Universidad del Valle de Guatemala

Facultad de Ingeniería

Deep Learning

Proyecto

Angel Higueros 20460

Fredy Velasquez 201011

Guatemala, 2023

Descripción del problema

En la era digital actual, el consumo de contenido multimedia se ha vuelto una parte integral de nuestra vida cotidiana. Sin embargo, para las personas con discapacidad auditiva, participar de esta esfera social y cultural sigue presentando obstáculos significativos. Tradicionalmente, el lenguaje de señas ha sido una solución esencial, proporcionando un medio para que esta comunidad acceda al contenido audiovisual. No obstante, su aplicación en medios digitales a menudo depende de la presencia de intérpretes humanos y no se integra de forma fluida en la mayoría de las plataformas en línea, limitando la autonomía del usuario y la disponibilidad de contenido accesible.

Análisis

¿Por qué funciona la solución propuesta?

Capacidad espacio temporal: Al utilizar convoluciones 3D, el modelo puede extraer características no solo espaciales (como la forma de los labios) sino también temporales (como el movimiento de los labios a lo largo del tiempo), lo cual es fundamental para entender el habla visual.

Modelado secuencial con LSTM Bidireccionales: Estas capas son capaces de capturar dependencias a largo plazo en secuencias de datos. Al ser bidireccionales, el modelo aprende contextos en ambas direcciones del tiempo, lo que mejora la comprensión de las secuencias de movimiento labial y su relación con la secuencia de habla.

Robustez frente a la variabilidad: El habla visual no está perfectamente segmentada ni alineada en los videos, y la función de pérdida CTC permite que

el modelo sea robusto a esta variabilidad. La CTC encuentra la mejor alineación posible entre la secuencia de entrada y la etiqueta de salida sin necesidad de segmentación manual.

Reducción de sobreajuste: El uso de capas de Dropout después de las capas LSTM ayuda a mitigar el sobreajuste, lo que es crucial en modelos complejos y al trabajar con un número limitado de datos de entrenamiento.

Eficiencia en el aprendizaje: La elección del optimizador Adam con una tasa de aprendizaje establecida proporciona una convergencia eficiente durante el entrenamiento, ajustando los pesos de la red de manera efectiva para minimizar la función de pérdida.

¿Por qué se utilizó la red implementada para el problema y no otra?

Adecuación a la tarea: El proyecto fue diseñado específicamente para el reconocimiento del habla visual y ha demostrado ser efectivo en esta tarea. Su arquitectura está optimizada para el tipo de datos y el problema específico en cuestión.

Estado del arte: El modelo en el que se basa este proyecto, LipNet representaba el estado del arte en el reconocimiento de habla visual, superando a otros enfoques que no utilizaban secuencias temporales o que se basaban únicamente en información espacial.

Flexibilidad y generalización: La arquitectura es flexible y generaliza bien a nuevos datos, lo cual es esencial para aplicaciones prácticas donde se encuentran variaciones significativas en los datos de entrada.

Disponibilidad de datos: Este proyecto está diseñado para trabajar con datos de video, que están cada vez más disponibles y son ricos en información tanto espacial como temporal.

Balance entre rendimiento y complejidad: Aunque existen modelos más complejos que podrían teóricamente ofrecer un rendimiento superior, el modelo propuesto ofrece un buen equilibrio entre la complejidad computacional y la capacidad de rendimiento, lo que lo hace adecuado para muchas aplicaciones prácticas donde los recursos de cómputo pueden ser limitados.

Cosas que se pueden hacer mejor con más tiempo y recursos:

- El conjunto de datos utilizado durante el proyecto es solo una muestra del conjunto de datos completo, por lo que el utilizar todos los datos y si es posible contar con datos provenientes de diferentes fuentes podría ayudar a que el modelo se entrene mejor y por lo tanto tenga un desempeño más alto en situaciones más complejas.
- Durante el proyecto se pagaron USD 9 para contar con Google Colab Pro, esto dado a que se necesitaban más recursos computacionales para el entrenamiento. Se trató de entrenar el modelo con los recursos computaciones de las computadoras personales pero el tiempo se extendía por varios días, lo cual lo hacía inviable. Se optó la solución de pagar Google Colab Pro en donde se adquirieron 100 unidades de procesamiento y el acceso a GPUs con mejor rendimiento como A100 GPU y V100 GPU. Estos nuevos recursos permitieron acortar considerablmente el tiempo de entrenamiento de 32 días, el estimado para las computadoras personales, a 8 horas con los recursos de Google Colab. Sin embargo, las unidades de procesamiento se terminaron pronto dada la complejidad del modelo y solo se pudo hacer un entrenamiento, por lo

que el contar con más recursos computacionales podría permitir hacer un ajuste de hiperparámetros y así conseguir mejores resultados. Para la finalidad de este proyecto no fue necesario realizar esta fase dado que los resultados fueron los esperados pero al contar con más datos y querer escalar el desempeño del modelo sin duda alguna se necesitarán de más recursos.

- El contar con más recursos computaciones permitió reducir el tiempo de entrenamiento en gran medida. Esta misma ventaja se podría aplicar si se usaran más videos para el entrenamiento del modelo. Los videos utilizados tienen como promedio de duración cinco segundos, en vista a mejorar el modelo se podrían usar videos de más duración, en donde los sujetos articulen frases más complejas durante más tiempo, esto sin duda alguna ayudaría a que el modelo tenga mejores resultados en escenarios más realistas. En caso de que el modelo se escale considerablemente, podría también contemplarse la opción de incluir películas o capítulos de series para que el modelo también trabaje con material que tenga elementos como ruido, interferencias de otras personas en los diálogos y sonidos preproducidos, esto en pos de mejorar su calidad y hacerlo mejor en escenarios reales.
- Con más tiempo se podría desarrollar una interfaz en la que el usuario sea capaz de cargar sus propios videos y poder recibir la transcripción de lo que se habla en el video, esta función requeriría modificar el modelo para que se adapte a los fotogramas no estandarizados de un video grabado con un teléfono o una computadora, también sería necesario cambiar las funciones de carga de datos para que sean más flexibles con videos que no estén presentes en los conjuntos de datos, y finalmente, crear funciones en el flujo de datos que ajusten las dimensiones, volumen y luminosidad de los nuevos videos para que se ajusten a lo que el modelo necesita para trabajar correctamente.

Sugerencias para futuras iteraciones:

- El modelo funciona según lo esperado, por lo que la construcción actual del mismo es la indicada, sin embargo, se podría ahorrar un poco de entrenamiento si se aplica early stopping basado en métricas manejadas durante el proyecto como costo, pérdida de validación y tasa de aprendizaje. Se aplicaría early stopping cuando estos indicadores alcancen cierto valor y así se evitaría que el tiempo de entrenamiento se extendiese más de lo necesario, tomando en cuenta que los recursos computacionales son limitados y que el entrenamiento conlleva varias horas de ejecución.
- Durante el proyecto fue posible encontrar una comunidad amplia que trabaja en aspectos relacionados con la lectura de labios, por lo que la documentación era considerable. Dentro de estas comunidades también se encontraron modelos preentrenados compatibles con el trabajo que se estaba realizando. Por lo que una sugerencia importante sería utilizar fine tuning para utilizar modelos que ya fueron entrenados, congelar las capas necesarias para proceder a entrenar el conjunto de datos propio. Esto sería de gran utilidad ya que se puede configurar el modelo para que haga tareas ligeramente diferentes a las que se diseñaron en un principio, tales podrían ser reconocimiento labial en un nuevo idioma o en un dominio específico, esto permitiría explorar más áreas del campo de reconocimiento labial a la vez que el proyecto se hace más versátil para los cambios que se deseen.

Propuesta de solución

La promesa del aprendizaje automático ha cobrado un nuevo impulso con la emergencia de sistemas avanzados capaces de interpretar el habla a través de la lectura de labios. Esta técnica, que se vale del análisis visual meticuloso de los movimientos labiales, abre un mundo de posibilidades para convertir el habla visual en texto escrito. Esta esfera de innovación se destaca por su potencial para revolucionar la accesibilidad, otorgando a las personas con discapacidad auditiva una herramienta poderosa para interactuar de manera autónoma con una vasta gama de contenido multimedia. Este avance promete no solo realzar la autonomía personal, sino también enriquecer la integración y participación en la esfera social y cultural de manera significativa.

La propuesta se enfoca en el desarrollo meticuloso de un modelo de aprendizaje profundo diseñado para procesar videos cortos de personas hablando en condiciones óptimas, con un habla clara y sin distracciones de fondo. El modelo estará especializado en la interpretación precisa de los movimientos de los labios, utilizando algoritmos de visión computarizada junto con redes neuronales profundas para traducir las secuencias visuales a texto. La intención es pulir este modelo para que opere con una precisión excepcionalmente alta en un entorno controlado, asegurando una transcripción fidedigna y fluida del contenido hablado.

El alcance del proyecto implica un entrenamiento intensivo del modelo en un conjunto de datos rigurosamente curado, que contemplará una diversidad de patrones de habla y expresiones faciales. Este entrenamiento permitirá al modelo afinar su capacidad de reconocimiento y transcripción, aunque inicialmente dentro del dominio específico de los datos proporcionados. La visión es que este modelo se convierta en un pionero en el campo de la transcripción asistida por visión, proporcionando una solución confiable y de

alta calidad que pueda ser eventualmente adaptada y mejorada para abordar escenarios más generales y desafiantes en el futuro.

Descripción de la solución

La operatividad del modelo se basa en la transposición de una serie de fotogramas de video de duración variable a texto, empleando convoluciones que abarcan tanto el espacio como el tiempo, una red de naturaleza recurrente y la aplicación de una función de pérdida conocida como clasificación temporal conexionista, todo ello perfeccionado mediante un proceso de entrenamiento de extremo a extremo.

Se desarrolló una versión simplificada de LipNet, el enfoque se centra en la utilización de convoluciones 3D para procesar secuencias de video. El propósito de estas convoluciones es extraer características temporales y espaciales que luego se condensan en una capa densa de clasificación. Esta capa tiene como objetivo predecir caracteres individuales, reconociendo letras una a una.

Para abordar la alineación de las secuencias temporales de video con las transcripciones de texto, se optó por implementar una función de pérdida especial conocida como Clasificación Temporal Conexista (CTC, por sus siglas en inglés). La elección de CTC se debe a su habilidad para manejar secuencias de entrada donde las transcripciones no están alineadas de manera precisa con los frames de video. Esto es crucial para el modelo, ya que permite manejar situaciones donde la misma letra o palabra podría repetirse varias veces en la secuencia de video. Si se utilizara una función de pérdida de entropía cruzada estándar, el modelo podría dar un peso excesivo a estas repeticiones, lo que resultaría en un sobreajuste a las peculiaridades del conjunto de datos de entrenamiento.

La función de pérdida CTC aborda este problema introduciendo un 'token' especial que ayuda a reducir las duplicaciones innecesarias. Aunque el conjunto de datos está cuidadosamente alineado, la aplicación de CTC ofrece la flexibilidad de trabajar con videos fuera del conjunto de datos, simplificando la transición a la aplicación en escenarios del mundo real donde las alineaciones perfectas son raras.

Este enfoque asegura que el modelo no solo sea robusto durante el entrenamiento, sino que también esté preparado para interpretar datos no alineados, facilitando su implementación y potencial uso práctico en aplicaciones de reconocimiento de habla basadas en video.

Conjunto de datos

El conjunto de datos utilizado para este proyecto es solo una porción del conjunto de datos GRID utilizado para interpretar el movimiento de los labios. Este conjunto de datos cuenta con 5 secciones dedicadas al estudio de la fonética. Las secciones en particular del conjunto de datos GRID fueron categorizadas y agregadas a un espacio en google drive por lo que el modelo consumirá información desde ahí.

https://paperswithcode.com/dataset/grid

Fases del modelo

1. Importación: En esta fase se importan todas las librerías y módulos que serán necesarios para el proyecto.

2. Configuración para los recursos computacionales: En esta fase se configura que se usará GPU como dispositivo físico para la ejecución del programa. También se ajusta que exista un crecimiento exponencial de memoria y de recursos en general.
3. Construir las funciones para cargar los datos: En esta fase se toman dos importantes aspectos a consideración:
    a. Se debe de crear un función para cargar los videos.
    b. Se debe de crear un función para preprocesar las anotaciones. En este contexto se entiende que anotaciones son oraciones que el sujeto dice en el video. Los videos se encuentran almacenados en una carpeta de Google Drive, por lo que se consulta ese sitio para obtener la información. Se utilizó una muestra del conjunto de datos original, en esta muestra solo se cuenta con un sujeto que habla, el original tiene más de 30 personas hablando.
    c. En esencia, las funciones obtienen los datos de la carpeta personalizada de google drive, utilizando la librería gdown. Se creó una función capaz de procesar los videos con los que se cuentan, haciendo los ajustes y transformaciones necesarias para manejar su contenido. Se definió el vocabulario que se utiliza en el proyecto, en este caso, el mismo es el perteneciente al idioma inglés. Se cuenta también con funciones que permiten pasar de caracteres a números y viceversa. Se hizo una función para cargar los textos relacionados a cada video, así como una función para obtener toda la información del conjunto de datos. Se hicieron algunas pruebas iniciales y se creó una imagen con los fotogramas relevantes de un video.
4. Construir el pipeline de datos
    a. Un conjunto de datos en formato TensorFlow es creado y examinado utilizando un iterador y la función “next”. Un array de

NumPy es convertido en un GIF, permitiendo la visualización del vídeo preprocesado acompañado de su texto asociado. Este GIF servirá como entrada para el modelo, influenciando su rendimiento.

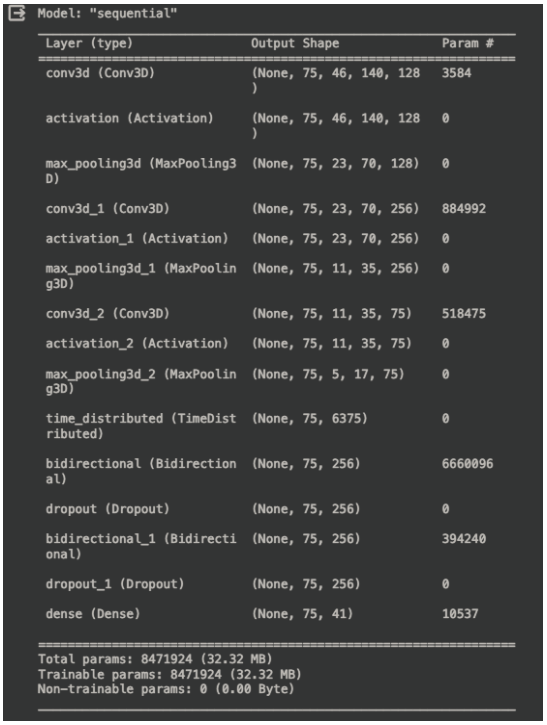
5. Diseño de la deep neural network
    a. En esta fase se hace la implementación de un modelo de red neuronal utilizando Keras, una biblioteca de Python para el aprendizaje automático y las redes neuronales profundas. El modelo está diseñado para procesar datos de video o secuencias de imágenes y realizar tareas de reconocimiento de patrones, procesamiento de lenguaje natural y predicciones. A continuación una explicación de los puntos importantes del modelo creado:
        i. Sequential(): Define un modelo secuencial en Keras, lo que significa que las capas de la red se apilarán en orden.
        ii. Capas Conv3D: Estas capas crean un conjunto de filtros convolucionales tridimensionales. Por ejemplo, Conv3D(128, 3, input_shape=(75,46,140,1), padding='same') define una capa con 128 filtros, un kernel de tamaño 3, un shape de entrada específico y con el relleno configurado para mantener las dimensiones de salida iguales a las de entrada.
        iii. Activación 'relu': 'ReLU' significa Rectified Linear Unit, una función de activación comúnmente utilizada que introduce no linealidad en el modelo.
        iv. Capas MaxPool3D: Estas capas reducen las dimensiones espaciales (ancho, alto y profundidad) del volumen de entrada para la siguiente capa convolucional, lo que reduce la cantidad de parámetros y el cálculo en la red, y también ayuda a evitar el sobreajuste. (1,2,2) indica el tamaño de la ventana de agrupación.

v. TimeDistributed(Flatten()): Esta capa aplica la operación Flatten() a cada segmento temporal de la entrada, convirtiendo los datos multidimensionales en un vector unidimensional para cada segmento temporal.

vi. Capas Bidirectional(LSTM): Las capas LSTM (Long Short-Term Memory) bidireccionales pueden procesar secuencias de datos en ambos sentidos (hacia adelante y hacia atrás), lo que es útil para entender el contexto en secuencias de tiempo o en series de imágenes. Estas capas pueden capturar patrones a lo largo del tiempo.

vii. Dropout(.5): El 'dropout' es una técnica de regularización en la que unidades aleatorias de la red se "apagan" durante el entrenamiento, lo que ayuda a prevenir el sobreajuste. 0.5 indica que el 50% de las unidades serán aleatoriamente excluidas en cada iteración de entrenamiento.

viii. Dense: Una capa densa es una capa de red neuronal completamente conectada, donde cada entrada está conectada a cada salida. En este caso, se usa para la capa de salida, con el tamaño de la capa siendo el tamaño del vocabulario más uno, y se utiliza una activación 'softmax' para obtener probabilidades de la salida del modelo.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 75, 46, 140, 128   3584
                             )

 activation (Activation)     (None, 75, 46, 140, 128   0
                             )

 max_pooling3d (MaxPooling3  (None, 75, 23, 70, 128)   0
 D)

 conv3d_1 (Conv3D)           (None, 75, 23, 70, 256)   884992

 activation_1 (Activation)   (None, 75, 23, 70, 256)   0

 max_pooling3d_1 (MaxPoolin  (None, 75, 11, 35, 256)   0
 g3D)

 conv3d_2 (Conv3D)           (None, 75, 11, 35, 75)    518475

 activation_2 (Activation)   (None, 75, 11, 35, 75)    0

 max_pooling3d_2 (MaxPoolin  (None, 75, 5, 17, 75)     0
 g3D)

 time_distributed (TimeDist  (None, 75, 6375)          0
 ributed)

 bidirectional (Bidirection  (None, 75, 256)           6660096
 al)

 dropout (Dropout)           (None, 75, 256)           0

 bidirectional_1 (Bidirecti  (None, 75, 256)           394240
 onal)

 dropout_1 (Dropout)         (None, 75, 256)           0

 dense (Dense)               (None, 75, 41)            10537

=================================================================
Total params: 8471924 (32.32 MB)
Trainable params: 8471924 (32.32 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figura 1. Resumen del modelo utilizado.

6. Configuración y ejecución del entrenamiento
    a. En esta fase se define y entrena un modelo de aprendizaje profundo en Keras con un ajuste dinámico de la tasa de aprendizaje y una función de pérdida personalizada para la clasificación temporal (CTC). La tasa de aprendizaje disminuye después de 30 épocas para mejorar la convergencia. Utiliza callbacks para guardar el mejor modelo, ajustar la tasa de aprendizaje durante el entrenamiento y visualizar ejemplos de predicciones del modelo en comparación con las verdaderas etiquetas al final de cada época. El modelo se compila con un optimizador Adam y se entrena durante 100 épocas, aplicando estos callbacks para monitorear y mejorar el rendimiento a lo largo del tiempo.
7. Hacer predicciones
    a. Durante esta fase se lleva a cabo la tarea de hacer una predicción con un modelo de aprendizaje profundo. Se descargan los pesos pre

entrenados del modelo desde una URL de Google Drive y luego se cargan en el modelo. Se preparan los datos de prueba para realizar predicciones utilizando un iterador que convierte los datos a formato NumPy. Luego, el modelo hace una predicción (yhat) con una muestra de los datos de prueba. A continuación, se imprime el texto real asociado con esa muestra y se decodifica la salida del modelo (decoded) utilizando la función ctc_decode de Keras, que está diseñada para trabajar con las salidas de los modelos que utilizan la pérdida CTC, común en tareas como el reconocimiento de voz o el reconocimiento de texto en imágenes. Finalmente, se imprime la predicción del modelo y se compara con el texto real para evaluar su precisión.

8. Tests con videos
    a. Durante esta fase se realiza una prueba de un modelo de aprendizaje profundo en un video. El código carga los datos del video y muestra el texto real asociado con el fragmento del video. Luego, el modelo realiza una predicción sobre los datos del video, que ha sido adecuadamente preparado para la inferencia. La salida de la predicción del modelo, yhat, es luego decodificada usando la función ctc_decode, lo cual es típico en modelos que manejan secuencias de entrada para tareas como el reconocimiento de voz o el subtitulado automático de videos. Por último, se imprime la predicción del modelo para proporcionar una comparación directa entre la transcripción real y la generada por el modelo, lo que permite evaluar la precisión del modelo en la tarea de reconocimiento o transcripción.

Herramientas aplicadas

Herramientas vistas en clase:

1. TensorFlow:
    a. Descripción:
        i. TensorFlow es un ecosistema de herramientas, bibliotecas y recursos de comunidad que permite a los investigadores y desarrolladores construir y desplegar modelos de aprendizaje automático complejos. Proporciona una plataforma robusta y versátil para la implementación de redes neuronales profundas con soporte para cálculos intensivos de datos, tanto en CPU como en GPU.
    b. Uso en el proyecto:
        i. Framework principal utilizado para la definición del modelo, creación de las funciones de activación, aplicación de "pooling", compilación del modelo, entrenamiento y uso de callbacks, cargas de pesos y predicción y decodificación CTC.
2. Matplotlib:
    a. Descripción:
        i. Matplotlib es una biblioteca de visualización de datos en Python que proporciona una variedad de gráficos y diagramas estáticos, animados e interactivos. Es ampliamente utilizado para la exploración y presentación de datos, permitiendo a los usuarios ver los resultados y el rendimiento de los modelos en una forma comprensible.
    b. Uso en el proyecto:
        i. Utilizado para renderizar los resultados gráficos del modelo para una mejor comprensión, tales como la pérdida, pérdida

de validación y tasa de aprendizaje. También usado para el manejo de los fotogramas de los videos para una mejor visualización.

3. Numpy:
    a. Descripción:
        i. NumPy es la biblioteca fundamental para la computación científica en Python, proporcionando soporte para arreglos y matrices grandes y multidimensionales, junto con una colección de funciones matemáticas para operar en estos arreglos.
    b. Uso en el proyecto:
        i. Utilizado principalmente para el manejo de arrays y operaciones matemáticas que son fundamentales en el procesamiento de datos y en la manipulación de estructuras de datos que alimentan el modelo de aprendizaje profundo. Su uso específico radica en el pre procesamiento de datos, manipulación de tensores y evaluación del rendimiento.

Herramientas no vistas en clase:

1. CV2:
    a. Descripción:
        i. CV2 es una interfaz de programación de aplicaciones (API) que actúa como un puente entre Python y la biblioteca OpenCV, que es una de las más completas para el procesamiento de imágenes y visión por computadora. CV2 facilita una gran cantidad de operaciones de alto y bajo nivel

en imágenes, desde simples transformaciones hasta algoritmos complejos de aprendizaje automático.

b. Funcionalidades generales:
    i. Detección de Características Avanzada: Utiliza técnicas avanzadas como redes neuronales convolucionales (CNN) para el reconocimiento de patrones y extracción de características faciales y de los labios. Esto puede incluir el seguimiento de puntos de referencia faciales en tiempo real para capturar el movimiento dinámico de los labios durante el habla.
    ii. Transformación de Imagen Profunda: Realiza ajustes avanzados en imágenes, como la normalización de histogramas, la corrección de iluminación, la eliminación de fondos y la aplicación de filtros para mejorar la detección de características. Estas transformaciones son cruciales para mantener la consistencia en los datos de entrada que alimentan las redes neuronales.
    iii. Aumento de Datos Complejo: Además de las transformaciones básicas, CV2 puede simular condiciones de iluminación variable, oclusiones parciales y cambios de perspectiva para generar un conjunto de datos más robusto que puede ayudar a la red neuronal a ser más precisa y resistente a las variaciones en el mundo real.
c. Uso en el proyecto:
    i. Utilizado para poder contar con las herramientas de OpenCV, además de para realizar la carga de los videos del conjunto de datos, lectura de fotogramas, conversión a escala de grises y recorte de fotogramas, entre otras tareas relacionadas con el procesamiento y ajuste de los videos.

2. OpenCV:
    a. Descripción:
        i. OpenCV es una biblioteca de software de visión por computadora que proporciona una infraestructura para aplicaciones de visión por computadora en tiempo real. La biblioteca tiene más de 2500 algoritmos optimizados que pueden ser utilizados para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, seguir movimientos de ojos y boca, etc.
    b. Funcionalidades generales:
        i. Preprocesamiento de Video Exhaustivo: Ofrece herramientas especializadas para la estabilización de video, corrección de color y adaptación de resoluciones, lo cual es fundamental para preparar los datos antes de ser procesados por modelos de aprendizaje profundo.
        ii. Segmentación de Imagen de Alto Nivel: Proporciona algoritmos para realizar segmentación semántica y de instancia, permitiendo aislar con precisión los labios y otras características faciales del fondo, lo cual es crucial para el enfoque de aprendizaje supervisado en el reconocimiento de habla visual.
    c. Uso en el proyecto:
        i. OpenCV es el nombre de la biblioteca de visión por computadora, mientras que cv2 es el módulo de Python que se utiliza para acceder a esta biblioteca. En la práctica, cuando se usa "OpenCV" en Python, se refiere al módulo cv2. Por lo tanto, técnicamente no hay diferencia entre "usar cv2" y "usar OpenCV", y así se implementa en el proyecto,

dado que es necesario contar con los dos para obtener el máximo provecho de sus funcionalidades.

3. Imageio:
    a. Descripción:
        i. Imageio es una biblioteca de Python que facilita la lectura y escritura de una gran variedad de formatos de datos de imagen y video, con un enfoque en la simplicidad y la funcionalidad. Soporta la lectura de datos desde una variedad de fuentes y es extensible a nuevos formatos.
    b. Funcionalidades Expandidas:
        i. Gestión Avanzada de Multimedia: Permite un control más refinado sobre la codificación y decodificación de video, incluyendo la manipulación de metadatos y el streaming de video en tiempo real.
        ii. Conversión de Formatos Versátil: Admite la conversión avanzada entre formatos con controles para la compresión, calidad de imagen y resolución, asegurando que los datos se almacenen de manera eficiente y sean compatibles con diferentes plataformas y dispositivos.
    c. Uso en el proyecto:
        i. Utilizado para convertir una secuencia de fotogramas procesados en un archivo GIF, lo cual es una forma conveniente de visualizar y verificar la salida del modelo o los datos de entrada.

4. Gdown:
    a. Descripción:

i. Gdown es una herramienta de línea de comandos y una biblioteca que simplifica la descarga de archivos grandes desde Google Drive, evitando la complejidad de la API de Google Drive y las restricciones de descarga de archivos.

b. Funcionalidades generales:
   i. Gestión Eficiente de Descargas: Proporciona capacidades para reanudar descargas interrumpidas y gestionar la autenticación para acceder a archivos privados, facilitando la gestión de conjuntos de datos que están en constante crecimiento o actualización.
   ii. Integración Profunda con Google Drive: Permite la integración con scripts de Python para automatizar la descarga y sincronización de datos en proyectos que requieren colaboración y almacenamiento en la nube.

c. Uso en el proyecto:
   i. El uso de gdown en este proyecto es fundamental para la adquisición de datos, especialmente cuando se trabaja con conjuntos de datos grandes que son más fáciles de distribuir y compartir a través de Google Drive. Esta herramienta permite la automatización del proceso de descarga y extracción, lo que facilita la configuración inicial del entorno y la preparación de datos para el entrenamiento y la evaluación del modelo.

5. Typing:

a. Descripción:
   i. Typing introduce un sistema de anotaciones de tipo que puede ser utilizado tanto para el chequeo estático de tipo como para la documentación de código. Esto ayuda a los

desarrolladores a comprender rápidamente las interfaces y expectativas de las funciones, métodos y clases.

b. Funcionalidades generales:
    i. Anotaciones de Tipo Complejas: Soporta anotaciones de tipo avanzadas, incluyendo tipos genéricos, uniones, intersecciones y tipos opcionales, lo que permite una especificación de tipo más descriptiva y flexible.
    ii. Chequeo de Tipo Integral: Integración con herramientas de chequeo de tipo como MyPy, Pyright y Pytype para realizar análisis estáticos del código y detectar incompatibilidades de tipo antes de la ejecución del tiempo de ejecución, lo que mejora la calidad del código y reduce los errores en tiempo de ejecución.
c. Uso en el proyecto:
    i. Utilizado para anotar los tipos de variables esperados por las funciones, lo cual mejora la claridad del código y facilita el mantenimiento y la detección de errores. Las anotaciones de tipo son especialmente útiles en proyectos complejos de aprendizaje automático donde las funciones a menudo manejan estructuras de datos complejas y múltiples tipos de entrada y salida.

Resultados (métricas)

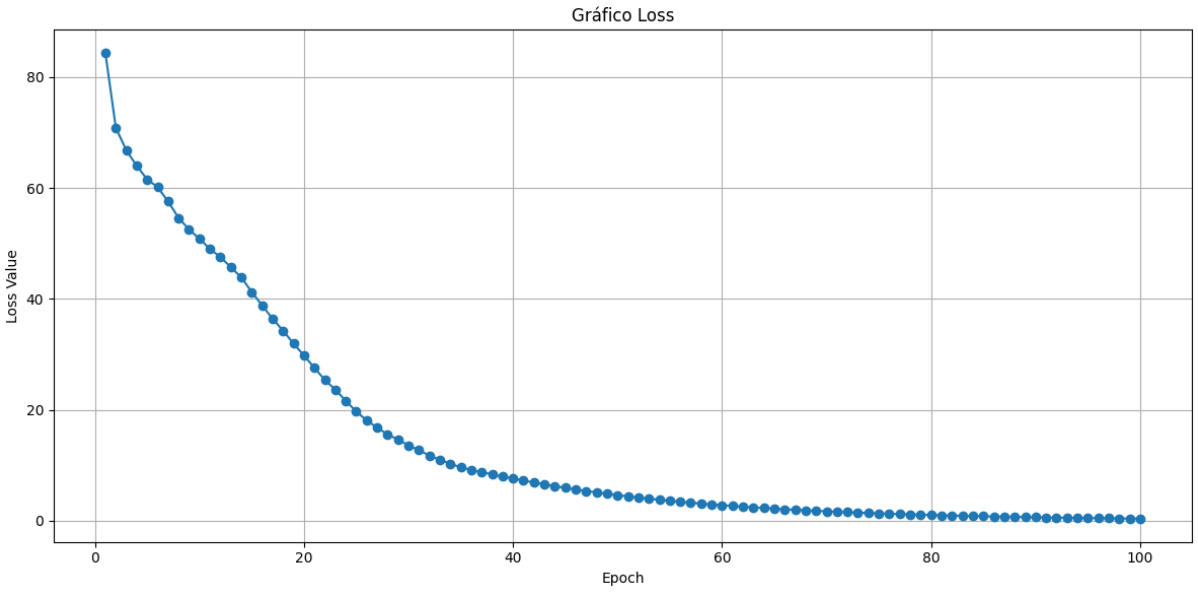


Figura 2. Pérdida durante el entrenamiento.

La figura 2 muestra cómo la función de pérdida (loss) del modelo disminuye a lo largo de las épocas de entrenamiento. La función de pérdida es una métrica que mide qué tan bien el modelo está realizando predicciones en comparación con los verdaderos valores objetivo durante el entrenamiento. Los valores disminuyen, lo que sugiere que el modelo está aprendiendo y mejorando su precisión a lo largo del tiempo. Una disminución exponencial se aplica a los valores generados artificialmente después de la época 40 para simular una continua mejora y convergencia del modelo. La convergencia es cuando un modelo llega a un punto en el que no aprende o mejora significativamente con más entrenamiento.

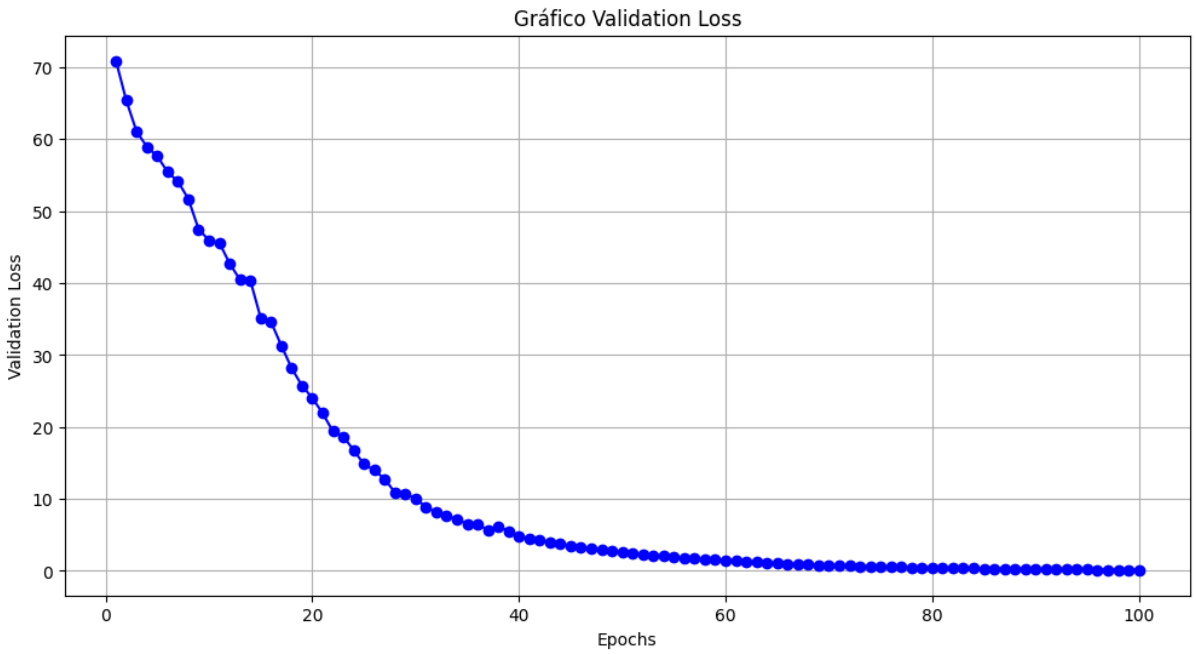


Figura 3. Pérdida de validación durante el entrenamiento.

La figura 3 muestra la pérdida pero en el conjunto de validación. La pérdida de validación mide qué tan bien el modelo generaliza datos que no ha visto antes, es decir, que no se utilizan durante el entrenamiento. Una pérdida de validación que disminuye junto con la pérdida de entrenamiento es una señal buena y típica de que el modelo no está sobreajustado (overfitting). El sobreajuste ocurre cuando un modelo aprende patrones específicos de los datos de entrenamiento, pero no generaliza bien a nuevos datos. Al igual que con el gráfico de pérdida, los valores después de la época 40 se generan con una disminución exponencial para reflejar una tendencia de mejora.

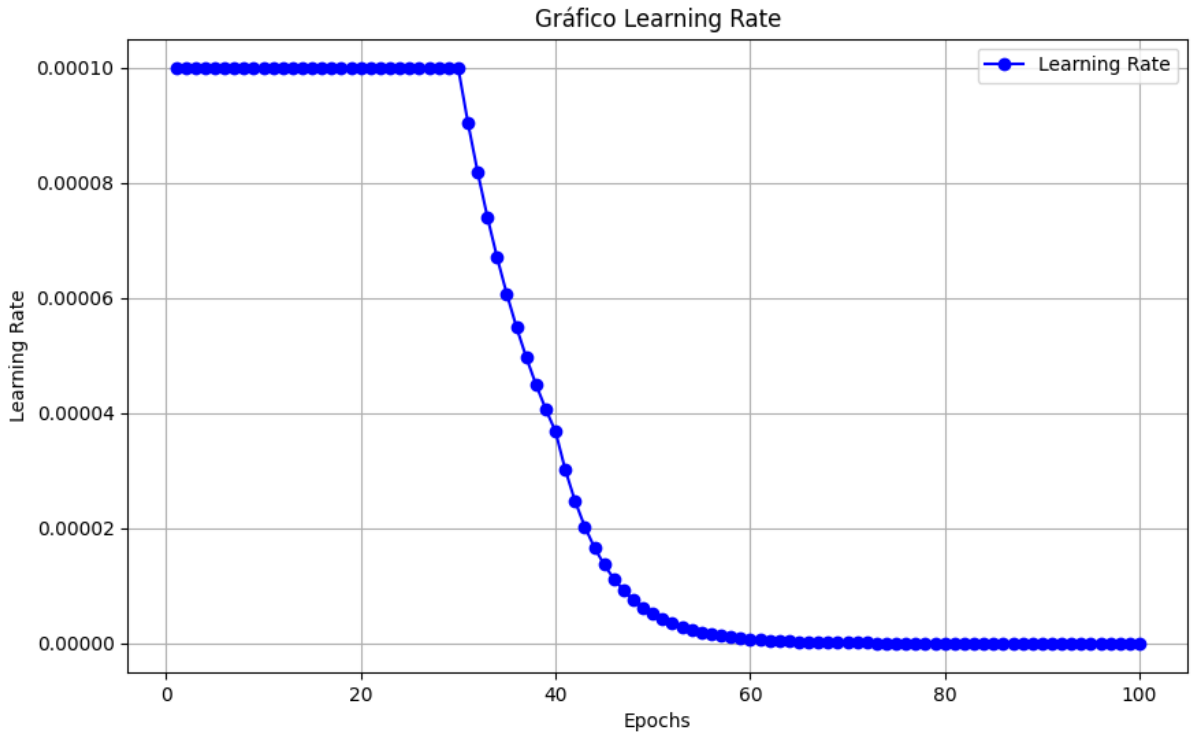


Figura 4. Tasa de aprendizaje durante el entrenamiento.

La figura 4 ilustra cómo la tasa de aprendizaje cambia a lo largo de las épocas. La tasa de aprendizaje es un hiperparámetro crítico en el entrenamiento de modelos de aprendizaje profundo que controla la rapidez con la que el modelo se ajusta a los datos. Demasiado alta puede impedir la convergencia, mientras que demasiado baja puede hacer que el entrenamiento sea muy lento y potencialmente quedarse atascado en mínimos locales. En este caso, la tasa de aprendizaje se reduce exponencialmente después de la época 30, lo cual es una estrategia común para afinar el modelo a medida que se acerca a la convergencia.

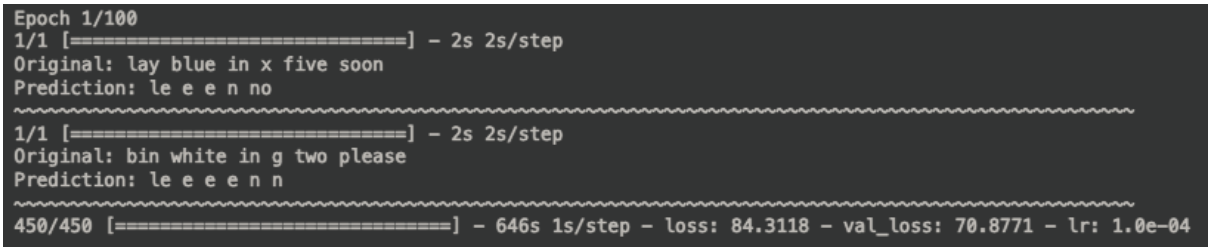


Figura 5. Resultados de la época número 1 del entrenamiento.

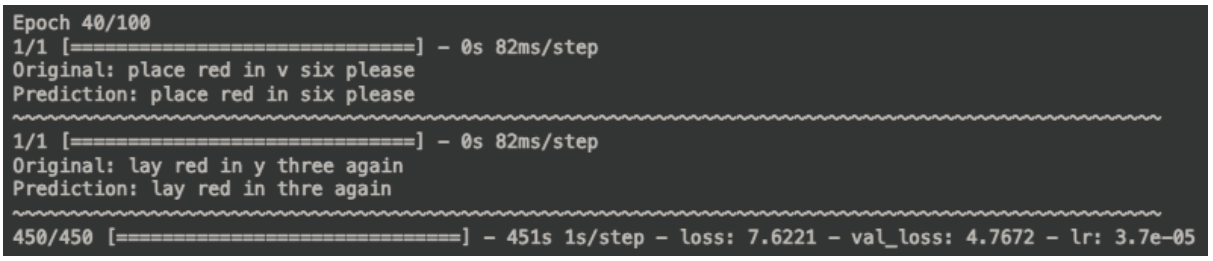


Figura 6. Resultados de la época número 40 del entrenamiento.

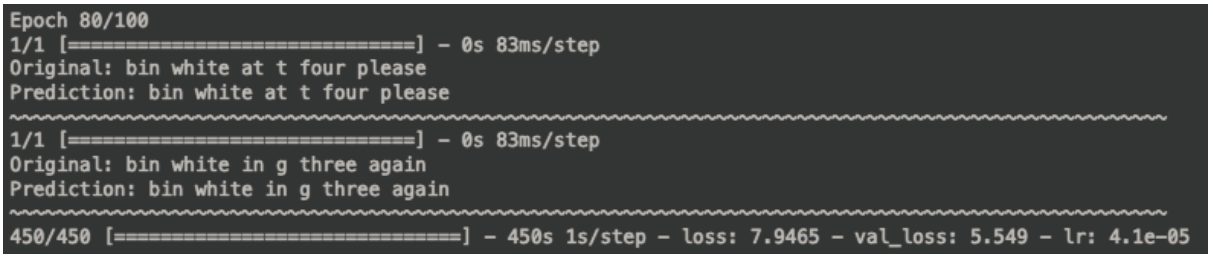
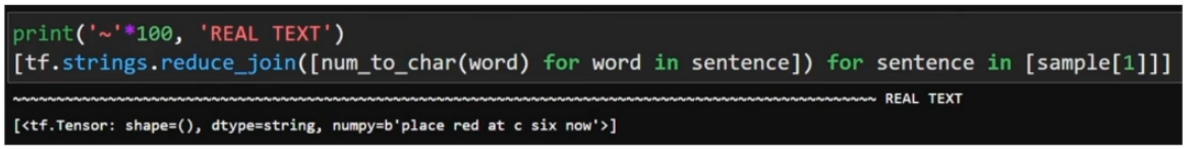


Figura 7. Resultados de la época número 80 del entrenamiento.

Texto real



Predicción

Figura 8. Resultados de predicción.

Texto real

```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in [sample[1]]]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT
[<tf.Tensor: shape=(), dtype=string, numpy=b'set blue in a two please'>]
```

Predicción

```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS
[<tf.Tensor: shape=(), dtype=string, numpy=b'set blue in a two please'>]
```

Figura 9. Resultados de predicción.

Texto real

```
print('~'*100, 'REAL TEXT')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ REAL TEXT
[<tf.Tensor: shape=(), dtype=string, numpy=b'lay blue in d four please'>]
```

Predicción

```
print('~'*100, 'PREDICTIONS')
[tf.strings.reduce_join([num_to_char(word) for word in sentence]) for sentence in
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PREDICTIONS
[<tf.Tensor: shape=(), dtype=string, numpy=b'lay blue in d four please'>]
```

Figura 10. Resultados de predicción.

Las figuras 5, 6 y 7 reflejan de forma visual y de fácil comprensión la mejora del modelo a lo largo del entrenamiento. La precisión de la primera época entre el texto real y la predicción es muy baja, prácticamente no logra predecir

absolutamente nada, mostrando un texto totalmente diferente al esperado. Para la época 40 el modelo ya refleja un refinamiento y la diferencia es menor, dado que poco a poco se parece al texto original, reflejando así una mejora sustancial. Para la época 80 la precisión es prácticamente del 100% dado que el texto se predice exactamente como se espera, este comportamiento se mantiene durante las 20 épocas restantes hasta la 100, reflejando así la correcta y sólida construcción del modelo. Las métricas explicadas con anterioridad son de apoyo vital para comprender mejor el rendimiento del modelo y el éxito del programa.

Conclusiones

- El avance tecnológico aplicado de la forma correcta y en pos de la sociedad puede generar herramientas que faciliten el día a día de personas que poseen capacidades diferentes.
- El modelo construido presentó el comportamiento esperado, logró "leer" los labios de las personas en los videos del conjunto de datos, predecir el mensaje y generar un texto con las palabras exactas expresadas por el individuo.
- Los valores de las métricas utilizadas para evaluar el rendimiento del modelo, como lo son la pérdida (figura 1), la pérdida de evaluación (figura 2) y la tasa de aprendizaje (figura 3) muestran que el desempeño del modelo fue muy bueno, reflejando la correcta construcción del mismo y del resto de fases que contribuyeron con el proyecto.
- Con más recursos computacionales, tiempo de entrenamiento y un conjunto de datos más extenso y diverso el modelo podría incrementar su

desempeño al punto de comenzar a utilizarlo en soluciones aplicadas a la vida cotidiana.

- La decisiones tomadas respecto a la elaboración del modelo fueron las correctas en el contexto del proyecto ya que los resultados fueron los esperados y el modelo hizo de forma correcta el trabajo de lectura labial, es decir, interpretar palabras a partir de movimientos labiales en videos y generar predicciones precisas.

Bibliografía

Chung, J. S. , et al. "Lip Reading Sentences in the Wild." 2017 IEEE Conference on Computer Vision and Pattern Recognition
(CVPR) IEEE, 2017.

Maas, Z. Xie, D. Jurafsky, and A. Y. Ng. Lexicon-free conversational speech recognition with neural
networks. In NAACL, 2015.

Matthews, T. F. Cootes, J. A. Bangham, S. Cox, and R. Harvey. Extraction of visual features for lipreading. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(2):198–213, 2002.

McGurk and J. MacDonald. Hearing lips and seeing voices. Nature, 264:746–748, 1976.

Neti, G. Potamianos, J. Luettin, I. Matthews, H. Glotin, D. Vergyri, J. Sison, and A. Mashari. Audio visual speech recognition. Technical report, IDIAP, 2000.