Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

Detección y clasificación de palabras a partir de la lectura de labios mediante técnicas de aprendizaje profundo y aprendizaje máquina.

TESIS PARA LA OBTENCIÓN DEL GRADO DE MAESTRÍA EN SISTEMAS COMPUTACIONALES

Presenta: **EDGAR ESAU MONTES DIAZ**

Asesor: **DR. VÍCTOR HUGO MARTÍNEZ SÁNCHEZ**

Co-asesor: **DR. IVÁN ESTEBAN VILLALÓN TURRUBIATES**

Tlaquepaque, Jalisco. Julio 2024

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a mi director de tesis, el Dr. Víctor Hugo Martínez Sánchez, cuya paciencia y apoyo fueron fundamentales. Su conocimiento y experiencia fueron cruciales durante el proceso de realización y planeación de este trabajo, acompañándome en momentos de incertidumbre y proporcionando herramientas para la resolución de problemas que, sin su apoyo, habrían sido determinantes para el éxito del proyecto.

Al CONAHCYT, agradezco por brindarme a mí y a miles de estudiantes en México la oportunidad de cursar una maestría, con el objetivo de difundir el conocimiento y formar personas capaces de aportar a la sociedad con lo aprendido.

Al ITESO y sus representantes, quienes siempre muestran un insaciable deseo por el conocimiento y la enseñanza, y que constantemente buscan opciones para hacer la educación accesible y de calidad.

También quiero agradecer al Dr. Iván Esteban Villalon Turrubiates, quien siempre estuvo pendiente de mi avance. Me brindó gran conocimiento y retroalimentación como asesor y maestro, y de quien he obtenido aprendizajes valiosos que me servirán toda la vida.

Agradezco especialmente a la Dra. Mildreth Isadora Alcaraz Mejía y al Dr. Luis Enrique González Jiménez que dedicaron su tiempo y esfuerzo para revisar y proporcionar valiosos comentarios sobre este trabajo. Sus contribuciones fueron fundamentales para mejorar la calidad y claridad de la investigación.

Deseo expresar mi gratitud a mis maestros, quienes me proporcionaron las bases del conocimiento necesario para alcanzar este logro. Su dedicación y compromiso con la enseñanza han sido esenciales para mi formación académica y profesional.

A las personas que colaboraron en la adquisición y análisis de los datos de video, así como a los participantes y a los encargados de la producción. Su ayuda fue clave para el desarrollo y éxito de este trabajo.

A mis amigos y compañeros de estudio, quienes siempre me brindaron su apoyo emocional y con quienes compartí grandes experiencias, frustraciones y conocimientos que fueron esenciales a lo largo de estos años de estudio.

A mi familia, a mi novia y mis amigos, quienes siempre me motivaron y alentaron. Su apoyo incondicional fue la base emocional que me permitió estudiar la maestría y concluir este trabajo.

Finalmente, agradezco a todas las personas que, directa o indirectamente, contribuyeron a la realización de este trabajo y dejaron una huella en este logro.

A todos y cada uno, muchas gracias.

Dedicatoria

Dedico este trabajo a mis padres, mi hermana y mi tía, cuyo amor y apoyo incondicional me han brindado la fortaleza y los valores necesarios para enfrentar las adversidades. Su guía y ejemplo han sido fundamentales para forjar mi carácter y darme el coraje para alcanzar mis metas.

A mi novia, por su infinita paciencia y constante motivación. Gracias por mantenerme centrado y por alentarme a perseguir mis sueños con entusiasmo y determinación.

Resumen

La información contenida en los labios y el rostro proporciona datos significativos para el reconocimiento del habla y el procesamiento del lenguaje natural. Aunque existen diversas metodologías para abordar este problema, la mayoría se basan en conjuntos de datos en chino e inglés. Este trabajo se enfoca en la implementación de un modelo alimentado por un conjunto de datos en español, desarrollado específicamente para este proyecto. El objetivo es generar una herramienta útil capaz de clasificar palabras en español latino, sirviendo como referencia para futuras investigaciones orientadas a la creación de subtítulos automáticos para personas con discapacidad auditiva.

Utilizando técnicas de Aprendizaje Máquina, por sus siglas en inglés (ML) y Aprendizaje Profundo, por sus siglas en inglés (DL), se busca extraer características clave para la Lectura Automática de Labios, por sus siglas en inglés (ALR), a partir de videos. El modelo se alimenta de un conjunto de datos propio. Este conjunto de datos está diseñado para incorporar técnicas de extracción de características utilizando Redes Neuronales Convolucionales, por sus siglas en inglés (CNN) y memoria a corto plazo. Se pretende utilizar un modelo recurrente como las Redes Neuronales de Memoria de Corto-Largo Plazo, por sus siglas en inglés (LSTM).

La arquitectura del modelo permitirá utilizar la información temporal en videos, donde eventos anteriores ayudarán a mejorar la predicción futura de palabras. De esta manera, se ofrece una solución avanzada y accesible para el reconocimiento del habla en español latino.

Tabla de contenido

Lista de figuras

Lista de tablas

Lista de acrónimos y abreviaturas

3DCvT		3D Convolutional Vision Transformer
3DCvT-III		3D Convolutional Vision Transformer model version III
Adam		Adaptative Moments
ALR		Automatic Lip Reading
ANN		Artificial Neural Network
API		Application Programming Interface
ASST-GCN		Adaptative Semantic-Spatio-Temporal Graph Convolutional Network
Bbox		Bounding Box
BiGRU		Bidirectional Gated Recurrent Unit
CER		Character Error Rate
CNN		Convolutional Neural Network
CPU		Central Processing Unit
DataFrame		Panda's Series indexed by a value
DL		Deep Learning
FN		False Negatives
FP		False Positives
GPU		Graphics Processing Unit
HD		High Definition
HMM		Hidden Markov Model
IA		Inteligencia Artificial
ITESOWordNet		Modelo propio de lectura automática de labios para clasificación de palabras
IOU		Intersection Over Union
JAX		Just After Execution
LabelImg		Graphical image annotation tool
LRLPs		Lip Reading related Landmark Points
LRW		Lip Reading in the Wild
LRW-1000		Lip Reading in the Wild 1000
LSTM		Long Short-Term Memory
LTU		Linear Threshold Unit
MEXLR2023		Conjunto de datos propio para lectura automática de labios en español latino
MIRACL-VC1		MIRACL-VC1 dataset
ML		Machine Learning
MLP		Multilayer Perceptron
Ndarray		Matriz multidimensional en NumPy
NumPy		Numerical Python
Pandas		Python Data Analysis Library
PascalVOC		Pascal Visual Object Classes
PER		Phoneme Error Rate
ResNet50		Residual Network of fifty convolutional layers
RNNs		Recurrent Neural Networks
ROI		Region Of Interest
SVM		Support Vector Machine
TN		True Negatives
TNR		True Negative Rate
TOG		Trabajo de Obtención de Grado
TP		True Positives

TPR		True Positive Rate
V100		NVIDIA V100 Tensor Core GPU
VGG16		Visual Geometry Group Network of sixteen convolutional layers
VGG19		Visual Geometry Group Network of nineteen convolutional layers
VLRF		Visual Lip Reading Feasibility
VSR		Visual Speech Recognition
VUs		Visual Units
WER		Word Error Rate
XML		Extensible Markup Language

1. INTRODUCCIóN

La comunicación es fundamental en la interacción humana y se lleva a cabo mediante diferentes medios, siendo uno de los más importantes el lenguaje. Sin embargo, existen limitaciones que pueden reducir la capacidad de comunicación asertiva, como la discapacidad auditiva o el ruido ambiental. En este contexto, el ALR se presenta como una herramienta. Esta herramienta permite reconocer el habla de una persona a través del procesamiento del movimiento de sus labios.

El ALR es el proceso que permite reconocer el habla de una persona a través del procesamiento del movimiento de sus labios. Existen diferentes enfoques del ALR. En [1] se emplea el reconocimiento del habla como un punto de datos extra en conjunto del sonido para predicción del habla utilizando una metodología multimodal. Otra aplicación detallada en [2], integra el ALR al procesamiento de la lengua de señas en ruso para mejorar su desempeño.

El avance en la investigación del ALR ha permitido una ampliación en las áreas de aplicación de esta tecnología, lo cual podría beneficiar a la sociedad en general. Entre las posibles aplicaciones se encuentran la robótica y la enseñanza de idiomas. También incluye las plataformas de streaming y los videojuegos. Además, proporciona una herramienta de apoyo para personas con discapacidad auditiva y trabajadores en ambientes ruidosos.

En México, según la Secretaría de Salud, al menos 2.3 millones de personas sufren de discapacidad auditiva hasta el año 2021. Aunque algunos medios utilizan intérpretes visuales para proporcionar información a personas con discapacidad auditiva, pocas industrias lo implementan de manera regular. Personas con discapacidad auditiva podrían beneficiarse de una herramienta que les permita mejorar su capacidad de comunicación. Además, trabajadores que necesitan comunicarse en ambientes ruidosos y estudiantes que desean aprender español también se beneficiarían mediante la generación automática de subtítulos a partir del reconocimiento de labios.

Actualmente existe una variedad de metodologías basadas en ML y DL para atacar el problema del ALR. El enfoque mayormente utilizado y probado es la división de las tareas para la predicción; en un frente se recolectan las características del rostro para su procesamiento y por el otro, se realiza el tratamiento de la información secuencial.

Para la recolección de características en el estado del arte los autores de [3] proponen un modelo. Este modelo es llamado por sus siglas en inglés Adaptative Semantic-Spatio-Temporal Graph Convolutional Network (ASST-GCN). Esta técnica se basa en el procesamiento de la semántica y las características espaciotemporales de los datos de entrenamiento. En [4] se emplean CNNs, basadas en codificador-decodificador. Estas CNNs se utilizan para tareas de predicción a nivel de píxel en la detección de objetos salientes.

En [5] se utilizan CNNs para extraer características faciales. Estas características se usan para el reconocimiento de rostros. Como parte de la recolección de características en el estado del arte, se emplean

algunas arquitecturas de redes neuronales. Por ejemplo, en [6] se implementan las arquitecturas VGG19 y ResNet50 para la extracción de características faciales, incluyendo la boca.

El procesamiento del movimiento de los labios requiere del tratamiento de fotogramas continuos. Por esta razón, algunos trabajos enfocados en el procesamiento del lenguaje natural [7] utilizan el modelo Hidden Markov Model (HMM), por sus siglas en inglés. Este modelo reconoce secuencias partiendo del estado anterior y clasifica nuevas observaciones. Otras investigaciones [6], [8] implementan LSTM como una alternativa para extraer información temporal, probando su efectividad en tareas de predicción de secuencias. Existen trabajos [8] donde se emplea LSTM junto con Support Vector Machine (SVM), por sus siglas en inglés. Este enfoque se utiliza para localizar la región de la boca, alcanzando un porcentaje de exactitud de localización del 88%. Los autores de [1] con enfoque en el audio. Emplean Redes Neuronales Recurrentes (RNNs), por sus siglas en inglés. Tienen la capacidad de mantener una memoria interna, lo que les permite recordar información relevante al tratar secuencias de datos.

Una de las tareas cruciales para la implementación del ALR es la extracción de información significativa del rostro. La velocidad del habla, iluminación, pronunciación, resolución e incluso el tono de la piel [9] agregan complejidad al procesamiento del lenguaje natural y a la detección de características visuales. En la investigación [9] se habla acerca del tratamiento del lenguaje natural a partir de visemas, los cuales se usan en la teoría de la articulación del habla. En esta teoría, se considera que la pronunciación se puede definir en términos de cómo interactúan los diferentes elementos articulatorios, como los labios y la lengua, para producir sonidos específicos. En el trabajo [2] se menciona el concepto de la hiperarticulación. La hiperarticulación es una técnica de pronunciación exagerada que, según los autores, aumenta las posibilidades de que personas con alguna discapacidad auditiva sean capaces de reconocer el habla con mayor facilidad.

Los autores de [10] prueban su modelo usando conjuntos de datos en otros idiomas diferentes al español. Algunos de estos conjuntos de datos incluyen el Lip Reading in the Wild (LRW) y Lip Reading in the Wild 1000 (LRW-1000), por sus siglas en inglés. LRW es un conjunto de datos masivo basado en el idioma inglés, mientras que LRW-1000 está basado en el chino mandarín. El uso de estos conjuntos de datos a gran escala permite utilizar técnicas de DL. El conjunto de datos LRW utilizado en [10] consta de videos provenientes de programas de radio y televisión. En LRW se integran 500 palabras de las más comunes en el idioma de 1000 hablantes y 550 millones de discursos. Otro conjunto de datos estudiado también en el trabajo [10] es el LRW-1000 el cuál proviene de programas de la televisión que contiene 1000 clases, cada una con una palabra en el lenguaje chino mandarín, obtenido de más de 2000 hablantes. Algunos modelos [5], [11] emplean conjuntos de datos obtenidos de la web, basados en videos abstraídos de la plataforma YouTube.

Debido a la diversidad de lenguajes y acentos alrededor del mundo, los datos con los que trabaja el estado del arte que mayormente abarcan el idioma inglés, chino [10], ruso [2] y árabe [7]. Sin embargo, estos datos son irrelevantes para la creación de un modelo de predicción automático enfocado en personas hablantes del español latino en México. Por lo tanto, reutilizar un conjunto de datos ya existente sería ineficiente para el trabajo actual. Actualmente existe un conjunto de datos llamado Visual Lip Reading Feasibility (VLRF), por sus siglas en inglés. Este conjunto fue creado por los autores de [12] y contiene 3 horas de video de personas hablantes del castellano. Sin embargo, la ambigüedad en los acentos, la

pronunciación y utilización de palabras diferentes entre el español latino y el español castellano hacen que este conjunto de datos sea poco eficiente para el propósito de este trabajo.

Dado las limitaciones que conlleva la aplicación de conjuntos de datos ajenos al español latino, el objetivo del modelo propuesto requiere generar un nuevo conjunto de datos. Estos datos se obtendrán de grabaciones de video realizadas en un entorno controlado.

El entrenamiento del modelo implica tomar un conjunto de videos etiquetados a manera de palabras individuales. Estos videos son procesados utilizando una combinación de CNN para la obtención de características del rostro y LSTM para la predicción y el tratamiento de secuencias de fotogramas. El trabajo propuesto difiere del enfoque multi modalidad [2], ya que se centra únicamente en el entrenamiento y procesamiento de datos visuales. Esto hace que la capacidad de extender el conjunto de datos en el futuro sea más flexible.

El enfoque actual se centra en características derivadas del movimiento de la boca y las áreas circundantes a ella [3]. En contraste, el modelo propuesto utiliza la barbilla, las mejillas, el movimiento de la mandíbula, los labios y su contorno, abarcando así completamente la mitad inferior del rostro. Este enfoque busca evaluar si la integración de estas características, previamente inexploradas en el estado del arte, puede proporcionar datos significativos para tareas de ALR.

1.1. Antecedentes

La discapacidad auditiva afecta a aproximadamente 2.3 millones de personas en México, según datos de la secretaria de salud [13]. A pesar de su alta incidencia, la discapacidad auditiva recibe menos atención en comparación con otras discapacidades. Esto debido a que no es fácilmente perceptible. Como resultado las personas con discapacidad auditiva enfrentan dificultades para ser incluidas en la sociedad y acceder a servicios esenciales.

En el ámbito de la tecnología, se han desarrollado diversas herramientas para apoyar a las personas con discapacidad auditiva, como audífonos, implantes cocleares y aplicaciones de traducción de lenguaje de señas. Sin embargo, estas soluciones tienen limitaciones, especialmente en contextos donde la lectura labial es una habilidad crítica para la comprensión verbal. Algunas personas con problemas auditivos han aprendido a entender el habla observando los movimientos de los labios y otras señales faciales. Esto se conoce como lectura labial, y es fundamental para que las personas con discapacidad auditiva puedan comunicarse de manera más efectiva con quienes los rodean.

Los avances recientes en tecnología como la Inteligencia Artificial (IA), el procesamiento de imágenes y el DL han abierto nuevas posibilidades para mejorar el ALR. El uso de CNN, LSTM y algoritmos de aprendizaje automático permite el análisis y reconocimiento de patrones complejos, facilitando la traducción visual del habla a texto. Estos sistemas pueden capturar y procesar características faciales como el movimiento de la lengua, los labios y la mandíbula, proporcionando una interpretación precisa del lenguaje hablado.

1.2. Justificación

Las tendencias actuales de investigación se centran primordialmente en la adquisición de características lingüísticas en idiomas como el inglés, chino mandarín y ruso. En contraste, el presente trabajo propone sentar las bases para el desarrollo de una herramienta de apoyo al aprendizaje de la lectura de labios para individuos con capacidad auditiva reducida. Este propósito se materializará a través de la creación de un modelo capaz de clasificar palabras de forma automática, con la capacidad de integrarse en diversos medios de comunicación y plataformas de streaming. Para ello, se recurrirá a técnicas avanzadas de ML y DL, adaptadas específicamente al movimiento de los labios de los hablantes mexicanos del español latino.

Con base en los datos proporcionados por la Secretaría de Salud de México, se estima que aproximadamente 2.3 millones de personas padecen algún tipo de discapacidad auditiva en el país. La eventual implementación de esta herramienta en medios digitales que carecen de intérpretes podría conllevar beneficios significativos para esta población. Además de facilitar la comunicación en las actividades cotidianas, este avance podría tener un impacto sustancial en la calidad de vida de los afectados.

1.3. Problema

En México, aproximadamente 2.3 millones de personas tienen discapacidad auditiva. Esto constituye una barrera para la comunicación efectiva en el hogar, en entornos educativos, laborales y en cualquier situación donde la comunicación verbal sea esencial. La lectura labial es una habilidad crucial para estas personas. La mayoría de las soluciones tecnológicas actuales de asistencia se centran en idiomas diferentes al español latino, reduciendo su aplicabilidad y efectividad para los hablantes de esta región.

Los investigadores destacan la importancia de desarrollar herramientas de asistencia basadas en el procesamiento del lenguaje natural y DL para mejorar la lectura labial. Estudios previos han demostrado que técnicas que incorporan el uso de CNNs y LSTMs, son efectivas para el reconocimiento del habla a partir de datos visuales. Sin embargo, la mayoría de estos estudios se han realizado en el contexto de idiomas como el inglés, chino y ruso, dejando un vacío significativo en la investigación para el español latino.

En el ámbito de la lectura labial, modelos basados en DL utilizan grandes conjuntos de datos en inglés y chino para entrenar algoritmos de reconocimiento del habla. Aunque estas soluciones han tenido éxito en otros idiomas, no abordan las necesidades de los hablantes de español latino.

Este trabajo propone resolver la falta de herramientas efectivas para la lectura labial en hablantes de español latino en México. Por ello, destaca la importancia de desarrollar un modelo de reconocimiento del habla basado en un conjunto de datos específico de esta región. La efectividad del modelo se logrará utilizando técnicas avanzadas de ML y DL. El procesamiento del lenguaje de señas y la creación de hardware especializado está fuera del alcance de la investigación actual.

Resolver este problema de comunicación es crucial para mejorar la inclusión social y la calidad de vida de las personas con discapacidad auditiva en México. Proporcionar una herramienta efectiva para la lectura labial facilitará la interacción y comunicación de las personas afectadas en diversos entornos. De esta manera se busca promover la integración social y tener un impacto significativo en su calidad de vida. La carencia de una solución adecuada resultará en desafíos significativos para las personas con discapacidad auditiva, lo cual dificultará su plena integración en la sociedad. Este contexto propiciará la continuación del aislamiento social, así como la restricción en su capacidad para acceder a la información y a oportunidades laborales. Estas circunstancias acarrearán un impacto negativo en su calidad de vida y bienestar general.

1.4. Objetivos

1.4.1. Objetivo general:

Desarrollar un modelo avanzado de clasificación de palabras en español latino, capaz de detectar y procesar el área de la boca y sus alrededores. Utilizando el marco de referencia PyTorch y alimentado por el conjunto de datos propio MEXLR2023, se busca generar un modelo competitivo para tareas de ALR. Este modelo no solo pretende ser una herramienta efectiva en la clasificación de palabras, sino también sentar las bases para futuras investigaciones que faciliten la integración social de personas con discapacidad auditiva.

1.4.2. Objetivos específicos:

- Indagar acerca de herramientas avanzadas para el procesamiento de imágenes faciales con Python.
- Analizar el estado del arte del ALR.
- Recopilar información sobre la presentación y formato de datos utilizados en modelos de referencia similares.
- Definir el formato y la estructura del conjunto de datos propio MEXLR2023.
- Crear un conjunto de datos basado en el español latino.
- Limpiar y normalizar el conjunto de datos para estandarizar la duración de los videos.
- Etiquetar el área de la boca en cada fotograma del conjunto de datos.
- Desarrollar un modelo para la localización precisa del área de la boca utilizando las etiquetas generadas.
- Entrenar y validar el modelo de localización utilizando el fragmento de datos de entrenamiento y validación en MEXLR2023.
- Evaluar el modelo de localización utilizando los datos de prueba de MEXLR2023.
- Desarrollar un modelo de clasificación de palabras a partir del conjunto de datos.
- Entrenar y validar el modelo de clasificación con los datos de entrenamiento y validación de MEXLR2023.
- Evaluar el modelo de clasificación utilizando los datos de prueba de MEXLR2023.
- Optimizar el modelo de clasificación con base en los resultados obtenidos durante las pruebas.
- Documentar los resultados y hallazgos de la investigación mediante el desarrollo de un Trabajo de Obtención de Grado (TOG).

1.5. Innovación tecnológica

El trabajo propuesto introduce una innovación significativa al campo del reconocimiento del habla y la lectura de labios. Esto se logra mediante el desarrollo de un modelo específicamente diseñado para hablantes de español latino, utilizando un conjunto de datos propio denominado MEXLR2023. Esta propuesta se distingue por abordar una necesidad no satisfecha. La mayoría de los estudios sobre ALR se centran en idiomas con fonética diferente al español latino, lo que implica diferencias en el movimiento y la posición de los labios y áreas circundantes a la boca. Esto deja un vacío significativo en la tecnología disponible para los hablantes del español latino. Al crear un modelo basado en un conjunto de datos específico de esta región, el proyecto responde directamente a las necesidades de una población significativa que hasta ahora ha sido subatendida en términos de desarrollo tecnológico.

La innovación principal radica en la generación y utilización del conjunto de datos MEXLR2023. Este nuevo conjunto de datos incorpora características faciales y lingüísticas propias de personas hablantes del español latino. Este enfoque proporciona una base más precisa y relevante para entrenar modelos de DL basados en palabras, mejorando la exactitud y eficiencia del ALR para esta comunidad lingüística.

Además, la metodología propuesta incluye el uso combinado de CNN para la extracción de características faciales y LSTM para el procesamiento de secuencias temporales. Al utilizar un enfoque dual es posible incrementar la posibilidad de una mayor precisión en la detección y clasificación de palabras. Esto se logra aprovechando tanto las capacidades de las CNN para el análisis de imágenes, como las fortalezas de LSTM para el manejo de información temporal. La inclusión de técnicas avanzadas de procesamiento de imágenes y datos secuenciales, junto con la personalización del conjunto de datos, representa una innovación significativa en el campo.

El modelo resultante no solo busca mejorar la tecnología de reconocimiento del habla, sino también ofrecer una herramienta accesible y eficaz para personas con discapacidad auditiva en México. Al proporcionar una solución tecnológica y automática adaptada a sus necesidades, se promueve la inclusión social y se mejora la calidad de vida de una población vulnerable. Subrayando la relevancia y el impacto social del proyecto.

2. ESTADO DEL ARTE

2.1 Estado del arte del ALR

El ALR se ha convertido en un área de investigación de creciente interés debido a su potencial para mejorar significativamente la precisión del reconocimiento del habla en entornos ruidosos y su amplia aplicación en tecnologías de asistencia. Esta sección examina los avances recientes en el campo del Reconocimiento Automático del Habla, con un enfoque particular en los diversos métodos y estrategias para la extracción de características. Además, se analiza el uso de conjuntos de datos con distintas características visuales y lingüísticas, proporcionando una visión integral de las técnicas más innovadoras y efectivas que están moldeando el futuro del ALR.

2.1.1 Un método basado en Transformador de Visión Convolucional 3D

El estudio [10] aborda el problema de la lentitud en el procesamiento de varios cuadros continuos de imágenes. Este problema se resuelve al proponer un método basado en transformadores de visión 3D y convoluciones que capturan las características del movimiento de los labios.

El enfoque que utilizan los autores se denomina 3D Convolutional Vision Transformer, por sus siglas en inglés (3DCvT). Las características resultantes del procesamiento de 3DCvT se utilizan como entrada para la siguiente etapa.

Esta nueva etapa integra una Bidirectional Gated Recurrent Unit, por sus siglas en inglés (BiGRU), que se encarga del modelado de secuencias temporales. BiGRU es un tipo de RNN bidireccional que procesa la secuencia en ambas direcciones, hacia adelante y hacia atrás [14]. Finalmente, incorporan una capa totalmente conectada que se encarga de la clasificación resultante.

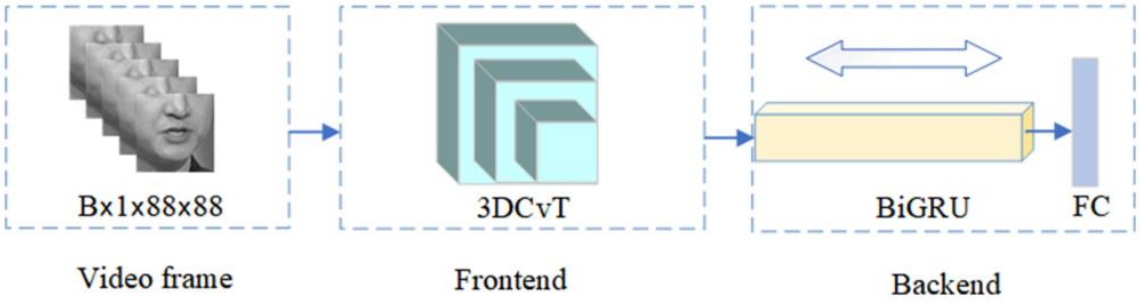

Figure 1 Vista de las diferentes etapas propuestas destacando la sección de 3DCvT en el trabajo A Lip Reading Method Based on 3D Convolutional Vision Transformer

A diferencia del trabajo propuesto, en [10] se prueba el modelo en conjuntos de datos en inglés y chino mandarín, resaltando la importancia y relevancia de la exploración de datos en español latino.

Los autores prueban su mejor modelo, llamado 3DCvT-III, basado en 3DCvT, utilizando los conjuntos de datos LRW y LRW1000. Los resultados obtenidos para estos dos conjuntos de datos son 88.5% de

exactitud para LRW y 57.5% para LRW-1000. La diferencia en los porcentajes se destaca debido a las diferentes características lingüísticas de los dos conjuntos de datos. Esta premisa resalta el impacto que puede tener un conjunto de datos en un modelo y la variabilidad en los resultados obtenidos al utilizar diferentes lenguajes como entrada.

2.1.2 Grafos adaptativos en tareas de ALR

En [3] se introduce el uso de grafos de CNNs para mejorar la precisión del ALR. Utilizan dos tipos de grafos adaptativos: el grafo semántico, derivado de la adyacencia compartida entre las matrices en el conjunto de datos, y el grafo dinámico espaciotemporal, basado en la matriz de adyacencia de los datos de entrada.

Una característica interesante sobre el tratamiento de los datos en este trabajo es el procesamiento que efectúan sobre las imágenes. A diferencia de la investigación propia, los autores añaden a su modelo una capa paralela a la encargada de extraer características visuales. Esta capa paralela se encarga de procesar el contorno de la boca y las áreas circundantes a ella, partiendo de puntos denominados Lip Reading related Landmark Points, por sus siglas en inglés (LRLPs).

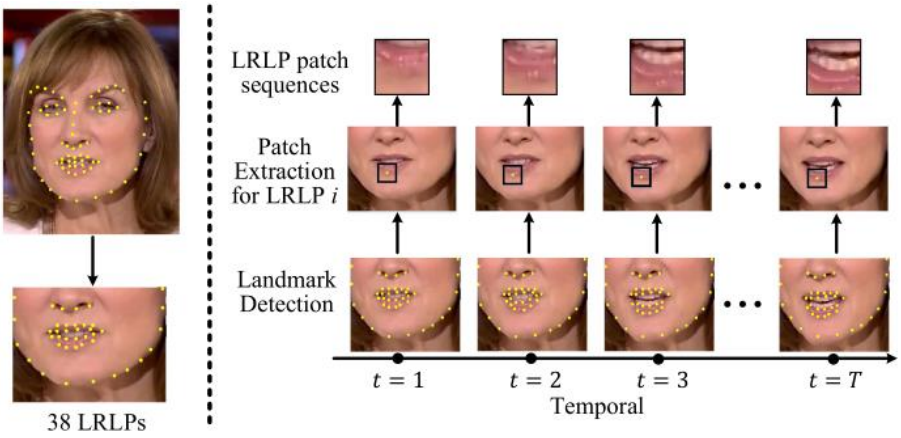

Figure 2 Ejemplo de extracción de secuencias de LRLPs en el trabajo Adaptative Semantic-Spatio-Temporal Graph Convolutional Network for Lip Reading

Finalmente, se utiliza una red de dos canales buscando combinar la información complementaria de la apariencia visual y el contorno de la boca. El modelo ASST-GCN se enfoca en el reconocimiento dinámico de la deformación del contorno de la boca, extrayendo información temporal y espacial.

Además, la integración de LRLPs enriquece el procesamiento semántico, permitiendo experimentación tanto a nivel de palabra como de oraciones completas. Por otro lado, el procesamiento visual utiliza CNNs para extraer las características visuales del rostro y la boca.

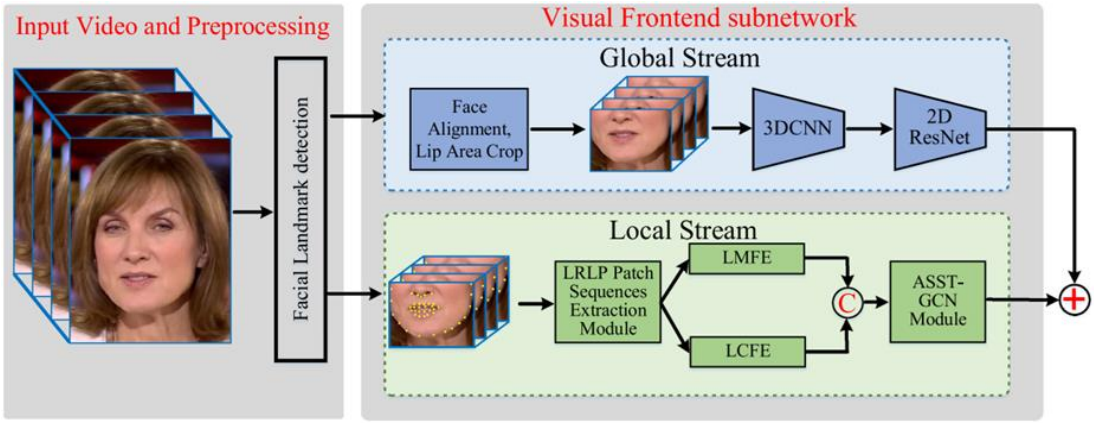

Figure 3 Red paralela de procesamiento de características visuales y del contorno de la boca en el trabajo Adaptative Semantic-Spatio-Temporal Graph Convolutional Network for Lip Reading

Un punto de datos a considerar en esta investigación, además de los ya mencionados con anterioridad, es el conjunto de datos utilizado. En este caso, al igual que en la investigación previa, se utiliza LRW, que recordemos se basa en el idioma inglés. El resultado final sobre LRW presenta un porcentaje de exactitud de 85.7% para su mejor modelo.

2.1.3 Sistema de reconocimiento de labios basado en visión y DL

El trabajo [6] propone un sistema ALR basado exclusivamente en video, empleando técnicas de DL. Los autores subrayan la necesidad del preprocesamiento del video de entrada, seleccionando solo fotogramas clave para aumentar la efectividad del modelo y eliminar redundancias.

Otro de los puntos clave mencionados es la minimización del procesamiento de características. Los autores señalan que, al utilizar píxeles para el procesamiento como en otras investigaciones, los datos pueden ser extensos y redundantes. Por lo tanto, la metodología propuesta en [6] consiste en realizar un análisis de seguimiento de labios en tiempo real. Posteriormente, se identifican formas y movimientos de la boca a través de puntos de referencia de la Región de Interés, por sus siglas en inglés (ROI).

Al enfocarse en ROIs, el sistema es capaz de localizar únicamente los puntos clave en cada fotograma, evitando así la necesidad de abstracción de características a nivel de píxel.

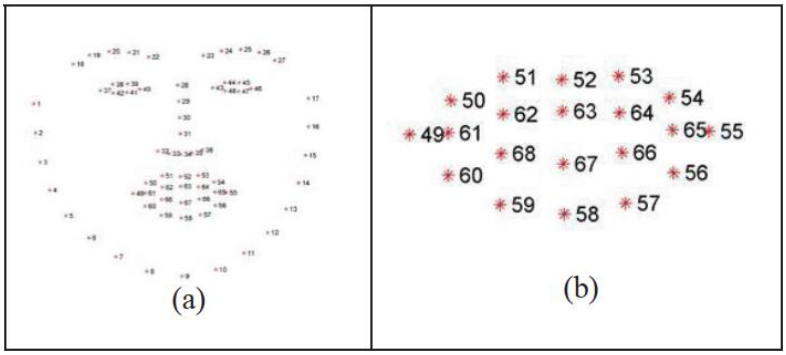

Figure 4 Detección de puntos referencia faciales y seccionado de la región de los labios en el trabajo Vision based Lip Reading System using Deep Learning

Entre otras aplicaciones, los autores proponen el enfoque de este tipo de tecnologías de asistencia relacionadas con el ALR en entornos ruidosos y en el aprendizaje de idiomas.

Como parte de la arquitectura, utilizan CNN y LTSM. Para las CNN presentan dos tipos de redes: VGG19 y ResNet50, alcanzando una efectividad del 85% al implementar ResNet50.

2.1.4 Limitación de datos y propuestas alternativas

En [12] se aborda la limitación de las aplicaciones de ALR al idioma inglés debido a la disponibilidad de datos de entrenamiento a gran escala en otros idiomas. Otra característica importante en este contexto es la calidad de los datos enfocados en idiomas diferentes al inglés, ya que la limitante de que no son suficientemente grandes para cubrir un vocabulario amplio afecta la capacidad de generalización de los modelos. Los autores de este trabajo utilizan un conjunto de datos en español castellano de solo 3 horas de duración, destacando la dificultad de trabajar con datos significativamente reducidos en comparación con investigaciones en otros idiomas, que disponen de conjuntos de datos más extensos.

Además de utilizar arquitecturas clásicas en el campo como CNN en conjunto con RNN. Proponen añadir el uso de Visual Units, por sus siglas en inglés (VUs), en las tareas de clasificación. Las VUs se refieren a una colección de imágenes con características visuales similares pero que son diferentes en la lingüística.

También implementan y enfatizan en la relevancia de considerar el tanto el contexto del habla (fonemas, sílabas, palabras) como el contexto corto (secuencias de fonemas previos y consecuentes) para mejorar el rendimiento de los modelos.

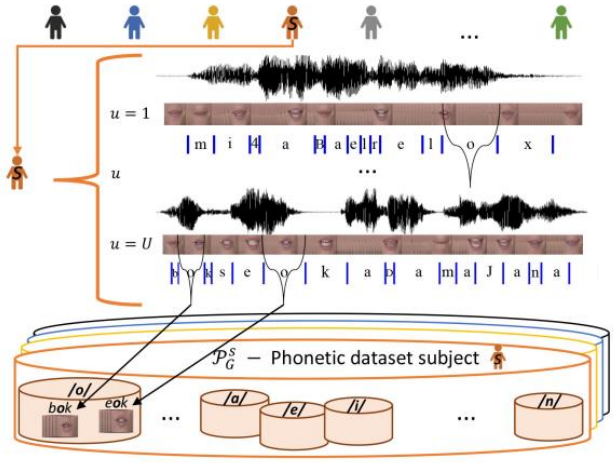

Figure 5 Ejemplo de distribución fonética por sujeto en el trabajo End-to-End Lip-Reading Without Large-Scale Data

La arquitectura del sistema propuesto consiste en dos módulos uno visual y otro temporal, donde el módulo temporal está conformado por un modelo basado en atención secuencia a secuencia utilizando LSTM. Finalmente, la precisión del modelo sobre el conjunto de datos VLRF alcanzó un Character Error Rate, por sus siglas en inglés (CER), del 44.77% y un Word Error Rate, por sus siglas en inglés (WER) del 72.90%.

2.1.5 Características visuales independientes del hablante

El artículo [9] se enfoca en las características visuales para el reconocimiento del habla, destacando la diferencia entre sistemas robustos que integran multimodalidad, es decir que combinan las características visuales y auditivas. En búsqueda de la generalización, los autores presentan dos enfoques: reconocimiento del habla dependiente del hablante y reconocimiento del habla independiente del hablante.

Introducen el concepto de visemas, que describen la pronunciación en términos de la interacción de elementos articulatorios como labios y lengua. Proponen la concatenación de visemas para la clasificación de texto en lugar del mapeo de visemas individuales. Además, mencionan factores que agregan complejidad a las tareas de Visual Speech Recognition por sus siglas en inglés (VSR) como la velocidad al hablar, el tono de la piel en inclusive la pronunciación.

Se incorpora al igual que en otras investigaciones el uso de LRLPs a partir del rostro. La arquitectura utilizada en este trabajo está dividida en dos módulos que trabajan en conjunto. El primero se encarga de extraer las características del video de entrada y el segundo se enfoca en el procesamiento de secuencias. El modelo resultante tiene una exactitud del 76.89% sobre el conjunto de datos MIRACL-VC1 [15] en inglés.

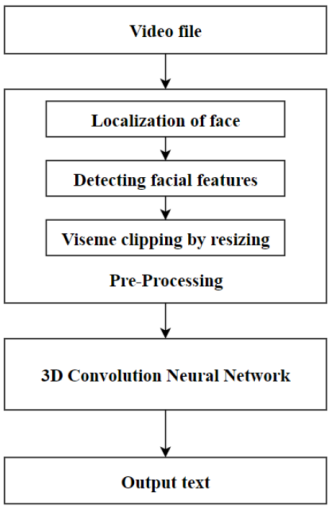

Figure 6 Diagrama de flujo de la arquitectura propuesta en el trabajo Speaker-Independent Speech Recognition using Visual Features

3. MARCO TEÓRICO

3.1 IA y conceptos relacionados

A pesar de no existir aún una definición formal, la IA se puede definir como un campo de estudio dentro de la informática que busca comprender y construir entidades inteligentes, es decir, sistemas que pueden percibir su entorno y llevar a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea. Según Russell y Norvig [16], la IA abarca desde el desarrollo de algoritmos y modelos que permiten a las máquinas aprender y razonar, hasta la creación de sistemas que pueden interactuar con el entorno de manera autónoma. La IA se puede enfocar desde diferentes perspectivas, como la simulación de la inteligencia humana, la creación de sistemas que piensen de manera lógica, o la construcción de agentes que actúen racionalmente.

3.1.1 Algunas perspectivas y definiciones de la AI.

Pensar humanamente: Se refiere a la capacidad de las computadoras para resolver problemas o tomar decisiones basándose en un razonamiento similar al humano.

Pensar racionalmente: Implica que las computadoras puedan razonar y percibir el entorno utilizando modelos computacionales que emulan procesos lógicos y coherentes.

Actuar humanamente: Se enfoca en la habilidad de las computadoras para ejecutar acciones o funciones que normalmente requieren un alto grado de inteligencia humana, imitando comportamientos y respuestas humanas.

Actuar racionalmente: Para un sistema, actuar racionalmente significa tener la capacidad de percibir el entorno, adaptarse a cambios, mejorar su desempeño y tomar acciones efectivas en situaciones de incertidumbre.

3.1.2 ML aplicado

El ML es una subdisciplina de la IA que se centra en la ciencia del desarrollo de algoritmos y modelos que permiten a las computadoras aprender de la experiencia en lugar de seguir instrucciones rígidas, encontrando patrones a partir de datos, y mejorando su rendimiento en tareas específicas sin haber sido explícitamente programadas para ello [17].

Dentro de ML existen diferentes tipos de aprendizaje, los principales se pueden categorizar de la siguiente manera.

Aprendizaje supervisado: Los algoritmos se entrenan utilizando un conjunto de datos etiquetados, es decir, para cada ejemplo de entrenamiento existe una entrada y una salida esperada.

Aprendizaje no supervisado: Para este enfoque, los algoritmos se entrenan con datos sin etiquetas, es decir, el sistema debe descubrir patrones y estructuras por sí mismo.

Aprendizaje semi-supervisado: En este tipo de aprendizaje se combinan datos etiquetados con una gran cantidad de datos no etiquetados durante el entrenamiento, se utiliza cuando el costo de etiquetar datos es alto y laborioso.

Aprendizaje por refuerzo: Un agente es capaz de aprender mediante el aprendizaje por refuerzo a través de la recompensa o castigo que va en función de los resultados obtenidos al realizar ciertas acciones. El objetivo del algoritmo en este tipo de aprendizaje es maximizar la recompensa.

3.1.3 Redes Neuronales Artificiales (ANN) Perceptrón Multicapa (MLP) y DL

Las ANNs son sistemas de computación inspirados en la estructura y funcionamiento del cerebro humano. Intentan imitar la manera de procesar la información de las redes neuronales naturales presentes en nuestros cerebros. Las ANNs replican el mecanismo de generación y transporte de señales de las neuronas biológicas, creando un sistema computacional capaz de realizar tareas complejas a través de la interconexión y el ajuste de nodos, denominados neuronas artificiales [18].

El perceptrón es la unidad básica de las ANNs y fue uno de los primeros modelos de aprendizaje automático desarrollados. Está formado por una neurona llamada Linear Threshold Unit, por sus siglas en inglés (LTU), en donde cada neurona se conecta a todas las entradas disponibles.

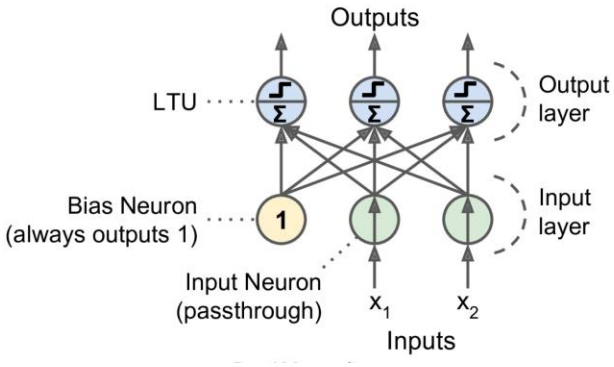

Figure 7 Diagrama de ejemplo de un perceptrón del libro Hands-On Machine Learning with Scikit-Learn and TensorFlow

El proceso de entrenamiento de un perceptrón implica recibir un dato de entrenamiento a cada momento, de tal manera que por cada neurona que produzca una predicción correcta, éste sea capaz de reforzar la conexión de las neuronas que generan una predicción adecuada [17].

Por otro lado, un MLP está compuesto por una capa de neuronas de entrada, una o más capas de LTUs como parte de las capas ocultas y finalmente otra capa de LTUs a la salida, el perceptrón multicapa se usa generalmente para tareas de clasificación de clases binarias [17].

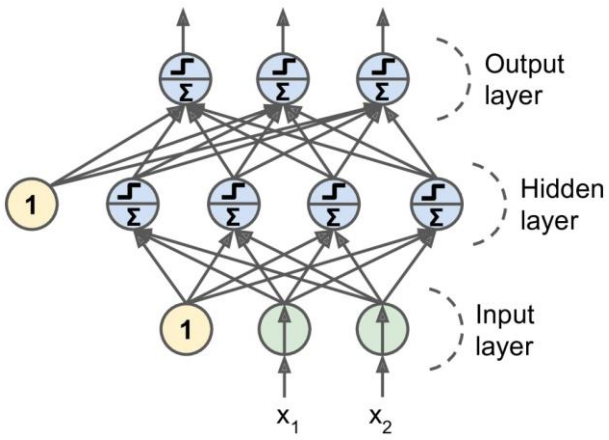

Figure 8 Diagrama de ejemplo de un MLP del libro Hands-On Machine Learning with Scikit-Learn and TensorFlow

El DL es una subdisciplina del ML, representa una evolución significativa en el campo del aprendizaje automático. Se enfoca en el uso de ANNs con múltiples capas para modelar y extraer representaciones complejas y abstractas de datos a gran escala. Tienen la capacidad de aprender jerarquías de características de los datos, donde la captura de características simples en las capas inferiores y la combinación de esas características en las capas superiores, permiten generar representaciones algorítmicas más complejas [19].

3.1.4 Capas de una red neuronal

Las redes neuronales están compuestas por una serie de capas que transforman la entrada de datos a través de una serie de pasos intermedios para finalmente producir una salida. Cada capa en la red neuronal realiza una transformación específica de los datos de entrada y pasa el resultado a la siguiente capa. La arquitectura de estas capas y sus interconexiones es crucial para el funcionamiento y la capacidad de la red para aprender y generalizar a partir de los datos [20].

Existen diferentes tipos de capas en una red neuronal. La capa de entrada es la primera capa de la red y su función principal es recibir los datos de entrada y distribuirlos a las capas posteriores. Las capas ocultas se sitúan entre la capa de entrada y la capa de salida; cada nodo en estas capas aplica una función de activación a una combinación de valores provenientes de la capa anterior, permitiendo la transformación y el aprendizaje de características complejas. Finalmente, la capa de salida es responsable de producir el resultado final de la red neuronal, que puede ser una predicción, una clasificación o cualquier otro tipo de resultado deseado.

3.2 Métricas de evaluación para modelos de clasificación

La evaluación de modelos de clasificación es fundamental para comprender su rendimiento y tomar decisiones informadas sobre su uso y mejora. Este apartado proporciona una visión detallada de las métricas esenciales utilizadas para evaluar modelos de clasificación.

3.2.1 Verdaderos positivos, verdaderos negativos, falsos positivos, falsos negativos y matriz de confusión

Para evaluar un modelo de clasificación, es necesario comprender los conceptos relacionados con el contraste que existe entre las predicciones positivas y negativas respecto al valor de salida esperado, en

este contexto existen verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos, a continuación, se expone la descripción de cada uno de esos conceptos.

Un Verdadero Positivo, por sus siglas en inglés (TP), ocurre cuando el modelo predice de manera positiva una instancia realmente positiva. Los Verdaderos Negativos, por sus siglas en inglés (TN), se consideran como las predicciones donde el modelo categoriza correctamente las clases negativas, es decir un verdadero positivo ocurre cuando una instancia es realmente negativa y el modelo la clasifica como tal. Un Falso Positivo, por sus siglas en inglés (FP), se produce cuando el modelo clasifica incorrectamente una instancia negativa como positiva. Los Falsos Negativos, por sus siglas en inglés (FN), suceden si el modelo predice incorrectamente una instancia positiva como negativa. Una matriz de confusión puede ser creada a partir de estos 4 conceptos [21].

Table 1 Matriz de confusión de predicciones positivas y negativas a partir del valor esperado de las clases reales

	Predicción Positiva	**Predicción Negativa**
Clase Positiva	TP	FN
Clase Negativa	FP	TN

3.2.2 Precisión y error

Es crucial definir el objetivo de un modelo en términos de las métricas a emplear. En este contexto, la precisión, en inglés accuracy, y el error son métricas esenciales para evaluar el rendimiento de un modelo de clasificación. Estas métricas permiten medir cuán bien un modelo predice en comparación con las etiquetas reales de los datos.

En términos sencillos, la precisión (1) nos indica la proporción de predicciones correctas sobre el total de predicciones realizadas, y se calcula mediante la siguiente fórmula:

$$\text{Precisión} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

La contraparte de la precisión es el error (2), que se puede definir como una medida complementaria debido a que mide la proporción de predicciones incorrectas realizadas por el modelo. Se calcula sumando los FPs y los falsos negativos FNs, y dividiendo el resultado por el número total de instancias.

$$\text{Error} = \frac{FP + FN}{TP + TN + FP + FN} \quad (2)$$

3.2.3 Métricas de sensibilidad-especificidad

En contextos donde los costos de los diferentes tipos de errores varían significativamente. La sensibilidad, también conocido como recall o por sus siglas en inglés True Positive Rate (TPR), mide la proporción de instancias correctamente identificadas como positivas por el modelo, esta métrica proporciona la capacidad para detectar casos positivos. Por otro lado, la especificidad, o True Negative Rate por sus siglas en inglés (TNR), calcula la proporción de instancias negativas correctamente identificadas, indicando la capacidad del modelo para excluir casos negativos [17].

3.2.4 Métricas de precisión-sensibilidad

Es habitual y efectivo combinar la precisión y la sensibilidad o recall en una única métrica denominada F1 score (3). La integración de estas dos métricas otorga mayor peso a los valores bajos, lo que significa que el F1 score solo será alto cuando tanto la precisión como el recall sean elevados. En otras palabras, esta métrica busca equilibrar y maximizar ambas dimensiones simultáneamente [17]. El F1 score esta dado por la siguiente formula.

$$\mathrm{F1} = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}} \quad (3)$$

$$\mathrm{F1} = \frac{2 \times precision \times recall}{precision + recall} \quad (4)$$

3.2.5 Intersección Sobre la Unión (IOU)

IOU es una métrica utilizada en el campo de la detección de objetos y visión computacional para medir la superposición entre dos recuadros compuestos por coordenadas Xmin, Ymin, Xmax y Ymax. Su objetivo es cuantificar el área común entre el recuadro predicho y el recuadro real. Específicamente, se emplea para evaluar la precisión de la predicción de un recuadro (Y) en un modelo en comparación con el recuadro de referencia (X). La idea fundamental detrás de esta métrica es lograr que el recuadro predicho Y se ajuste lo más posible al recuadro esperado X o en inglés ground truth [22].

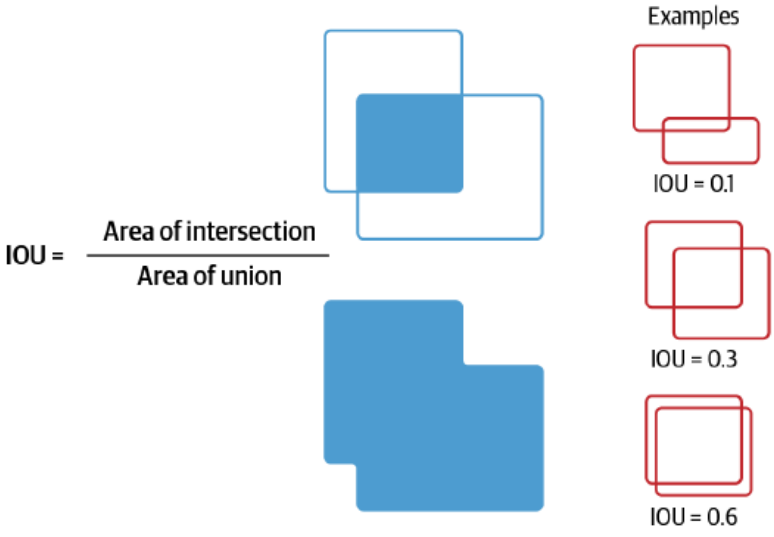

Figure 9 Fórmula para calcular el IOU del libro Practical Machine Learning for Computer Vision

3.3 Librerías principales de Python

Python es un lenguaje de programación de propósito general de cuarta generación, caracterizado por su gran semejanza al lenguaje humano. Una de sus características destacadas es su implementación multiparadigma, lo que le permite adaptarse a diferentes enfoques según el problema a resolver. Entre estos paradigmas se incluyen el imperativo, el procedural, el orientado a objetos y el funcional. La gramática de Python es distintiva debido a su sintaxis sencilla, clara y directa.

Python se ha convertido en un referente en el ámbito científico gracias a su capacidad para manejar grandes volúmenes de datos y su simplicidad en la programación, lo que lo hace cada vez más popular entre los usuarios. Gracias a estos factores positivos y a que su código es libre y gratuito, muchas organizaciones han desarrollado módulos y bibliotecas para la gestión de datos y diversas aplicaciones en la ciencia de datos y la IA [23]. A continuación, se describen algunas de esas bibliotecas y su uso.

3.3.1 NumPy

Numerical Python (NumPy) es una biblioteca universal de Python, diseñada para manejar datos numéricos y realizar operaciones eficientes. NumPy es utilizada actualmente en diversas áreas científicas como la ingeniería y la computación científica. Proporciona soporte para procesar matrices y arreglos multidimensionales (Ndarray) y también incluye una extensa colección de funciones matemáticas de alto nivel que permiten operar con diferentes tipos de datos. Además de ser una biblioteca ampliamente utilizada para el procesamiento directo de datos, también sirve como base para otras bibliotecas en Python [24].

3.3.2 Pandas

Pandas es una biblioteca de Python construida sobre NumPy, diseñada para la manipulación y análisis de datos. Ofrece estructuras de datos rápidas y flexibles, optimizadas para trabajar con datos relacionales. Dos de las estructuras más importantes y ampliamente utilizadas en Pandas son Series y DataFrame. Las Series facilitan el manejo de datos unidimensionales, mientras que los DataFrames permiten manipular datos bidimensionales, siendo ideales para aplicaciones estadísticas, de ingeniería y diversas áreas de la ciencia [25].

3.3.3 Scikit Learn

Scikit Learn es una libreria de código abierto diseñada para trabajar con ML. Proporciona herramientas simples y eficientes para el análisis predictivo de datos, incluyendo algoritmos de clasificación, regresión, agrupamiento, reducción de dimensionalidad y selección de modelos. Además, ofrece diversos algoritmos de ML llamados estimadores, en inglés estimators, que proveen la capacidad de ser utilizados con conjuntos de datos propios [26]. La implementación de Scikit Learn permite trabajar con bibliotecas como NumPy, y es ampliamente utilizada en el campo de la IA.

3.3.4 Tensorflow y Keras

Tensorflow es una biblioteca que proporciona un marco de referencia para el ML y desarrollo de DL. Desarrollada originalmente por el equipo de Google Brain, se utiliza ampliamente en investigación como en aplicaciones comerciales debido a su facilidad para construir y entrenar modelos de DL eficientes.

Además, Tensorflow facilita la ejecución de estos modelos en diversos dispositivos, como teléfonos, tabletas, sistemas distribuidos a gran escala y sistemas computacionales [27].

Keras es un marco de referencia que funciona como una Interfaz de Programación de Aplicaciones, por sus siglas en inglés (API), para DL. Trabaja con Tensorflow, Just After Execution, por sus siglas en inglés (JAX), un marco de referencia de Google para tareas de ML [28] y PyTorch, simplificando la construcción de modelos complejos gracias a su sintaxis intuitiva y amigable [29].

3.3.5 PyTorch

PyTorch es una biblioteca optimizada para trabajar con tensores para tareas de DL enfocada en el uso optimizado de hardware como Unidades Graficas de Procesamiento por sus siglas en inglés (GPUs) y Unidades Centrales de Procesamiento por sus siglas en inglés (CPUs) [30]. Gracias al manejo de tensores, PyTorch acelera las operaciones matemáticas. Además, provee una extensa lista de funciones para el entrenamiento distribuido, así como el manejo de procesos para eficientizar las tareas de DL [31].

4. DESARROLLO Y METODOLOGÍA

4.1 Conjunto de datos

Los datos utilizados en el estado del arte y los datos públicos son irrelevantes para la creación de un modelo de predicción automático con un enfoque geográfico diferente a los ya propuestos. Aunque existe diversidad de idiomas y acentos en todo el mundo, los conjuntos de datos públicos abarcan principalmente el inglés, chino [10], ruso [2] y árabe [7]. Por lo tanto, la reutilización de un conjunto ya existente es ineficiente para la investigación propuesta.

Actualmente, existe un conjunto de datos llamado VLRF, implementado y creado por los autores de [12], que contiene tres horas de video de personas hablantes del español castellano. Sin embargo, la ambigüedad en acentos, pronunciación y uso de palabras entre el español latino y el español castellano lo convierte en un conjunto de datos poco eficiente para el propósito de este trabajo de investigación. Esto genera la necesidad de la creación de un conjunto de datos completamente nuevo para alimentar un modelo enfocado en hablantes del español latino en México.

4.1.1 Descripción del conjunto de datos MEXLR2023

Como se describe anteriormente, dado que ningún conjunto de datos existente satisface completamente las necesidades del proyecto, es necesario crear uno propio. Este conjunto de datos debe contener videos de diversas personas pronunciando las cinco palabras objetivo que se presentan más adelante. Los videos requieren conversión a fotogramas e inclusión de una variedad de léxicos, con participantes de ambos sexos y diversas características visuales. Además, es necesario que la resolución de los videos sea superior a 256 por 256 píxeles para asegurar la captura de detalles y que sean filmados en un ambiente controlado.

La estrategia de recolección y preparación de datos para el conjunto MEXLR2023 consta de cuatro etapas, las cuales se describen a continuación.

4.1.1.1 Selección de las palabras de interés

Con el objetivo de generar una diversidad léxica en el modelo, se seleccionaron cinco palabras que presentan variedad en su pronunciación y número de sílabas. Las expresiones que fueron seleccionadas para generar el conjunto de datos MEXLR2023 son las siguientes:

1. Acción.
2. Fuerza.
3. Número.
4. Ciencia.
5. Música.

4.1.1.2 Búsqueda de participantes para la recolección de muestras y captura de video

Para la recolección del conjunto de datos de video, se implementó una estrategia que incluyó la colaboración de Víctor Daniel Pacheco Gómez, un profesional en cinematografía y estudiante de la carrera de cine con experiencia en la captura y dirección de videos.

Se seleccionaron cinco hablantes nativos de español residentes en Guadalajara, Jalisco, México, para participar en el proyecto. Los participantes seleccionados fueron Luis Enrique Pulido de Alba, Fátima Pacheco Gómez, Juan Jesús Rocha Arellano, Juan Martín Vidrio Rodríguez y Viviana Muñoz. Esta selección permitió enriquecer el conjunto de datos con una representación equilibrada de ambos sexos.

Figure 10 Participantes en la captura de videos para el conjunto de datos MEXLR2023

Cada participante fue posicionado a una distancia de un metro y medio de la cámara para pronunciar las cinco palabras objetivo. Se filmaron 20 videos por participante para cada palabra, lo que resultó en 100 videos por palabra y un conjunto de 500 videos en total. Cada video tuvo una duración aproximada de 2 segundos y se capturó en color RGB a una velocidad de 30 fotogramas por segundo, con un formato HD de 1280 por 720 píxeles.

4.1.1.3 Abstracción de los fotogramas de cada video

Además de la captura de los videos, se desarrolló una aplicación que descompone cada video en fotogramas. Dado que los videos tenían duraciones variables, la cantidad total de fotogramas variaba significativamente. Algunos videos alcanzaban los 60 fotogramas o más, mientras que otros tenían un mínimo de 45 fotogramas. Para estandarizar la cantidad de fotogramas por video, se implementó un proceso de normalización que ajusta cada video a secuencias de exactamente 50 imágenes. El algoritmo de esta aplicación consiste en añadir o eliminar fotogramas de manera equilibrada al inicio y al final de la secuencia hasta alcanzar los 50 fotogramas deseados.

Finalmente, esta aplicación generó una secuencia de 50 imágenes por palabra, facilitando así el uso del conjunto de datos para tareas de DL y ML.

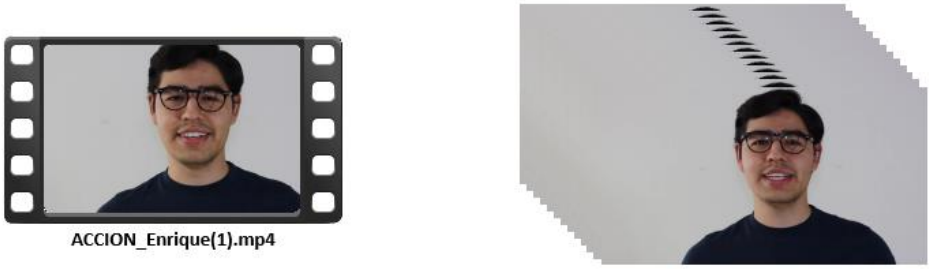

Figure 11 Descomposición de videos en secuencias de fotogramas para el conjunto de datos MEXLR2023

4.1.1.4 Obtención de las regiones de interés

Después de descomponer los videos en secuencias de fotogramas, se empleó la herramienta Labelimg [32] para llevar a cabo el proceso de etiquetado manual para cada imagen. En este proceso, se crearon carpetas individuales para cada secuencia de fotogramas y se asignaron Cuadros Delimitadores, por sus siglas en inglés (Bbox) a la región de la boca en cada fotograma. El formato utilizado en la creación de etiquetas fue Pascal Visual Object Classes, por sus siglas en inglés (PascalVOC) [33], debido a su representación en Extensible Markup Language, por sus siglas en inglés (XML). Este etiquetado detallado de las regiones de interés del rostro es fundamental para la generación de nuevos videos que servirán como entrada para un modelo recurrente LSTM.

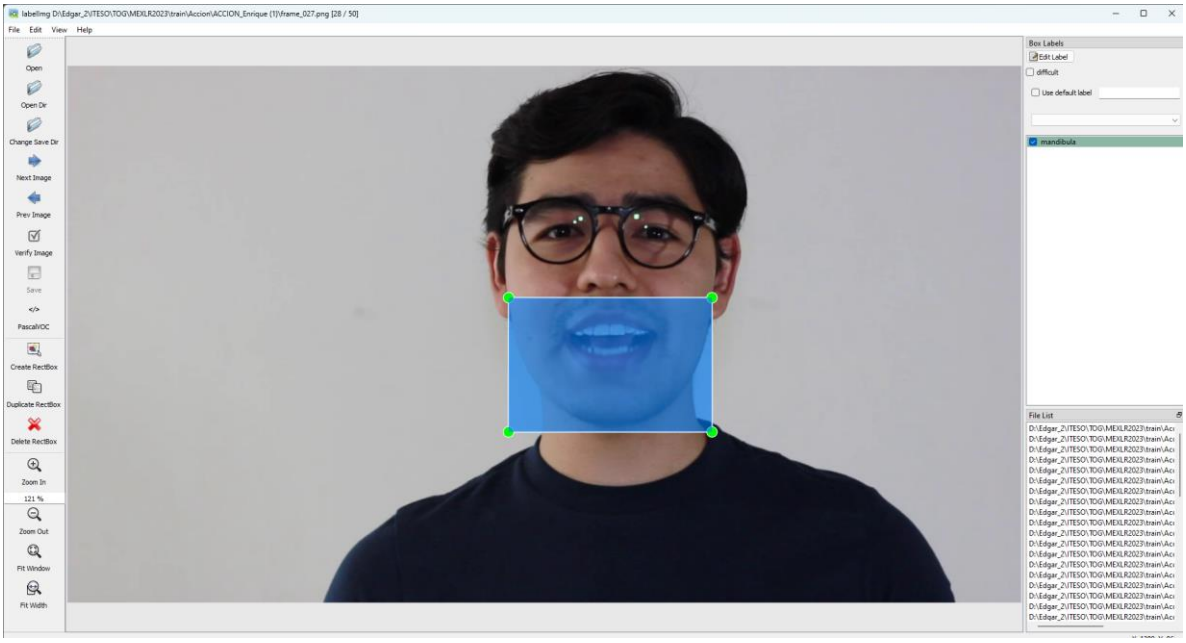

Figure 12 Etiquetado de ROI a partir de Bboxes en el conjunto de datos MEXLR2023 utilizando la herramienta Labelimg

4.1.1.5 Distribución y jerarquía de MEXLR2023

El conjunto de datos MEXLR2023 está estructurado en un esquema de carpetas específico diseñado para facilitar su uso y acceso. La organización de MEXLR2023 sigue la siguiente jerarquía dentro de las carpetas de entrenamiento, validación y prueba:

```
MEXLR2023/
└── Train/
    ├── Accion/
    │   ├── secuencia_1/
    │   ├── secuencia_2/
    │   └── ...
    ├── Ciencia/
    │   ├── secuencia_1/
    │   ├── secuencia_2/
    │   └── ...
    ├── Fuerza/
    │   ├── secuencia_1/
    │   ├── secuencia_2/
    │   └── ...
    ├── Musica/
    │   ├── secuencia_1/
    │   ├── secuencia_2/
    │   └── ...
    └── Numero/
        ├── secuencia_1/
        ├── secuencia_2/
        └── ...
```

Cada carpeta corresponde a una de las palabras objetivo y contiene subcarpetas que representan diferentes secuencias de fotogramas. Esta organización facilita la navegación y el acceso a las secuencias específicas de cada palabra.

El conjunto de datos está dividido en tres subconjuntos: entrenamiento (Train), validación (Validation) y prueba (Test), con la siguiente distribución porcentual:

- Train: 78.4%

- Validation: 19.6%
- Test: 2.0%

En total, el conjunto de datos consta de 500 videos. Esta distribución garantiza un equilibrio adecuado para el entrenamiento y la evaluación del modelo, asegurando suficientes datos para cada fase del proceso de desarrollo.

4.2 Definición del modelo ITESOWordNet

La arquitectura propuesta ITESOWordNet consta de dos etapas principales; la primera tiene como objetivo la localización y segmentación de las ROIs para cada fotograma, por otro lado, la segunda etapa se encarga de la extracción de características, el procesamiento secuencial de imágenes previamente segmentadas y la clasificación final de palabras.

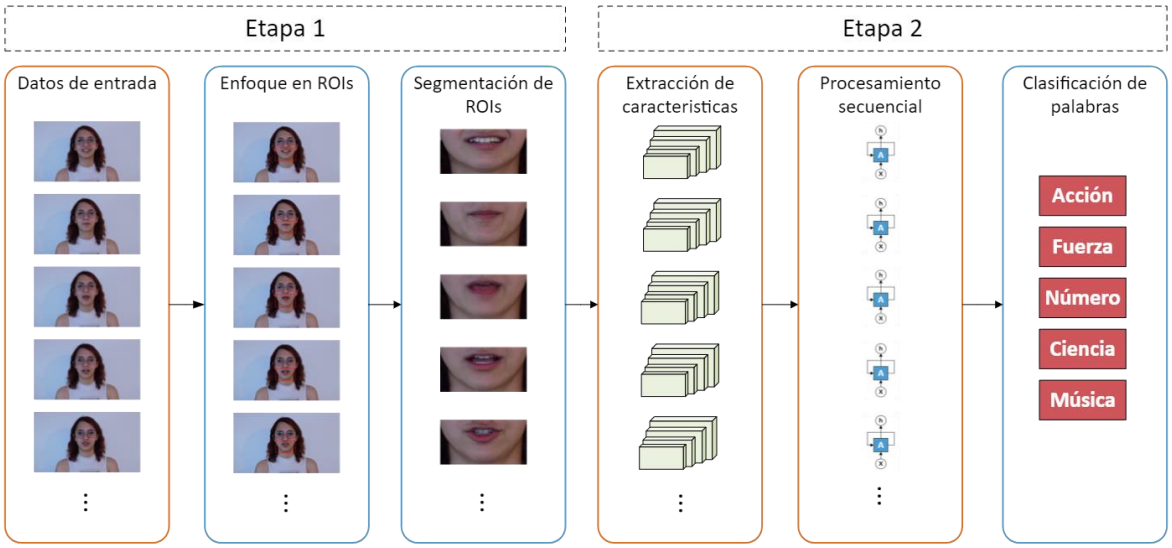

Figure 13 Arquitectura y etapas del modelo de clasificación de palabras ITESOWordNet

4.2.1 Etapa 1: Enfoque en las ROIs

Originalmente, se exploró la manera de hacer más efectivo el procesamiento de datos, definiendo la mejor estrategia para extraer la mayor cantidad de características de cada imagen. Se contempló la posibilidad de utilizar LRLPs para filtrar las características más relevantes para tareas de ALR. Sin embargo, se planteó la hipótesis de que existen detalles que aportan datos significativos en el procesamiento del conjunto de píxeles que forman una imagen. Esos detalles podrían estar contenidos en la variación de tonalidad y sombras en el rostro al verbalizar. Por lo tanto, se decidió que limitarse a los LRLPs incrementaría la posibilidad de perder información relevante para el modelo, contenida a nivel píxel. En consecuencia, se decidió enfocarse en el procesamiento a partir de imágenes completas a color, marcando un contraste con las metodologías que utilizan puntos que contemplan únicamente la posición de la boca y las áreas circundantes a ella.

Para lograr un procesamiento adecuado de las imágenes y los Bboxes, se crearon funciones que se encargan de preprocesar y redimensionar los datos para ajustarlos al tamaño requerido por el modelo.

```python
def get_normalized_img_bbox(img_path, xmin=None, ymin=None, xmax=None, ymax=None, width=320, height=200):
    """
    Normaliza a un ancho y alto deseado, una imagen dada junto con su bbox.

    xmin(int): xmin En caso de ausencia, solamente se devuelve la imagen reescalada.
    ymin(int): ymin.
    xmax(int): xmax.
    ymax(int): ymax

    Regresa:
      [img, Xmin, Ymin, Xmax, Ymax]
    """

    #Definimos el nuevo tamaño deseado para las imágenes
    new_width = width
    new_height = height

    #Leemos la imagen
    img = cv2.imread(img_path)

    #Ajustamos color
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    #Redimensionamos la imagen al nuevo tamaño deseado
    img_new = cv2.resize(img, (new_width, new_height))

    if xmin is not None:
      #Calculamos la proporción de cambio de tamaño (escala) para ajustar las coordenadas
      scale_x = new_width / img.shape[1]
      scale_y = new_height / img.shape[0]

      #Ajustamos las coordenadas del bounding box
      Xmin = int(xmin * scale_x)
      Ymin = int(ymin * scale_y)
      Xmax = int(xmax * scale_x)
      Ymax = int(ymax * scale_y)

      if Xmin >= Xmax:
        raise Exception("Xmin igual o mayor a Xmax en {}".format(img_path))

      if Ymin >= Ymax:
        raise Exception("Ymin igual o mayor a Ymax en {}".format(img_path))

      return [img_new, Xmin, Ymin, Xmax, Ymax]

    return img_new
```

Figure 14 Código de la función get_normalized_img_bbox para la etapa 1 de ITESOWordNet

La función anterior `get_normalized_img_bbox` recibe la imagen y coordenadas del Bbox correspondiente. Posteriormente, se encarga de redimensionar tanto la imagen como las coordenadas del Bbox, dado un ancho y alto deseados, y devuelve la imagen resultante.

```python
class CustomDataset(Dataset):
    """
    Esta clase recibe un directorio raiz de donde itera el arbol de clases para obtener videos de cada
clase dividido en fotogramas.
    Posteriormente genera un dataset que contiene a cada uno de los frames de todos los videos en el
directorio y lo divide en batches.
    """
    def __init__(self, root_dir, bbox=True):
        self.root_dir = root_dir
        self.bbox = bbox
        self.classes = sorted(os.listdir(root_dir))
        self.video_folders = []
        for cls in self.classes:
            class_folder = os.path.join(self.root_dir, cls)
            self.video_folders.extend([os.path.join(class_folder, video_folder) for video_folder in
sorted(os.listdir(class_folder))])

        self.total_images = sum(len([f for f in os.listdir(video_folder) if f.endswith('.png')]) for
video_folder in self.video_folders)

    def __len__(self):
        return self.total_images

    def __getitem__(self, idx):
        video_folder_idx = 0
        while idx >= len([f for f in os.listdir(self.video_folders[video_folder_idx]) if
f.endswith('.png')]):
            idx -= len([f for f in os.listdir(self.video_folders[video_folder_idx]) if
f.endswith('.png')])
            video_folder_idx += 1

        video_folder = self.video_folders[video_folder_idx]

        image_files = sorted([f for f in os.listdir(video_folder) if f.endswith('.png')])
        xml_files = sorted([f for f in os.listdir(video_folder) if f.endswith('.xml')])

        # Obtenemos el index de la imagen dentro del folder de video
        image_idx = idx % len(image_files)

        # Obtenemos el index del XML dentro del folder de video
        xml_idx = idx % len(xml_files)

        # Leemos el XML para obtener las coordenadas de la boca
        tree = ET.parse(os.path.join(video_folder, xml_files[xml_idx]))
        root = tree.getroot()

        # Encontramos el objeto
        xml_obj = root.find('object')

        # Obtenemos los valores de xmin, ymin, xmax, ymax
        xmin = int(xml_obj.find('bndbox/xmin').text)
        ymin = int(xml_obj.find('bndbox/ymin').text)
        xmax = int(xml_obj.find('bndbox/xmax').text)
        ymax = int(xml_obj.find('bndbox/ymax').text)

        # Leemos la imagen
        image_path = os.path.join(video_folder, image_files[image_idx])
        # Normalizamos tamaño de imagen y bounding boxes
        if self.bbox:
          img, Xmin, Ymin, Xmax, Ymax = get_normalized_img_bbox(image_path, xmin, ymin, xmax, ymax)
        else:
          img = get_normalized_img_bbox(image_path)

        img_tensor = torch.tensor(img, dtype=torch.float32) / 255.0
        img_tensor = img_tensor.permute(2, 0, 1)

        # bboxes y labels
        if self.bbox:
          bbox_list = [Xmin, Ymin, Xmax, Ymax]
          bbox_tensor = torch.tensor([bbox_list], dtype=torch.float64)
          labels_tensor = torch.ones((bbox_tensor.shape[0]), dtype = torch.int64)

          target = {}
          target['boxes'] = bbox_tensor
          target['label'] = labels_tensor

          # Regresamos la imagen en tensor y target con xmin, ymin, xmax, ymax
          return img_tensor, target
        else:
          # Regresamos la imagen en tensor
          return img_tensor
```

Figure 15 Código de la clase CustomDataset para la etapa 1 de ITESOWordNet

También se desarrolló una clase personalizada `CustomDataset`, que itera sobre el árbol de directorios de clases, obteniendo cada video dividido en fotogramas. Esta clase recibe un directorio raíz y procesa los fotogramas de cada video por cada clase. Además, utiliza la función `get_normalized_img_bbox` para normalizar las imágenes. Devuelve una imagen junto a su Bbox a manera de tensor para alimentar una red neuronal basada en el marco de referencia PyTorch.

```
def get_data_loader(data_path, batch_size=20, pin_memory=True, bbox=True):
  """
  Esta funcion crea un data loader a partir de la clase CustomDataset, con la cantidad de elementos por
batch deseados
  """
  #Creamos conjunto de datos custom a partir de clase CustomDataset
  custom_dataset = CustomDataset(data_path, bbox)
  #definimos el tamaño del batch
  BATCH_SIZE = batch_size

  #Creamos el data_loader
  data_loader = DataLoader(
      custom_dataset,
      batch_size=BATCH_SIZE,
      shuffle=True,
      num_workers=1,
      pin_memory=pin_memory
  )

  return data_loader
```

Figure 16 Código de la función get_data_loader para la etapa 1 de ITESOWordNet

Finalmente, la función `get_data_loader` genera un cargador de datos a partir de la clase `CustomDataset`, lo que permite dividir los fotogramas en lotes y producir un cargador independiente para los conjuntos de entrenamiento, validación y prueba.

Partiendo de los Bboxes y etiquetas transformadas a tensores, se creó un modelo con el objetivo de localizar las ROIs en cada imagen. Este modelo está basado en Faster R-CNN ResNet50 FPN [34]. Para lograr un manejo adecuado de los datos, se adecuaron las entrada y salidas según el conjunto de datos MEXLR2023. Además, se ajustaron los pesos de la red utilizando aprendizaje transferido, teniendo en cuenta las dimensiones de los tensores de entrada y salida.

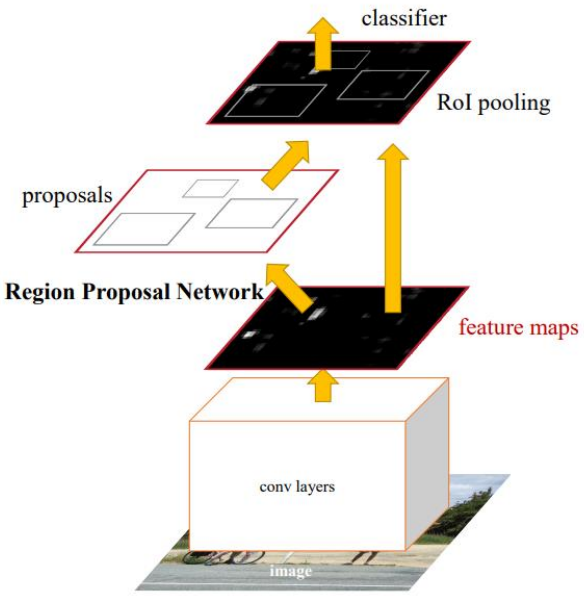

Figure 17 Representación gráfica del modelo Faster R-CNN del trabajo Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

La etapa 1 del modelo ITESOWordNet finaliza con el retorno de los nuevos tensores calculados. Estos tensores resultantes corresponden a las proposiciones de la región segmentada de la boca y sus alrededores. De esta manera, somos capaces de obtener las ROIs de cada uno de los fotogramas presentes en cada uno de los videos.

4.2.2 Etapa 2: Extracción de características de las ROIs, procesamiento secuencial y clasificación final

El modelo propuesto para la etapa 2 consta de un procesamiento modular dividido en dos apartados. El primer módulo se encarga de la extracción de características de cada una de las imágenes de entrada. Este apartado se lleva a cabo empleando un modelo preentrenado VGG16. Así mismo, toma la salida procesada por el modelo en la etapa 1 y alimenta al segundo módulo de la etapa 2 con vectores de características.

Posteriormente, el segundo módulo trata las secuencias de imágenes empleando RNNs del tipo LSTM. Finalmente, se clasifica la secuencia utilizando una capa totalmente conectada cuya salida llega a cinco neuronas, cada una correspondiente a una clase.

La implementación del modelo para la etapa 2 comienza con la definición de una función que utiliza el modelo de la etapa 1 para obtener las ROIs de cada imagen.

```python
def predict_mouth(image_path, model, width=64, height=64, show_img=False):
    """
    Esta función carga una imagen, la procesa y muestra el cuadro delimitador.
    """

    # Cargar la imagen y preprocesarla
    img = cv2.imread(image_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Realizar la predicción
    pred_box = predict_image(model, img_tensor)

    # Dibujar el cuadro delimitador en la imagen
    xmin, ymin, xmax, ymax = map(int, pred_box)

    # Recortar la región de interés (ROI)
    img = img[ymin:ymax, xmin:xmax]

    # Redimensionar la imagen recortada
    img = cv2.resize(img, (width, height))

    if show_img:
        # Mostramos la imagen resultante
        plt.imshow(img)
        plt.axis('off')  # Desactivar los ejes
        plt.show()

    # Regresamos un ndarray de la imagen resultante
    return img
```

Figure 18 Código de la función predict_mouth para la etapa 2, módulo 1 de ITESOWordNet

La función `predict_mouth` recibe la imagen a procesar, el modelo, y las dimensiones de ancho y largo deseadas. Luego, realiza la predicción de la ROI de cada entrada utilizando el modelo de la etapa 1. Por defecto, la salida de la función tiene un tamaño de 64 por 64 pixeles, resultando en una imagen cuadrada con 3 canales de color en forma de Ndarray.

```python
def reduce_and_crop_video(video_path, mouth_model, effective_frames, from_xml=False):
    """
    This function will return a list of np.arrays, each member of the list
    will be one frame of the video
    """
    frames = list()

    files = os.listdir(video_path)

    # Obtenemos todos los archivos del folder de video
    image_files = sorted([file for file in files if file.endswith('.png')])

    # Calculamos el salto entre fotogramas para reducir el número de fotogramas
    step = len(image_files) // effective_frames

    # Iteramos sobre los archivos de imagen y seleccionamos los fotogramas deseados
    for i, filename in enumerate(image_files):
        # Si el fotograma actual debe ser incluido
        if i % step == 0:
            image_path = os.path.join(video_path, filename)

            if from_xml:
                cropped_image = get_mouth_from_xml(image_path)
            else:
                cropped_image = predict_mouth(image_path, mouth_model) # image = ndarray

            frames.append(cropped_image)#Modificar

    return frames
```

Figure 19 Código de la función reduce_and_crop_video para la etapa 2, módulo 2 de ITESOWordNet

A un nivel de abstracción superior, se implementó la función `reduce_and_crop_video` como parte del módulo 2, que se encarga de procesar cada carpeta que contiene una secuencia de imágenes correspondientes a un video. Esta función recibe como entrada el video, el modelo que detecta el área de la boca y la cantidad de fotogramas efectivos.

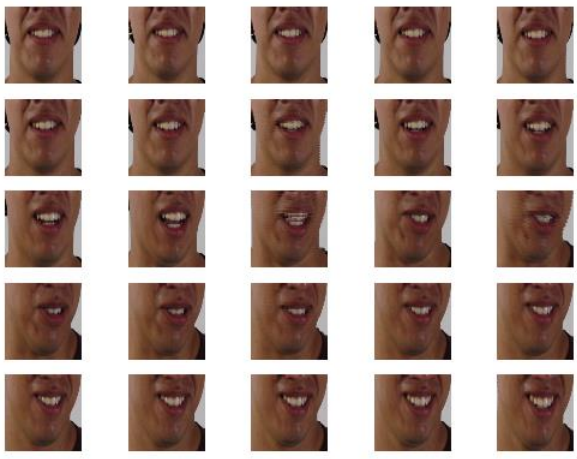

Figure 20 Ejemplo del efecto de la función reduce_and_crop_video sobre una muestra de video

Se decidió utilizar 25 de las 50 imágenes totales por video. Esto se basa en la premisa de que es posible obtener resultados favorables al considerar solamente una cantidad de significativa de fotogramas para reducir la dimensionalidad de los datos durante el procesamiento de secuencias. De esta manera, la cantidad de fotogramas efectivos se refiere al número de imágenes a considerar durante la creación de lotes a partir del total contenido en un video.

```python
def get_reduced_videos(dataset_path, mouth_model, effective_frames, from_xml=False):
    videos = list()
    labels = list()
    label_map = {}
    label_num = 0

    # Iteramos sobre cada una de las clases en el dataset
    for cl in sorted(os.listdir(dataset_path)):
        cl_path = os.path.join(dataset_path, cl) #Class path

        if cl not in label_map:
            label_map[cl] = label_num
            label_num += 1

        # Iteramos sobre cada archivo dentro del path de cada clase
        for video_file in os.listdir(cl_path):
            video_path = os.path.join(cl_path, video_file)

            # Cargamos y tomamos los frames deseados de cada video
            # ya recortado con el modelo que detecta la boca o con el xml
            reduced_video = reduce_and_crop_video(video_path, mouth_model, effective_frames, from_xml)

            # Agregamos videos y eriquetas a la lista
            videos.append(reduced_video)
            labels.append(label_map[cl])

    return np.array(videos), np.array(labels), label_map
```

Figure 21 Código de la función get_reduced_videos para la etapa 2, módulo 2 de ITESOWordNet

La función `get_reduced_videos` se encarga de iterar sobre el conjunto de datos recibido para aplicar la función `reduce_and_crop_video` a cada uno de los videos. Recibe un conjunto de datos, el modelo encargado de localizar la ROI y la cantidad de fotogramas efectivos. Devuelve dos Ndarrays, uno correspondiente a los videos y otro a las etiquetas, además del mapa de etiquetas.

```python
class CustomDataset(Dataset):
    def __init__(self, dataset_path, mouth_model=None, effective_frames=25):
        """
        Esta clase crea un dataset de una secuencia de imagenes a partir
        de un modelo que detecta la boca

        Args:
            dataset_path(str): El path donde se tienen los datos ej. train path
            mouth_model(model): Modelo que detecta la boca
            effective_frames(int): Numero de frames a considerar por video
        """
        data_splitted = get_reduced_videos(dataset_path, mouth_model, effective_frames)
        self.videos = data_splitted[0]
        self.labels = data_splitted[1]
        self.label_map = data_splitted[2]
        self.transform = transforms.Compose([
          transforms.ToTensor(),# Convertimos imágenes de Numpy arrays a tensores de PyTorch
          transforms.Normalize(mean=[0.5], std=[0.5])])  # Normalizamos los tensores

    def __len__(self):
        return len(self.videos)

    def __getitem__(self, idx):
        video = self.videos[idx]
        label = self.labels[idx]

        # Convertimos el video a lista de frames, aplicar transformación y volver a apilar
        video = [self.transform(frame) for frame in video]
        video = torch.stack(video)

        # label es un solo valor, no necesita transformación
        return video, label
```

Figure 22 Código de la clase CustomDataset para la etapa 2, módulo 2 de ITESOWordNet

Al igual que para la etapa 1, en la 2 se desarrolló una clase personalizada `CustomDataset` que itera sobre el conjunto de datos. Utiliza la función de `get_reduced_videos` para procesar todos los videos contenidos en el conjunto. Esta clase recibe un directorio raíz y procesa la secuencia de fotogramas de cada video por clase. Devuelve una lista de tensores correspondientes a un video, y la etiqueta del video.

```python
def get_data_loader(data_path, batch_size=10, pin_memory=True):
  # Creamos conjunto de datos custom a partir de clase CustomDataset
  custom_dataset = CustomDataset(data_path)
  # definimos el tamaño del batch
  BATCH_SIZE = batch_size

  # Creamos el data_loader
  data_loader = DataLoader(
      custom_dataset,
      batch_size=BATCH_SIZE,
      shuffle=True,
      num_workers=1,
      pin_memory=pin_memory
  )

  return data_loader, custom_dataset.label_map
```

Figure 23 Código de la función get_data_loader para la etapa 2, módulo 2 de ITESOWordNet

Como parte final de la preparación de los datos para la fase de entrenamiento, la función `get_data_loader` genera un cargador de datos a partir de la clase `CustomDataset`. Esta función permite dividir los tensores correspondientes a los videos y sus respectivas etiquetas en lotes. De esta manera, somos capaces de producir un cargador de datos independiente para los conjuntos de entrenamiento, validación y prueba.

En este punto, ya están definidas las etapas de preprocesamiento de los videos para la segmentación de las ROIs de cada uno de los videos. También se ha completado la conversión de las secuencias de imágenes a tensores junto con sus etiquetas.

La siguiente etapa consiste en la definición del modelo que será entrenado posteriormente a partir de los tensores resultantes. Como se describió al inicio de la etapa 2, el modelo encargado del procesamiento de las secuencias de información está conformado por VGG16 y LSTM. Finalmente, este modelo se encarga de clasificar cada video con la palabra correspondiente a la predicción, véase Figure 24 Diagrama a bloques de la arquitectura propuesta para la etapa 2 de ITESOWordNet

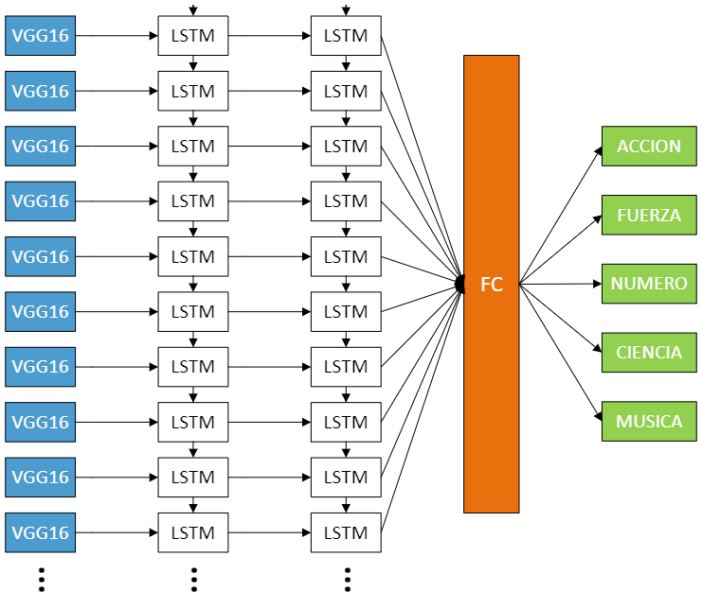

Figure 24 Diagrama a bloques de la arquitectura propuesta para la etapa 2 de ITESOWordNet

4.3 Selección de función de pérdida y optimización

En el diseño y entrenamiento de modelos de DL, la elección de la función de pérdida y el algoritmo de optimización es crucial para asegurar el desempeño óptimo. La manera en que somos capaces de medir el error existente entre las salidas predichas por el modelo y las esperadas se calcula a partir de la función de pérdida, también conocida como función de costo. En este sentido, a menor valor de función de pérdida, menor es el error en las predicciones [31].

Por otro lado, el algoritmo de optimización se encarga de la tarea del ajuste de pesos en el modelo. Al igual que en la teoría de control, en tareas de DL la optimización tiene como objetivo obtener el mejor resultado en el ajuste de un algoritmo [19]. De esta manera, busca minimizar la función de pérdida de manera eficiente.

Para ITESOWordNet se ha seleccionado la función de pérdida de entropía cruzada al tratarse de una tarea de clasificación. La entropía cruzada (4) es adecuada para problemas de clasificación multiclase, ya que mide la discrepancia entre la distribución de probabilidad predicha por el modelo y la distribución real de las clases.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\ \log(p_{i,j}) \qquad (5)$$

En la ecuación (4), N es el número de ejemplos, C el número de clases, $y_{i,j}$ son las etiquetas reales de la clase j para la muestra i y $p_{i,j}$ es la probabilidad predicha de la clase j para la muestra i [35].

Se eligió el optimizador Adaptative Moments, por sus siglas en inglés (Adam), que calcula dinámicamente tasas de aprendizaje adaptativas para cada parámetro del modelo, basándose en los primeros y segundos

momentos del gradiente [20]. A partir del optimizador Adam, se actualizan los parámetros del modelo buscando la manera de obtener un rendimiento óptimo en la tarea de clasificación de palabras en base a la tasa de aprendizaje seleccionada, que para ITESOWordNet fue de 0.001.

4.4 Entrenamiento

La etapa de entrenamiento es conocida también como la fase de aprendizaje. En este punto, la forma exacta del algoritmo de predicción es determinada a partir del conjunto de datos seleccionado para este propósito [36]. La ejecución del entrenamiento se lleva a cabo utilizando el conjunto de datos dividido en lotes. Estos lotes alimentan el modelo a través de varias épocas. Durante cada época, el modelo ajusta sus parámetros utilizando un optimizador en función de la pérdida calculada.

Para cada iteración en el ciclo de entrenamiento, se realiza una serie de pasos listados en Table 2 Listado de pasos en un ciclo de entrenamiento

Table 2 Listado de pasos en un ciclo de entrenamiento

Propagación hacia adelante
Cálculo de la pérdida
Propagación hacia atrás
Actualización de los parámetros

4.4.1 Entrenamiento del modelo ITESOWordNet

Para entrenar el modelo ITESOWordNet, se implementó la función `train_model`. Esta función recibe el cargador de datos para entrenamiento y validación, la función de pérdida a utilizar, el optimizador y el número de épocas.

```python
def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs=5,
model_path=model_path):
    best_val_loss = float('inf')  #Aqui guardaremos el mejor loss hasta el momento
    train_acc = 0
    val_acc = 0

    # Ploteamos barra de progreso
    pbar = trange(0, num_epochs, leave=False, desc='Epoch')

    # Entrenamiento del modelo
    for epoch in pbar:
        epoch_time_start = time.time() # Inicializamos contador tiempo por epoca
        pbar.set_postfix_str('Accuracy:  Train {:.2f}%, Val {:.2f}%'.format(train_acc, val_acc))

        current_epoch_loss = 0.0 # Aqui acumularemos la perdida de train por epoca
        correct = 0
        total = 0

        model.train()

        for batch_inputs, batch_labels in tqdm(train_loader, desc='Training', leave=False):
            batch_inputs = batch_inputs.to(device) # Movemos tensores a GPU
            batch_labels = batch_labels.to(device) # Movemos tensores a GPU
            batch_labels = batch_labels.long()

            optimizer.zero_grad() # Limpiamos gradientes

            outputs = model(batch_inputs) # Pasamos el batch de videos por el modelo y obtenemos sus
salidas
            _, predicted = torch.max(outputs, 1) # Obtenemos las clases predecidas

            total += batch_labels.size(0)
            correct += (predicted == batch_labels).sum().item() # Obtenemos las predicciones correctas

            loss = criterion(outputs, batch_labels) # Calculamos la perdida
            loss.backward() # Calculamos gradientes
            optimizer.step() # Actualizamos parametros del modelo

            current_epoch_loss += loss.item() * batch_inputs.size(0)

        # Calculamos el accuracy del entrenamiento
        train_acc = 100 * correct/total
        train_accuracies.append(train_acc)

        epoch_loss = current_epoch_loss/len(train_loader.dataset)
        train_losses.append(epoch_loss)

        print("Epoch {}, TRAIN Loss: {}, TRAIN Accuracy: {}".format(epoch+1, epoch_loss, train_acc))


        # Validacion del modelo
        model.eval()

        current_epoch_loss = 0.0 # Aqui acumularemos la perdida de val por epoca
        correct = 0
        total = 0
```

Figure 25 Código de la función train_model parte 1 (Entrenamiento)

Por cada época, la función se encarga de mover los tensores de datos al hardware disponible. En el caso del entrenamiento de ITESOWordNet, se utilizaron GPUs del tipo V100 disponibles en Google Colab. Posteriormente, por cada uno de los lotes, se calcula la pérdida del modelo. Una vez que se tiene la pérdida, el optimizador se encarga de actualizar los parámetros del modelo.

```
        with torch.inference_mode():
            for batch_inputs, batch_labels in tqdm(val_loader, desc='Validation', leave=False):
                batch_inputs = batch_inputs.to(device) # Movemos tensores a GPU
                batch_labels = batch_labels.to(device) # Movemos tensores a GPU
                batch_labels = batch_labels.long()

                outputs = model(batch_inputs) # Pasamos el batch de videos por el modelo y obtenemos sus
salidas
                _, predicted = torch.max(outputs, 1) # Obtenemos las clases predecidas

                total += batch_labels.size(0)
                correct += (predicted == batch_labels).sum().item() # Obtenemos las predicciones
correctas

                loss = criterion(outputs, batch_labels) # Calculamos la perdida

                current_epoch_loss += loss.item() * batch_inputs.size(0)

        # Calculamos el accuracy de la validacion
        val_acc = 100 * correct/total
        val_accuracies.append(val_acc)

        epoch_loss = current_epoch_loss/len(val_loader.dataset)
        val_losses.append(epoch_loss)

        print("Epoch {}, VAL Loss: {}, VAL Accuracy: {}".format(epoch+1, epoch_loss, val_acc))

        # Guardamos el modelo solo si la pérdida de validación actual es la mejor hasta ahora
        if epoch_loss < best_val_loss:
            best_val_loss = epoch_loss

            torch.save({
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                'loss': epoch_loss}
                , model_path)
            print("GUARDANDO MODELO...")
            print("Mejor modelo guardado con pérdida de validación {} en {}".format(epoch_loss,
model_path))

            # Cargamos el modelo desde el punto de control al comienzo de la siguiente época
            checkpoint = torch.load(model_path, map_location=device)
            model.load_state_dict(checkpoint['model_state_dict'])
            optimizer.load_state_dict(checkpoint['optimizer_state_dict'])

        # Liberamos la memoria del GPU
        torch.cuda.empty_cache()

        epoch_time_end = time.time() # Finalizamos contador tiempo por epoca
        delta_time_epoch = epoch_time_end-epoch_time_start
        print("End Epoch {}, Elapsed time: {} Minutos".format(epoch+1, round(delta_time_epoch/60, 2)))

    print("Entrenamiento Completado")
```

Figure 26 Código de la función train_model parte 2 (Validación)

Finalmente, para la validación que se realiza por cada época, se configura el modelo en modo inferencia e iteramos sobre los lotes del conjunto de validación. El conjunto de validación, en este punto, es desconocido para el modelo, por lo que nos permite ejercitar la capacidad de generalizar sobre datos desconocidos.

4.5 Métricas de evaluación seleccionadas

La evaluación del desempeño de un modelo de DL no puede basarse únicamente en la pérdida durante la fase de entrenamiento y validación. Es crucial utilizar métricas de evaluación que proporcionen una visión más completa del rendimiento del modelo en la fase de prueba. Para el modelo ITESOWordNet, se seleccionaron las métricas de evaluación descritas a continuación.

Exactitud: Para calcular la proporción de predicciones correctas entre el total de predicciones realizadas.

Sensibilidad: La sensibilidad mide la proporción de verdaderos positivos que fueron correctamente identificados por el modelo [22].

F1-Score: Esta medida combina precisión y sensibilidad para obtener un valor más objetivo. Se utiliza principalmente en problemas donde el conjunto de datos está desbalanceado [37]. Aunque este no es el caso para este modelo, sin embargo, es un punto de datos interesante a analizar.

Matriz de Confusión: Con el objetivo de representar de manera gráfica y detallada el rendimiento del modelo al mostrar el número de predicciones correctas e incorrectas por cada clase.

5. RESULTADOS Y DISCUSIóN

Los resultados del entrenamiento y validación del modelo a través de 30 épocas se reflejan a continuación en las gráficas de pérdida y exactitud.

Figure 27 Gráficas de pérdida y exactitud del modelo ITESOWordNet durante la fase de entrenamiento a través de 30 épocas

En la gráfica de pérdida podemos observar que tanto la pérdida de entrenamiento como la de validación presentan una tendencia decreciente. En la validación se muestran algunas fluctuaciones que incrementan un poco más a partir de la época 10. Esas fluctuaciones podrían deberse a la variabilidad en el conjunto de datos o al ajuste de los parámetros. Sin embargo, en general, podemos apreciar que la pérdida de validación sigue de cerca a la de entrenamiento, lo cual es un buen indicador de que el modelo tiene la capacidad de generalizar de manera adecuada.

La gráfica de exactitud muestra una tendencia creciente y constante, alcanzando casi el 100% para el entrenamiento. Al final del entrenamiento, se puede observar que la validación se mantiene en torno al 90%. También existen algunas fluctuaciones en la exactitud de la prueba y validación; sin embargo, la tendencia que marcan y la cercanía con la que se mueven, indican también una alta capacidad de generalización.

En términos de sobreajuste, la pérdida de validación no muestra una tendencia de aumento significativo, y tanto el comportamiento del entrenamiento como el de validación parecen seguir un patrón muy similar. Por lo anterior, no existe un indicio de sobreajuste en el comportamiento del modelo a través de las épocas.

La prueba del modelo se realizó sobre el conjunto de datos de prueba compuesto por 10 videos. Estos están balanceados, con dos correspondientes a cada una de las clases. A continuación, se presentan los resultados obtenidos en términos de las diferentes métricas seleccionadas.

5.1 Exactitud, Precisión y Sensibilidad del modelo

5.1.1 Exactitud del modelo ITESOWordNet

$$Exactitud = \frac{\sum TP}{\sum Total}$$

TP = 9
Total = 10 (6)

$$Exactitud = \frac{9}{10} = 0.9$$

5.1.2 Precisión del modelo ITESOWordNet

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

TP = 9
TN = 0
FP = 1
FN = 0 (7)

$$\text{Precisión} = \frac{9 + 0}{9 + 0 + 1 + 0} = \frac{9}{10} = 0.9$$

5.1.3 Sensibilidad del modelo ITESOWordNet

$$Sensibilidad = \frac{TP}{TP + FN}$$

TP = 9
FN = 0 (8)

$$Sensibilidad = \frac{9}{9 + 0} = 1$$

Como se puede apreciar en los cálculos (5), (6), (7) el modelo logra alcanzar una alta exactitud y precisión similar. Además, dado que no hubo clasificaciones de FN, la sensibilidad alcanza un valor de 1. Esto indica que el modelo es muy efectivo en identificar correctamente las clases sin omitir ninguna instancia positiva.

5.2 F1 Score

$$\text{F1} = \frac{2 \times precisión \times sensibilidad}{precisión + sensibilidad}$$

Precisión = 0.9
Sensibilidad = 1 (9)

$$\text{F1} = \frac{2 \times 0.9 \times 1}{0.9 + 1} = \frac{1.8}{1.9} = 0.947$$

Partiendo del concepto armónico que utiliza el F1 score para comparar la precisión y sensibilidad, se puede observar que el modelo presenta buenos resultados. La diferencia entre ambas métricas no es significativa. Esto indica que el modelo ITESOWordNet mantiene un buen desempeño en términos de F1 score, reflejando un equilibrio adecuado entre la precisión y sensibilidad.

5.3 Matriz de confusión

A continuación, se presenta la matriz de confusión, que grafica la clasificación efectuada por del modelo ITESOWordNet sobre el conjunto de datos de prueba.

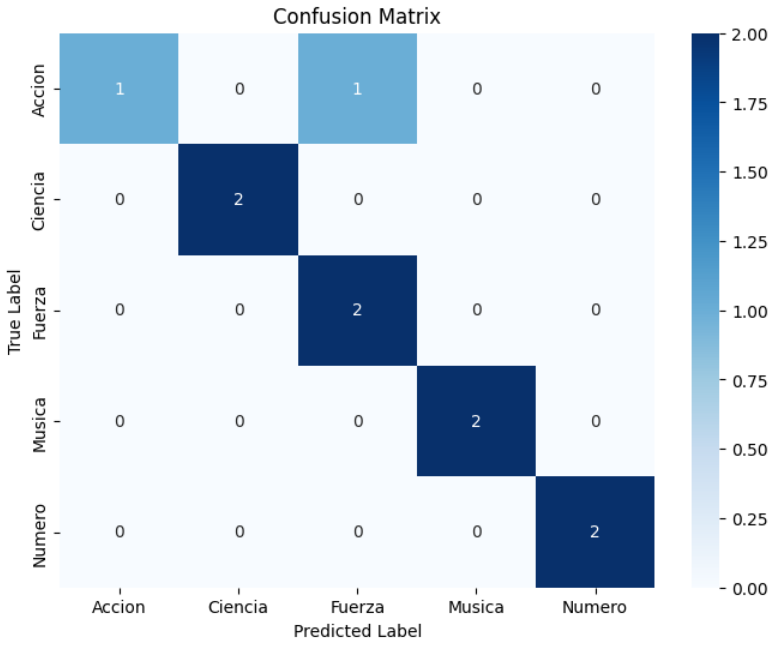

Figure 28 Matriz de confusión que muestra el desempeño de clasificación del modelo ITESOWordNet sobre el conjunto de datos de prueba

La matriz de confusión proporciona una representación detallada del rendimiento del modelo ITESOWordNet en la tarea de clasificación de palabras. A partir de ella, se puede observar que el modelo clasifica correctamente la mayoría de las instancias en sus respectivas clases.

Para la clase "Acción", el modelo realizó una clasificación correcta en una instancia, mientras que hubo una instancia clasificada incorrectamente como "Fuerza". En las clases "Ciencia", "Fuerza", "Música" y "Número", el modelo logró una clasificación perfecta, con dos instancias correctamente clasificadas en cada una de estas clases y sin ningún error.

Estos resultados indican que el modelo tiene un alto rendimiento en la clasificación de la mayoría de las clases, con un pequeño error en la clase "Acción". La matriz de confusión demuestra la capacidad del modelo para generalizar adecuadamente sobre los datos en un ambiente controlado y clasificar con precisión la mayoría de los videos en sus categorías correctas.

5.4 Contraste contra el estado del arte

Los resultados obtenidos con el modelo ITESOWordNet demuestran su efectividad para clasificar las palabras seleccionadas, alcanzando una exactitud del 90%. No obstante, es importante comparar este rendimiento con otros modelos del estado del arte en tareas similares de clasificación como [12].

Una diferencia clave entre nuestro modelo y los modelos del estado del arte radica en el idioma y la cantidad de datos disponibles. Muchos de los modelos más avanzados se han desarrollado y entrenado

utilizando grandes volúmenes de datos en inglés, mientras que ITESOWordNet se enfoca en el español latino.

Esta distinción lingüística puede influir significativamente en el rendimiento, ya que los modelos en inglés suelen beneficiarse de mayores recursos y datasets más extensos. A pesar de estas diferencias, el modelo ITESOWordNet muestra un desempeño competitivo. La exactitud alcanzada es comparable a la de los modelos del estado del arte cuando se ajustan a contextos de datos limitados y específicos de un idioma. Además, ITESOWordNet destaca por su capacidad de manejar variaciones y particularidades del español latino, una característica que no siempre está presente en modelos entrenados principalmente en inglés.

6. CONCLUSIONES

En este trabajo se desarrolló el modelo ITESOWordNet para la detección y clasificación de palabras a partir de la lectura de labios, utilizando técnicas de DL y ML. El modelo fue entrenado y validado con un conjunto de datos específico, MEXLR2023, diseñado y capturado en un entorno controlado con hablantes nativos del español latino en México.

Uno de los principales retos enfrentados durante el desarrollo de este proyecto fue la cantidad de datos disponibles. La creación y etiquetado manual de las 25,000 imágenes resultantes de los 500 videos representó un desafío significativo. Este proceso no solo fue laborioso, sino que también destacó la necesidad de contar con conjuntos de datos más amplios y variados para mejorar la robustez y capacidad generalización del modelo.

A pesar de estas limitaciones, el modelo ITESOWordNet demostró un desempeño competitivo, alcanzando una exactitud del 90% en la clasificación de palabras. Estos resultados son comparables a los modelos del estado del arte, especialmente considerando el enfoque en un idioma y contexto regional específicos. Además, muestran la viabilidad de desarrollar una aplicación que integre un modelo capaz de generar subtítulos. Esta aplicación tendría un desempeño sobresaliente y beneficiaría a personas con discapacidad auditiva en México.

Para futuras investigaciones, se recomienda incrementar la cantidad de datos de entrenamiento. Ampliar el conjunto de datos no solo podría mejorar la capacidad de generalización del modelo. También permitiría el entrenamiento de arquitecturas más complejas que puedan captar mejor las sutilezas del lenguaje visual. Además, es crucial expandir el enfoque del modelo a entornos no controlados. Esto permitirá evaluar y mejorar el desempeño del modelo en condiciones más variadas y realistas, aumentando su aplicabilidad práctica en situaciones cotidianas.

7. REFERENCIAS

[1] Y.-H. Goh, K.-X. Lau and Y.-K. Lee, "Audio-Visual Speech Recognition System Using Recurrent Neural Network," in *International Conference on Information Technology*, Bangkok, Thailand, 2019.

[2] I. D., R. Dmitry and K. Alexey, "AUTOMATIC LIP-READING OF HEARING IMPAIRED PEOPLE," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Moscow, Russia, 2019.

[3] S. Changchong, Z. Xinzhong, X. Huiying, P. Matti and L. Li, "Adaptive Semantic-Spatio-Temporal Graph Convolutional Network for Lip Reading," *IEEE TRANSACTIONS ON MULTIMEDIA*, vol. 24, pp. 3545 - 3557, 2022.

[4] J. Yuzhu, Z. Haijun, Z. Zhao and L. Ming, "CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances," *Science Direct*, 2020.

[5] A. Soad and E. Lamiaa, "Deep Convolutional Neural Network-Based Approaches for Face Recognition," *MDPI*, 2019.

[6] D. Nikita, A. Anamika, H. B. Smriti, M. Apurva and W. Kalyani, "Vision based Lip Reading System using Deep Learning," in *International Conference on Computing, Communication and Green Engineering*, Pune, India, 2021.

[7] E. Mohamed, M. M. Ayman and A. N. Abdurrahman, "A Silent Password Recognition Framework Based on Lip Analysis," *IEEE Access*, vol. 8, pp. 55354 - 55371, 2020.

[8] R. G, A. P, L. P, M. P. P and K. L., "Mouth Gesture Classification using Computational Intelligence," in *International Conference on Computing Methodologies and Communication*, Erode, India, 2022.

[9] G. Pooventhiran, A. Sandeep, K. Manthiravalli, D. Harish and R. D. Karthika, "Speaker-Independent Speech Recognition using Visual Features," *International Journal of Advanced Computer Science and Applications*, vol. 11, 2020.

[10] W. Huijuan, P. Gangqiang and C. Tingyu, "A Lip Reading Method Based on 3D Convolutional Vision Transformer," *IEEE Access*, vol. 10, pp. 77205 - 77212, 2022.

[11] K. B. Aleksey and A. I. Sergey, "Reconstruction of the Face Image from Speech Recording: a Neural Networks Approach," in *International Conference on Quality Management, Transport and Information Security, Information Technologies*, Yaroslavl, Russian Federation, 2021.

[12] A. Fernandez-Lopez and F. M. Sukno, "End-to-End Lip-Reading Without Large-Scale Data," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 30, pp. 2076 - 2090, 2022.

[13] S. d. Salud, "www.gob.mx," 28 Noviembre 2021. [Online]. Available: https://www.gob.mx/salud/prensa/530-con-discapacidad-auditiva-2-3-millones-de-personas-instituto-nacional-de-rehabilitacion?idiom=es. [Accessed 29 Mayo 2024].

[14] T. Vaj, "GRU vs. Bi-GRU: which one is going to win?," Medium, 2 February 2023. [Online]. Available: https://vtiya.medium.com/gru-vs-bi-gru-which-one-is-going-to-win-58a45ede5fba. [Accessed 21 June 2024].

[15] A. Rekik, B. Hamadou and W. Mahdi, "MIRACL-VC1," 2014. [Online]. Available: https://sites.google.com/site/achrafbenhamadou/-datasets/miracl-vc1. [Accessed 6 June 2024].

[16] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, New Jersey: Pearson, 2010.

[17] A. Gerón, Hands-On Machine Learning with Scikit-Learn and TensorFlow, CA: O'Reilly Media, Inc., 2017.

[18] R. Rojas, Neural Networks: A Systematic Introduction, Berlin: Springer, 1996.

[19] A. Charu C., Neural Networks and Deep Learning: A Textbook, NY, USA: Springer, 2018.

[20] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Massachusetts: MIT Press, 2016.

[21] S. Ahlawat, Reinforcement Learning for Finance: Solve Problems in Finance with CNN and RNN Using the TensorFlow Library, NJ, USA: Appress Media, LLC., 2023.

[22] V. Lakshmanan, M. Görner and R. Gillard, Practical Machine Learning for Computer Vision: End-to-End Machine Learning for Images, CA, United Statess: O'Reilly Media, Inc., 2021.

[23] Ó. Ramírez Jiménez, Python a fondo: Domine el lenguaje del presente y del futuro, Ciudad de México, México: Alfaomega, 2021.

[24] "What is Numpy?," numpy.org, 2008. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html#whatisnumpy. [Accessed 08 June 2024].

[25] "Pandas package overview," pandas.pydata.org, 2008. [Online]. Available: https://pandas.pydata.org/docs/getting_started/overview.html. [Accessed 8 June 2024].

[26] "Scikit Learn Getting Started," scikit-learn.org, 2007. [Online]. Available: https://scikit-learn.org/stable/getting_started.html. [Accessed 8 June 2024].

[27] "Citando a TensorFlow," tensorflow.org, 21 January 2022. [Online]. Available: https://www.tensorflow.org/about/bib?hl=es-419. [Accessed 8 June 2024].

[28] R. Shankar, "What is Google JAX? Everything You Need to Know," geekflare, 29 October 2022. [Online]. Available: https://geekflare.com/google-jax/. [Accessed 8 June 2024].

[29] F. Chollet, "Introduction to Keras for engineers," keras.io, 10 July 2023. [Online]. Available: https://keras.io/getting_started/intro_to_keras_for_engineers/. [Accessed 8 June 2024].

[30] "PyTorch documentation," pytorch.org, 2023. [Online]. Available: https://pytorch.org/docs/stable/index.html#. [Accessed 8 June 2024].

[31] E. Stevens and L. Antiga, Deep Learning with PyTorch: Escential Excerpts, Shelter Island, NY: Manning Publications Co., 2019.

[32] Tzutalin, "LabelImg," 2015. [Online]. Available: https://github.com/HumanSignal/labelImg. [Accessed 24 June 2024].

[33] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The PASCAL Visual Object Classes Challenge - A Retrospective," *Internation Journal of Computer Vision manuscript*, 2014.

[34] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances In Neural Information Processing Systems*, vol. 28, 2015.

[35] "What is Cross-Entropy Loss Function?," geeksforgeeks.org, 3 January 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-cross-entropy-loss-function/. [Accessed 27 June 2024].

[36] C. M. Bishop, Pattern Recognition and Machine Learning, New York, USA: Springer, 2006.

[37] R. Díaz, "Métricas de Clasificación," themachinelearners.com, 2020. [Online]. Available: https://www.themachinelearners.com/metricas-de-clasificacion/. [Accessed 27 June 2024].