿Qué horario está disponible?
Lsec_18	Perdón no sé nada al respecto
Lsec_19	¿Como te puedo ayudar?
Lsec_20	Perdón por llegar atrasado

7.2. Anexo II: Etiquetado de imágenes primera iteración

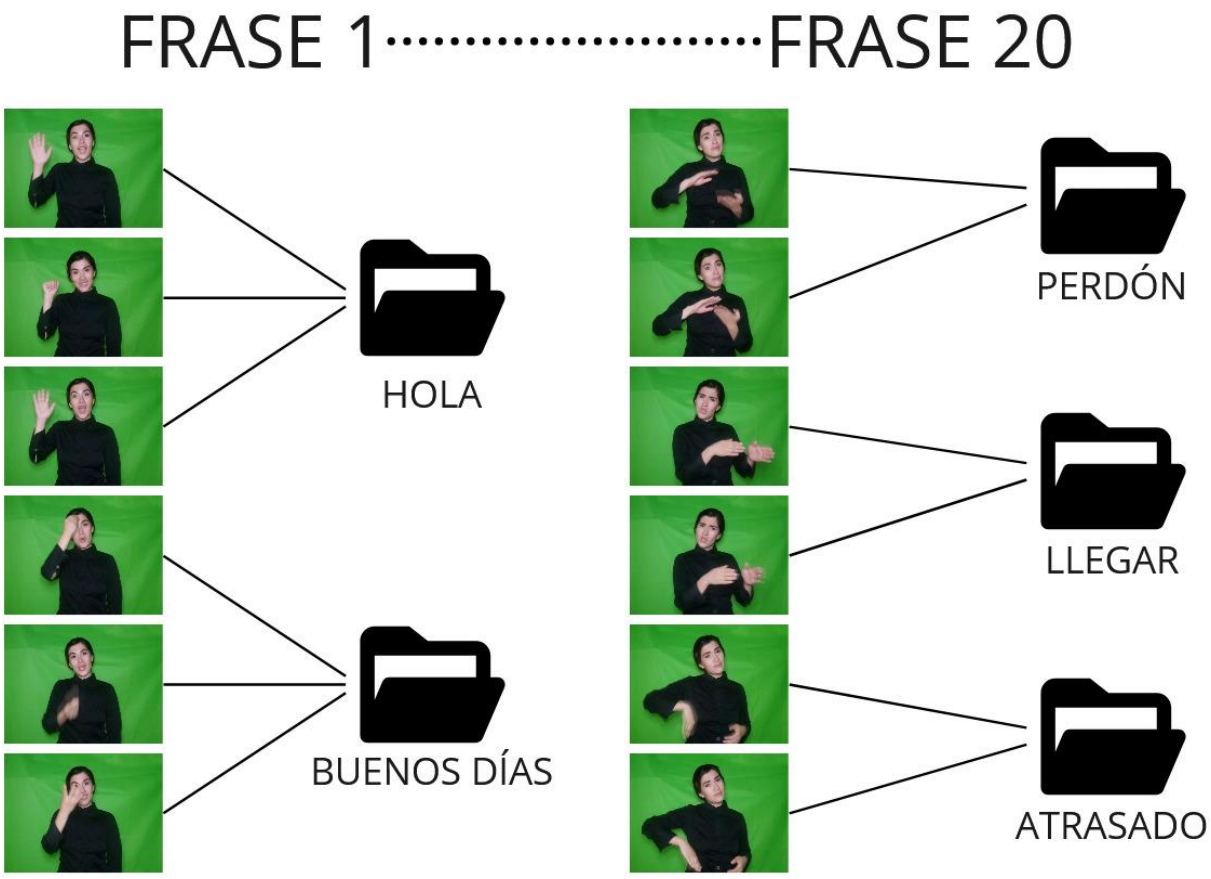


Figura 34. Etiquetado de imágenes primera iteración

Fuente: elaboración propia

7.3. Anexo III: Etiquetado de imágenes segunda iteración

Figura 35. Etiquetado de imágenes segunda iteración

Fuente: elaboración propia

7.4. Anexo IV: Matriz de confusión datos de entrenamiento arquitectura CNN-BiLSTM

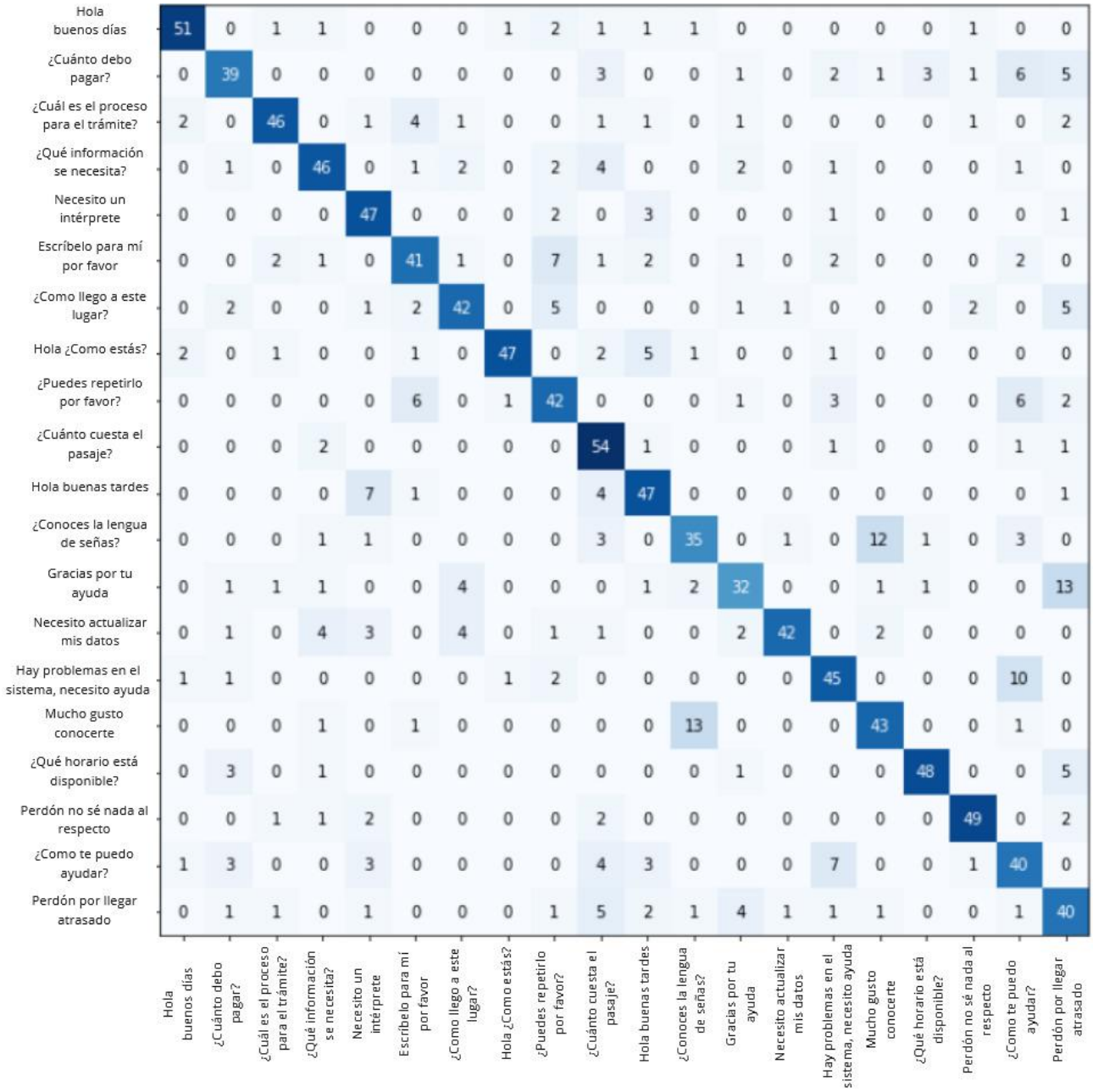


Figura 36. Matriz de confusión CNN-BiLSTM entrenamiento

Fuente:elaboración propia

7.5. ANEXO V: Matriz de confusión datos de prueba arquitectura CNN-BiLSTM

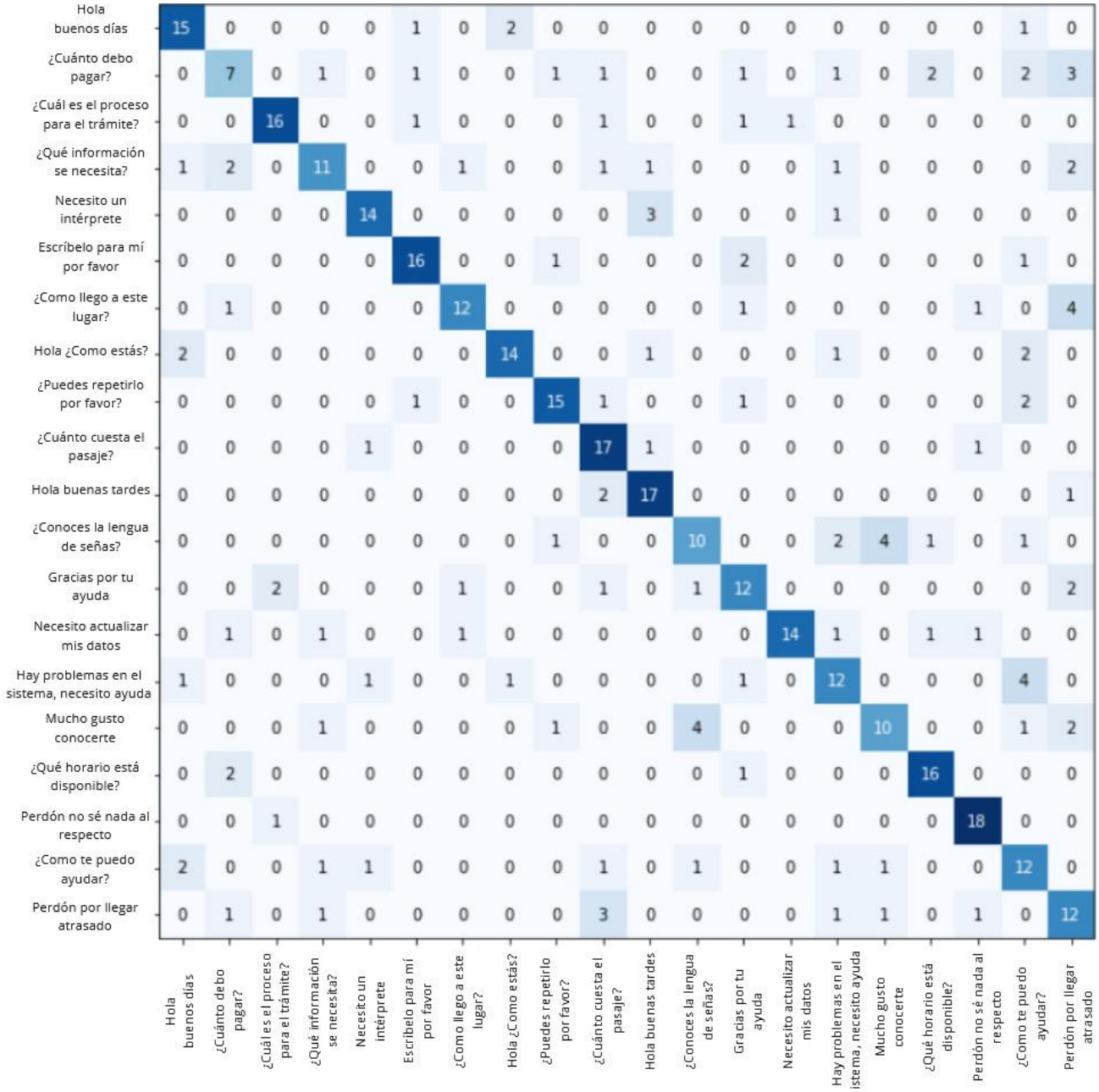


Figura 37. Matriz de confusión CNN-BiLSTM pruebas

Fuente: elaboración propia

7.6. ANEXO VI: Matriz de confusión datos de entrenamiento arquitectura ResNet-LSTM

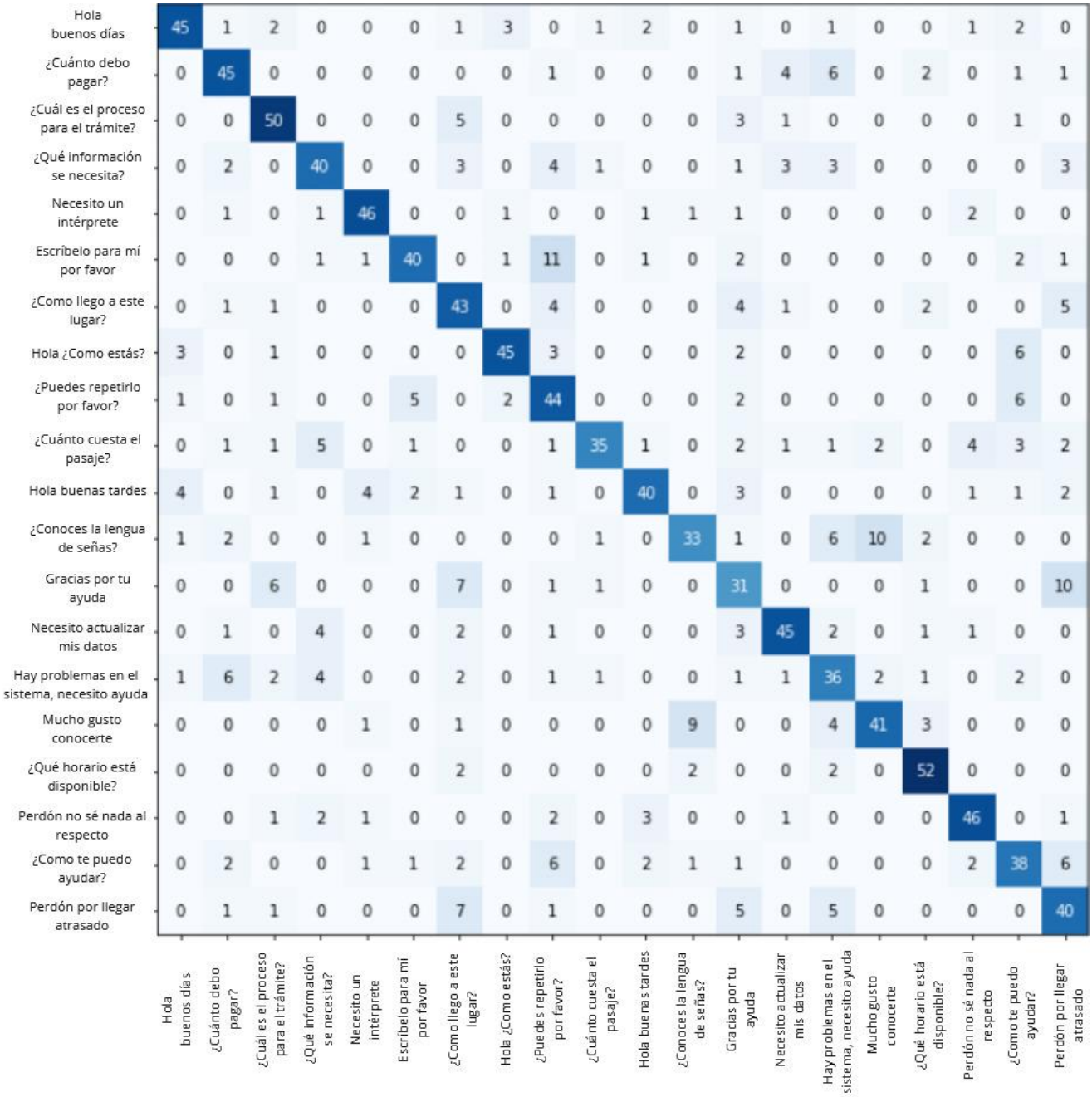


Figura 38. Matriz de confusión ResNet-LSTM entrenamiento

Fuente: elaboración propia

7.7. ANEXO VII: Matriz de confusión datos de prueba arquitectura ResNet-LSTM

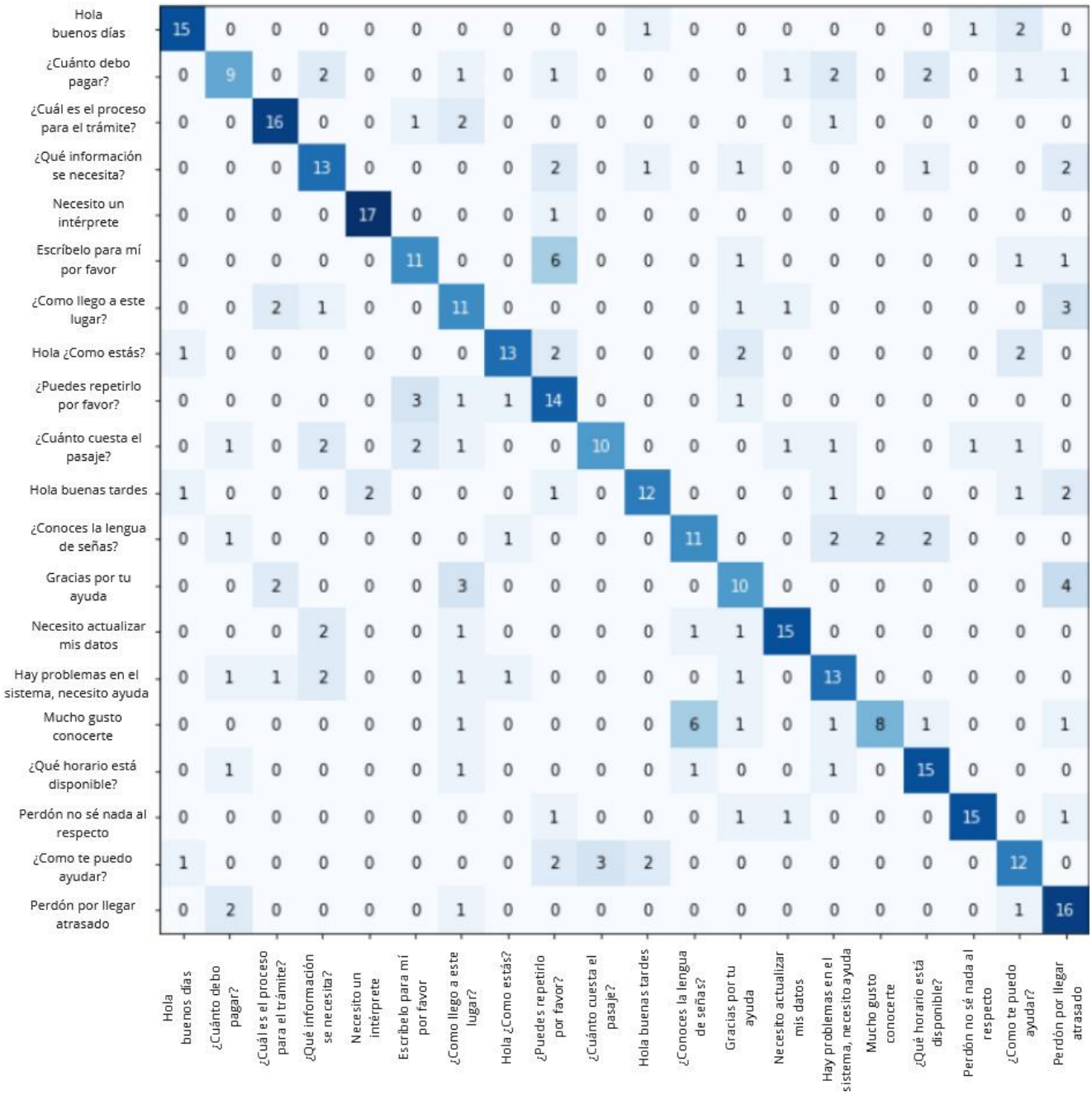


Figura 39. Matriz de confusión ResNet-LSTM pruebas

Fuente: elaboración propia